

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

Факультет
 Вычислительной техники

Кафедра
 МОиПЭВМ

Направление подготовки 09.03.02 «Информационные системы и технологии»

БАКАЛАВРСКАЯ РАБОТА

на тему

Информационно-обучающая система для
проведения занятий по физике

Студент Орлушин Андрей Александрович
 (подпись, дата) (ФИО полностью)
 Руководитель Прошкина Е.Н.
 (подпись, дата) (фамилия, инициалы)
 Нормоконтролёр Такташкин Д.В.
 (подпись, дата) (фамилия, инициалы)

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

Заведующий кафедрой Макарычев П.П.
 (подпись) (фамилия, инициалы)

Работа защищена с отметкой 4 (протокол заседания ГЭК от 14.06.21 № 2)

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

Пенза, 2021

Федеральное государственное бюджетное образовательное учреждение
высшего образования

Пензенский государственный университет
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
Вычислительной техники

Кафедра
МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
П.П. Макарычев
«28» декабря 2020 г

ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ БАКАЛАВРА

1. Студент Орлушин Андрей Александрович гр. 17ВИ1 факультета ВТ
<Фамилия, имя, отчество полностью>
2. Тема работы Информационно-обучающая система для проведения занятий по физике
Тема утверждена приказом ИГУ № от " 28 " декабря 2020 г
3. Руководитель работы к.т.н., доцент кафедры «МОиПЭВМ» Прошкина Е.Н.
4. Задание на работу (назначение разработки, исходные данные и т.п.)
 - 1) Разработать информационно обучающую систему для проведения занятий по физике в виде клиент-серверного приложения
 - 2) Среда выполнения:
Языковые средства разработки: C#, SOL
Инструментальные средства разработки: Microsoft Visual Studio Management Studio
Технологии разработки: ООП
 - 3) Основные функции информационной системы:
 - а) регистрация и авторизация пользователей;
 - б) предоставить возможность учителю добавлять в базу новые классы;
 - в) предоставить возможность учителю добавлять новых учеников;
 - г) предоставить возможность учителю добавлять в базу лекции и лабораторные работы;
 - д) предоставить возможность учителю выставлять результаты за выполненные задания;
 - е) предоставить возможность ученикам выполнять лабораторные работы;
 - ж) предоставить возможность ученикам использовать лекционные

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	<i>Анализ предметной области и постановка задачи</i>	10%	15.04.2021 – 20.04.2021	<i>Громов</i>
2	<i>Проектирование</i>	25%	20.04.2021 – 30.04.2021	<i>Громов</i>
3	<i>Реализация</i>	30%	30.04.2021 – 19.05.2021	<i>Громов</i>
4	<i>Контроль качества программного обеспечения</i>	5%	19.05.2021 – 15.05.2021	<i>Громов</i>
5	<i>Оформление пояснительной записки</i>	10%	15.05.2021 – 10.06.2021	<i>Громов</i>

Дата выдачи задания " 28 " декабря 2021г

Руководитель бакалаврской работы *Громов* 14.06.21г.
(подпись, дата)

Задание к исполнению принял студент *Громов* 14.06.21г.
(подпись, дата)

Работу к защите допустить
 Декан факультета *А.Р. Громов*
 (подпись, дата)
 24.06.2021

Реферат

Пояснительная записка содержит 84 листа, 33 рисунка, 19 таблиц, 15 использованных источников, 2 приложения.

ИНФОРМАЦИОННО-ОБУЧАЮЩАЯ СИСТЕМА, ФИЗИКА, БАЗА ДАННЫХ, SQL, C#, UML 2.0.

Объектом разработки является информационно-обучающая система для проведения занятий по физике.

Цель работы – разработка информационно-обучающей системы для проведения занятий по физике.

В результате проведенной работы создана система, которая позволяет проводить лекционные занятия, лабораторные работы с большей эффективностью, анализировать успеваемость учащихся за выполненные работы.

Разработка проводилась на языке программирования C#, в качестве сред разработки использовались среда программирования Visual Studio 2017.

					ПГУ 09.03.02 – 08БР171.14 ПЗ				
Изм.	Лист	№ докум.	Подп.	Дата	1. Информационно-обучающая система для проведения занятий по физике. Пояснительная записка.	Лит.	Лист	Листов	
Разраб.		Орлушин А.А.		14.06.2017					
Пров.		Прошкина Е.Н.		14.06.2017			5	84	
Реценз.						Гр. 17ВИ1			
Н. контр.		Такташкин Д.В.		22.06.2017					
Утв.		Макарычев П.П.							

Содержание

Введение.....	12
1 Анализ предметной области	15
1.1 Понятие виртуальной лаборатории	15
1.2 Анализ существующих информационно-обучающих систем.....	16
1.3 Анализ требований.....	22
1.3.1 Функциональные требования.....	22
1.3.2 Нефункциональные требования	24
2 Проектирование системы	25
2.1 Анализ бизнес-процессов образовательной организации.....	25
2.2 Проектирование логической модели базы данных.....	29
2.3 Проектирование ПО	32
2.3.1 Диаграмма вариантов использования	32
2.3.2 Диаграмма классов.....	34
2.3.3 Диаграмма состояний	38
2.3.4 Диаграмма деятельности	40
2.3.5 Диаграмма последовательности	42
2.3.6 Диаграмма кооперации	43
2.4 Проектирование пользовательского интерфейса.....	44
2.5 Планирование проекта и оценка бюджета.....	46
3 Реализация системы	52
3.1 Физическая модель базы данных.....	52
3.2 Диаграмма компонентов.....	53
3.3 Диаграмма развертывания.....	55
3.4 Реализация пользовательского интерфейса	56

	11
4 Контроль качества программного обеспечения.....	61
4.1 Функциональное тестирование.....	61
4.2 Метрики кода.....	62
Заключение	67
Список использованных источников	68
Приложение А. Листинг программы.....	70
Приложение Б. Результаты функционального тестирования.....	84

Введение

Современное общество находится на таком уровне развития, когда уже необходимо отходить от обучения как заучивания фактов. Следует использовать инновационные технологии с целью донесения необходимого материала до обучающихся. Становится очевидным необходимость изменения подхода к процессу образования. Нужно вносить инструментальные технологии и интерактивность в обучение [1]. Следует обучать находить информацию, понимать её и правильно обрабатывать. Эффективным решением может стать использование обучающих систем, которые в последнее время очень активно развиваются и внедряются в высшие учебные заведения [2].

Обучающие системы – это программные средства профессиональной подготовки обучающихся, состоящие из электронных учебников и набора специализированных тренажеров. Обучающие системы – это одно из наиболее эффективных средств интенсификации обучения при повышении квалификации специалистов, изучении языка, освоении предметной области, темы и др. Системы позволяют в оригинальной методической форме выработать у обучаемых необходимые навыки и умения, а также закрепить лекционный материал.

Стандартный функционал обучающей системы, как правило, содержит два режима: обучение и экзамен [3].

Режим обучения состоит из ряда независимо функционирующих тем обучения. Каждая тема может включать в себя цветные наглядные рисунки, схемы с элементами мультипликации, текстовый теоретический материал, контрольные вопросы и учебно-тренировочные задачи. Все темы обучения работают в следующих режимах:

- входного контроля, включающего опрос обучаемого по данной теме и теоретическое обучение с помощью текстовой информации;
- тренировки, обеспечивающие демонстрацию работы отдельных систем;

- контроля, в котором обучаемый самостоятельно выбирает порядок действий по выполнению задачи;
- подсказки (при контроле знаний) с указанием на очередное правильное действие при решении задачи;
- фиксации ошибок, позволяющие накапливать допущенные обучаемым ошибки при входном контроле и решении задач, входящих в тему обучения, а также выводить их на печать с указанием имени обучаемого, названий выполненных задач, количества ошибок и подсказок.

Режим «экзамен» включает в себя контрольные вопросы. После ответа на предлагаемые контрольные вопросы печатается протокол экзамена с указанием даты, имени экзаменуемого, количества вопросов и неправильных ответов. Данная статистика позволяет понять, над какими темами стоит поработать экзаменуемому. В каких разделах необходимо поправить материал, если ошибка носит систематический характер у большего количества экзаменуемых.

При работе в электронной среде обучаемый имеет основную цель – получение качественных знаний, навыков, умений, а обучающая система поддерживает и помогает ему в достижении этой цели. Преподавателю инновационные технологии дают возможность реализовывать свои собственные методические приемы и свой творческий стиль работы.

С развитием информационных технологий стало проще и удобнее осуществлять сбор, хранение и обработку информации в цифровом виде. Для этого начали создавать различные информационные системы, направленные на автоматизацию или упрощение обработки данных. Это привело к тому, что бумажной работы стало намного меньше.

Целью данной квалификационной работы является разработка информационно-обучающей системы проведения занятий по физике, которая позволяет проводить лекционные занятия, лабораторные работы с большей эффективностью, анализировать успеваемость учащихся за выполненные работы.

Данная работа является актуальной, так как информационно-обучающая система, в состав которой входит виртуальная лаборатория предоставляет возможность наглядной визуализации на экране компьютера, что позволяет пронаблюдать процессы, трудноразличимые в реальных условиях без использования дополнительной техники. Также виртуальная лаборатория, в сравнении со старым оборудованием безопасна, так как устраняется угроза в случаях, когда идет работа, например, с химическими веществами. Виртуальная лаборатория позволяет экономить время и ресурсы, так как некоторые работы требуют достаточно больших массивов информации, а также даёт возможность использования лаборатории в дистанционном формате, что позволяет проводить опыты без непосредственного контакта. Разрабатываемая в рамках выпускной квалификационной работы информационная система должна позволить облегчить и улучшить процесс проведения лабораторных работ по физике.

1 Анализ предметной области

1.1 Понятие виртуальной лаборатории

Виртуальная лаборатория – это среда, имитирующая на экране планшета, смартфона, компьютера или интерактивной панели инструменты учебной лаборатории. Они дают возможность создать электрические цепи, построить чертежи или графики, выполнить измерения.

Виртуальные лаборатории могут использовать не только учителя, но и школьники. Применяя их на уроках, педагоги могут создавать с их помощью собственные интерактивные задания. А дети могут проводить эксперименты в онлайн-режиме не только находясь в школе, но и дома. Таким образом, учащимся легче изучать и повторять материал занятия.

Виртуальная лаборатория должна обладать функциями, которые сделают работу с ней удобнее для пользователя.

Логика представления материала в виртуальной лабораторной работе отличается от реальной работы более детальным описанием процесса исследования, обилием подсказок и ссылок, а также наличием анимации.

Не менее актуальными могут быть такие особенности, как:

- большое количество лабораторных работ;
- общедоступность;
- наличие теоретической базы.

Для сравнения были выбраны следующие системы виртуальных лабораторий:

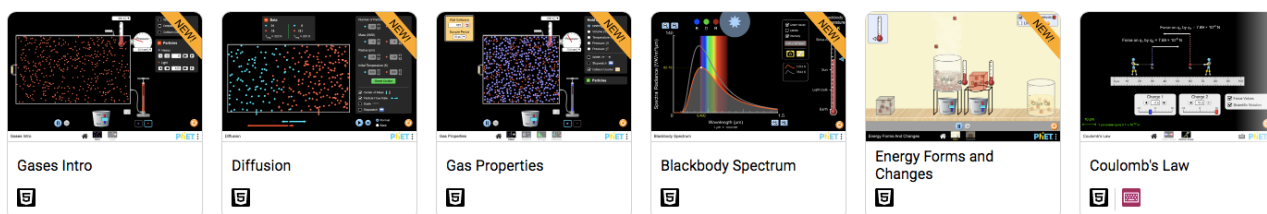
- комплекс виртуальных лабораторий PhET;
- виртуальные лаборатории Томского политехнического института;
- виртуальные лаборатории VirtuLab;
- программа Algodoo.

Рассмотрим каждую из перечисленных систем.

1.2 Анализ существующих информационно-обучающих систем

PhET – проект, разработанный Университетом Колорадо [4]. Проект включает большое множество виртуальных лабораторий, демонстрирующих различные явления в области физики, биологии, химии, математики, наук о Земле.

Chemistry All Chemistry (53)



Math All Math (42)

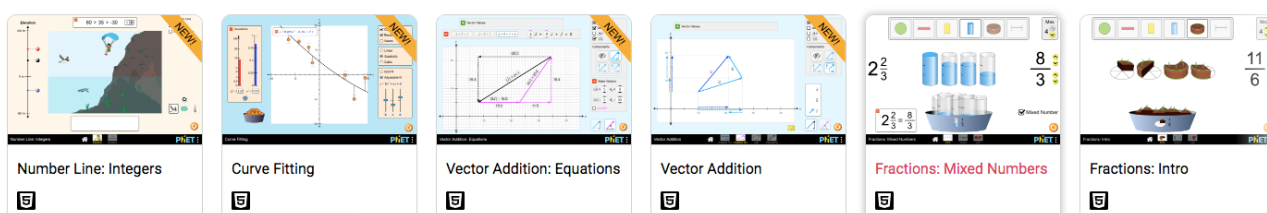


Рисунок 1 – Интерфейс PhET

Возможности PhET:

- большое множество виртуальных лабораторий по различным областям науки;
- платная интернет-платформа;
- дружелюбный интерфейс;
- большое количество языков;
- отсутствие теоретической подготовки;
- опыты выполняются онлайн.

PhET – проект, содержащий в себе бесплатные интерактивные математические и научные симуляции. Симуляторы PhET основаны на обширных исследованиях в области образования и привлекают студентов через интуитивно понятную игровую среду, в которой студенты учатся через исследования и открытия.

Виртуальные лаборатории Томского политехнического института. более двухсот виртуальных лабораторных работ (химия, экология,

гидрогазодинамика, методы анализа, электрические машины, энергетика, физика, микробиология, метрология, термодинамика).



Рисунок 2 – Виртуальная лаборатория ТПИ

Параметры сравнения виртуальной лаборатории ТПИ:

- более двухсот лабораторных работ по различным областям науки;
- наличие 3D лабораторий;
- поддержка только русского языка;
- возможность проводить уроки в компьютерной аудитории по сети, с последующим анализом успеваемости ученика;
- поддержка технологий виртуальной реальности;
- увлекательные интерактивные уроки и лабораторные работы;
- возможность выполнения лабораторных онлайн;
- сложный интерфейс.

Виртуальные лаборатории Томского политехнического института представляют собой комплекс компьютерных программ (виртуальных лабораторных работ), реализующий имитационное моделирование реальных явлений, технологических процессов, измерительных и рабочих методик [5]. Виртуальная работа содержит интерактивные виртуальные модели и элементы графической визуализации результатов моделирования. В зависимости от методических предпосылок виртуальная лаборатория может содержать элементы входного, итогового и промежуточного контроля в тестовой форме, задания для проведения исследования и модуль полностью автоматического или комбинированного формирования отчёта. Комплексы с доступом on-line выполняются в виде исполняемого swf-документа. При необходимости может обладать функциями фиксации в централизованной БД хода и результатов выполнения экспериментов.

Виртуальный лабораторный комплекс состоит, как правило, из нескольких лабораторных работ. Ориентировочное время производства такого комплекса, включая предварительную подготовку автора (консультация по подготовке методических указаний) – 9 месяцев.

VirtualLab – проект по разработке виртуальных лабораторных работ для учащихся по физике, химии, биологии, экологии [6]. Виртуальные лабораторные работы реализованы при помощи технологии Flash. Отличаются узкой специализацией, в большинстве случаев линейностью опыта (вся последовательность действий и результаты опыта заданы заранее). Продукты VirtualLab имеют познавательную ценность и решают задачу проведения лабораторных работ при отсутствии необходимого оборудования.

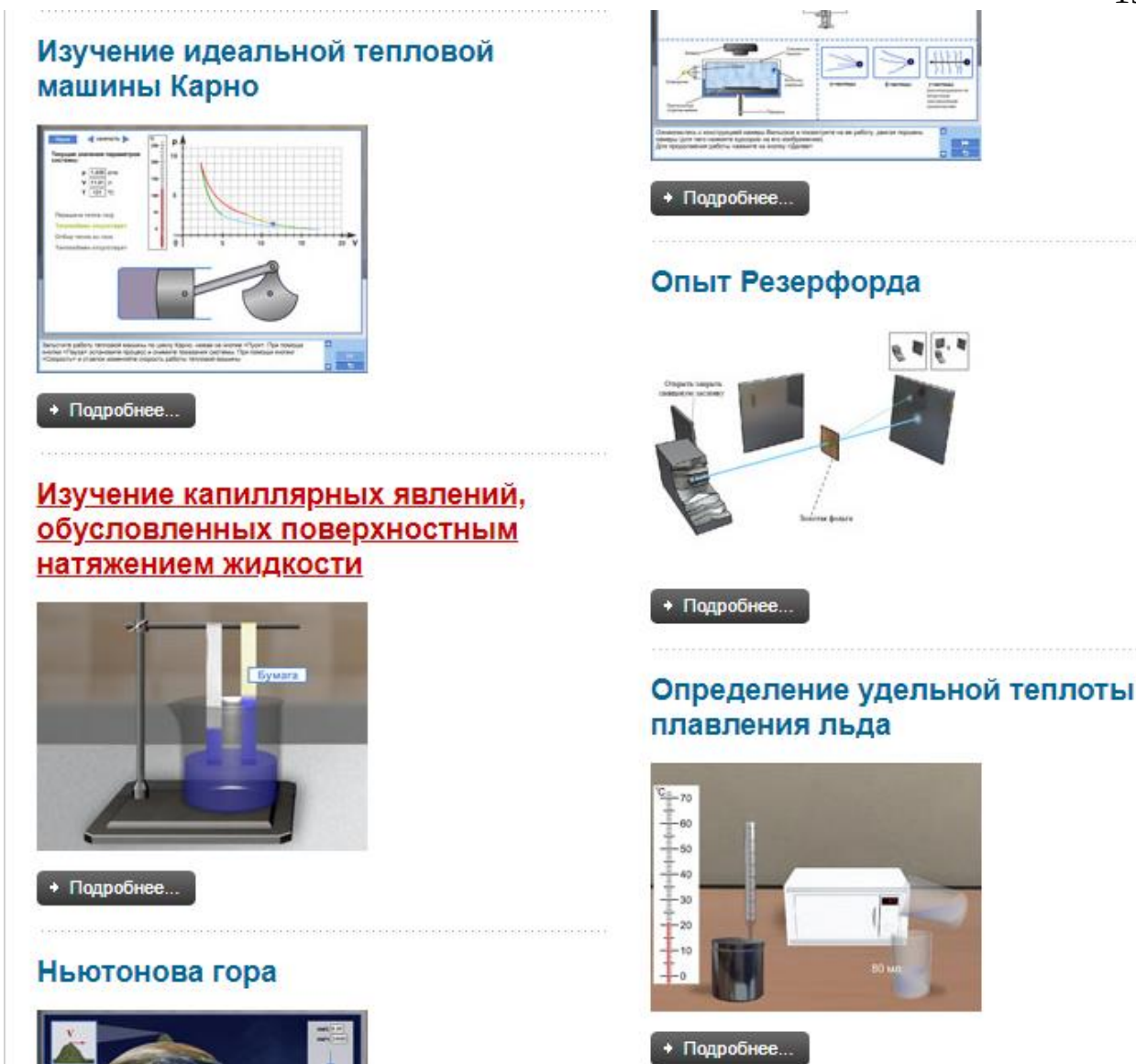


Рисунок 3 – Виртуальные лаборатории VirtualLab

Параметры сравнения VirtualLab:

- работа через браузер;
- наличие лабораторных работ по различным видам наук;
- бесплатное использование;
- возможность отслеживания результатов опыта;
- поддержка только русского языка;
- наличие 3D опытов;
- использование достаточно устаревшей технологии Flash Player.

VirtualLab позволяет проводить опыты прямо в окне браузера с помощью анимационных flash-роликов. Лабораторные работы разделены на

группы по тематике - физика, химия, биология и экология и далее по темам или классам. В каждой работе предлагается изучить некоторый материал и ответить на вопросы или выполнить задания/тесты. Спектр предлагаемых работ очень широк и интересен, ряд работ выполняются в 3D-режиме.

Лаборатория очень удобна тем, что не требует скачивания и установки на компьютер, а значит платформенно-независима и бесплатна. Для корректной работы только требуется установить необходимые плагины.

Algodoo – программа, предназначенная для физических 2D симуляций [7]. Имеет очень богатый инструментарий для создания различных объектов, механизмов и систем с целью моделирования их физического взаимодействия и свойств.

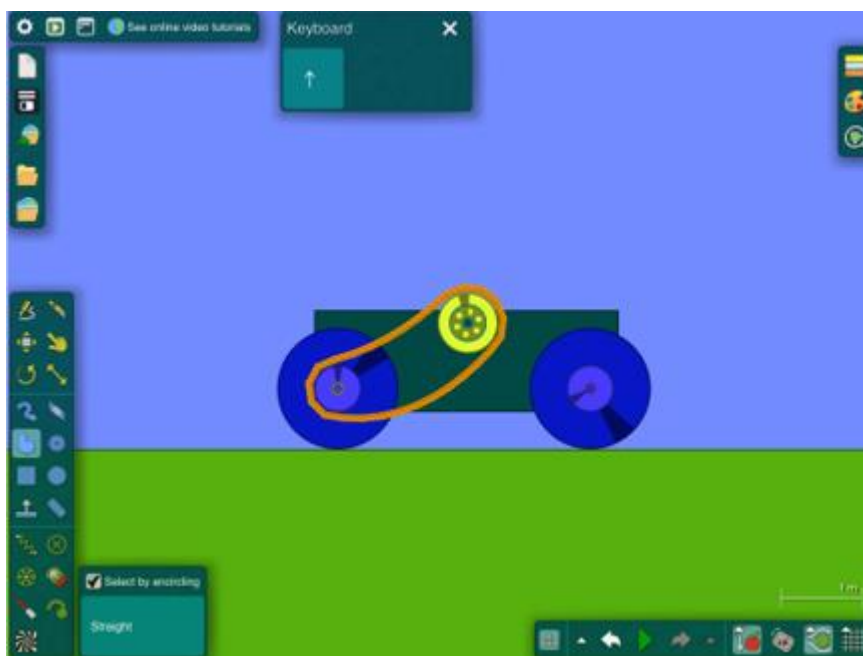


Рисунок 4 – Интерфейс Algodoo

Параметры сравнения Algodoo:

- способность создания своих опытов;
- 2D графика;
- прикладная программа;
- бесплатная;
- отсутствие русского языка;

– внутренняя база с опытами.

Программа способна симулировать не только механические процессы, но и оптические, а возможность программирования при помощи скриптового языка Thyme позволяет создавать объекты с оригинальными физическими свойствами, различные функции, эффекты и явления. Также имеется возможность загружать рисунки: рисунок становится объектом симуляции и ему можно задать любые физические свойства.

Согласно характеристикам, представленным в описании систем аналогов, мне удалось составить сравнение этих систем с разрабатываемой. Для удобства сравнение систем приведено в таблице 1.

Таблица 1 – Сравнение аналогов

Критерий	PhET	Виртуальные лаборатории ТПИ	Виртуальные лаборатории и VirtualLab	Algodoo	Разрабатываемая система
Качество пользовательского интерфейса	высокое	низкое	низкое	среднее	среднее
Количество лабораторных работ	высокое	среднее	среднее	низкое	среднее
Качество лабораторных работ	высокое	среднее	низкое	низкое	среднее
Сложность в освоении	низкая	высокая	средняя	высокая	низкая
Возможность работы по локальной сети	нет	да	нет	нет	да

Продолжение таблицы 1

Наличие русского языка	да	да	высокая	нет	да
Стоимость	платная	бесплатная	бесплатная	бесплатная	бесплатная

В результате анализа были выявлены достоинства и недостатки существующих систем, а также составлена таблица сравнения.

Разрабатываемая система имеет усредненные характеристики в количестве и качестве лабораторных работ, а также отличается низкой степенью сложности в освоении и простым интерфейсом.

1.3 Анализ требований

Анализ требований является частью процесса разработки программного обеспечения, которая включает в себя сбор требований, выявление взаимосвязей, их систематизацию и документирование.

Полнота и качество требований играют важную роль в успехе реализации проекта. Требования к ПО должны быть выполнимыми, тестируемыми и детализованными. Функциональные требования – список задач, которые должны решать само программное обеспечение. Они определяют действия системы и какова реакция системы на различные входные данные [8].

1.3.1 Функциональные требования

В результате анализа предметной области и продуктов-аналогов были выделены основные функциональные требования к системе.

Для преподавателя требования представлены на рисунке 5.

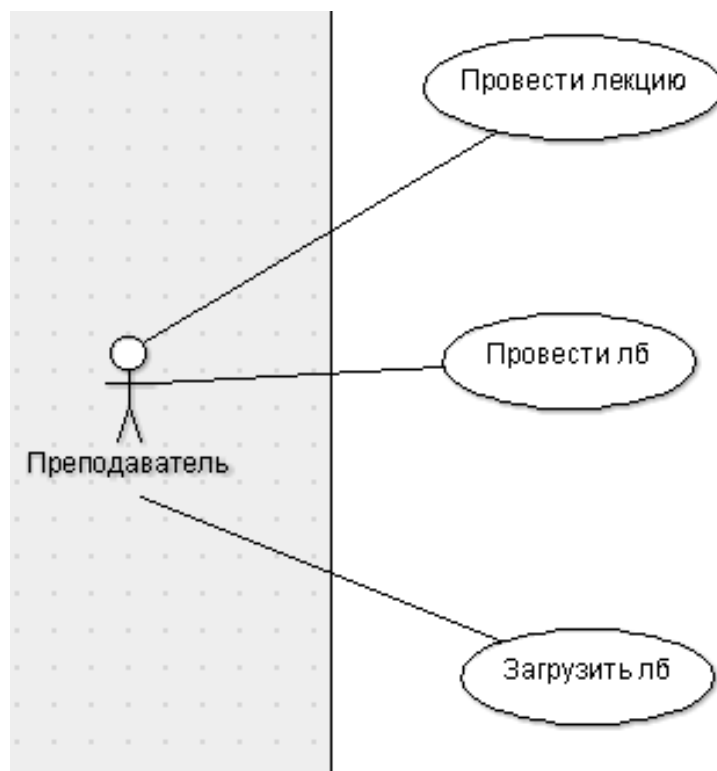


Рисунок 5 – Функциональные требования для преподавателя

Для преподавателя были выделены функциональные требования, такие как:

- провести лабораторную работу;
- провести лекционную часть занятия;
- загрузить необходимые данные в базу.

Для ученика требования представлены на рисунке 6.

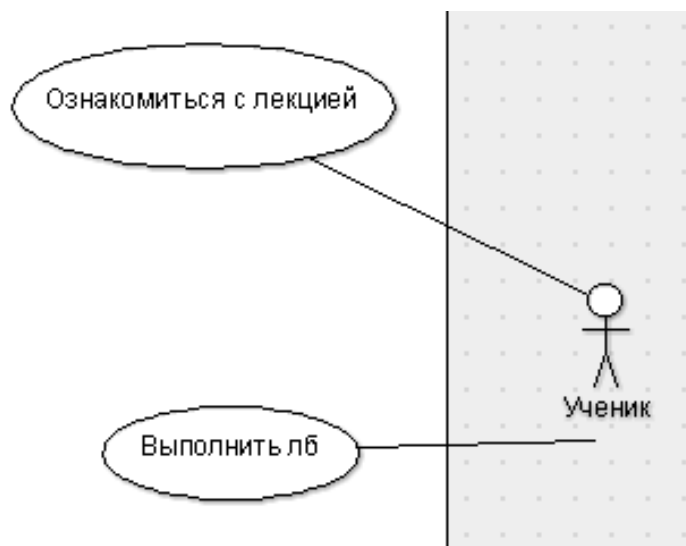


Рисунок 6 – Функциональные требования для ученика

Для ученика были выделены функциональные требования, такие как:

- ознакомиться с лекцией;
- выполнить лабораторную работу.

В результате проектирования были определены функциональные требования к системе.

1.3.2 Нефункциональные требования

Нефункциональные требования описывают, как именно должен работать программный продукт или система, и какими свойствами или характеристиками она должна обладать.

Разработку можно разделить на два этапа: разработка серверной части и разработка клиентской части. Составим требования к каждой из них.

Требования к клиентской части:

- клиентская часть будет являться настольным приложением;
- система должна обладать интерфейсом, удобным и понятным для пользователя;
- приложение должно быть совместимо с операционной системой Windows.

Требования к серверной части приложения:

- в качестве СУБД должна использоваться СУБД Microsoft Sql Server 2008r2 Express (сервер SQLEXPRESS).

2 Проектирование системы

2.1 Анализ процессов образовательной организации

Важным этапом проектирования является построение модели бизнес-процессов. На начальных этапах создания ИС необходимо понять, как работает система, которую необходимо автоматизировать. Поэтому для описания информационно-обучающей системы необходимо построить модель, которая будет соответствовать предметной области и содержать в себе знания всех участников процессов системы [9].

Нотация IDEF0 поддерживает последовательную декомпозицию процесса до требуемого уровня детализации. Дочерняя диаграмма, создаваемая при декомпозиции, охватывает ту же область, что и родительский процесс, но описывает ее более подробно. Согласно методологии IDEF0 при декомпозиции стрелки родительского процесса переносятся на дочернюю диаграмму в виде граничных стрелок.

Таким образом, IDEF0 может применяться на ранних этапах создания широкого круга систем. В то же время она может быть использована для анализа функций существующих систем и выработки решений по их улучшению.

Контекстная диаграмма (диаграмма верхнего уровня), являясь вершиной древовидной структуры диаграмм, показывает назначение системы (основную функцию) и ее взаимодействие с внешней средой. В каждой модели может быть только одна контекстная диаграмма. После описания основной функции выполняется функциональная декомпозиция, т.е. определяются функции, из которых состоит основная.

Контекстная диаграмма процессов информационно-обучающей системы для проведения занятий по физике в методологии IDEF0 представлена на рисунке 7.

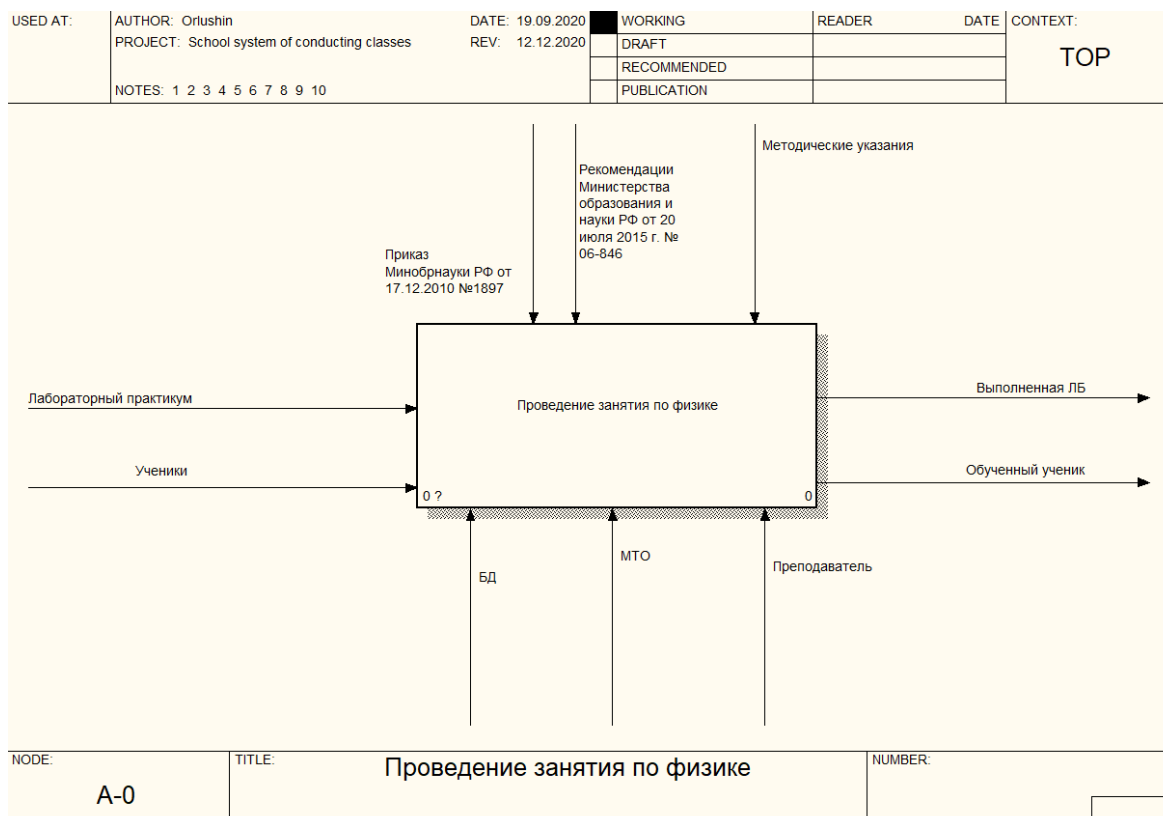


Рисунок 7 – Контекстная диаграмма

В данной контекстной диаграмме входными данными является:

- лабораторный практикум;
- ученики.

Выходным данным для процесса «Проведение занятия по физике» являются:

- выполненная ЛБ;
- обученный ученик.

Механизмы и управления – это действующие приказы, рекомендации, методические указания. Ресурсами являются: база данных, материально-техническое обеспечение, преподаватель.

На основе созданной контекстной диаграммы в результате детализации обобщенного процесса «Проведение занятия по физике» была разработана детализирующая диаграмма, представленная на рисунке 8.

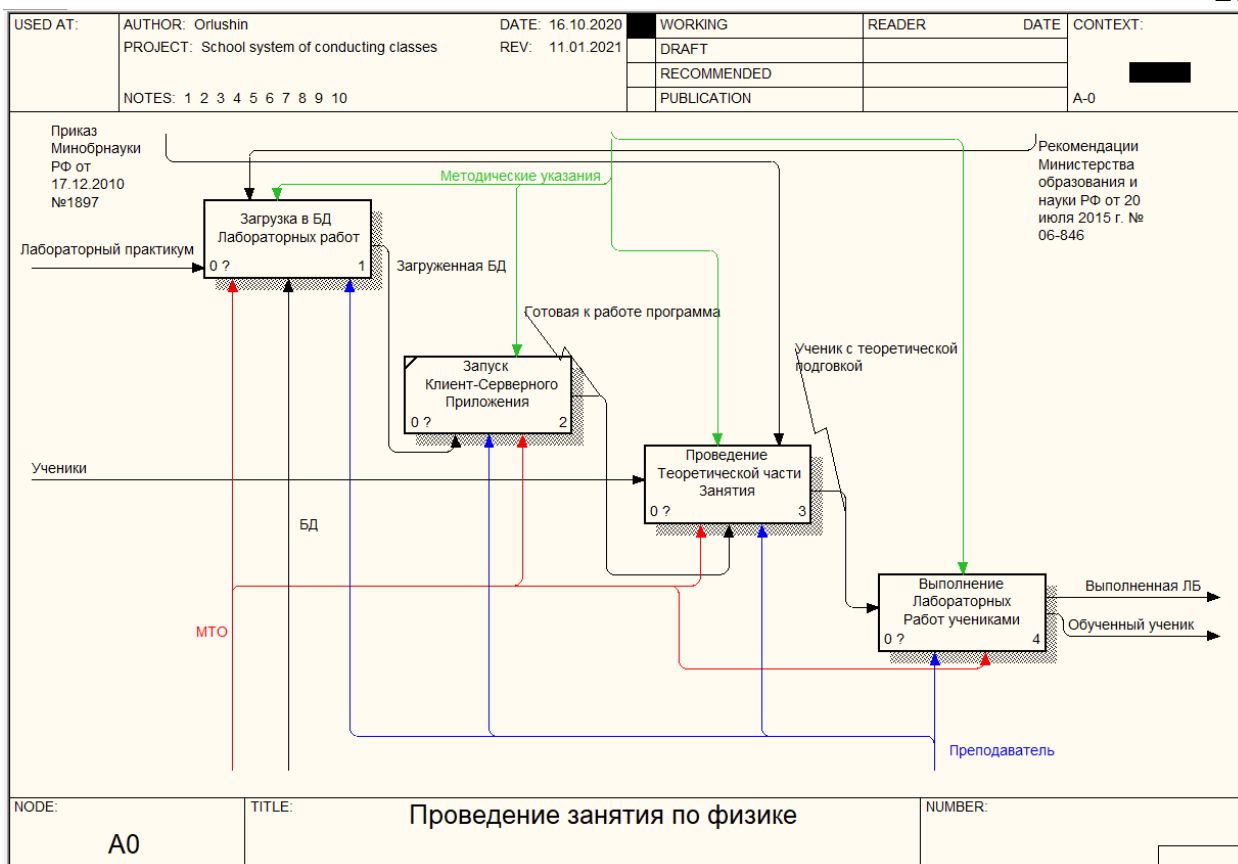


Рисунок 8 – Детализирующая диаграмма процесса «Проведение занятия по физике» в нотации IDEF0

В процессе декомпозиции процесса системы «Проведение занятия по физике» выделены 4 процесса: «Загрузка в БД лабораторных работ», «Запуск клиент-серверного приложения», «Проведение теоретической части занятия», «Выполнение лабораторных работ учениками».

Процесс «Загрузка в БД лабораторных работ» предназначен для загрузки необходимых файлов в базу данных. На данном этапе на сервер осуществляется загрузка файлов необходимых для лабораторных работ.

Процесс «Запуск клиент-серверного приложения» предназначен для запуска приложения на персональных компьютерах. На данном этапе пользователь осуществляет вход в систему под своим логином и паролем.

Процесс «Проведение теоретической части занятия» предназначен для преподнесения теории ученикам. Данный процесс подразумевает, что ученик читает лекционные файлы, находящиеся в приложении.

Процесс «Выполнение лабораторных работ» предназначен для проведения практической части занятия. На данном этапе ученик

осуществляет выполнение лабораторной работы. После выполнения лабораторной работы учитель оценивает эту работу и проставляет в системе результаты.

Декомпозиция процесса «Проведение теоретической части занятия» в нотации IDEF0 представлена на рисунке 9.

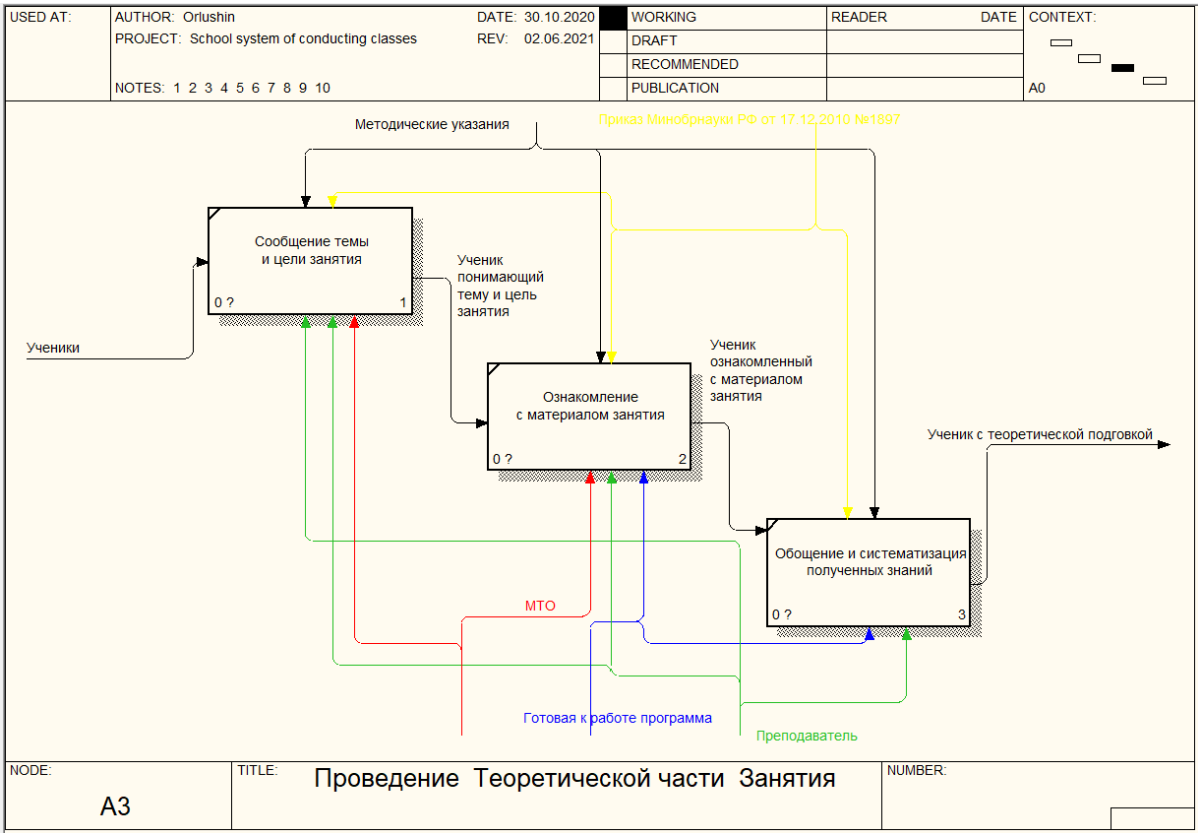


Рисунок 9 – Детализирующая диаграмма процесса «Проведение теоретической части занятия»

Первоначально преподаватель сообщает ученикам тему занятия, указывает какие материалы будут использованы на уроке, которые были подобраны согласно методическим рекомендациям. Далее в приложении осуществляется ознакомление с лекцией занятия. После этого производится обобщение и систематизация полученных после прочтения лекции знаний. В итоге ученик готов к выполнению практической части занятия.

Декомпозиция процесса «Загрузка в БД лабораторных работ» в нотации DFD представлена на рисунке 10.

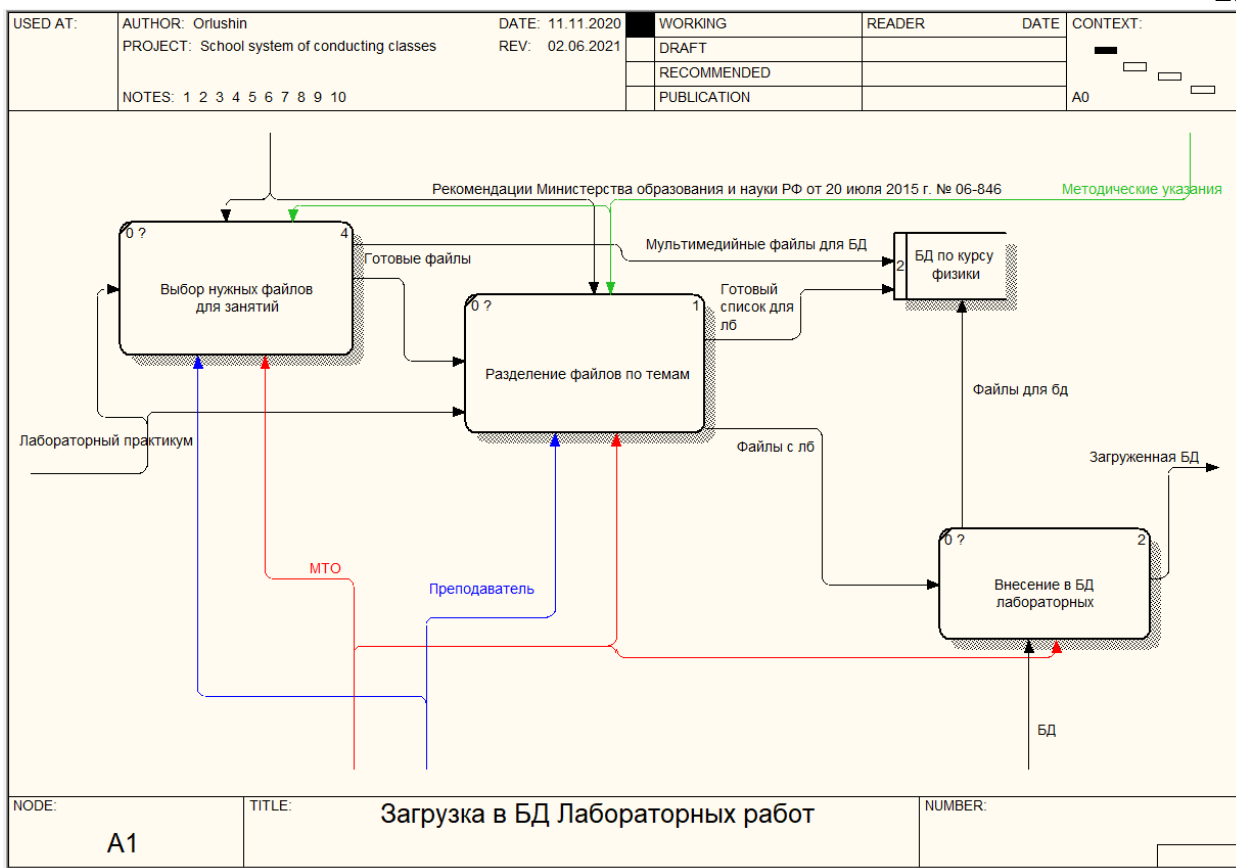


Рисунок 10 – Детализирующая диаграмма процесса «Загрузка в БД лабораторных работ»

Первоначально преподаватель подготавливает нужные файлы для проведения занятия с учетом методических указаний и рекомендаций. Далее осуществляется разделение файлов по темам в определенную папку в файловой системе. После этого происходит автоматическая загрузка в базу данных этих файлов. Теперь пользователи могут использовать в приложении готовые для занятия файлы.

2.2 Проектирование логической модели базы данных

Проектирование структуры данных осуществлялось с помощью Case-средства ERwin Data Modeler. ERwin предназначен для разработки структуры БД и является наиболее известным и доступным средством моделирования данных [10].

Логическая модель представляет собой абстрактный взгляд на данные и может иметь аналоги в реальном мире. Объектами логической модели

являются сущности, атрибуты и отношения между сущностями. Данная модель является универсальной [11].

На основании описанной логической модели, проектировщик может выбрать необходимую СУБД, и Erwin автоматически создаст соответствующую физическую модель. По физической модели ERwin может сгенерировать соответствующий SQL-скрипт [12].

Таким образом была спроектирована логическая модель базы данных для информационно-обучающей системы проведения занятий по физике. Данная модель содержит 6 таблиц, связанных между собой логическими связями.

Спроектированная логическая модель базы данных представлена на рисунке 11.

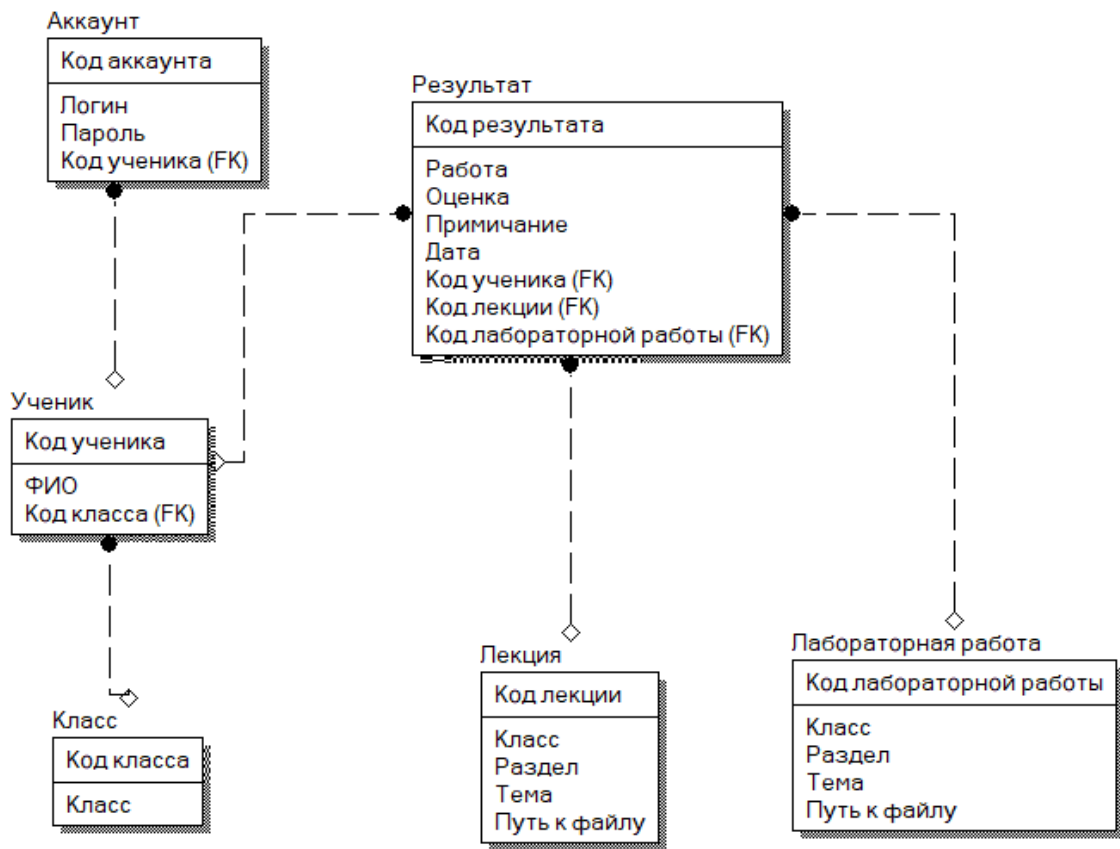


Рисунок 11 – Логическая модель данных

Описание сущностей и их атрибутов логической модели представлены в таблице 2.

Таблица 2 – Описание сущностей и их атрибутов логической модели

Сущность	Атрибуты	Роль атрибута	Описание атрибута
Аккаунт	Код аккаунта	РК	Уникальный код аккаунта в системе
	Логин		Логин пользователя
	Пароль		Пароль пользователя
	Код ученика	FK	Уникальный код ученика
Ученик	Код ученика	РК	Уникальный код ученика
	ФИО		Фамилия, имя и отчество ученика
	Код класса	FK	Уникальный код класса
Класс	Код класса	РК	Уникальный код класса
	Класс		Номер класса
Результат	Код результата	РК	Уникальный код результата
	Работа		Номер работы
	Оценка		Оценка за выполненную работу
	Примечание		Дополнительное примечание к выполненной работе
	Дата		Дата выполнения работы
	Код ученика	FK	Уникальный код ученика
	Код лекции	FK	Уникальный код лекции
	Код лабораторной	FK	Уникальный код работы
Лекция	Код лекции	РК	Уникальный код лекции
	Класс		Номер класса
	Раздел		Раздел предмета
	Тема		Тема лекции
	Путь к файлу		Путь к файлу с лекцией
Лабораторная работа	Код ЛБ	РК	Уникальный код работы
	Класс		Номер класса
	Раздел		Раздел предмета
	Тема		Тема лекции
	Путь к файлу		Путь к файлу с лабораторной работой

После определения сущностей и установления отношений между ними необходимо привести структуру БД в соответствие ограничениям целостности, которые вводятся для исключения любого дублирования, а также потенциальной противоречивости хранимых данных.

Так как спроектированная логическая модель базы данных соответствует следующим условиям:

- содержит простые атрибуты;
- каждый неключевой атрибут зависит от идентификатора;
- каждый неключевой атрибут нетранзитивно зависит от идентификатора.

Таким образом, данная модель является нормализованной и находится в третьей нормальной форме.

2.3 Проектирование ПО

В процесс проектирования ПО входят разработка моделей, диаграмм, отражающих структуру программы, их текстовое описание. Все это составляет проектную документацию. Для ее создания может быть использован язык UML – унифицированный язык моделирования, предназначенный для описания программных систем.

Недостатком языка является изначальная ориентированность на объектно-ориентированные системы, что накладывает ограничения на выполнение основной своей функции – предоставить, с одной стороны, достаточно формальное, с другой стороны, достаточно удобное, и, с третьей стороны, достаточно универсальное средство, позволяющее до некоторой степени снизить риск расхождений в толковании спецификаций [13].

Проектирование системы осуществляется с использованием UML-диаграмм с помощью CASE-средства ArgoUML.

2.3.1 Диаграмма вариантов использования

Диаграмма вариантов использования (сценариев поведения, прецедентов) является исходным концептуальным представлением системы в процессе ее проектирования и разработки. Данная диаграмма состоит из актеров, вариантов использования и отношений между ними. При построении диаграммы могут использоваться также общие элементы нотации: примечания и механизмы расширения.

Диаграмма вариантов использования представлена на рисунке 12.

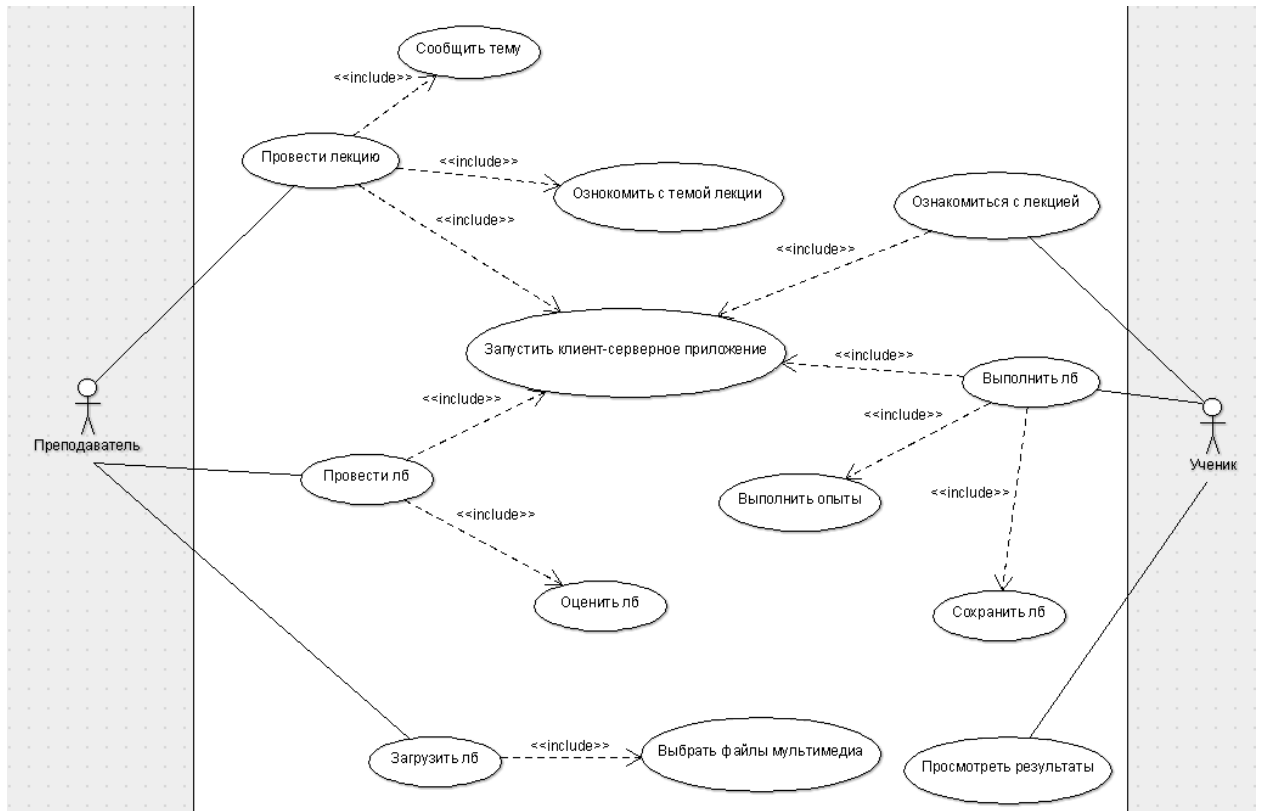


Рисунок 12 – Диаграмма вариантов использования

Описание действующих лиц представлено в таблице 3, а краткое описание вариантов использования в таблице 4.

Таблица 3 – Список действующих лиц

Наименование лица	Профиль, подготовка и навыки
Преподаватель	Человек, проводящий учебное занятие по определенной дисциплине.
Ученик	Человек, обучаемый преподавателем.

Таблица 4 – Описание вариантов использования

Вариант использования	Контекст использования
Провести лекцию	Проведение лекционной части занятия с целью получения учеником теоретических знаний
Провести ЛБ	Проведение лабораторной работы
Загрузить ЛБ	Загрузить в базу данных лабораторные работы
Ознакомиться с лекцией	с Прослушать лекцию и усвоить материал

Продолжение таблицы 4

Вариант использования	Контекст использования
Сообщить тему	Сообщить ученикам тему занятия
Выполнить ЛБ	Выполнить лабораторную работу по указанной теме.
Запустить клиент-серверное приложение	Запуск приложения для проведения занятий
Оценить ЛБ	Провести оценку готовых лабораторных работ
Выбрать файлы мультимедиа	Учитель выбирает нужный материал для лабораторной работы
Выполнить опыты	Выполнить задания лабораторной работы
Сохранить ЛБ	Сохранить выполненную лабораторную работу
Просмотреть результаты	Ученик может посмотреть результаты выполненных работ

В результате анализа были выявлены и описаны прецеденты системы, а также построена диаграмма вариантов использования.

2.3.2 Диаграмма классов

Диаграмма классов определяет типы классов системы и различного рода статические связи, которые существуют между ними. На диаграммах классов изображаются также атрибуты классов, операции классов и ограничения, которые накладываются на связи между классами.

Вид и интерпретация диаграммы классов существенно зависит от точки зрения (уровня абстракции): классы могут представлять сущности предметной области (в процессе анализа) или элементы программной системы (в процессах проектирования и реализации).

В соответствии с моделью предметной области были спроектированы следующие классы, представленные на диаграмме классов (рисунок 13).

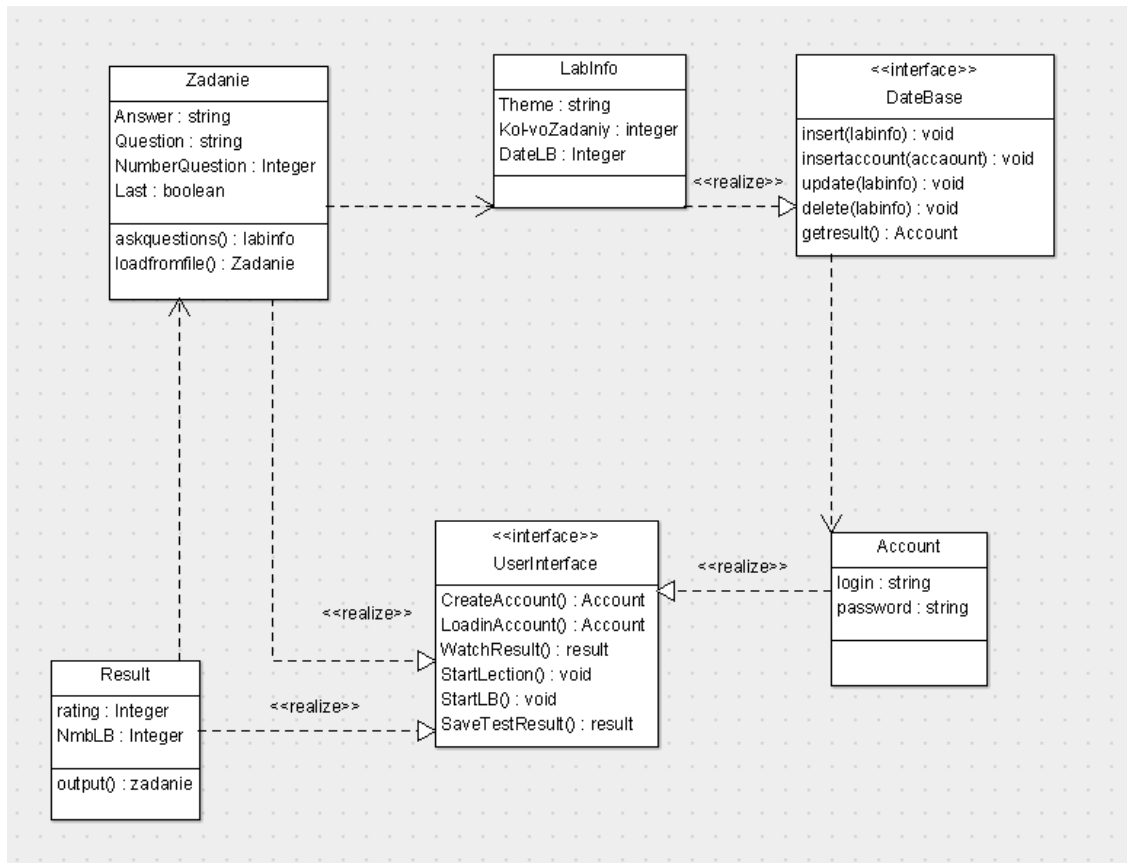


Рисунок 13 – Диаграмма классов

Краткое описание классов приведено ниже.

Labinfo. Хранит данные о лабораторных работах.

Описание атрибутов:

- theme: string – тема лабораторной работы;
- kol-vozadaniy – для количества заданий лабораторной работы.

Описание связей:

- реализация от labinfo к database – позволяет отделить методы взаимодействия объектов класса labinfo с БД в отдельный интерфейс;
- зависимость от zadanie к labInfo – позволяет использовать объекты класса labinfo классом zadanie.

Datebase. Используется для управления БД и формирования запросов.

Описание методов:

- insert(labinfo): void – добавляет в БД новые записи о ЛБ;
- insertaccount(account): void – добавляет в БД данные о зарегистрированном аккаунте;

- update(labinfo): void – изменяет запись, аналогичную объекту labinfo в БД;
- delete(): void – удаляет запись, аналогичную объекту labinfo в БД;
- getresult(): account – выводит полученные результаты учащегося по ЛБ.

Описание связей:

- Реализация от labinfo к database – позволяет отделить методы взаимодействия объектов класса labinfo с БД в отдельный интерфейс;
- Зависимость от database к account – позволяет классу database использовать объекты класса account для сохранения созданных аккаунтов и результатов прохождения ЛБ.

Account. Хранит данные об аккаунте пользователя.

Описание атрибутов:

- login: string – логин пользователя;
- password: string – пароль пользователя.

Описание связей:

- реализация от account к userinterface – позволяет отделить методы регистрации в системе и входа в аккаунт пользователем в отдельный интерфейс;
- зависимость от database к account – позволяет классу database использовать объекты класса account для сохранения созданных аккаунтов.

Zadanie. Используется для проведения лабораторных работ и сохранении ответов.

Описание методов:

- askquestions(): labinfo – последовательно выводит задания ученику;
- loadfromfile(): zadanie – загружает лабораторную работу из файла;
- writevopros(): void – записывает ответ пользователя на задание.

Описание атрибутов:

- answer: string – ответ ученика на задание;

- question: string – задание для ученика;
- last: boolean – если значение true – задание последнее;
- number: integer – хранит номер задания для последовательного вывода.

Описание связей:

- реализация от zadanie к userinterface – позволяет разделить методы задания в отдельный интерфейс;
- зависимость от zadanie к labinfo – позволяет использовать объекты класса labinfo классом zadanie;
- зависимость от result к zadanie – позволяет использовать объект класса zadanie классом result.

Result. Используется для вывода результатов лабораторной работы в удобной для пользователя форме.

Описание методов:

- output(): void – выводит результаты лабораторной в удобной для пользователя форме.

Описание атрибутов:

- rating: integer – оценка за лабораторную работу;
- nmbldb – номер лабораторной работы.

Описание связей:

- реализация от result к userinterface – позволяет разделить методы просмотра и сохранения результатов пройденных пользователем тестов в отдельный интерфейс;
- зависимость от result к zadanie – позволяет использовать объект класса zadanie классом result.

UserInterface. Используется для обеспечения интерфейса пользователя.

Описание методов:

- createaccount(): account – создает аккаунт пользователя путем заполнения объекта account;
- loadaccount(): account – осуществляет вход в аккаунт пользователя;

- startlb(): void – запускает лабораторную работу;
- startbection(): void – запускает лекцию;
- savetestresult(): result – сохраняет результаты теста в БД;
- watchresult(): result – выводит результаты пройденных лабораторных работ.

Описание связей:

- реализация от account к userinterface – позволяет отделить методы регистрации в системе и входа в аккаунт пользователем в отдельный интерфейс;
- реализация от result к userinterface – позволяет отделить методы просмотра и сохранения результатов выполненных пользователем заданий в отдельный интерфейс;
- реализация от zadanie к userinterface – позволяет отделить методы выбора заданий для лабораторной работы.

С помощью диаграммы классов были описаны классы, атрибуты, методы и зависимости между ними.

2.3.3 Диаграмма состояний

Состояние – это одна из характеристик объектов системы наряду с поведением. Каждый объект представляет собой набор атрибутов, выполняемых функций и имеет взаимосвязи с другими объектами [14]. Состояние определяется значениями вышеперечисленных параметров в определенный момент работы системы. Изменение состояния происходит при наступлении какого-либо события (например, после отправки сообщения). Диаграмма состояний показывает, как объект переходит из одного состояния в другое и какое событие при этом произошло.

В данной системе диаграмма подробно отображает процесс проведения лабораторной работы. Диаграмма состояния в нотации UML представлена на рисунке 14.

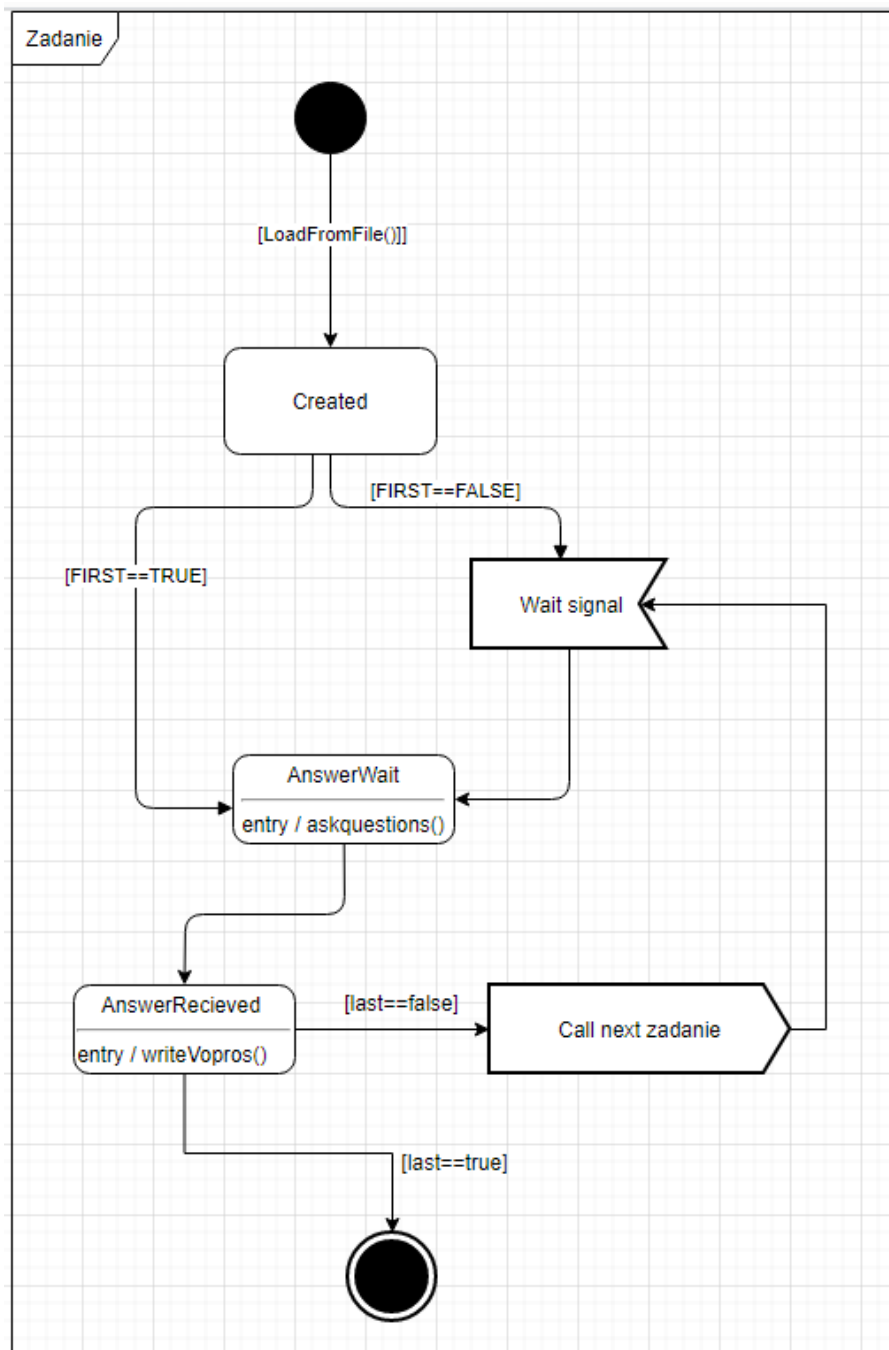


Рисунок 14 – Диаграмма состояния объекта «Задание»

Выделены следующие состояния объекта:

- объект создан;
- ожидает вызова;
- ожидает ответа;
- ответ получен;
- послать сигнал следующему объекту;
- финальное состояние.

2.3.4 Диаграмма деятельности

Для варианта использования «Выполнить ЛБ» составим диаграмму деятельности (рисунок 15).

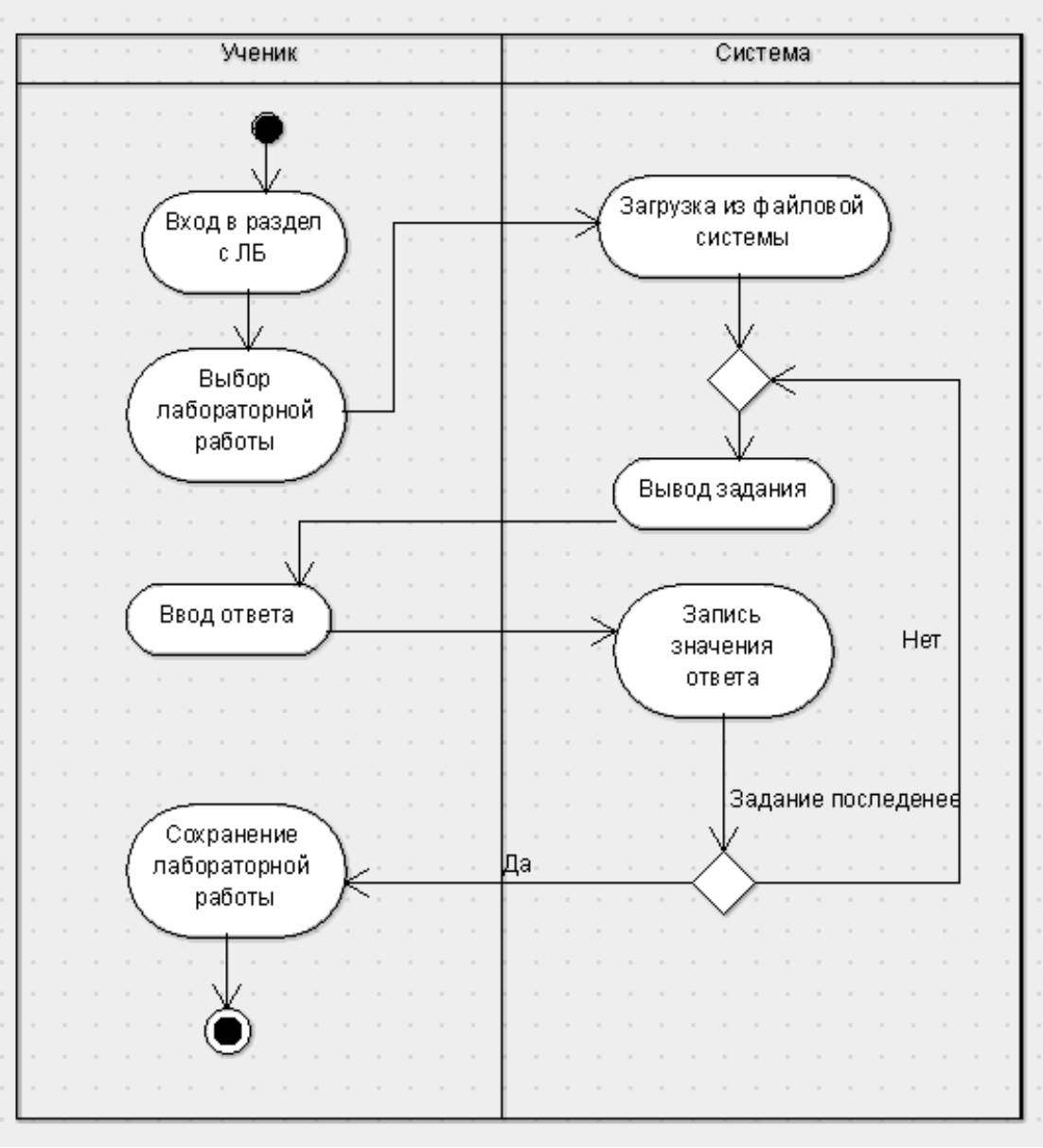


Рисунок 15 – Диаграмма деятельности для прецедента «Выполнить ЛБ»

Описание узлов диаграммы деятельности представлена в таблице 5.

Таблица 5 – Описание сообщений.

Узел	Тип	Описание	Входной поток	Выходной поток
Вход в раздел с ЛБ	Узел действия	Пользователь входит в раздел со списком лабораторных работ	Нет	Выбор лабораторной работы
Выбор лабораторной работы	Узел действия	Пользователь выбирает нужную лабораторную работу	Вход в раздел с ЛБ	Загрузка из файловой системы
Загрузка из файловой системы	Узел действия	Начинается загрузка лабораторной работы из файла	Выбор лабораторной работы	Вывод задания
Вывод задания	Узел действия	Выводит задание лабораторной работы	Загрузка из файловой системы	Ввод ответа
Ввод ответа	Узел действия	Ввод ответа на задание лабораторной работы	Вывод задания	Записывает значение ответа
Запись значения ответа	Узел действия	Записывает ответ в writevopros()	Ввод ответа	Задание последнее?

Ученик сначала входит в приложение: в раздел, где хранятся лабораторные работы. Выбирает нужную работу из каталога. Система выводит задания, на которое ученик должен дать ответ. После получения от ученика ответа, система проверяет последнее ли это задание, если ответ положительный, то система сохраняет результаты лабораторной работы.

2.3.5 Диаграмма последовательности

Диаграмма последовательности описывает сценарий работы приложения. На диаграмме отображаются экземпляры объектов и сообщения, которыми они обмениваются в рамках одного варианта использования.

В данной системе выделим 5 объектов (Пользователь, интерфейс, лабораторная, интерфейс базы данных). Диаграмма отображает процесс добавления лабораторной работы в базу данных. Диаграмма последовательности в нотации UML представлена на рисунке 16.

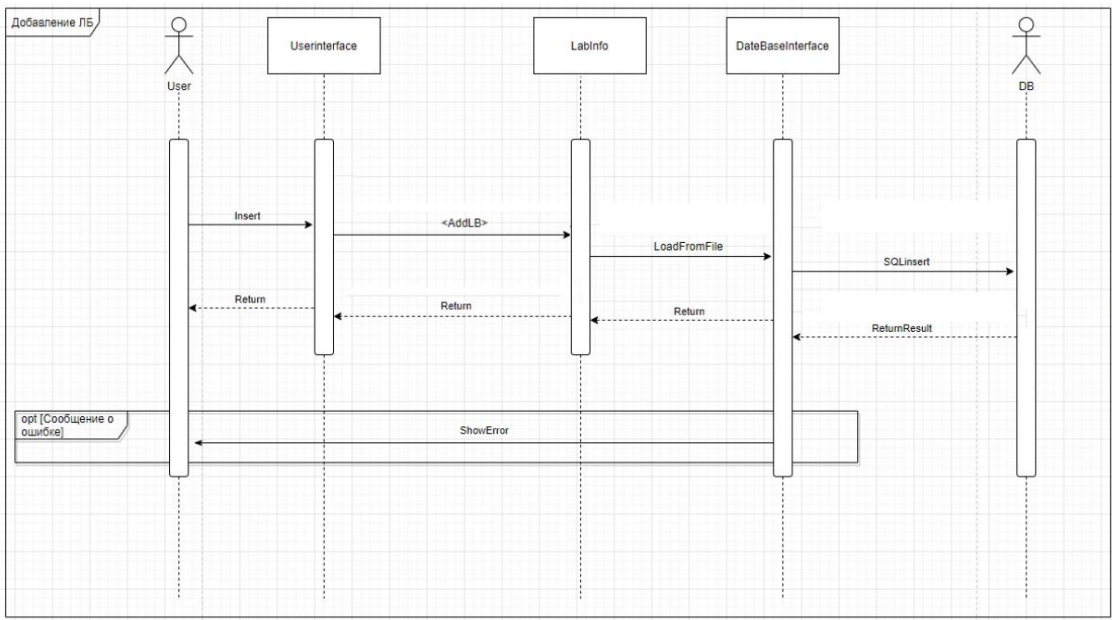


Рисунок 16 – Диаграмма последовательности

Описание диаграммы последовательности представлено в таблице 6 и 7.

Таблица 6 – Описание линий жизни

Линии жизни	Описание
User	Отображает пользователя (преподавателя) в системе. Пользователь (преподаватель) может добавлять, удалять, изменять данные лабораторных работ в БД
UserInterFace	Осуществляет взаимодействие пользователя с системой
LabInfo	Служит для заполнения данными лабораторные работы

Продолжение таблицы 6

Линии жизни	Описание
DateBaseInterface	Осуществляет взаимодействие системы и базы данных
DB	Отображает БД в системе

Таблица 7 – Описание сообщений

Сообщение	Описание	Инициатор	Получатель	Тип
Insert	Добавление ЛБ	User	UserInterface	Синхронное
AddLB	Иницирует создание ЛБ	Userinterface	LabInfo	Синхронное
LoadFromFile	Формирует команду для добавления объекта в БД	LabInfo	DateBaseInterface	Синхронное
SQLInsert	Передает текст команды в БД	DateBaseInterface	DB	Синхронное
Show error	Возвращает код ошибки в интерфейс пользователя	DateBaseInterface	User	Асинхронное

Все объекты, представленные на диаграмме, существуют на протяжении всего сеанса общения.

2.3.6 Диаграмма кооперации

Понятие кооперации (collaboration) является одним из фундаментальных понятий в языке UML. Оно служит для обозначения множества взаимодействующих с определенной целью объектов в общем контексте моделируемой системы. Цель самой кооперации состоит в том, чтобы специфицировать особенности реализации отдельных наиболее значимых операций в системе. Кооперация определяет структуру поведения системы в терминах взаимодействия участников этой кооперации.

Диаграмма кооперации в нотации UML представлена на рисунке 17.

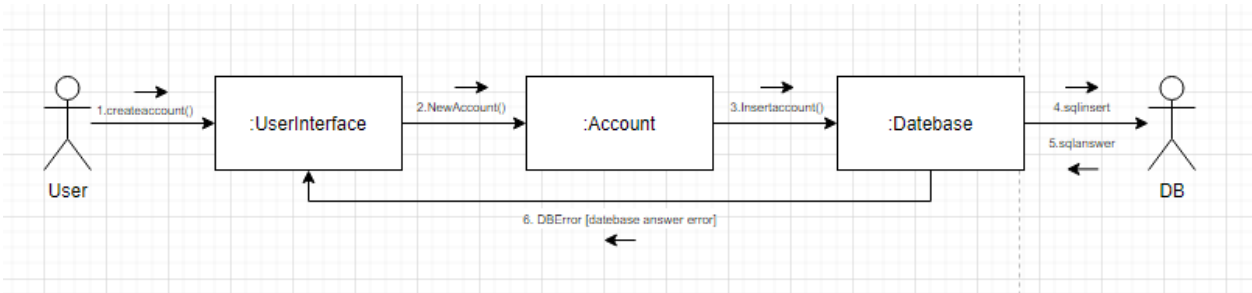


Рисунок 17 – Диаграмма кооперации

Описание диаграммы кооперации представлена в таблице 8.

Таблица 8 – Описание объектов диаграммы кооперации

Объект	Описание	Входной поток	Выходной поток
UserInterface	Интерфейс предназначен для работы пользователя с системой.	createaccount()	newaccount()
Account	Класс для хранения данных пользователя	newaccount()	insertaccount()
Datebase	Класс для осуществления связи системы с БД	insertaccount()	DBError [datebase answer error]
DB	Класс для отображения базы данных	sqlinsert	sqlanswer

В данной системе выделено 5 объектов: «User», «UserInterface», «Account», «DateBase», «DB». Также были определены связи между объектами.

2.4 Проектирование пользовательского интерфейса

Пользовательский интерфейс должен быть интуитивно понятным, чтобы пользователь мог эффективно решать свои задачи сразу после запуска программы без необходимости изучать ее или читать инструкции.

Проектирование интерфейса происходит исходя из ранее разработанных функциональных требований.

В интерфейсе приложения должны присутствовать следующие элементы:

- окно авторизации пользователя;
- главное окно приложения;
- строка основного меню;
- кнопки добавления, удаления, редактирования;
- модули поиска и сортировки;
- кнопки закрытия приложения и перехода в другие окна.

План интерфейса окна авторизации пользователя представлен на рисунке 18.

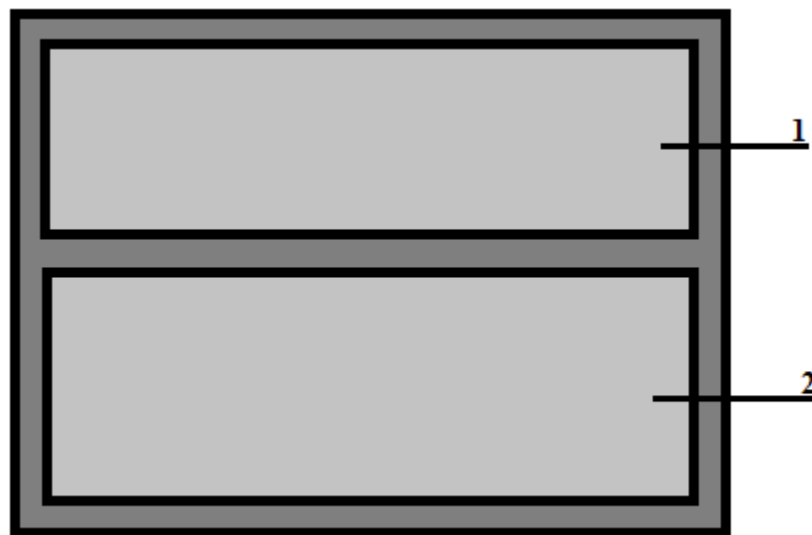


Рисунок 18 – План интерфейса окна авторизации пользователя

В области под номером 1 планируется модуль выбор пользователя (преподаватель, ученик).

В области под номером 2 будет осуществляться ввод логина и пароля.

План интерфейса главного окна приложения представлен на рисунке 19.

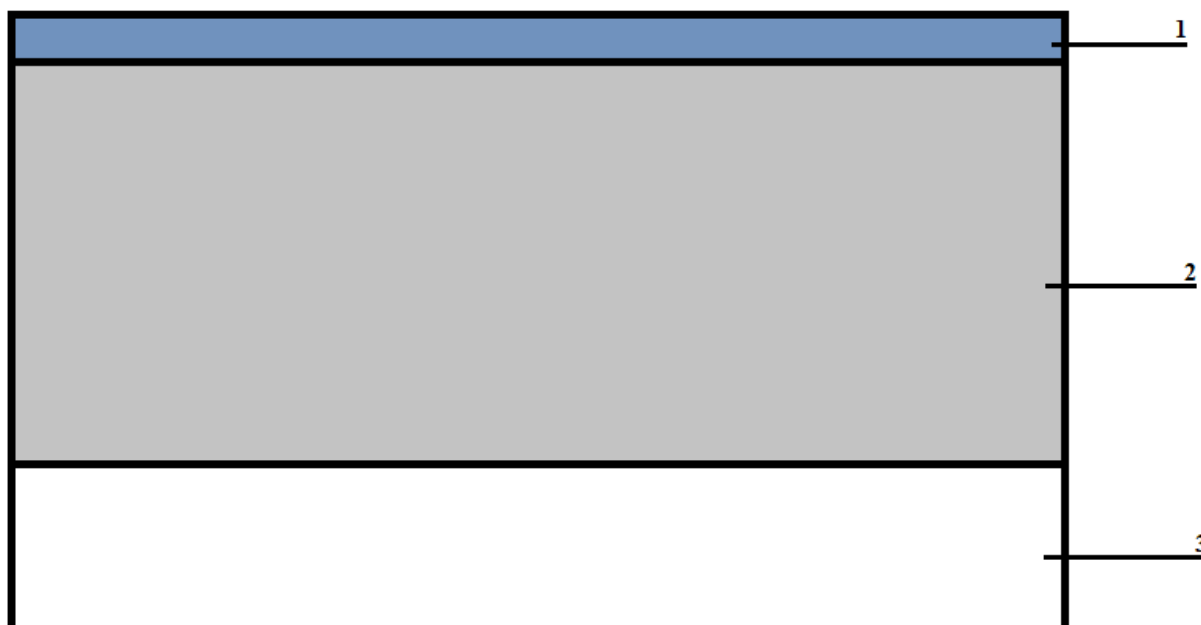


Рисунок 19 – План интерфейса главного окна приложения

В области под номером 1 будет размещена главная строка меню, в которой будет находиться выпадающий список с основными разделами системы, кнопка выхода из приложения.

Область номер 2 будет выделена под вывод данных.

В области под номером 3 планируется разместить модули сортировки, поиска, добавления, удаления и редактирования данных.

2.5 Планирование проекта и оценка бюджета

Управление проектами – это совокупность процессов по планированию, координации и контролю работ для реализации целей проектов с учетом ограничений на ресурсы, бюджет и требований качества. В более широком смысле под управлением проектами понимается применение знаний, практического опыта, инструментальных средств и методов для удовлетворения потребностей заинтересованных лиц проекта.

В ходе планирования проекта было выделено 19 задач. Список задач проекта, их виды, длительность и предшественники приведены в таблице 9.

Таблица 9 – Задачи проекта

Номер	Название задачи	Длительность (дни)
1	Изучение предметной области	3
2	Изучение готовых решений	2
Номер	Название задачи	Длительность (дни)
3	Сравнение готовых решений	1
4	Составление требований к подсистеме реактивного программирования	2
5	Составление требований к подсистеме построения интерфейса	2
6	Анализ функциональных требований	2
8	Выбор технологий и средств разработки	1
9	Проектирование подсистемы реактивного программирования	2
10	Проектирование подсистемы построения интерфейса	2
11	Реализация подсистемы реактивного программирования	10
12	Реализация подсистемы построения интерфейса	14
13	Тестирование методом причинно-следственных связей	2
14	Автоматизированное тестирование	1
15	Структурное тестирование	1
16	Тестирование интерфейса	1
17	Вычисление метрик качества кода	2
18	Составление глоссария	1
19	Составление отчета	4

В MS Project 2013 создадим проект с начальной датой 13.04.21. Занесем вышеперечисленные задачи в созданный проект.

Выделим необходимые ресурсы для выполнения каждой из задач данного проекта. Ресурсы и затраты представлены в таблице 10.

Таблица 10 – Ресурсы проекта

Номер	Название ресурса	Ставка
1	Программист 1	30 тысяч руб/мес
2	Программист 2	30 тысяч руб/мес
3	Тестировщик	25 тысяч руб/мес
4	Документовед	20 тысяч руб/мес

Для каждой задачи были назначены ресурсы (таблица 11).

Таблица 11 – Назначения ресурсов задачам

Номер ресурса	Номера задач
1	1, 2, 3, 4, 6, 7, 8, 9, 11, 13, 14, 17
2	1, 2, 3, 5, 6, 7, 8, 10, 12, 15, 16, 17
3	13, 14, 15, 16
4	18, 19

Распределение ресурсов по задачам представлено на рисунке 20.

	Название задачи	Длительн	Начало	Окончание	Названия ресурсов
0	Библиотека для построения реактивного пользовательского интерфейса	40 дней	Пн 13.04.21	Пт 05.06.21	
1	Начало проекта	0 дней	Пн 13.04.21	Пн 13.04.21	
2	Анализ предметной области	12 дней	Пн 13.04.21	Вт 28.04.21	1
3	Изучение предметной области	3 дней	Пн 13.04.21	Ср 15.04.21	Программист 1 ;Программист 2
4	Анализ продуктов-аналогов	3 дней	Чт 16.04.21	Пн 20.04.21	3
5	Изучение готовых решений	2 дней	Чт 16.04.21	Пт 17.04.21	Программист 1 ;Программист 2
6	Сравнение готовых решений	1 день	Пн 20.04.21	Пн 20.04.21	5 ;Программист 2
7	Постановка задачи	2 дней	Вт 21.04.21	Ср 22.04.21	4
8	Составление требований к подсистеме реактивного программирования	2 дней	Вт 21.04.21	Ср 22.04.21	Программист 1
9	Составление требований к подсистеме построения интерфейса	2 дней	Вт 21.04.21	Ср 22.04.21	Программист 2
10	Анализ требований	3 дней	Чт 23.04.21	Пн 27.04.21	7
11	Анализ функциональных требований	2 дней	Чт 23.04.21	Пт 24.04.21	Программист 1 ;Программист 2
12	Анализ нефункциональных требований	1 день	Пн 27.04.21	Пн 27.04.21	11 ;Программист 2
13	Выбор технологий и средств разработки	1 день	Вт 28.04.21	Вт 28.04.21	10 ;Программист 2
14	Проектирование	2 дней	Ср 29.04.21	Чт 30.04.21	2
15	Проектирование подсистемы реактивного программирования	2 дней	Ср 29.04.21	Чт 30.04.21	Программист 1
16	Проектирование подсистемы построения интерфейса	2 дней	Ср 29.04.21	Чт 30.04.21	Программист 2
17	Реализация	14 дней	Пт 01.05.21	Ср 20.05.21	14
18	Реализация подсистемы реактивного программирования	10 дней	Пт 01.05.21	Чт 14.05.21	Программист 1
19	Реализация подсистемы построения интерфейса	14 дней	Пт 01.05.21	Ср 20.05.21	Программист 2
20	Тестирование	7 дней	Чт 21.05.21	Пт 29.05.21	17
21	Тестирование методом причинно-следственных связей	2 дня	Чт 21.05.21	Пт 22.05.21	Тестирующий ;Программист 1
22	Автоматизированное тестирование	1 день	Пн 25.05.21	Пн 25.05.21	21 ;Программист 1
23	Структурное тестирование	1 день	Вт 26.05.21	Вт 26.05.21	22 ;Программист 2
24	Тестирование интерфейса	1 день	Ср 27.05.21	Ср 27.05.21	23 ;Программист 2
25	Вычисление метрик качества кода	2 дня	Чт 28.05.21	Пт 29.05.21	24 ;Программист 1
26	Документирование	5 дней	Пн 01.06.21	Пт 05.06.21	20
27	Составление глоссария	1 день	Пн 01.06.21	Пн 01.06.21	Документовед
28	Составление отчета	4 дня	Вт 02.06.21	Пт 05.06.21	27 Документовед
29	Конец проекта	0 дней	Пт 05.06.21	Пт 05.06.21	26

Рисунок 20 – Список задач проекта

Занесем вышеперечисленные ресурсы в проект. Результаты представлены на рисунке 21.

По составленному плану разработки была проанализирована загруженность ресурсов и составлен график трудозатрат. Загруженность ресурсов представлена на рисунке 23.

Название	Начало	Окончание	Оставшиеся трудозатраты
Программист 1	Пн 13.04.21	Пт 29.05.21	232 часов
Программист 2	Пн 13.04.21	Пт 29.05.21	256 часов
Тестировщик	Чт 21.05.21	Ср 27.05.21	40 часов
Документовед	Пн 01.06.21	Пт 05.06.21	40 часов

Рисунок 23 – Загруженность ресурсов

График трудозатрат представлен на рисунке 24.

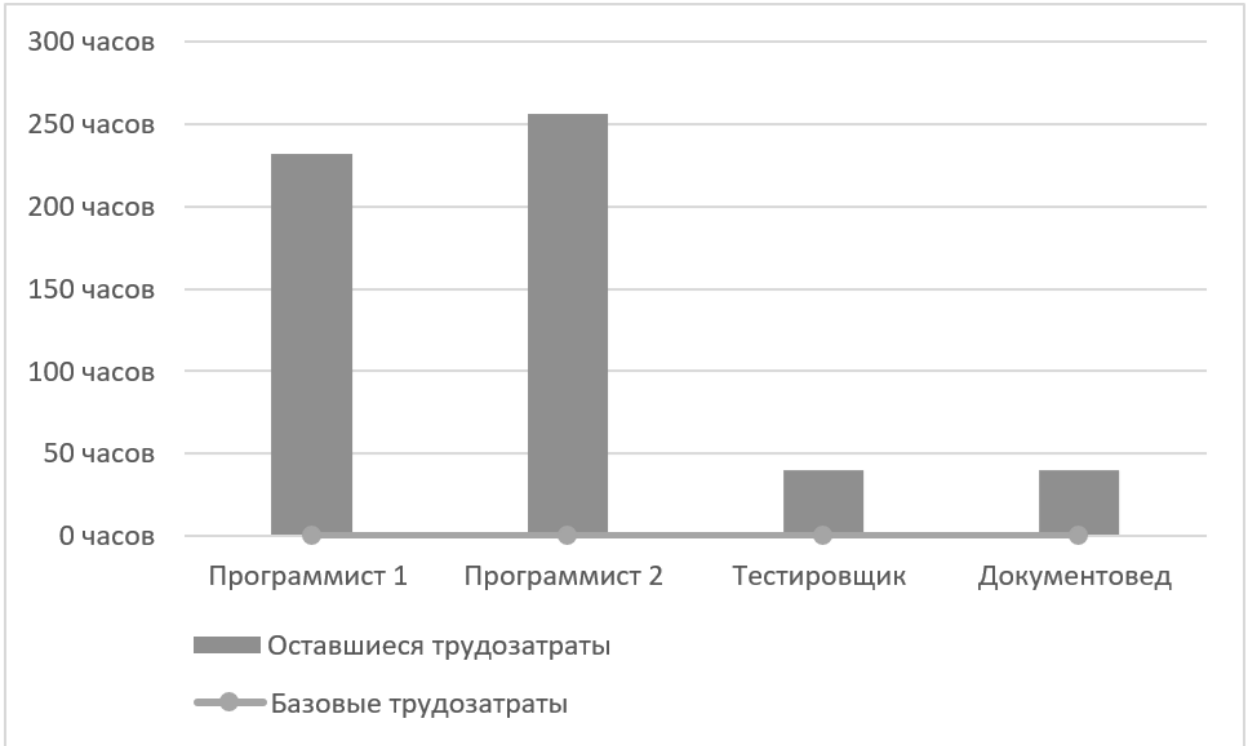


Рисунок 24 – График трудозатрат

Был составлен график затрат на выполнение отдельных задач первого уровня, благодаря чему можно оценить стоимость реализации отдельных этапов разработки библиотеки. Затраты на выполнение отдельных задач представлены на рисунке 25.

Название	Фактически затраты	Оставшиеся затраты	Базовые затраты	Затраты	Отклонение по стоимости
Начало проекта	0,00 ₽	0,00 ₽	0,00 ₽	0,00 ₽	0,00 ₽
Анализ предметной области	0,00 ₽	36 000,00 ₽	0,00 ₽	36 000,00 ₽	36 000,00 ₽
Проектирование	0,00 ₽	6 000,00 ₽	0,00 ₽	6 000,00 ₽	6 000,00 ₽
Реализация	0,00 ₽	36 000,00 ₽	0,00 ₽	36 000,00 ₽	36 000,00 ₽
Тестирование	0,00 ₽	19 750,00 ₽	0,00 ₽	19 750,00 ₽	19 750,00 ₽
Документирование	0,00 ₽	5 000,00 ₽	0,00 ₽	5 000,00 ₽	5 000,00 ₽
Конец проекта	0,00 ₽	0,00 ₽	0,00 ₽	0,00 ₽	0,00 ₽

Рисунок 25 – Затраты на задачи

Стоимость реализации проекта составляет 102 750 руб. Разработка будет идти с 13 апреля 2021 года по 6 июня 2021 года. Длительность разработки составит 40 рабочих дней. Трудозатраты на реализацию составят 568 часов.

3 Реализация системы

3.1 Физическая модель базы данных

Физическая модель строится на основе логической модели данных и учитывает особенности целевой СУБД. В качестве СУБД был выбран продукт MySQL.

При помощи case-средства ERWin Data Modeler была разработана физическая модель, представленная на рисунке 26.

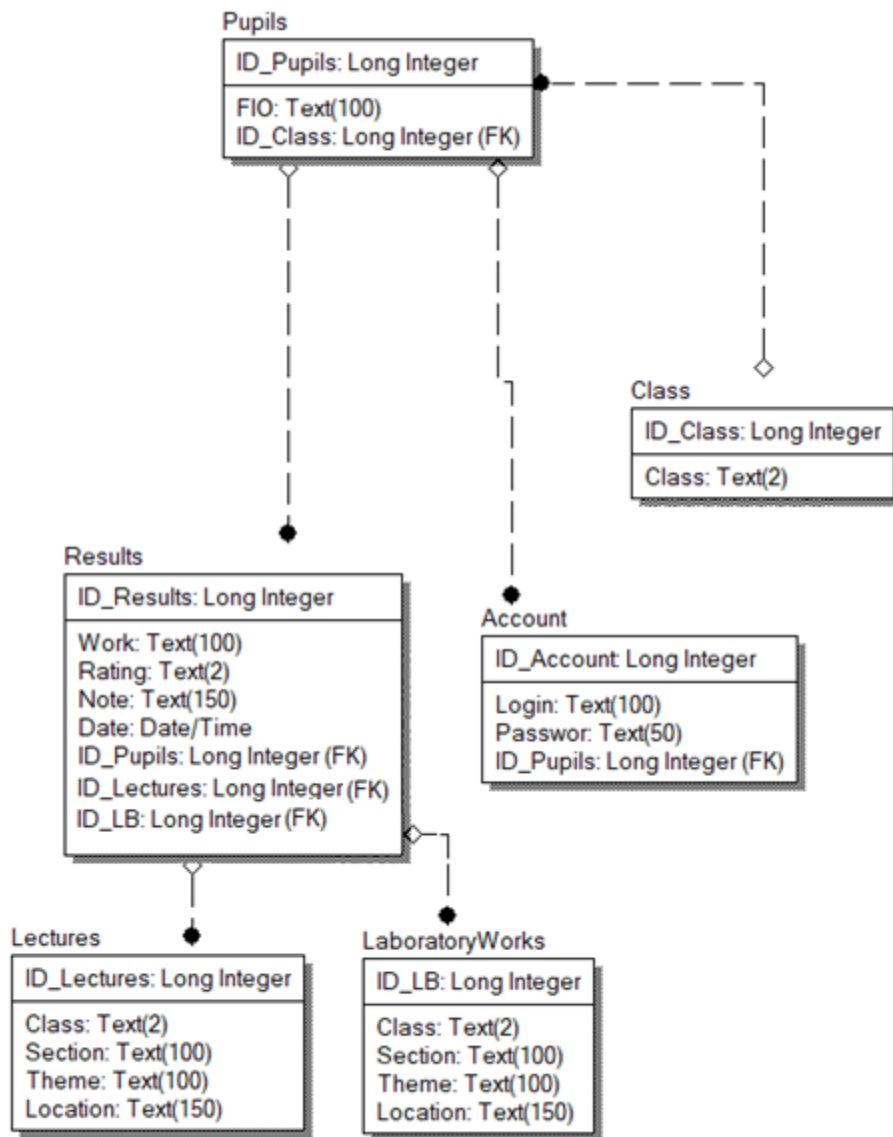


Рисунок 26 – Физическая модель базы данных

Сгенерированный скрипт DDL в ERWin Data Modeler был выполнен в СУБД Microsoft SQL.

3.2 Диаграмма компонентов

Чтобы показать, как осуществляется взаимодействие между программными компонентами, была разработана диаграмма компонентов (рисунок 27).

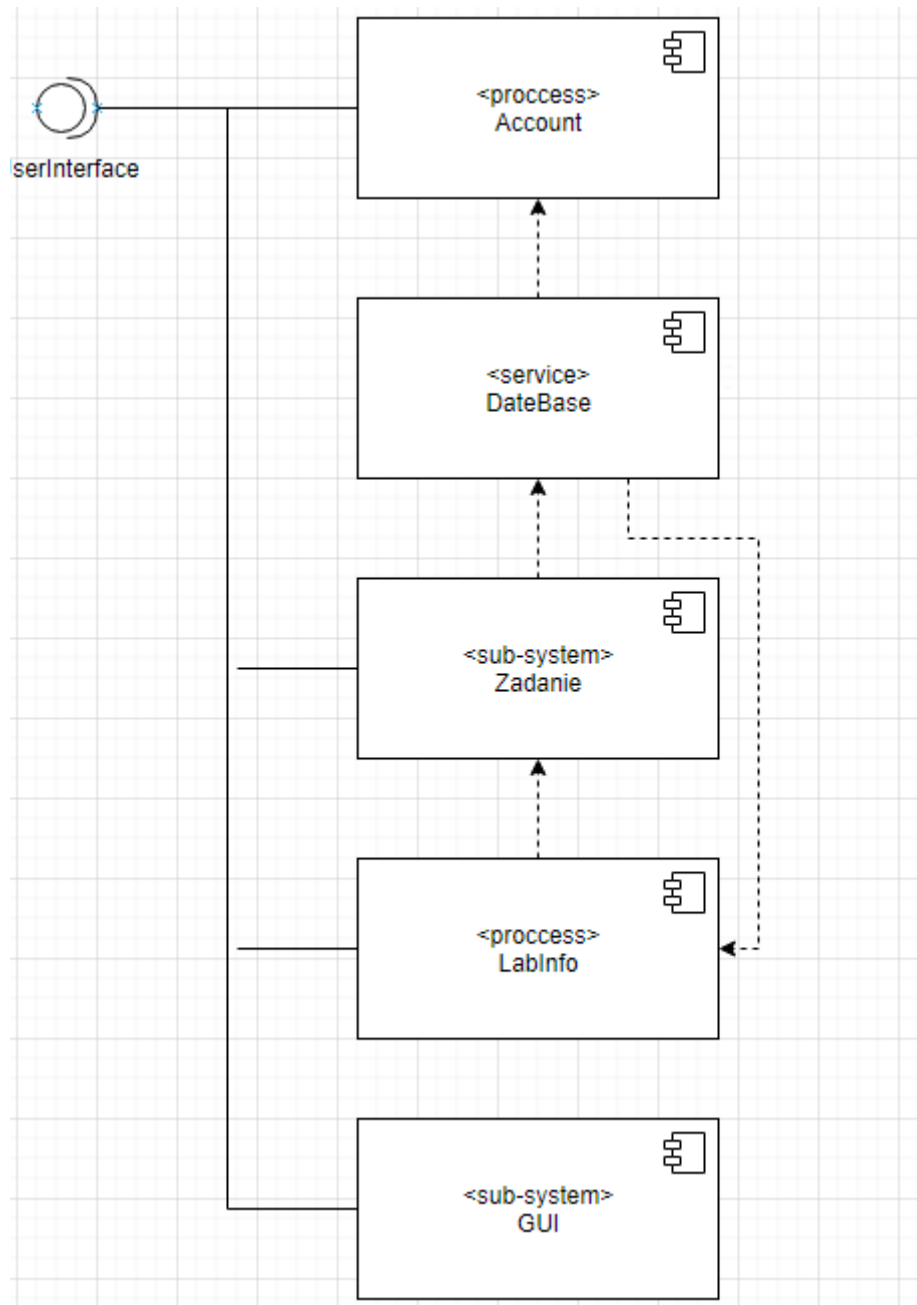


Рисунок 27 – Диаграмма компонентов

Описание диаграммы компонентов представлено в таблицах 12 - 15.

Таблица 12 – Описание компонента Labinfo

Название компонента	LabInfo
Описание	Используется для хранения лабораторных работ.
Тип	process
Предоставляемые интерфейсы	-
Используемые интерфейсы	-
Зависимости	-

Таблица 13 – Описание компонента Zadanie

Название компонента	Zadanie
Описание	Подсистема, осуществляющая проведение лабораторной работы. Выводится задание из Labinfo и пользователь записывает ответ в поле ответа.
Тип	s
Предоставляемые интерфейсы	UserInterface
Используемые интерфейсы	-
Зависимости	LabInfo

Таблица 14 – Описание компонента Account

Название компонента	Account
Описание	Хранит данные об аккаунте в процессе работы пользователя с системой.
Тип	process
Предоставляемые интерфейсы	UserInterface
Используемые интерфейсы	-
Зависимости	-

Таблица 15 – Описание компонента GUI

Название компонента	GUI
Описание	Подсистема отображения графического интерфейса пользователя.

Продолжение таблицы 15

Название компонента	GUI
Тип	sub-system
Предоставляемые интерфейсы	-
Используемые интерфейсы	UserInterface
Зависимости	-

Описание компонента DataBaseController представлено в таблице 16.

Таблица 16 – Описание компонента DataBaseController

Название компонента	DataBaseController
Описание	Осуществляет работу системы с базой данных.
Тип	service
Предоставляемые интерфейсы	-
Используемые интерфейсы	-
Зависимости	Account LabInfo

Данная диаграмма обеспечивает согласованный переход от логического к физическому представлению системы в виде программных компонентов.

3.3 Диаграмма развертывания

Диаграмма развертывания – это диаграмма, на которой представлены узлы выполнения программных компонентов реального времени, а также процессов и объектов.

Диаграмма развертывания применяется для представления общей конфигурации и топологии распределенной программной системы и содержит изображение размещения компонентов по отдельным узлам системы. Кроме того, диаграмма развертывания показывает наличие физических соединений – маршрутов передачи информации между аппаратными устройствами, задействованными в реализации системы. Диаграмма развертывания представлена на рисунке 28.

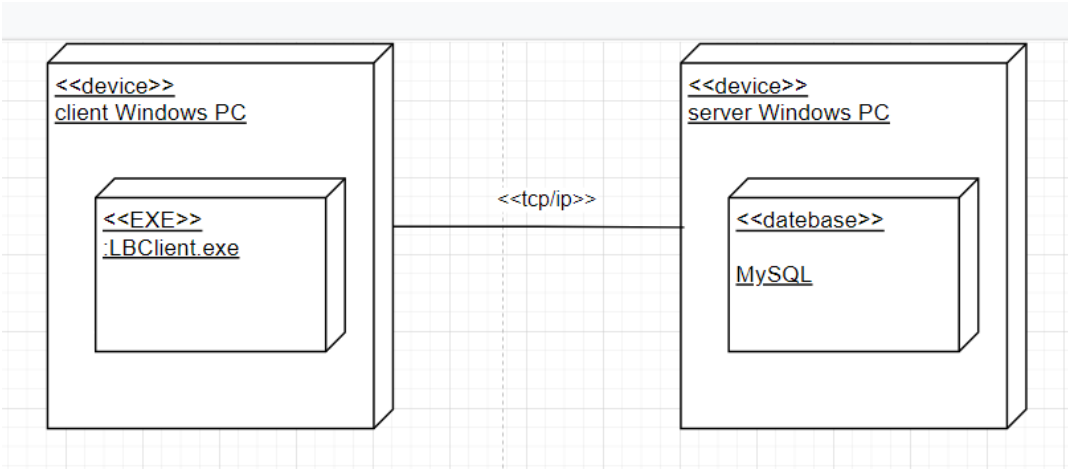


Рисунок 28 – Диаграмма развертывания

Описание узлов диаграммы развертывания представлена в таблице 17.

Таблица 17 – Описание узлов диаграммы развертывания

Узел	Описание	Стереотип	Связи
LBClient.exe	Программа для проведения занятий по физике	Execution environment	-
server: Windows PC	ПК, на котором разворачивается серверная часть системы	device	client: Windows PC [tcp/ip]
MySQL	СУБД с открытым исходным кодом для хранения данных, необходимых для работы системы	-	-
client: Windows PC	Клиентский ПК с ОС Windows.	device	server: Windows PC [tcp/ip]

Диаграмма развертывания предназначена для визуализации элементов и компонентов программы, существующих только на этапе ее исполнения.

3.4 Реализация пользовательского интерфейса

На основе анализа требований был спроектирован простой и интуитивно понятный пользовательский интерфейс с помощью Windows Forms.

Windows Forms – интерфейс программирования приложений (API), отвечающий за графический интерфейс пользователя и являющийся частью Microsoft .NET Framework [15].

На рисунке 29 представлен графический интерфейс формы авторизации пользователей.

1

2

Рисунок 29 – Форма авторизации

Рассмотрим основные элементы интерфейса формы авторизации, представленной на рисунке 29:

- 1 – выбор типа пользователя;
- 2 – ввод логина пользователя.

При заполнении формы пользователь нажимает кнопку «Отправить», после чего проверяется наличие данного пользователя в базе данных. Если пользователь не найден, будет выведена соответствующая ошибка, иначе, пользователь будет перенаправлен на форму ввода пароля которая представлена на рисунке 30.

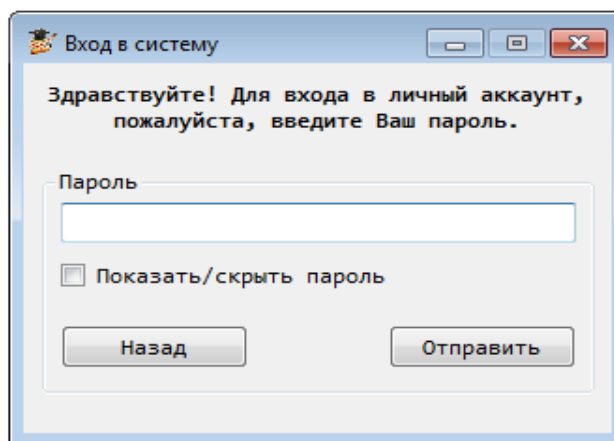


Рисунок 30 – Графический интерфейс формы ввода пароля

После успешного ввода пароля пользователь будет перенаправлен на основную страницу приложения (рисунок 31).

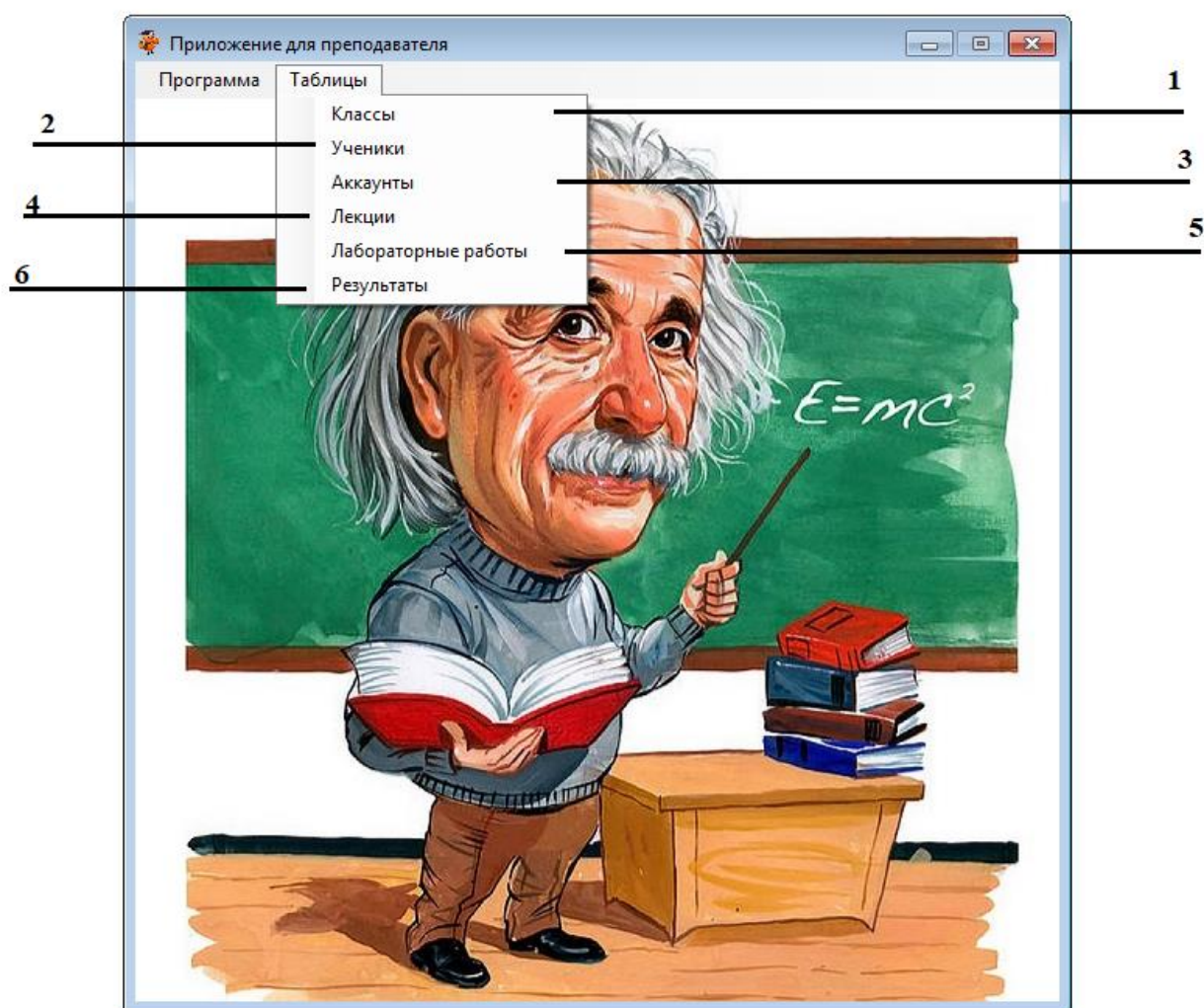


Рисунок 31 – Основная страница приложения

Рассмотрим основные элементы интерфейса формы авторизации, представленной на рисунке 32:

- 1 – добавление или изменение класса;
- 2 – добавление или изменение ученика;
- 3 – добавление или изменение паролей и логинов учеников;
- 4 – добавление или изменение лекций;
- 5 – добавление или изменение лабораторных работ;
- 6 – добавление результатов за выполненные работы.

Рассмотрим типовые страницы приложения. На рисунке 32 изображена страница отображения списка учеников. Цифрами на рисунке обозначены:

- 1 – форма ввода данных ученика;
- 2 – кнопка добавления в базу;
- 3 – кнопка изменения данных ученика;
- 4 – кнопка удаления из базы.

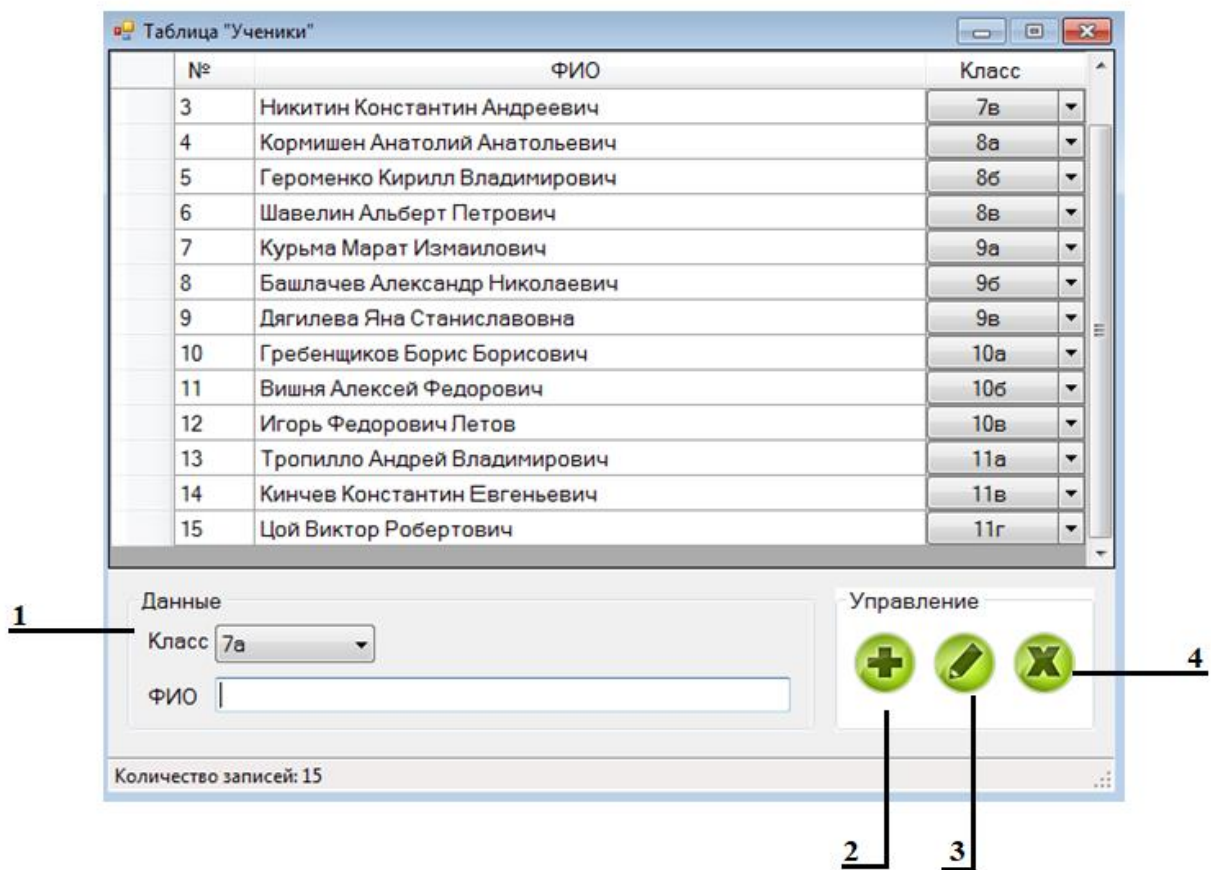


Рисунок 32 – Страница отображения списка учеников

На рисунке 33 изображена страница добавления новой лекции.

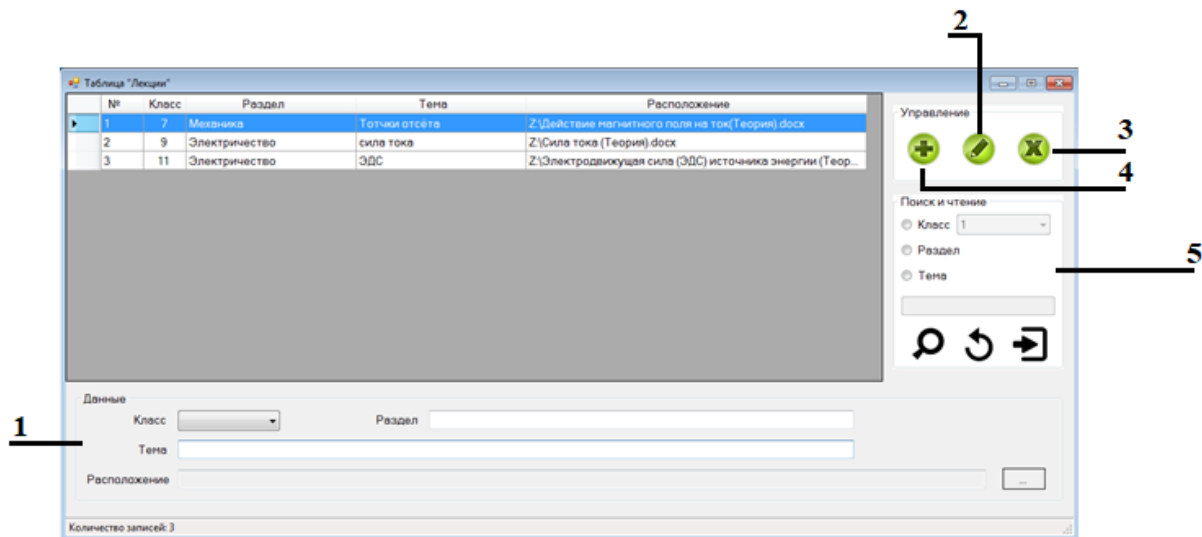


Рисунок 33 – Страница добавления новой лекции

На рисунке 33 цифрами обозначены:

- 1 – данные для лекции;
- 2 – кнопка изменения данных лекции;
- 3 – кнопка добавления в базу лекции;
- 4 – кнопка удаления из базы лекции;
- 5 – поиск и чтение файла с лекцией.

В модуле «Поиск и чтение» была реализована сортировка лекций для удобного поиска в базе, также из этой формы можно сразу открывать текстовый файл с лекцией по выбранной пользователем темой.

4 Контроль качества программного обеспечения

Контроль качества программного обеспечения – неотъемлемый этап в разработке приложений. Одним из важнейших способов контроля качества является тестирование программного обеспечения.

Тестирование – способ обеспечения качества программного продукта, которое представляет собой ряд мероприятий, связанных с различного рода испытаниями тестируемого объекта для установления соответствия или несоответствия техническим требованиям и направленных на выявление дефектов.

Под дефектами понимаются ошибки в программном обеспечении, которые ведут к нарушению работы приложения. Необнаруженные на стадии разработки дефекты способны принести заказчику материальные убытки, а также исправление таких ошибок при эксплуатации программного обеспечения может быть затруднено и экономически невыгодно. Поэтому тестирование является неотъемлемой частью разработки приложений.

Для выпускной квалификационной работы было выбрано тестирование методом черного ящика, которое подразумевает по собой осуществление функционального тестирования.

4.1 Функциональное тестирование

Функциональное тестирование – способ проверки программного обеспечения путем составления плана тестирования, который строится на основе прецедентов, описанных в диаграмме вариантов использования. Следовательно, план тестирования включает в себя не только описание самих тестов, но и их предполагаемые результаты выполнения.

Функциональное тестирование является одним из ключевых видов тестирования, задача которого установить соответствие разработанного программного обеспечения исходным функциональным требованиям заказчика.

План функционального тестирования приведен в таблице 18, а его результаты в приложении А.

Таблица 18 – План функционального тестирования

№	Вариант использования	Тест	Результаты
1	Авторизация	Авторизация с неверными данными	Тест выполнен. Система не нашла пользователя в базе и выдала ошибку (рисунок Б1)
2		Авторизация с верными данными	Тест выполнен. Система перенаправила пользователя на главную страницу (рисунок Б2)
3	Добавить ученика	Открыть форму для добавления ученика. Не заполняя полей, нажать на кнопку «Сохранить».	Тест выполнен. Система вывела сообщения об ошибке (рисунок Б3)
4		Открыть форму для добавления ученика. Заполнить поля и нажать на кнопку «Сохранить».	Тест выполнен. Новый пользователь добавлен в систему. Система вывела сообщение об успешном добавлении новой записи (рисунок Б4)
5	Добавление лекций и лабораторных работ	Добавление лекции, не заполняя форму	Тест выполнен. Лекция не была добавлена. Система вывела сообщение об ошибке (рисунок Б5)
6		Добавление лабораторной работы, не заполняя форму	Тест выполнен. Лабораторная работа не была добавлена. Система вывела сообщение об ошибке (рисунок Б6)
7	Добавление результатов	Добавление результата, не заполняя форму	Тест выполнен. Результат не был добавлен. Система вывела сообщение об ошибке (рисунок Б7)

Таким образом, проанализировав результаты функционального тестирования можно сделать вывод о том, что разработанное программное обеспечение информационно-обучающей системы работает корректно и соответствует исходным функциональным требованиям заказчика.

4.2 Метрики кода

Метрики кода представляют собой набор оценок программного обеспечения, которые дают разработчикам более глубокое представление о разрабатываемом коде и служат рекомендательными характеристиками. Они

позволяют руководителям проектов и предприятий изучить сложность разработанного или разрабатываемого проекта, оценить объем работ, стилистику разрабатываемой программы и усилия, потраченные каждым разработчиком для реализации решения.

Примером часто используемой метрики является комплексный показатель качества кода – Maintainability Index. Рассчитывается метрика по формуле:

$$MI = \max \left(0, (171 - 5.2 * \ln(HV) - 0.23 * CC - 16.2 * \ln(LoC)) * \frac{100}{171} \right),$$

где HV – Halstead Volume, вычислительная сложность: чем больше операторов, тем больше значение этой метрики;

CC – Cyclomatic Complexity, показатель структурной сложности кода, т.е. количество различных ветвей в коде - чем больше этот показатель, тем больше тестов должно быть написано для полного покрытия кода;

LoC – количество строк кода.

Комплексный показатель качества кода может принимать значения от 0 до 100 и показывает относительную сложность поддержки кода. Чем больше значение этой метрики, тем легче поддерживать код.

Для определения вычислительной сложности программы сначала следует рассчитать словарь (1) и длину программы (2), а затем вычислить на их основе вычислительную сложность (3).

Словарь программы:

$$\eta = \eta_1 + \eta_2, \quad (1)$$

где η_1 – число простых (отдельных) операторов и операций в программе;

η_2 – число простых (отдельных) операндов (констант и переменных) в программе.

Длина программы:

$$N = N_1 + N_2, \quad (2)$$

где N_1 – общее число простых (отдельных) операторов и операций в программе;

N_2 – общее число простых (отдельных) операндов в программе.

Вычислительная сложность:

$$HV = N \log_2 \eta. \quad (3)$$

Структурную сложность программы можно рассчитать таким образом:

$$CC = KOP + КОЦ + 1,$$

где КОР – количество операторов разветвления;

КОЦ – количество операторов цикла.

Если полученное значение СС больше 10, то это свидетельствует о структурной сложности программы.

Проведем анализ метрик кода для метода, который загружает данные из таблицы (листинг 1).

Листинг 1 – Метод загрузки данных из таблицы на языке C#.

```
public void download(string commandText, DataGridView table)
{
    try
    {
        if (sqlConnection.State ==
System.Data.ConnectionState.Closed)
            sqlConnection.Open();
        sqlCommand.Connection = sqlConnection;
        sqlCommand.CommandText = commandText;
        sqlDataReader = sqlCommand.ExecuteReader();

        if (sqlDataReader.HasRows)
        {
            table.Rows.Clear();
            int currentRow = 0;
            int rowsCounter = 0;
            while (sqlDataReader.Read())
            {
                currentRow = table.Rows.Add();
                ++rowsCounter;
                int columns = table.ColumnCount;
                // Количество столбцов в таблице.
```

```

        bool isNumber = false;

        // Смещение столбцов из-за столбца №.

        for (int column = 0; column <
columns; ++column)
        {
            if (column != 1)
            {
                if (!isNumber)

                    table.Rows[currentRow].Cells[column].Value =
sqlDataReader[column];

                else

                    table.Rows[currentRow].Cells[column].Value =
sqlDataReader[column - 1];

            }
            else
            {
                isNumber = true;

                table.Rows[currentRow].Cells[column].Value = rowsCounter;

            }

        }

    }
else
{
    table.Rows.Clear();
}
}

```

Произведем расчет комплексного показателя качества кода для метода (листинг 1).

Данные, необходимые для расчета формул 1 и 2, приведены в таблице 19.

Таблица 19 – Данные для расчета словаря и длины программы

Операторы			Операнды		
№	Оператор	Количество	№	Операнд	Количество
1	if	4	1	isNumber	3
2	++	2	2	currentRow	2
3	!=	1	3	rowsCounter	3
4	=	13	4	columns	2
5	<	1			
8	==	2			
9	;	18			
Итого	$\eta_1=9$	$N_1=39$		$\eta_2=4$	$N_2=11$

Словарь программы:

$$\eta = \eta_1 + \eta_2 = 10 + 4 = 14.$$

Длина программы:

$$N = N_1 + N_2 = 39 + 11 = 50.$$

Вычислительная сложность:

$$HV = 50 \times 3,807 = 190.35.$$

Количество строк кода:

$$LoC = 32.$$

Структурная сложность:

$$CC = 3 + 1 + 1 = 5.$$

Комплексный показатель качества кода:

$$MI = \max\left(0, (171 - 5.2 * \ln(190.35) - 0.23 * 5 - 16.2 * \ln(32)) * \frac{100}{171}\right) = 113.$$

Комплексный показатель качества равен 113.

На основе результатов расчёта можно сделать вывод о том, что структурная сложность данного метода невысока, следовательно, программное обеспечение не является структурно сложным. Комплексный показатель качества кода находится чуть выше среднего уровня, что говорит о том, что исходный код программного обеспечения сравнительно просто поддерживать.

Таким образом, программное обеспечение информационно-обучающей системы в целом имеет высокий уровень качества.

Заключение

В ходе выполнения выпускной квалификационной работы была разработана информационно-обучающая система проведения занятия физике.

После рассмотрения предметной области и постановки задач был проведен сравнительный анализ программных средств, выявлены их преимущества и недостатки. На его основе были составлены требования к разрабатываемой системе.

На этапе проектирования системы было осуществлено планирование разработки и оценка бюджета, которое показало, что длительность реализации проекта составляет 40 дней и на его реализацию следует затратить 102 750 руб. Также на этом этапе были созданы концепции окон пользовательского интерфейса приложения, разработана структура классов и логическая модель базы данных.

Разработка программного средства была разделена на две составляющие: разработку клиентской части и разработку серверной части. Клиентская часть должна быть представлена настольным приложением на языке C#. Серверная часть должна быть выполнена на языке SQL и использовать СУБД MySQL.

Тестирование клиентской части приложения проводилось методами функционального тестирования. Также были просчитаны метрики кода для оценки его качества.

Как видно из всего вышесказанного, поставленная задача по реализации информационно-обучающей системы проведения лабораторных работ по физике была выполнена.

Список использованных источников

1. Попова М.В. Информационные, коммуникационные технологии: Учебник / М.В. Попова, А.В. Сапожников, В.И. Сапожников – М.: РПА «АПР», – 296 с.
2. Хачатурова С.С. Необходимость использования электронных обучающих систем для повышения качества учебного процесса // Проблемы современной науки и образования. – 2016. – № 11(53). – С. 61–63.
3. Хачатурова С.С. Информационные технологии и их значение в современном мире // Вестник науки и образования. – 2016. – № 6 (18). – С.34–35.
4. PhET: Бесплатные онлайн-симуляторы физики, химии, биологии, наук о земле и математики [Электронный ресурс]. <https://phet.colorado.edu> (дата обращения: 24.04.2021).
5. Курс: Виртуальные лабораторные комплексы (ВЛК) [Электронный ресурс]. <http://lms.tpu.ru/course/view.php?id=8742> (дата обращения: 24.04.2021).
6. Виртуальная физика биология химия экология | Виртуальная лаборатория ВиртуЛаб [Электронный ресурс]. <http://www.virtulab.net/> (дата обращения: 24.04.2021).
7. Algodoo [Электронный ресурс] <http://www.algodoo.com> (дата обращения: 24.04.2021).
8. Анализ требований // Wikipedia [Электронный ресурс]. – URL: https://ru.wikipedia.org/wiki/Анализ_требований (дата обращения: 02.06.2021)
9. Горяшин Ю. В., Семухина Е. В., Кропова Т. Р. Сравнительный анализ существующих программ для сервисных центров // Научно-методический электронный журнал «Концепт». – 2015. – Т. 33. – С. 111–115.
10. Мокрозуб, В. Г. Реляционные базы данных в автоматизированных интеллектуальных информационных системах / В. Г. Мокрозуб. – Монография: Издательский дом «Спектр», 2011. - 108 с.

11. Конноли, Т. Базы данных: проектирование, реализация, сопровождение. Теория и практика / Т. Конноли, К. Бегг, А. Страчан.– М. : Издательский дом «Вильямс», 2000. – 1120 с.
12. Назарова, О. Б. Разработка реляционных баз данных с использованием CASE-средства All Fusion ERWin Data Modeler : учеб.-метод. пособие / О. Е. Масленникова, О. Б. Назарова // 2-е изд., стер. – М. : ФЛИНТА, 2013.
13. Арлоу, Д. UML 2 и унифицированный процесс. Практический объектно-ориентированный анализ и проектирование / Д. Арлоу, И. Нейштадт – СПб: СимволПлюс, 2007. – 624 с.
14. Прошкина, Е. Н. Методы и средства проектирования информационных систем и технологий. Методические указания к выполнению лабораторных работ / Е. Н. Прошкина. – Пенза: ПГУ, 2014. – 92с.
15. Windows Forms // Wikipedia [Электронный ресурс]. – URL: https://ru.wikipedia.org/wiki/Windows_Forms (дата обращения: 03.06.2021)

Приложение А

ЛИСТИНГ ПРОГРАММЫ

(рекомендуемое)

Листинг А.1 – Код файла TeacherVersion.sln

```
namespace TeacherVersion
{
    class Constants
    {
        public class Arguments
        {
            private const string initialArgument =
                "The_teacher's_life_is_more_important!";

            public static string InitialArgument
            {
                get
                {
                    return initialArgument;
                }
            }
        }

        public class WindowsNames
        {
            private const string warning = "Предупреждение";
            private const string error = "Ошибка";
            private const string information = "Информация";
            private const string selectLecture = "Выбрать файл с лекцией";
            private const string selectLaboratoryWorks = "Выбрать файл с лабораторной работой";

            public static string Warning
            {
                get
                {
                    return warning;
                }
            }

            public static string Error
            {
                get
                {
                    return error;
                }
            }

            public static string Information
            {
```

```
get
{
return information;
}
}
```

```
public static string SelectLecture
{
get
{
return selectLecture;
}
}
```

```
public static string SelectLaboratoryWorks
{
get
{
return selectLaboratoryWorks;
}
}
}
```

```
public class ToolTips
{
private const string insert = "Вставить запись в таблицу";
private const string update = "Обновить запись в таблице";
private const string delete = "Удалить запись из таблицы";
private const string search = "Осуществить поиск по выбранному критерию";
private const string cancel = "Отменить поиск";
private const string open = "Открыть выбранный документ";
```

```
public static string Insert
{
get
{
return insert;
}
}
```

```
public static string Update
{
get
{
return update;
}
```

```

}
}

public static string Delete
{
get
{
return delete;
}
}

public static string Search
{
get
{
return search;
}
}

public static string Cancel
{
get
{
return cancel;
}
}

public static string Open
{
get
{
return open;
}
}

public class Messages
{
private const string noTeacher = "Вы запустили программу не под
учётной записью преподавателя!\n" +
"В связи с этим функциональность приложения будет ограничена.";
private const string sqlException = "Не удалось подключиться к
базе данных, либо SQL-запрос содержит ошибку!";
private const string invalidOperationException = "Произошёл
вызов метода, недопустимого для текущего состояния объекта!";

```

```

private const string notSupportedException = "Была осуществлена
попытка чтения, поиска или записи в потоке, " +
"который не поддерживает функции вызова!";
private const string indexOutOfRangeException = "Индекс
находился вне границ массива!";
private const string win32Exception = "При открытии указанного
файла приложением произошла ошибка!";
private const string fileNotFoundException = "Файл с указанным
адресом не существует!";
private const string objectDisposedException = "Попытка
осуществления операции над удалённым объектом!";
private const string overflowException = "В результате
преобразования string в byte произошло переполнение!";
private const string formatException = "Формат аргумента string
(дата) не соответствует спецификациям параметра вызываемого
аргумента!";
private const string noClass = "Вы не заполнили поле \"Класс\".
\nПожалуйста, заполните поле \"Класс\", и попробуйте ещё раз.";
private const string classMatches = "Введённое Вами наименование
класса уже имеется в таблице.";
private const string noFIO = "Вы не заполнили поле \"ФИО\".
\nПожалуйста, заполните поле \"ФИО\", и попробуйте ещё раз.";
private const string noClassID = "Вы не выбрали значение в
списке выбора \"Класс\". \nПожалуйста, выберите значение в
списке выбора \"Класс\", и попробуйте ещё раз.";
private const string nothingPupils = "Вы не заполнили поле
\"ФИО\" и не выбрали значение в списке выбора \"Класс\"." +
"\nПожалуйста, заполните поле \"ФИО\", выберите значение в
списке выбора \"Класс\" и попробуйте ещё раз.";
private const string noFIOID = "Вы не выбрали значение в списке
выбора \"ФИО\"." +
"\nПожалуйста, выберите значение в списке выбора \"ФИО\" и
попробуйте ещё раз.";
private const string noLogin = "Вы не заполнили поле \"Логин\".
\nПожалуйста, заполните поле \"Логин\", и попробуйте ещё раз.";
private const string loginMatches = "Введённый Вами логин уже
имеется в таблице.";
private const string noPassword = "Вы не заполнили поле
\"Пароль\". \nПожалуйста, заполните поле \"Пароль\", и
попробуйте ещё раз.";
private const string nothing = "Вы не заполнили поля \"Логин\" и
\"Пароль\". \nПожалуйста, заполните поля \"Логин\" и \"Пароль\",
и попробуйте ещё раз.";
private const string nothingAccounts = "Вы не выбрали значение в
списке выбора \"ФИО\" и не заполнили поля \"Логин\" и
\"Пароль\"." +

```

```

"\nПожалуйста, выберите значение в списке выбора \"ФИО\",
заполните поля \"Логин\", \"Пароль\", и попробуйте ещё раз.";
private const string noClassList = "Вы не выбрали значение в
списке выбора \"Класс\"." +
"\nПожалуйста, выберите значение в списке выбора \"Класс\" и
попробуйте ещё раз.";
private const string noSection = "Вы не заполнили поле
\"Раздел\". \nПожалуйста, заполните поле \"Раздел\", и
попробуйте ещё раз.";
private const string noTheme = "Вы не заполнили поле \"Тема\".
\nПожалуйста, заполните поле \"Тема\", и попробуйте ещё раз.";
private const string noLecture = "Вы не выбрали файл с лекцией.
\nПожалуйста, выберите файл с лекцией и попробуйте ещё раз.";
private const string noLaboratoryWork = "Вы не выбрали файл с
лабораторной работой. \nПожалуйста, выберите файл с лабораторной
работой и попробуйте ещё раз.";
private const string nothingLectures = "Вы не выбрали значение в
списке выбора \"Класс\", не заполнили поля \"Раздел\", \"Тема\"
и не выбрали файл с лекцией." +
"\nПожалуйста, выберите значение в списке выбора \"Класс\",
заполните поля \"Раздел\", \"Тема\", выберите файл с лекцией, и
попробуйте ещё раз.";
private const string nothingLaboratoryWork = "Вы не выбрали
значение в списке выбора \"Класс\", не заполнили поля
\"Раздел\", \"Тема\" и не выбрали файл с лабораторной работой."
+
"\nПожалуйста, выберите значение в списке выбора \"Класс\",
заполните поля \"Раздел\", \"Тема\", выберите файл с
лабораторной работой, и попробуйте ещё раз.";
private const string noFIOResults = "Вы не выбрали значение в
списке выбора \"ФИО\". \nПожалуйста, выберите значение в списке
выбора \"ФИО\", и попробуйте ещё раз.";
private const string noWorkNumber = "Вы не заполнили поле \"№
работы\". \nПожалуйста, заполните поле \"№ работы\", и
попробуйте ещё раз.";
private const string excessWorkNumber = "Был введён слишком
большой № лабораторной работы." +
"\nМаксимально возможный № лабораторной работы – 255.";
private const string noWorkName = "Вы не заполнили поле
\"Название работы\". \nПожалуйста, заполните поле \"Название
работы\", и попробуйте ещё раз.";
private const string noRating = "Вы не выбрали значение в списке
выбора \"Оценка\". \nПожалуйста, выберите значение в списке
выбора \"Оценка\", и попробуйте ещё раз.";
private const string nothingResults = "Вы не выбрали значение в
списке выбора \"ФИО\", не заполнили поле \"№ работы\", не

```

```

заполнили поле \"Название работы\" и не выбрали значение в
списке выбора \"Оценка\").\" +
\"\\nПожалуйста, выберите значение в списке выбора \"ФИО\",
заполните поле \"№ работы\", заполните поле \"Название работы\",
выберите значение в списке выбора \"Оценка\", и попробуйте ещё
раз.\";
private const string successfulInsert = \"Данные успешно
добавлены в таблицу \";
private const string successfulUpdate = \"Выбранные Вами данные
успешно обновлены.\";
private const string successfulDelete = \"Выбранные Вами данные
успешно удалены.\";
private const string maxLengthClass = \"Максимально допустимая
длина наименования класса – 10 символов.\";
private const string maxLengthFIO = \"Максимально допустимая
длина ФИО – 100 символов.\";
private const string maxLengthLogin = \"Максимально допустимая
длина логина – 25 символов.\";
private const string maxLengthPassword = \"Максимально допустимая
длина пароля – 64 символа.\";
private const string maxLengthSection = \"Максимально допустимая
длина раздела физики – 100 символов.\";
private const string maxLengthTheme = \"Максимально допустимая
длина темы лекции – 256 символов.\";
private const string maxLengthWorkName = \"Максимально допустимая
длина названия работы – 100 символов.\";
private const string maxLengthNote = \"Максимально допустимая
длина темы лекции – 1000 символов.\";
private const string excess = \"\\nВы превысили указанное
требование на 1 символ.\\nДанный символ был автоматически
удалён.\";
private const string noPathToFile = \"Не указан путь к файлу.\";
private const string noFileExists = \"Файл с указанным адресом не
существует.\";

```

Листинг A.2 – Код файла Login.sln

```

namespace Login
{
    class Constants
    {
        public class WindowsNames
        {
            private const string warning = \"Предупреждение\";
            private const string error = \"Ошибка\";

            public static string Warning
            {

```

```

        get
        {
            return warning;
        }
    }

    public static string Error
    {
        get
        {
            return error;
        }
    }
}

public class Messages
{
    private const string noUserType = "Вы не выбрали тип  
пользователя (преподаватель или студент).\nПожалуйста, выберите  
тип пользователя, и попробуйте ещё раз.";
    private const string noLogin = "Вы не заполнили поле  
\"Логин\". Пожалуйста, заполните поле \"Логин\", и попробуйте  
ещё раз.";
    private const string noPassword = "Вы не заполнили  
поле \"Пароль\". Пожалуйста, заполните поле \"Пароль\", и  
попробуйте ещё раз.";
    private const string nothing = "Вы не выбрали тип  
пользователя (преподаватель или студент) и не заполнили поле  
\"Логин\".\n" +
        "Пожалуйста, выберите тип пользователя,  
заполните поле \"Логин\", и попробуйте ещё раз.";
    private const string sqlException = "Не удалось  
подключиться к базе данных, либо SQL-запрос содержит ошибку!";
    private const string objectDisposedException =
        "Попытка осуществления операции над удалённым объектом!";
    private const string invalidOperationException =
        "Произошёл вызов метода, недопустимого для текущего состояния  
объекта!";
    private const string notSupportedException = "Была  
осуществлена попытка чтения, поиска или записи в потоке, " +
        "который не поддерживает функции вызова!";
    private const string indexOutOfRangeException =
        "Индекс находился вне границ массива!";
    private const string noConnection = "Не удалось  
подключиться к базе данных!";

```

```

        private const string noHashFile = "Вход в систему
        невозможен, так как в папке с программой отсутствует файл " +
        FileNames.hashFileName + "!";
        private const string hashFileNotFound = "Файл " +
        FileNames.hashFileName + "не найден!";
        private const string unauthorizedAccessException =
        "Не удалось открыть файл " + FileNames.hashFileName + " из-за
        ошибки доступа!";
        private const string ioException = "Во время
        открытия файла " + FileNames.hashFileName + " произошла ошибка
        ввода-вывода!";
        private const string win32Exception = "При запуске
        клиентского приложения произошла ошибка!";
        private const string noAccount = "Не удалось найти
        указанный аккаунт!";
        private const string invalidPassword = "Неверный
        пароль! Повторите ввод пароля и попробуйте ещё раз.";

```

```

public static string NoUserType
{
    get
    {
        return noUserType;
    }
}

```

```

public static string NoLogin
{
    get
    {
        return noLogin;
    }
}

```

```

public static string NoPassword
{
    get
    {
        return noPassword;
    }
}

```

```

public static string Nothing
{
    get
    {
        return nothing;
    }
}

```

```

        }
    }

    public static string SqlException
    {
        get
        {
            return sqlException;
        }
    }

    public static string ObjectDisposedException
    {
        get
        {
            return objectDisposedException;
        }
    }

    public static string InvalidOperationException
    {
        get
        {
            return invalidOperationException;
        }
    }

    public static string NotSupportedException
    {
        get
        {
            return notSupportedException;
        }
    }

    public static string IndexOutOfRangeException
    {
        get
        {
            return indexOutOfRangeException;
        }
    }

    public static string NoConnection
    {
        get
        {

```

```
        return noConnection;
    }
}

public static string NoHashFile
{
    get
    {
        return noHashFile;
    }
}

public static string HashFileNotFound
{
    get
    {
        return hashFileNotFound;
    }
}

public static string UnauthorizedAccessException
{
    get
    {
        return unauthorizedAccessException;
    }
}

public static string IOException
{
    get
    {
        return ioException;
    }
}

public static string Win32Exception
{
    get
    {
        return win32Exception;
    }
}

public static string NoAccount
{
    get
```

```

        {
            return noAccount;
        }
    }

    public static string InvalidPassword
    {
        get
        {
            return invalidPassword;
        }
    }
}

public class FileNames
{
    public const string hashFileName = "hashes.txt";

    public static string HashFileName
    {
        get
        {
            return hashFileName;
        }
    }
}

public class PathNames
{
    private const string teacherPath =
@"\Teacher\TeacherVersion.exe";
    private const string studentPath =
@"\Student\StudentVersion.exe";
    public static string TeacherPath
    {
        get
        {
            return teacherPath;
        }
    }
    public static string StudentPath
    {
        get
        {
            return studentPath;
        }
    }
}

```

```

    }
    public class Sizes
    {
        private const ushort arraySize = 32;
        public static ushort ArraySize
        {
            get
            {
                return arraySize;
            }
        }
    }
    public class Arguments
    {
        private const string teacherArgument =
"The_teacher's_life_is_more_important!";
        private const string studentArgument =
"The_student's_life_is_more_important!";
        public static string TeacherArgument
        {
            get
            {
                return teacherArgument;
            }
        }
        public static string StudentArgumnet
        {
            get
            {
                return studentArgument;
            }
        }
    }
    public class Connections
    {
        private const string parametersODBC =
"Dsn=KnowledgeTestingServer;uid=EndUser;pwd=Connect,please!";
        private const string parametersSQL = "Data
Source=NONAME-PC\\SQLEXPRESS;Initial
Catalog=KnowledgeTesting;Persist Security Info=True;User
ID=EndUser;Password=Connect,please!";
        public static string ParametersODBC
        {
            get
            {
                return parametersODBC;
            }
        }
    }

```

```

    }
    public static string ParametersSQL
    {
        get
        {
            return parametersSQL;
        }
    }
}
public class requestsSQL
{
    private const string loginSearch = "SELECT * FROM
Accounts WHERE login = ";
    private const string passwordSearch = "SELECT * FROM
Accounts WHERE password = ";

    public static string LoginSearch
    {
        get
        {
            return loginSearch;
        }
    }

    public static string PasswordSearch
    {
        get
        {
            return passwordSearch;
        }
    }
}
}
}

```

Приложение Б

РЕЗУЛЬТАТЫ ФУНКЦИОНАЛЬНОГО ТЕСТИРОВАНИЯ

(рекомендуемое)

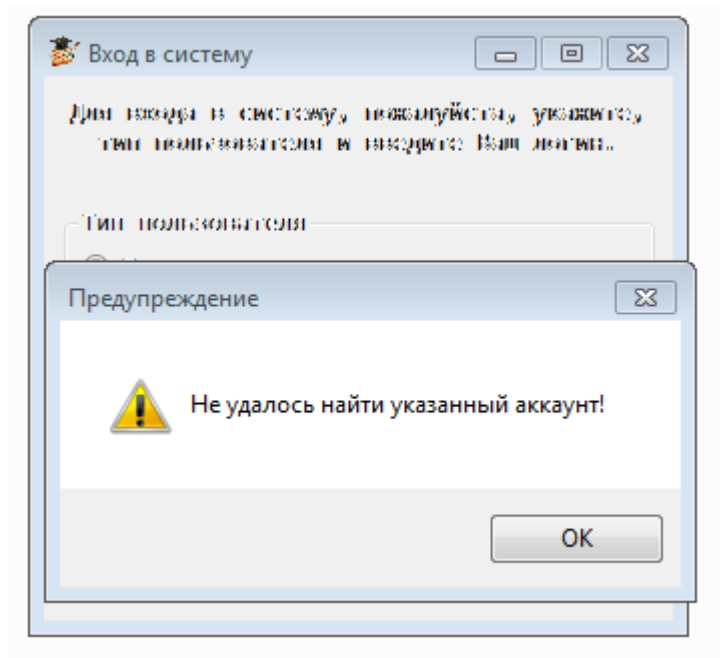


Рисунок Б.1 – Результат функционального теста №1

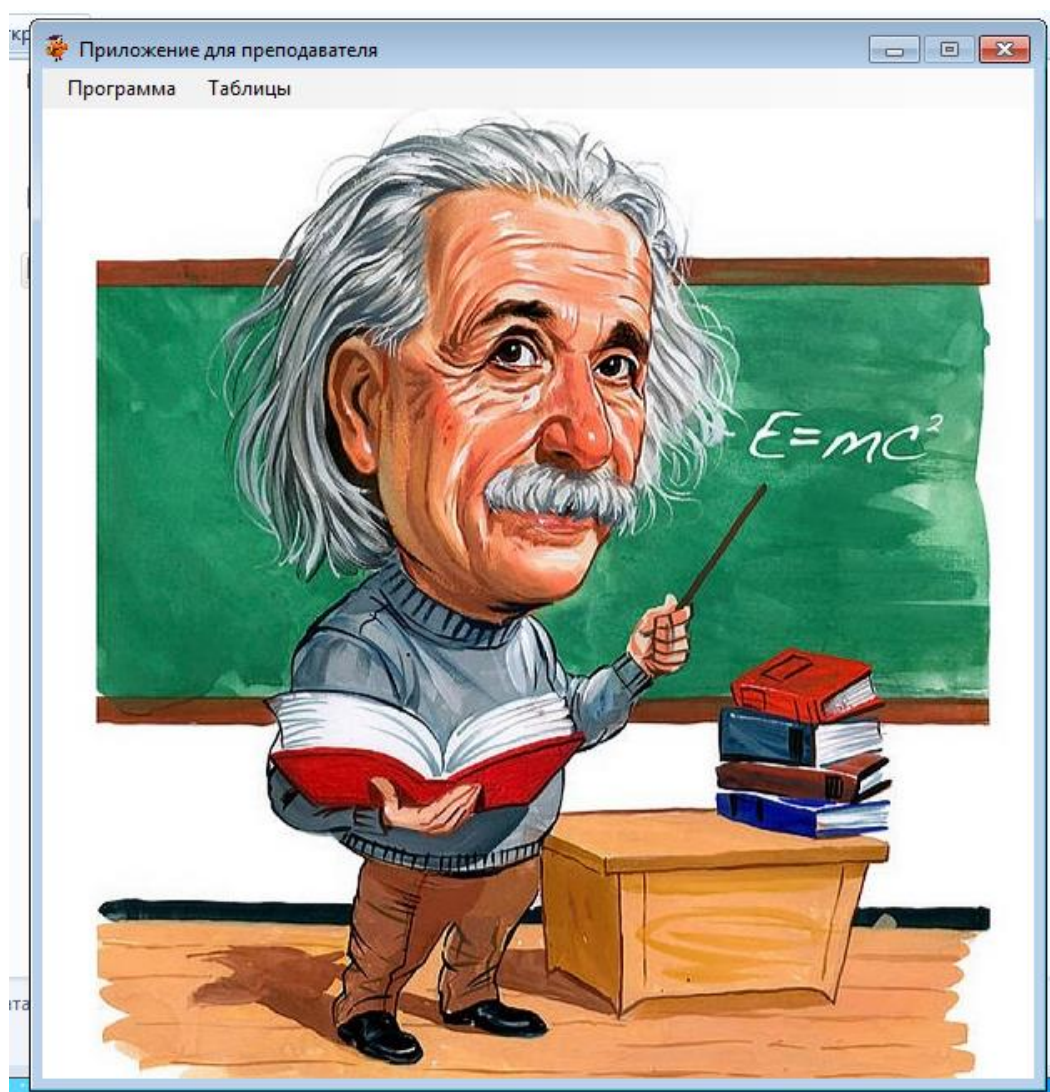


Рисунок Б.2 – Результат функционального теста №2

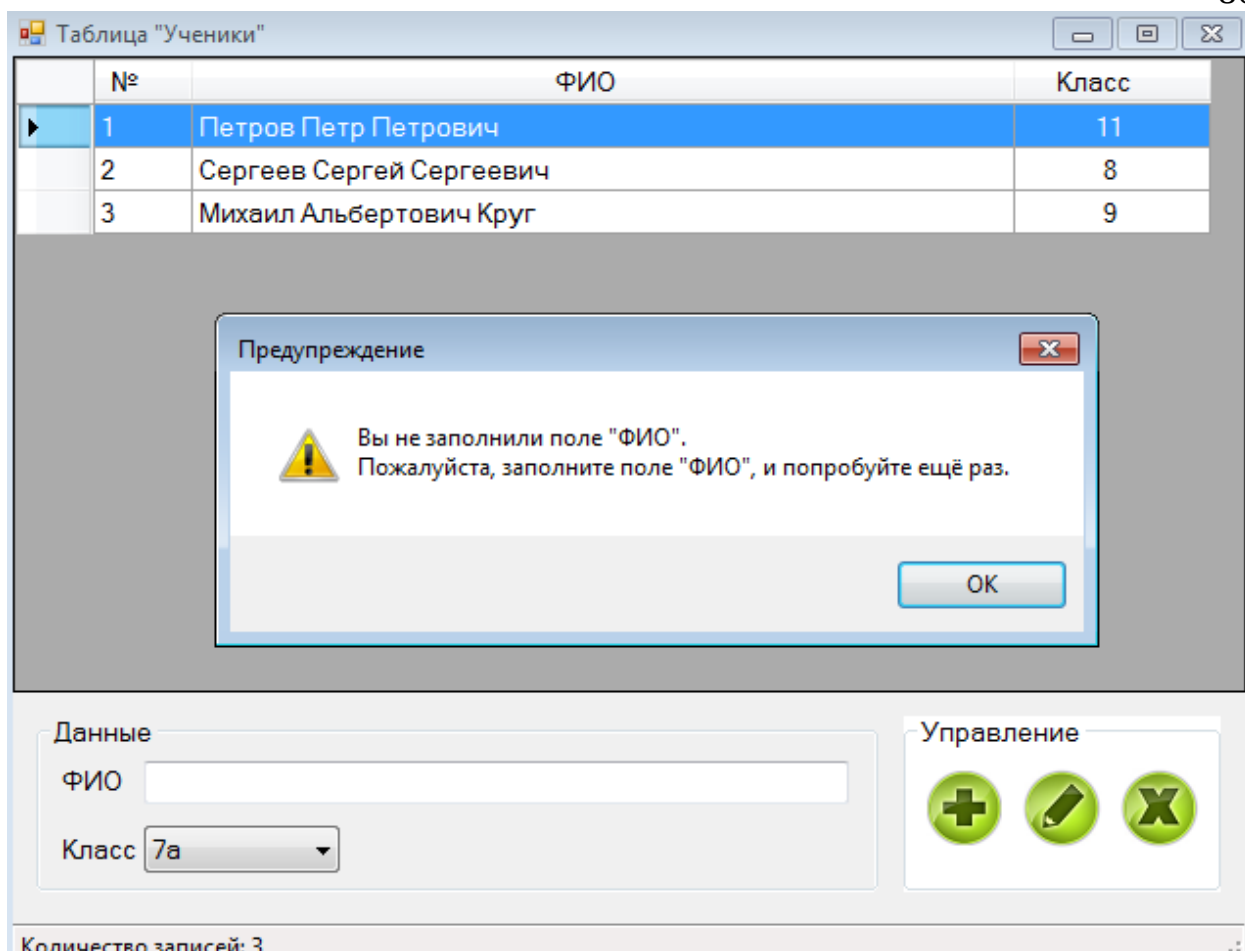


Рисунок Б.3 – Результат функционального теста №3

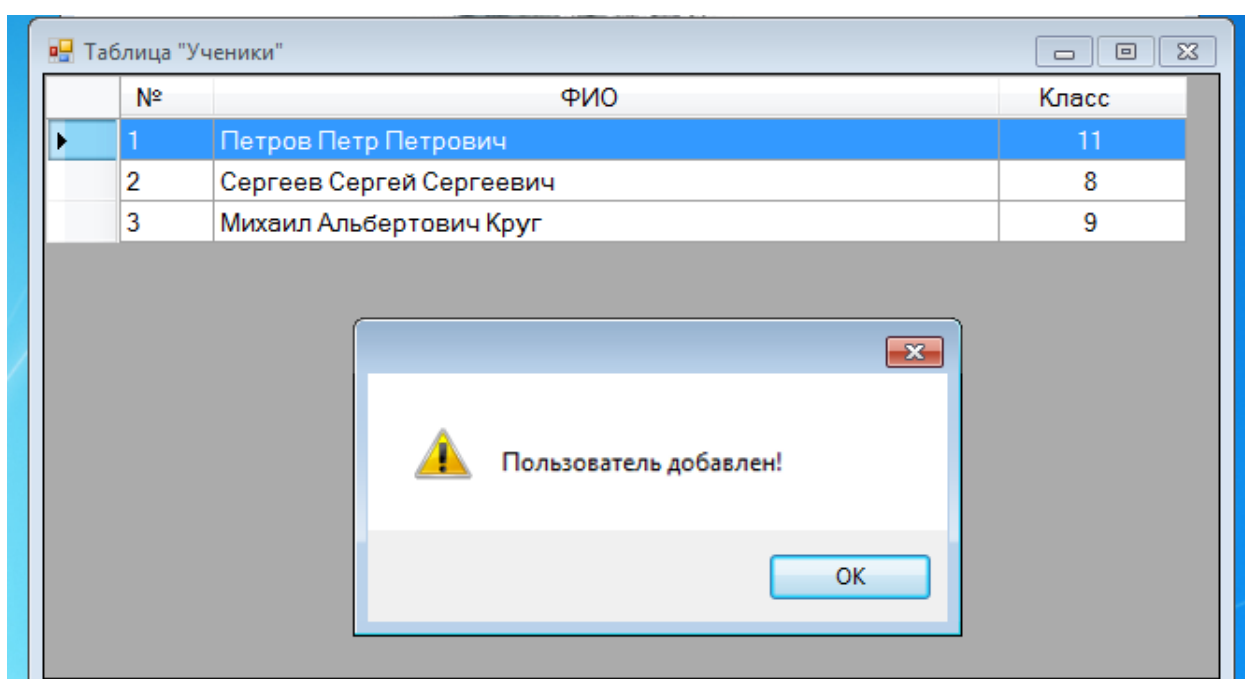


Рисунок Б.4 – Результат функционального теста №4

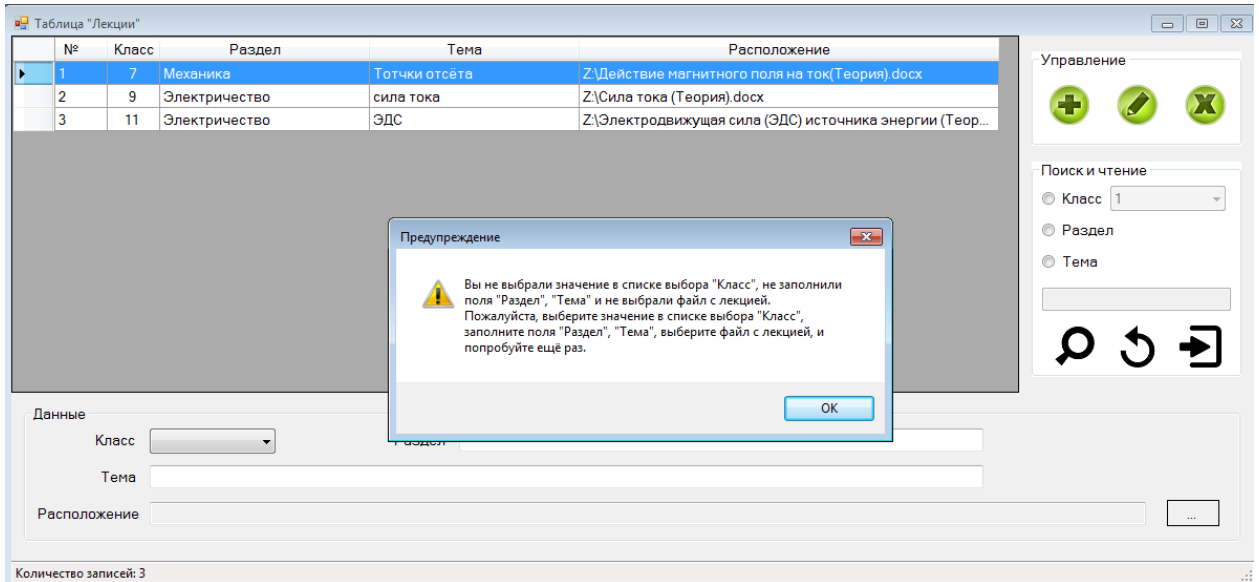


Рисунок Б.5 – Результат функционального теста №5

+

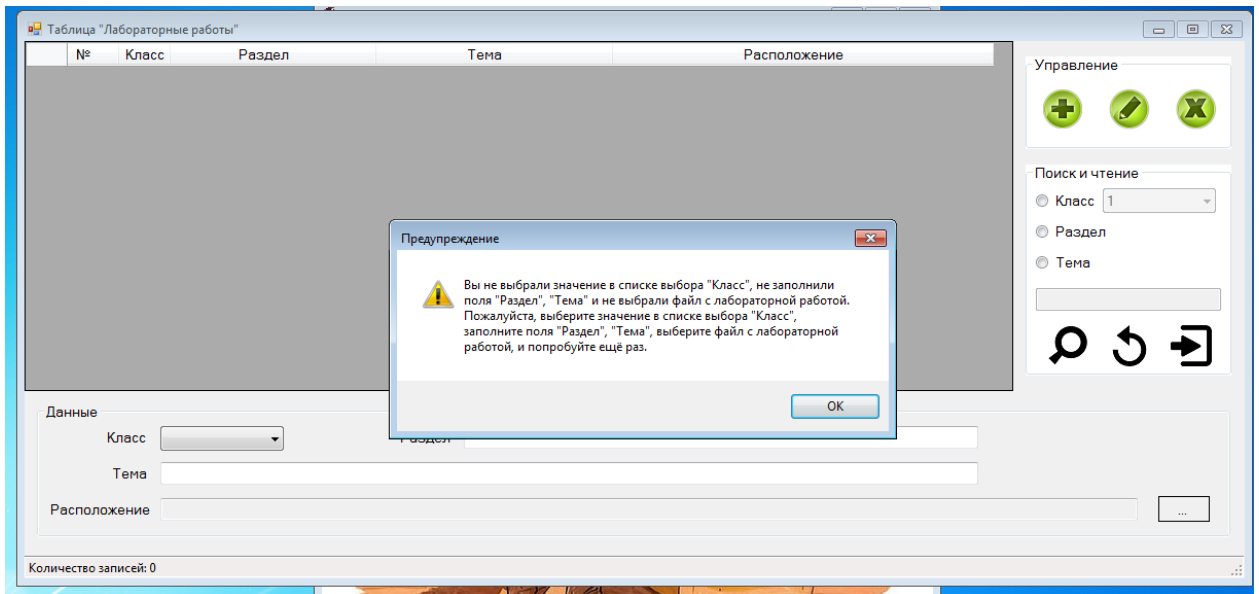


Рисунок Б.6 – Результат функционального теста №6

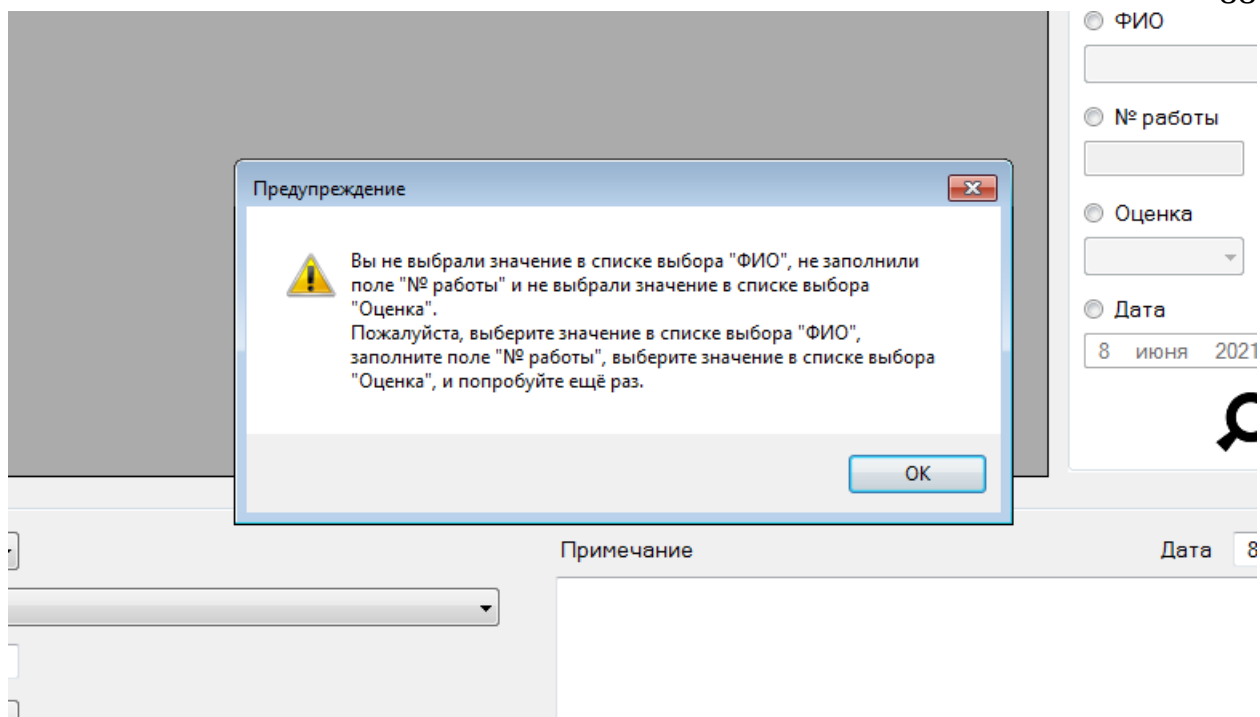


Рисунок Б.7 – Результат функционального теста №7