

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет

вычислительной техники

Кафедра

МОиПЭВМ

Направление подготовки	09.03.04 «Программная инженерия»
Профиль	Программное обеспечение вычислительной техники и автоматизированных систем

БАКАЛАВРСКАЯ РАБОТА

на тему

**Автоматизированная информационная система
тестирования знаний**

Студент	_____	<u>Семенов И.В.</u>
	(подпись, дата)	(ФИО полностью)
Руководитель	_____	<u>Волынская К.И.</u>
	(подпись, дата)	(фамилия, инициалы)
Нормоконтролёр	_____	<u>Попова Н.А.</u>
	(подпись, дата)	(фамилия, инициалы)

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

Заведующий кафедрой	_____	<u>Макарычев П.П.</u>
	(подпись)	(фамилия, инициалы)

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

Секретарь ГЭК	_____	<u>Попова Н.А.</u>
	(подпись)	(фамилия, инициалы)

Пенза, 2016

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
«__» _____ 2016 г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент _____ Семенов Игорь Владимирович _____ гр. 12ВП1 факультета _____ ВТ
<Фамилия, имя, отчество полностью>

2. Тема работы _____ Автоматизированная информационная система тестирования знаний

Тема утверждена приказом ПГУ № 566/0 от " 13 " _____ мая _____ 2016 _____

2. Руководитель работы _____ аспирант кафедры «МОиПЭВМ» Волынская К.И. _____

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

Назначение: автоматизированная информационная система тестирования
знаний

предназначены для выявления уровня знаний студента в ходе прохождения
тестов.

Основные функции:

а) Добавить тест

б) Редактировать тест

в) Удалить тест

г) Пройти тест

д) Ознакомиться с результатами

Технология разработки: ООП, UML, RUP, шаблон проектирования «Шлюз»,
Windows Forms.

Язык программирования: C#, SQL

Среда исполнения: операционные системы семейства Windows

4. Перечень подлежащих разработке вопросов
1) Анализ предметной области
2) Анализ функциональных требований
3) Проектирование
4) Реализация
5) Контроль качества ПО:
– Тестирование
– Анализ качества кода
6) План разработки и оценка бюджета

5. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и требований	10	06.06.2016 - 08.06.2016	
2	Проектирование	25	09.06.2016 - 12.06.2016	
3	Реализация	30	13.06.2016 - 15.06.2016	
4	Контроль качества	20	16.06.2016 - 17.06.2016	
5	План разработки, бюджет	5	18.06.2016 - 19.06.2016	
6	Оформление пояснительной записки	10	19.06.2016 - 20.06.2016	

Дата выдачи задания " ____ " _____ 20____

Руководитель бакалаврской работы _____

(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить
Декан факультета _____
(подпись, дата)

Реферат

Пояснительная записка содержит 105 лиса, 38 рисунков, 28 таблиц, 10 плакатов, 17 использованных источников, 6 приложений.

БАЗА ДАННЫХ, RUP, C#, IDEF1X, UML, VISUAL STUDIO 2013, MSSQLSERVER 2012, ADO.NET, СУБД.

Цель выпускной квалификационной работы – разработать автоматизированную информационную систему тестирования знаний.

Технология разработки – ООП, UML, RUP, шаблон проектирования «Шлюз», WindowsForms.

Результат работы – программное обеспечение Testing, отвечающее всем требованиям технического задания.

Для разработки программного продукта применялись языки программирования C# и SQL, интегрированная среда разработки VisualStudio 2013 и операционная система Windows 7. Для хранения данных была выбрана система управления базами данных SQLServer 2012.

					ПГУ 09.03.04 – 8ВР121.18 ПЗ						
Изм.	Лист	№ докум.	Подпись	Дата							
Разраб.	Семенов И.В				Автоматизированная информационная система тестирования знаний	Лит.		Лист	Листов		
Провер.	Юпова Н.А							4	105		
						Группа 12ВП1					
Н. Контр.											
Утверд.											

Содержание

Введение	7
1 Автоматизированная информационная система тестирования.....	8
1.1 Анализ предметной области и постановка задачи.....	8
1.1.1 Классификация тестов.....	8
1.1.2 Анализ программ аналогов	9
1.1.3 Постановка задачи.....	10
1.1.4 Выбор инструментальных средств	10
1.2 Анализ бизнес-процессов предметной области	12
1.2.1 Построение исходной концептуальной модели данных предметной об- ласти.....	14
1.3 Анализ требований.....	20
1.3.1 Анализ функциональных требований.....	20
2 Проектирование.....	28
2.1 Базовая архитектура системы	28
2.1.1 Логическая модель базы данных.....	29
2.1.1.2 Первая нормальная форма (1NF).....	29
2.1.1.3 Вторая нормальная форма (2NF).....	30
2.1.1.4 Третья нормальная форма (3NF)	30
2.1.2 Пользовательский интерфейс клиентского приложения.....	31
2.1.3 Проектирование программных средств.....	36
3 Реализация.....	38
3.1. Физическая модель и реализация базы данных системы.....	38
3.1.1 Серверная часть приложения системы.....	39
3.1.2 Хранимые процедуры.....	40
3.1.2.1 Реализация хранимых процедур	39
3.1.2.2 Результаты работы серверной части	42
3.1.3 Модель реализации клиентского приложения.....	44
3.2 Контроль качества программного обеспечения.....	50
3.2.1 Функциональное тестирование.....	50

3.2.2 Модульное тестирование.....	51
3.2.3 Метрики кода.....	53
3.3 Планирование разработки и оценка бюджета.....	56
3.3.1 Планирование проекта.....	56
Заключение	64
Список использованных источников	65
Приложение А Листинг программы.....	67
Приложение Б Результаты тестирования	89
Приложение В Листинг тестового проекта	100
Приложение Г Листинг скрипта DDL.....	104

Введение

Web-технологии сегодня позволяют создавать Интернет проекты самого разного типа сложности и целевой направленности. Они, как и любые другие разработки, постоянно совершенствуются и развиваются: добавляются новые, заменяются устаревшие. Иными словами идет естественный процесс эволюции Интернета, вообще, и web-технологий в частности.

Несмотря на то, что разработано достаточное количество программных продуктов, позволяющих автоматизировать процесс тестирования студентов, многие из них обладают недостатками, либо излишней функциональностью. Разработка нового продукта, ориентированного на конкретного пользователя, является важной и актуальной задачей

В данной выпускной квалификационной работе разрабатывается система автоматизации тестирования знаний студентов, подсистема студента. Внедрение данной системы позволит экономить время преподавателей и студентов, а также приведет к более объективному оцениванию знаний.

1 Автоматическая информационная система тестирования знаний

1.1 Анализ предметной области и постановка задачи

Предметной областью является тестирование знаний. Тестирование знаний в педагогике предназначено прежде всего для решения задач диагностики и обучения. Диагностическая функция заключается в выявлении знаний тестируемого [1]. Это основная функция тестирования. Обучающая функция состоит в мотивировании тестируемого к активизации работы по усвоению учебного материала. Тестированию, как методу, контроля знаний присущи следующие положительные стороны:

- стандартизированная процедура проведения;
- справедливость как в процессе контроля, так и в процессе оценки, практически исключая субъективизм преподавателя.
- возможность оценивать знания по всему курсу в целом, в отличие от устного и письменного экзамена;
- точность при оценке знаний, шкала оценки теста состоит из количества вопросов, в то время как, как обычная шкала оценки знаний — только из четырёх;
- экономия, как в плане средств, так и в плане времени;
- минимизация возможности нечестной демонстрации знаний.

1.1.1 Классификация тестов

Тесты можно классифицировать по различным признакам:

- по целям — информационные, диагностические, обучающие, мотивационные, аттестационные;
- по процедуре создания — стандартизованные, не стандартизованные;
- по способу формирования заданий — детерминированные, стохастические, динамические;

- по технологии проведения — бумажные, в том числе бумажные с использованием оптического распознавания, натурные, с использованием специальной аппаратуры, компьютерные;
- по форме заданий — закрытого типа, открытого типа, установление соответствия, упорядочивание последовательности;
- по наличию обратной связи — традиционные и адаптивные.

Традиционный тест содержит список вопросов и различные варианты ответов. Каждый вопрос оценивается в определенное количество баллов. Результат традиционного теста зависит от количества вопросов, на которые был дан правильный ответ. По мнению, Аванесова В. С., традиционный тест – система заданий, предъявляемая в порядке увеличения сложности в одно и то же время, с одинаковой системой оценивания для всех тестируемых. [2]

Особый вид теста, в котором каждое последующее задание выбирается в зависимости от ответов на предыдущие задания. Последовательность заданий и их количество в таком виде теста определяется динамически. Самыми значимыми преимуществами компьютерного адаптивного тестирования перед традиционным являются:

- возможность адаптации под уровень знаний тестируемого (не придется отвечать на слишком сложные или слишком простые вопросы);
- экономия времени и сил за счет сокращения количества заданий (длина теста может быть уменьшена до 60 процентов) без потери уровня достоверности.

1.1.2 Анализ программ аналогов

В данное время существует множество систем, предназначенных для автоматизации тестирования знаний студентов, как платных, так и бесплатных.

Рассмотрим программы, ориентированные на специфику обучения в отечественных ВУЗах. В таблице 1 приведена сравнительная характеристика автоматизированных систем тестирования.

Таблица 1 – Бизнес-процессы предметной области

№ п/п	Название	Работа через интернет	Платформа	Лицензия
1	Система «СИН-ТеЗ»	+	PHP, MySQL	Платная
2	«Конструктор тестов»	-	C++	Бесплатная
3	«OpenTest 2.0»	+	HTML, PHP, JavaScript	Платная

В ходе анализа программ аналогов были выявлены недостатки у вышеперечисленных систем тестирования, а именно функциональная перегруженность, платная лицензия, а также не все системы являются масштабируемыми и не работают через интернет.

Таким образом можем сделать вывод, что разработка собственного средства автоматизированного тестирования знаний лишнего выделенных у программ аналогов недостатков является целесообразной.

1.1.3 Постановка задачи

Данный программный продукт предназначен для автоматизированного тестирования знаний. Должны присутствовать следующие возможности: разграничение уровня доступа для преподавателя и студента, возможность загрузки новых тестов, их редактирования, удаления и добавления. Преподаватель также имеет возможность просматривать результаты тестирования студентов в сводной таблице. Студент также имеет возможность ознакомиться с результатами тестирования по завершении его прохождения.

1.1.4 Выбор инструментальных средств

Для структурированного хранения информации о сборках было принято решение использовать реляционную базу данных, в качестве СУБД – MS SQL Server 2012, а в качестве информационно – логического языка баз данных – SQL. Данный выбор обусловлен тем, что SQL – первый и пока единственный стандартный язык для работы с базами данных, который получил достаточно широкое распространение. Практически все крупнейшие разработчики СУБД в настоящее время создают свои продукты с использованием языка SQL либо с SQL – интерфейсом [3]. В него сделаны огромные инвестиции, как со стороны разработчиков, так и со стороны пользователей. Он стал частью архитектуры приложений, является стратегическим выбором многих крупных и влиятельных организаций [4].

База данных хранится на том же самом сервере, на котором происходит сборка программного продукта. Процесс начала сборки решено сделать согласно расписанию, ввиду того, что инициировать сборку при изменениях в хранилище системы контроля версий не представляется возможным. Это вызвано тем, что время сборки велико, и частота обновлений хранилища высокая.

Выбор интерфейса пал в пользу оконного, так как для него присутствуют множество автоматизированных средств проектирования, а также он предоставляет более широкий спектр возможностей по сравнению с web - интерфейсом.

В качестве языка программирования выбран C# версии 4.0. Выбор обусловлен тем, что данный язык обладает богатой библиотекой классов, позволяющей удобно проектировать, как графический интерфейс, так и взаимодействовать с базой данных. Также C# постоянно развивается и совершенствуется, добавляются новые функции, а работа со старыми становится проще. C/C++ не был выбран из-за более высокого порога вхождения [5].

Разработка проекта будет проводится на основе построения моделей. В качестве языка моделирования выбран UML. Язык UML основан на некотором

числе базовых понятий, которые могут быть применены для внедрения методов объектно-ориентированного анализа и программирования. В UML данный процесс называется Rational Unified Process (RUP) [6].

Суть концепции RUP заключается в последующей декомпозиции на отдельные этапы, на каждом из которых осуществляется разработка соответствующих типов диаграмм модели систем [7].

Для концептуального проектирования модели предметной области выбран Microsoft Office Visio, логическое и физическое проектирование информационной системы выполняется при помощи case - средства ERWin Data Modeler. Интегрированная среда разработки – Microsoft Visual Studio 2013.

Для планирования, распределения ресурсов по задачам и отслеживания прогресса в анализе объемов работ выбран Microsoft Project 2013. Выбор обусловлен тем, что MS Project обладает легкостью освоения, имеет богатые возможности в сфере планирования и оценки бюджета, а также является одной из наиболее популярных программ управления проектами.

1.2 Анализ бизнес-процессов предметной области

Выделим бизнес-процессы, протекающие в системе и занесем их в таблицу 2.

Таблица 2 – Бизнес-процессы предметной области

№ п/п	Бизнес-процесс	Исполнитель	Входные данные		Выходные данные	
			Поставщик	Содержание	Потребитель	Содержание
1	Добавить предмет	Клиентское приложение	Преподаватель	Новый предмет	База данных тестирования	Новый предмет
2	Добавить тест	Клиентское приложение	Преподаватель	Новый тест	База данных тестирования	Новый тест

Продолжение таблицы 2

№ п/п	Бизнес-процесс	Исполнитель	Входные данные		Выходные данные	
			Поставщик	Содержание	Потребитель	Содержание
3	Удалить тест	Клиентское приложение	Преподаватель	Тест, который необходимо удалить	База данных тестирования	Обновлённый список тестов
4	Редактировать тест	Клиентское приложение	Преподаватель	Отредактированный тест	База данных тестирования	Отредактированный тест
5	Добавить пользователя	Клиентское приложение	Преподаватель/студент	Данные о новом пользователе	База данных тестирования	Данные о новом пользователе
6	Получить список тестов	Клиентское приложение	База данных тестирования	Предмет, по которому необходимо получить тесты	Преподаватель/студент	Список тестов
7	Получить список вопросов	Клиентское приложение	База данных тестирования	Тест, по которому необходимо получить вопросы	Преподаватель/студент	Список вопросов
8	Добавить информацию о тестировании	Клиентское приложение	Клиентское приложение	Информация о проведенном тестировании	База данных тестирования	Информация о проведенном тестировании
9	Посмотреть результаты тестирования	Клиентское приложение База данных тестирования	Результаты тестирования	Преподаватель	Результаты тестирования	

Выделенные бизнес – процессы в дальнейшем будут использоваться при проектировании и реализации серверной части разрабатываемого программного продукта.

1.2.1 Построение исходной концептуальной модели данных предметной области

Концептуальное проектирование базы данных заключается в разработке концептуальной модели базы данных, которая не учитывает выбранную модель данных и особенности целевой СУБД [8]. Концептуальная модель, как правило, представлена в виде совокупности типов сущностей и связей между ними. Построение концептуальной модели данных предметной области выполнено в MS Office Visio.

Выделим типы сущностей и занесем результаты в таблицу 3.

Таблица 3 – Типы сущностей предметной области

№	Наименование	Описание	Категория	Количество экземпляров	Режим обновления
1	Ответ	Содержит возможные ответы на вопрос	Стержневая	1000	90% в год
2	Правильный ответ	Содержит правильный ответ на конкретный вопрос	Справочная	1000	90% в год
3	Вопрос	Содержит вопросы для тестов	Справочная	1000	90% в год
4	Студент	Содержит информацию о студенте	Стержневая	1000	90% в год
5	Предмет	Содержит возможные предметы для тестирования	Справочная	100	5% в год
6	Преподаватель	Содержит информацию о преподавателе Стержневая	1000	70% в год	
7	Тест	Содержит перечень тестов по конкретному предмету	Справочная	1000	90% в год
8	Тестирования	Содержит результаты прохождения тестирования	Стержневая	1000	90% в год

Выделим отношения между типами сущностей и занесем результаты в таблицу 4.

Таблица 4 – Типы связей между типами сущностей

№	Тип сущности 1	Тип сущности 2	Описание	Мощность	Обязательность	Ограничения
1	Вопрос	Ответ	Вопрос <i>Имеет</i> Ответ Ответ <i>Относится к</i> Вопросу	1:M	Обязательная (Вопрос не может не иметь ответа)	Без ограничений
2	Вопрос	Правильный ответ	Вопрос <i>Имеет</i> Правильный ответ Правильный ответ <i>Относится к</i> Вопросу	1:1	Обязательная (Вопрос не может не иметь правильного ответа)	Без ограничений
3	Тест	Вопрос	Тест <i>Имеет</i> Вопрос Вопрос <i>Относится к</i> Тесту	1:M Обязательная (Тест не может не иметь ни одного вопроса)	Без ограничений	
4	Предмет	Тест	Предмет <i>Имеет</i> Тест Тест <i>Относится к</i> Предмету	1:M	Необязательная (У предмета может не быть тестов)	Без ограничений
5	Студент	Тестирование	Студент <i>Прошёл</i> Тестирование Тестирование <i>Было пройдено</i> Студентом	M:M	Необязательная (Студент может не пройти ни одного теста)	Без ограничений

Продолжение таблицы 4

№	Тип сущности 1	Тип сущности 2	Описание	Мощность	Обязательность	Ограничения
6	Тестирование	Тест	Тестирование <i>По</i> Тесту Тест <i>Входит в</i> <i>сведения к</i> Тестированию	М : 1	Обязательная (Тестирование не может существовать без теста)	Без ограничений
7	Преподаватель	Предмет	Преподаватель <i>Преподает</i> Предмет Предмет <i>Преподается</i> Преподавателем	М : М	Обязательная (Преподаватель не может преподавать ни одного предмета)	Без ограничений

Выделим атрибуты типов сущностей и сформируем таблицу атрибутов для каждой сущности. Результаты приведены в таблицах 5 – 12.

Таблица 5 – Атрибуты типа сущности «Ответ»

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
ID_Ответ а	ID ответа	Целый	7 0-9999999	нет	РК	1	
Содержание	Содержание ответа	Символьный	250	'А-я', '0-9'	нет		Принцип программного управления
ID_Вопроса	ID вопроса, к которому относится ответ	Целый	7	0-9999999	нет	FK	2

Таблица 6 – Атрибуты типа сущности «Правильный ответ»

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
ID_Ответ а	ID правильно-го ответа	Целый	7	0-999999	нет	РК	1
ID_Ответ а	ID ответа	Целый	7	0-999999	нет	РК	1
ID_Вопр оса	ID вопроса, к которому относится пра-вильный ответ	Целый	7	0-999999	нет	FK	2

Таблица 7 – Атрибуты типа сущности «Вопрос»

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
ID_Вопро са	ID вопроса	Целый	7	0-999999	нет	РК	1
Содержа-ние	Содержание вопроса	Сим-вольный	250	'А-я', '0-9'	нет		Основная функция ЭВМ
ID_Тест а	ID теста, к которому относится во-прос	Целый	7	0-999999	нет	FK	2

Таблица 8 – Атрибуты типа сущности «Студент»

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
ID_Студе нта	ID студента	Целый	7	0-999999	нет	РК	623
Имя	Имя студента	Сим-вольный	20	'А-я'	нет		Иван
Фами-лия	Фамилия сту-дента	Сим-вольный	40	'А-я'	нет		Иванов
Отчество	Отчество сту-дента	Сим-вольный	20	'А-я'	нет		Иванович

Продолжение таблицы 8

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
Логин	Имя пользователя студента	Символьный	20	'A-z', '0-9'	нет		Login1
Пароль	Пароль студента	Символьный	20	'A-z', '0-9'	нет		Pass1

Таблица 9 – Атрибуты типа сущности «Преподаватель»

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
ID_Преподавателя	ID преподавателя	Целый	7	0-9999999	нет	РК	623
Имя	Имя преподавателя	Символьный	20	'А-я'	нет		Петр
Фамилия	Фамилия преподавателя	Символьный	40	'А-я'	нет		Петров
Отчество	Отчество преподавателя	Символьный	20	'А-я'	нет		Петрович
Логин	Имя пользователя преподавателя	Символьный	20	'A-z', '0-9'	нет		Login2
Пароль	Пароль преподавателя	Символьный	20	'A-z', '0-9'	нет		Pass2
ID_Предмета	ID преподаваемого предмета	Целый	7	0-9999999	нет	FK	1

Таблица 10 – Атрибуты типа сущности «Тест»

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
ID_Теста	ID теста	Целый	7	0-9999999	нет	РК	1
Наименование	Наименование теста	Символьный	30	'А-я', '0-9'	нет		Базовый тест

Продолжение таблицы 10

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
ID_Предмета	ID предмета к которому относится тест	Целый	7	0-999999	нет	FK	1

Таблица 11 – Атрибуты типа сущности «Группа»

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
ID_Предмета	ID предмета	Целый	7	0-999999	нет	FK	1
Наименование	Наименование предмета	Символьный	50	'А-я', '0-9'	нет		Информатика

Таблица 12 – Атрибуты типа сущности «Тестирование»

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример
ID_Тестирования	ID тестирования	Целый	7	0-999999	нет	FK	1
Количество правильных ответов	Количество правильных ответов во время тестирования	Целый	3	0-999	нет		10
Дата	Дата окончания тестирования	Дата/время	8	00:00:00 01.01.2016 – 00:00:00 31.12.2040	нет		09:02:32 04.05.2016
Оценка	Оценка по результатам тестирования	Целый	1	2-5	Нет		5
ID_Студента	ID студента	Целый	7	0-999999	нет	FK	623
ID_Теста	ID теста	Целый	7	0-999999	нет	FK	1

На основании выделенных сущностей, их атрибутов и связей построим концептуальную модель предметной области в нотации IDEF1X. Концептуальная модель представлена на рисунке 1.

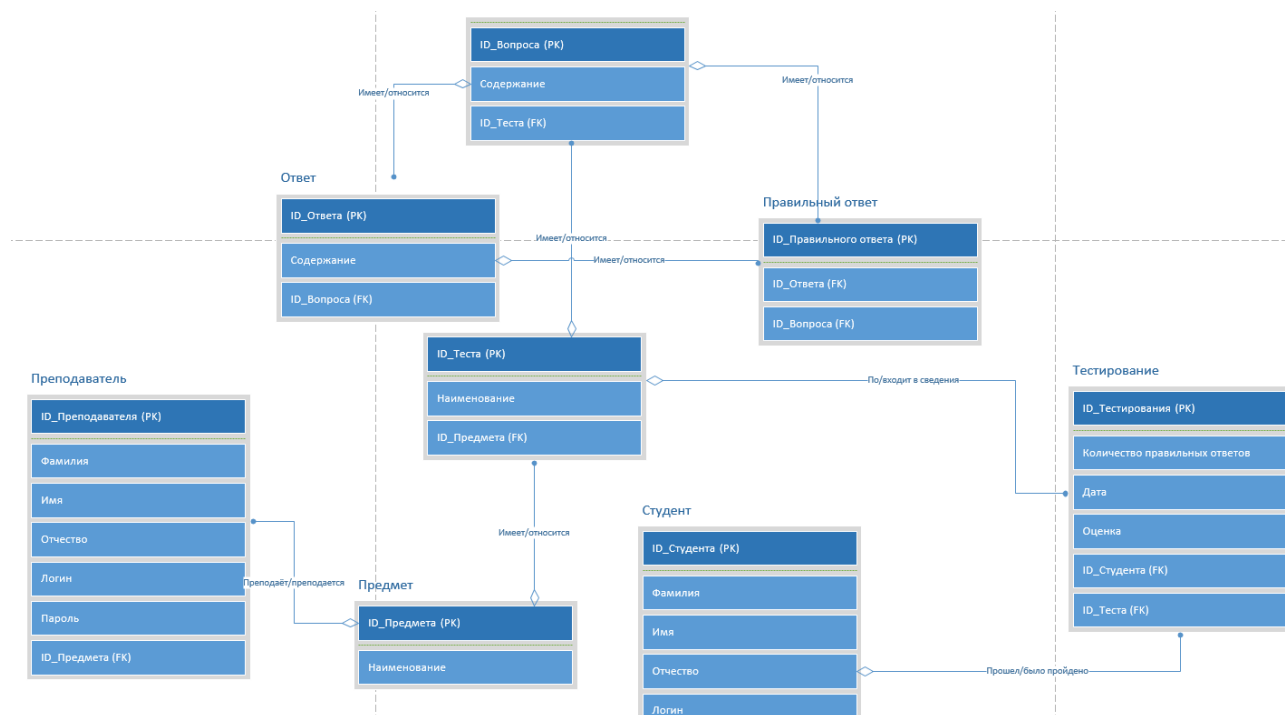


Рисунок 1 – Концептуальная модель системы

В ходе дальнейшей работы будет осуществлен переход от концептуальной модели к логической.

1.3 Анализ требований

1.3.1 Анализ функциональных требований

Требование можно определить, как «подробное описание того, что должно быть реализовано». В качестве пользователей разрабатываемого клиентского приложения выделяются 2 типа:

- преподаватель;
- студент.

Классы и характеристики пользователей представлены в таблице 13.

Таблица 13 – Классы и характеристики пользователей

Класс пользователей	Характеристика пользователей
Преподаватель	Преподаватель добавляет предметы и тесты по ним в базу данных, а также просматривает результаты тестирования. Ему разрешено производить модификацию и удаление вопросов и ответов теста.
Студент	Студент может выбрать для демонстрации своих знаний один из предметов, затем пройти тест за определенное время. Студенту разрешено видеть только результаты собственного тестирования.

В соответствии с поставленной задачей была построена модель прецедентов предметной области. Диаграмма вариантов использования представлена на рисунке 2.

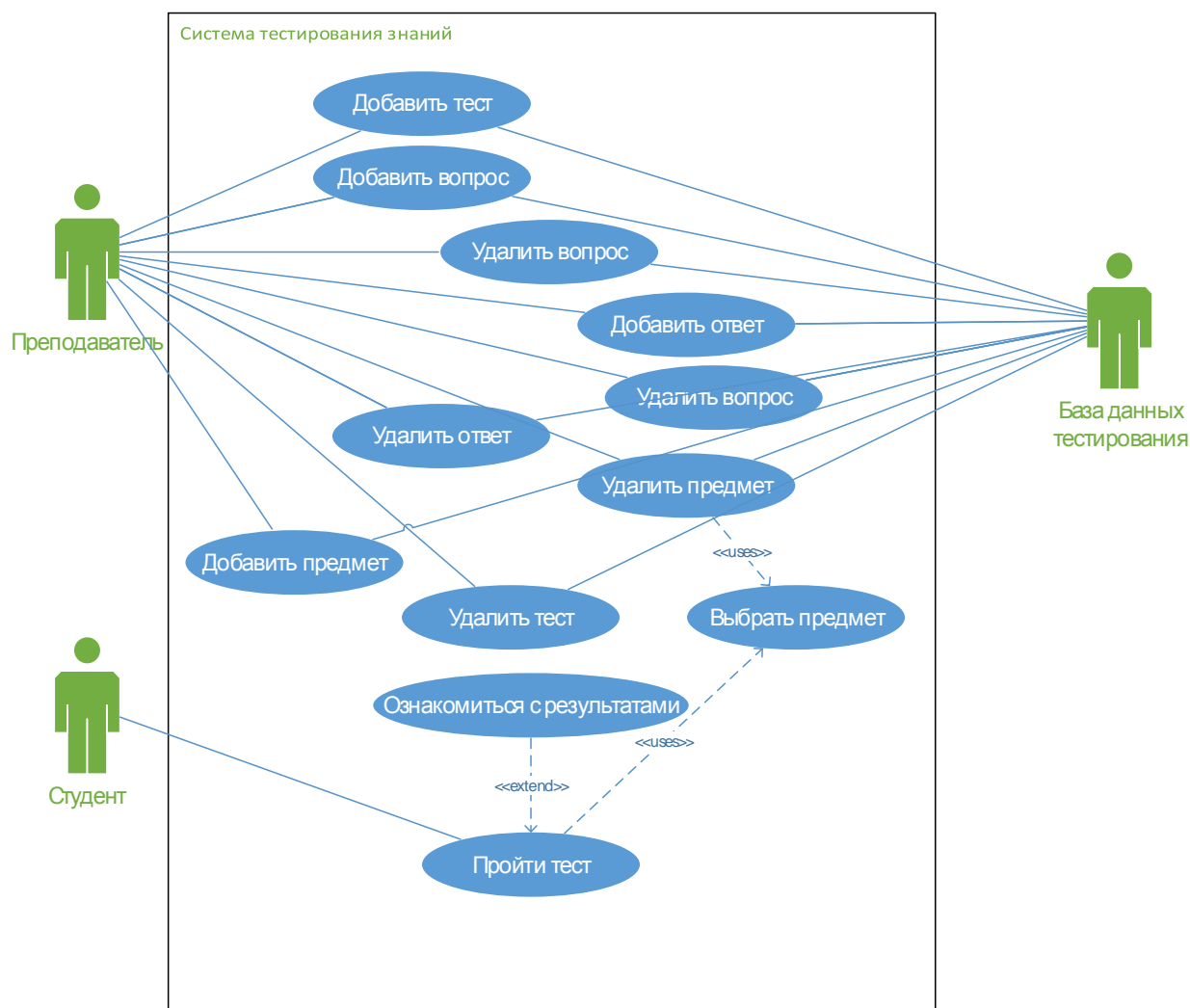


Рисунок 2 – Диаграмма вариантов использования

После создания диаграммы прецедентов и выявления актеров приступают к точному определению каждого прецедента. Этот процесс называется детализацией прецедентов [9]. Как итог этой деятельности будет получен более детализированный прецедент, включающий, как правило, имя прецедента и его спецификацию. Спецификации прецедентов представлены для прецедентов: добавить тест, добавить вопрос, пройти тест, выбрать предмет, а также для альтернативного прецедента выполнение теста прервано.

Прецедент: Добавить тест
ID: 1
Краткое описание: Добавление нового теста
Главные актёры: Преподаватель
Предусловия: Прецедент начинается, когда Преподаватель выбрал опцию "добавить тест".
Основной поток: 1. Система запрашивает у Преподавателя название теста. 2. Если тест с данным именем уже существует 2.1. Выдать сообщение об ошибке 3. Иначе 3.1. Система добавляет новый тест 4. Обновленный список тестов сохраняется в базе данных.
Постусловия: 1. Система обновляет список тестов.
Альтернативные потоки: нет

Рисунок 3 – Спецификация прецедента «Добавить тест»

Прецедент: Добавить вопрос
ID: 2
Краткое описание: Добавление нового вопроса
Главные актёры: Преподаватель
Предусловия: Прецедент начинается, когда Преподаватель выбрал тест, а также выбрал опцию "добавить вопрос".
Основной поток: 1. Система запрашивает у Преподавателя ввод данных. 2. Преподаватель вводит содержание вопроса, возможные варианты ответа и правильный ответ на вопрос. 3. Преподаватель подтверждает добавление нового вопроса. 4. Система проверяет корректность введенных данных, а именно наличие ответов, содержания вопроса и правильный ответ. 5. Если проверка данных не была завершена успешно 5.1. Выдать сообщение о некорректных данных 6. Иначе 6.1. Система добавляет новый вопрос 7. Новый вопрос сохраняется в базе данных.
Постусловия: 1. Система обновляет список вопросов на экране.
Альтернативные потоки: нет

Рисунок 4 – Спецификация прецедента «Добавить вопрос»

Прецедент: Пройти тест
ID: 3
Краткое описание: Прохождение тестирования студентом
Главные актёры: Студент
Нет.
Основной поток: 1. include (Выбрать предмет) 2. Система отображает список тестов по выбранному предмету. 3. Студент выбирает тест и выбирает опцию «выполнить». 4. Система инициирует начало тестирования и выдаёт Студенту вопросы, пока все вопросы не были заданы. 5. Студент должен ответить на текущий <u>вопрос</u> чтобы перейти к следующему. 6. По прохождении теста система выдаёт на экран результаты тестирования.
Постусловия: 1. Система фиксирует результат выполнения тестирования в базу данных.
Альтернативные потоки: Выполнение теста прервано.

Рисунок 5 – Спецификация прецедента «Пройти тест»

Прецедент: Выполнение теста прервано
ID: 3.1
Краткое описание: Студент отменяет выполнение теста.
Главные актёры: Студент.
Предусловия: Нет.
Основной поток: 1. Система запрашивает у Студента подтверждения прекращения тестирования. 2. Если Студент подтвердил выбор 2.1. Система сбрасывает результаты прохождения данного тестирования. 3. Иначе 3.1. Система возвращает пользователя к выполнению теста.
Постусловия: Нет.
Альтернативные потоки: нет

Рисунок 6 – Спецификация альтернативного прецедента «Выполнение теста прервано»

Прецедент: Выбрать предмет
ID: 4
Краткое описание: Выбор предмета для редактирования тестов либо для прохождения тестирования.
Главные актёры: Преподаватель. Студент.
Предусловия: Был произведен вход в систему.
Основной поток: 1. Система запрашивает у Преподавателя/Студента предмет для изменения либо прохождения теста. 2. Выбранный предмет сохраняется в системе для дальнейшей работы.
Постусловия: Нет.
Альтернативные потоки: нет

Рисунок 7 – Спецификация прецедента «Выбрать предмет»

Для наглядного представления поведения прецедентов удобно применять диаграмму деятельности. Диаграмма деятельности для прецедента «Добавить вопрос» (рисунок 8), диаграмма деятельности для прецедента «Пройти тест» (рисунок 9).

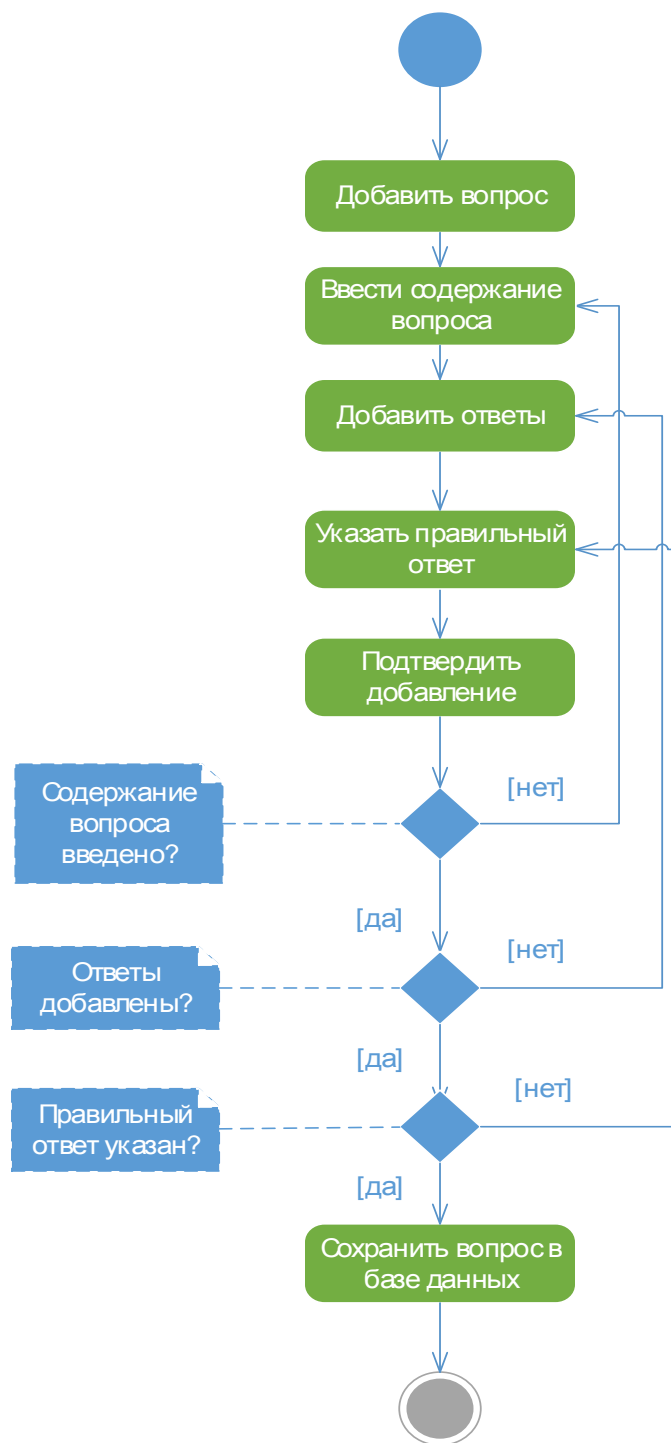


Рисунок 8 – Диаграмма деятельности прецедента «Добавить вопрос»

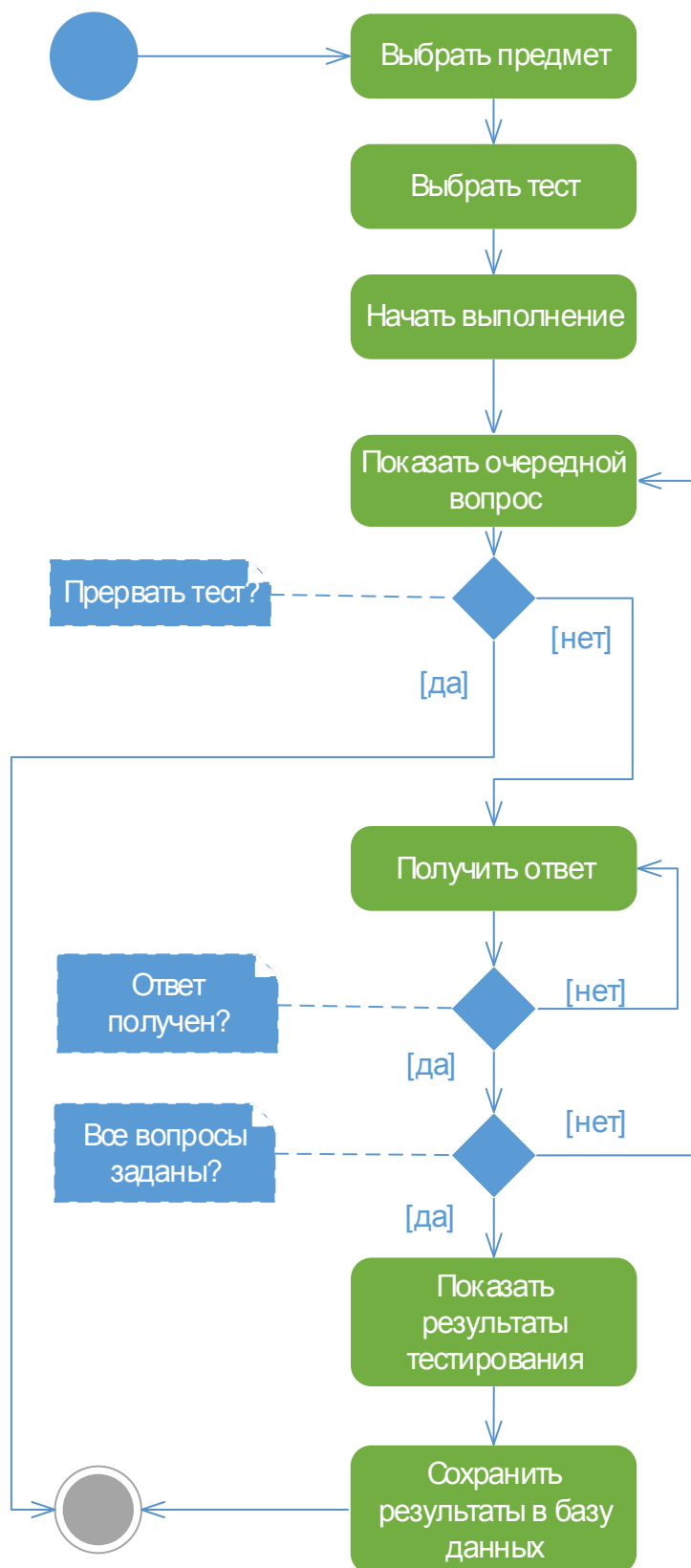


Рисунок 9 – Диаграмма деятельности прецедента «Пройти тест»

Для отображения взаимодействия между выделенными классами на этапе построения модели анализа рассмотрим диаграмму последовательностей для прецедента «Добавить вопрос» (рисунок 10).

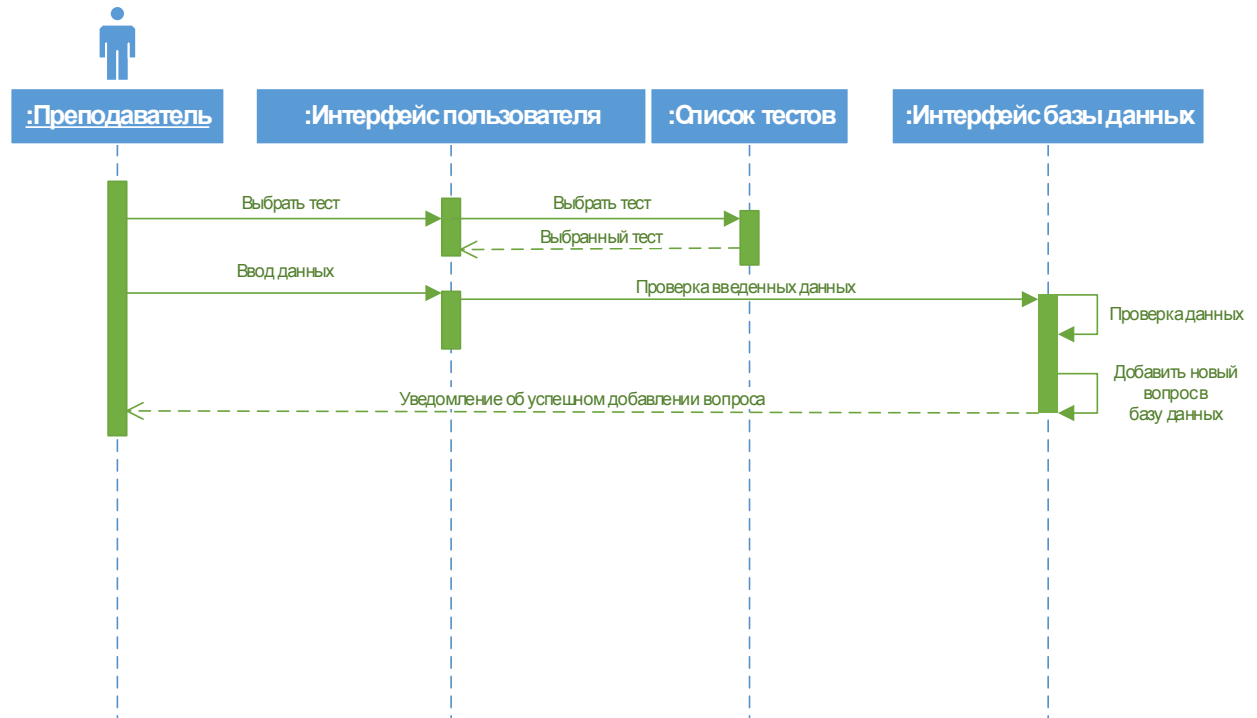


Рисунок 10 – Диаграмма последовательности прецедента «Добавить вопрос»

Диаграмма последовательности помогает установить временную взаимосвязь между классами, иными словами она лишь показывает реализацию поведения прецедента, но не является точным представлением каждого его шага. На диаграмме участвует пользовательский интерфейс, которым не занимаются всерьез вплоть до этапа проектирования, поэтому на диаграмме последовательностей он мог быть опущен. В анализе нас интересует только основное поведение классов анализа.

В данном случае рассматривается успешный вариант развития сценария, при котором преподаватель выбрал тест, ввел содержание вопроса, возможные ответы и указал правильный из них. Система проверила данные и по результатам проверки добавила новый вопрос в базу данных.

2 Проектирование

2.1.1 Базовая архитектура системы

Проектируемая система использует двухуровневую архитектуру (клиент-сервер). Архитектура системы представлена на рисунке 11.

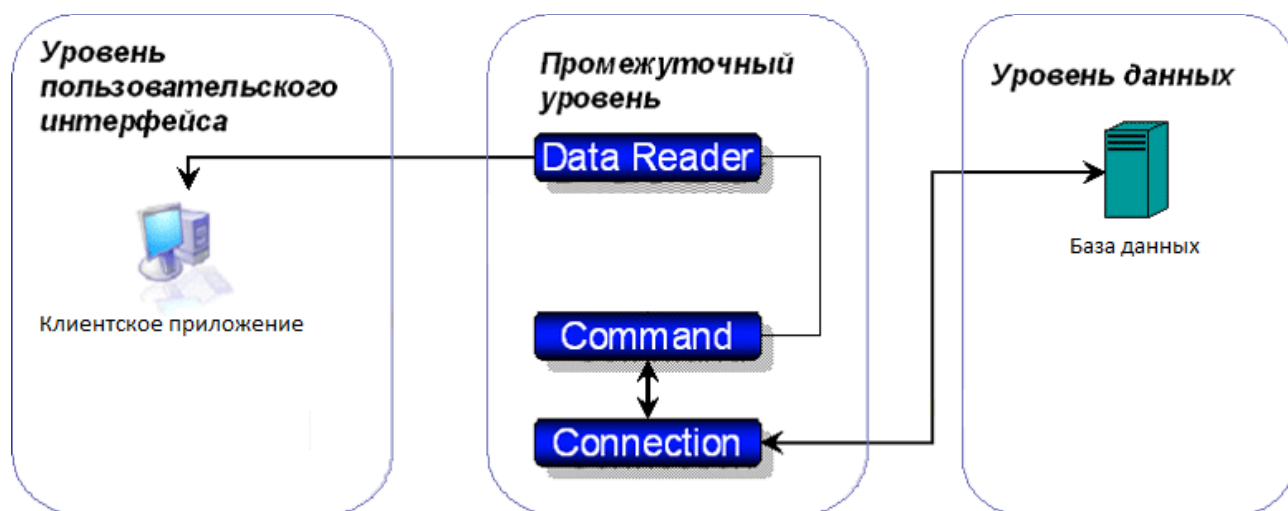


Рисунок 11 – Архитектура автоматизированной информационной системы тестирования

На данном рисунке изображены компоненты системы тестирования. Клиентское приложение разворачивается на рабочих станциях преподавателя и студента. Сервер хранит базу данных, содержащую информацию о тестах, пользователей и результатов тестирования. Взаимодействие с базой данных осуществляется с использованием технологии ADO.NET. Клиентское приложение подключается к базе данных с помощью строки подключения через класс `SqlConnection`, затем выполняет необходимую команду и получает результаты из базы данных в виде совокупности кортежей `SqlDataReader`.

Работу системы можно описать следующим образом: преподаватель входит в систему, наполняет ее тестами по соответствующим дисциплинам. В свою очередь студент при входе имеет возможность выбрать тест для прохождения. По результатам тестирования вся информация сохраняется в базе данных,

включая количество правильных ответов, оценку и дату тестирования, для дальнейшего анализа данной информации преподавателем.

2.1.2 Логическая модель базы данных

Логическая модель строится на основе концептуальной модели с учетом выбранной модели данных, но не учитывая особенности целевой СУБД.

Устранение избыточности модели производится, как правило, за счет декомпозиции отношений таким образом, чтобы в каждом отношении хранились только первичные факты. Процесс преобразования базы данных к виду, отвечающему нормальным формам, называется нормализацией.

Нормализация предназначена для приведения структуры базы данных к виду, обеспечивающему минимальную избыточность. Конечной целью нормализации является уменьшение потенциальной противоречивости хранимой в БД информации [10].

Нормальная форма – свойство отношения в реляционной модели данных, характеризующее его с точки зрения избыточности, которая потенциально может привести к логически ошибочным результатам выборки или изменения данных. Нормальная форма определяется как совокупность требований, которым должно удовлетворять отношение. Существует 6 нормальных форм, но обычно, для решения практических ограничиваются третьей нормальной формой (3НФ).

2.1.2.1 Первая нормальная форма (1НФ)

Переменная отношения находится в первой нормальной форме (1НФ) тогда и только тогда, когда в любом допустимом значении отношения каждый его кортеж содержит только одно значение для каждого из атрибутов.

В реляционной модели отношение всегда находится в первой нормальной форме по определению понятия отношение. Что же касается различных таблиц, то они могут не быть правильными представлениями отношений и, соответственно, могут не находиться в 1НФ.

2.1.2.2 Вторая нормальная форма (2NF)

Переменная отношения находится во второй нормальной форме тогда и только тогда, когда она находится в первой нормальной форме, и каждый неключевой атрибут зависит от ее потенциального ключа.

2.1.2.3 Третья нормальная форма (3NF)

Переменная отношения находится в третьей нормальной форме тогда и только тогда, когда она находится во второй нормальной форме, и отсутствуют транзитивные функциональные зависимости не ключевых атрибутов от ключевых.

В нашем случае каждый не ключевой атрибут каждого отношения зависит только от первичного ключа.

Построим логическую модель предметной области в нотации IDEF1X при помощи case-средства ERwin Data Modeler. Логическая модель представлена на рисунке 12.

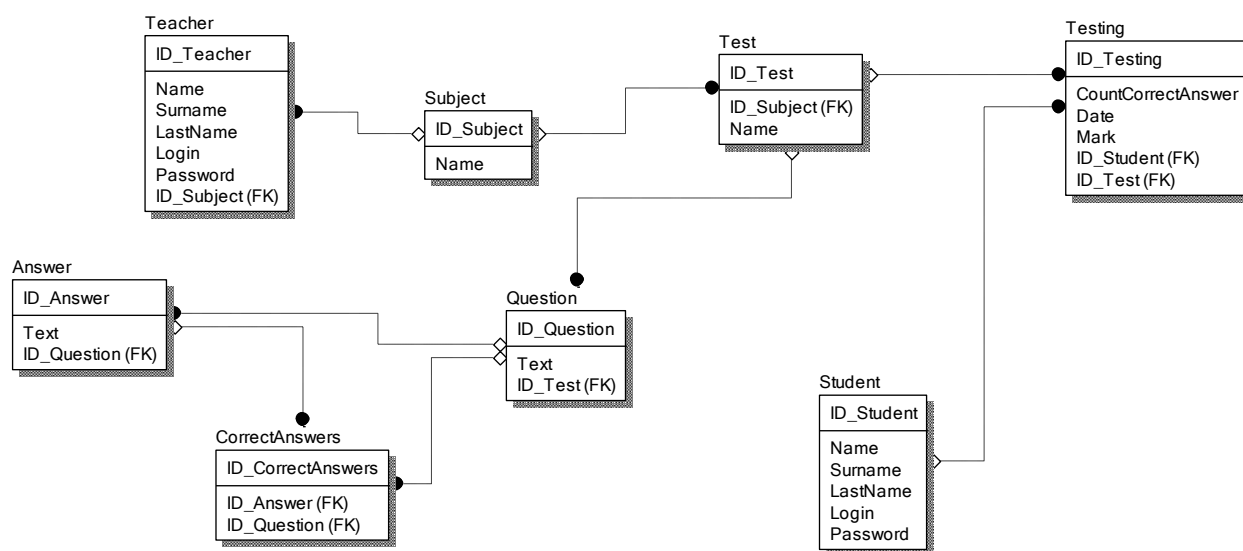


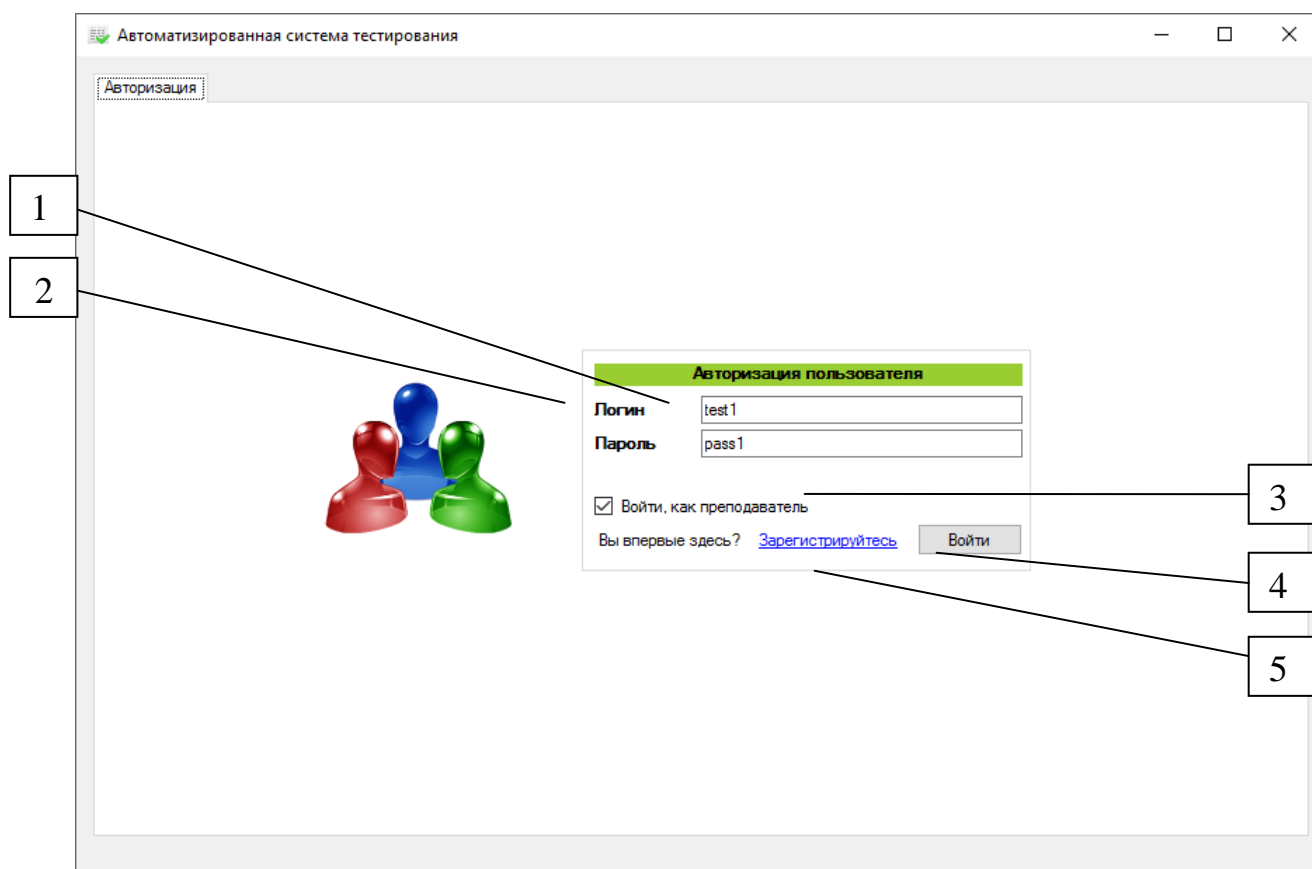
Рисунок 12 – Логическая модель системы

Разработанная логическая модель базы данных не содержит транзитивных зависимостей в отношениях. Таким образом, модель соответствует третьей нормальной форме.

2.1.3 Пользовательский интерфейс клиентского приложения

Исходя из анализа требований и модели анализа, был спроектирован простой и интуитивно понятный графический пользовательский интерфейс.

Главное окно содержит внутри себя панель с вкладками, отображение, которых, зависит от авторизованного пользователя. Рассмотрим вкладку главного окна с авторизацией в систему (рисунок 13).



- 1 – поля ввода логина и пароля пользователя, компонент `textBox`;
- 2 – текстовые метки для описания полей ввода, компонент `label`;
- 3 – флажок для входа от имени преподавателя, компонент `checkBox`;
- 4 – кнопка авторизации, компонент `button`;
- 5 – ссылка на форму регистрации, компонент `linkLabel`.

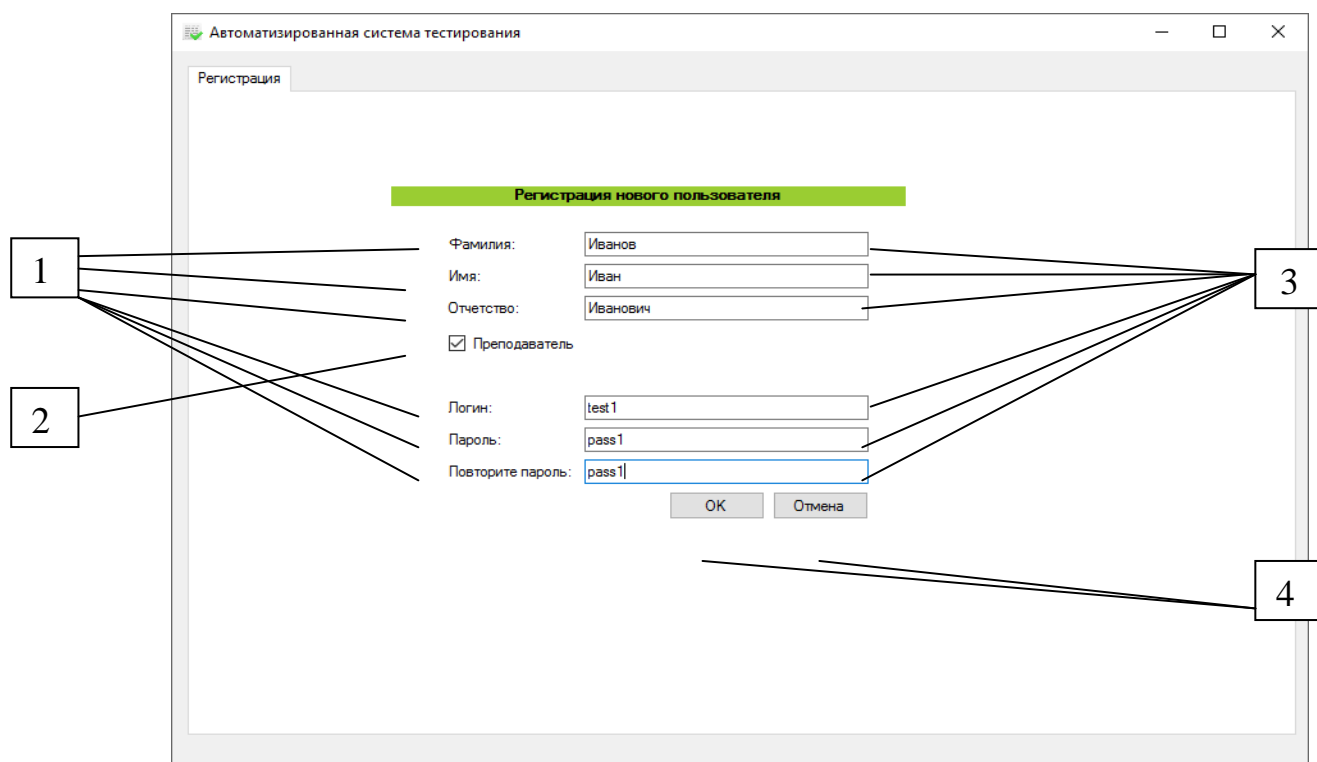
Рисунок 13 – Визуальные компоненты окна авторизации приложения

Данное окно предназначено для авторизации пользователя в системе. В дальнейшем пользователь сможет выйти из системы, сменить имя пользователя и пароль. Компоненты и их действия описаны в таблице 14.

Таблица 14 – Конструкции главного окна приложения

Наименование элемента формы	Тип элемента формы	Действие пользователя	Отклик системы
«Поле ввода логина и пароля»	Текстовое поле ввода	ввод с клавиатуры	-
«Флажок входа от имени преподавателя»	Флажок	одинарный щелчок левой кнопкой мыши	установка либо сброс в признака входа от имени преподавателя
«Зарегистрируйтесь»	Гиперссылка	одинарный щелчок левой кнопкой мыши	система отображает окно регистрации нового пользователя
«Войти»	Кнопка	одинарный щелчок левой кнопкой мыши	система проверяет введенное имя пользователя и пароль

Окно регистрации в системе представлено на рисунке 14.



- 1 – текстовые метки для описания полей ввода, компонент label;
- 2 – флажок для регистрации от имени преподавателя, компонент checkBox;
- 3 – поля ввода для информации о пользователе, компонент textBox;
- 4 – кнопка регистрации и отмены, компонент button;

Рисунок 14 – Визуальные компоненты окна регистрации

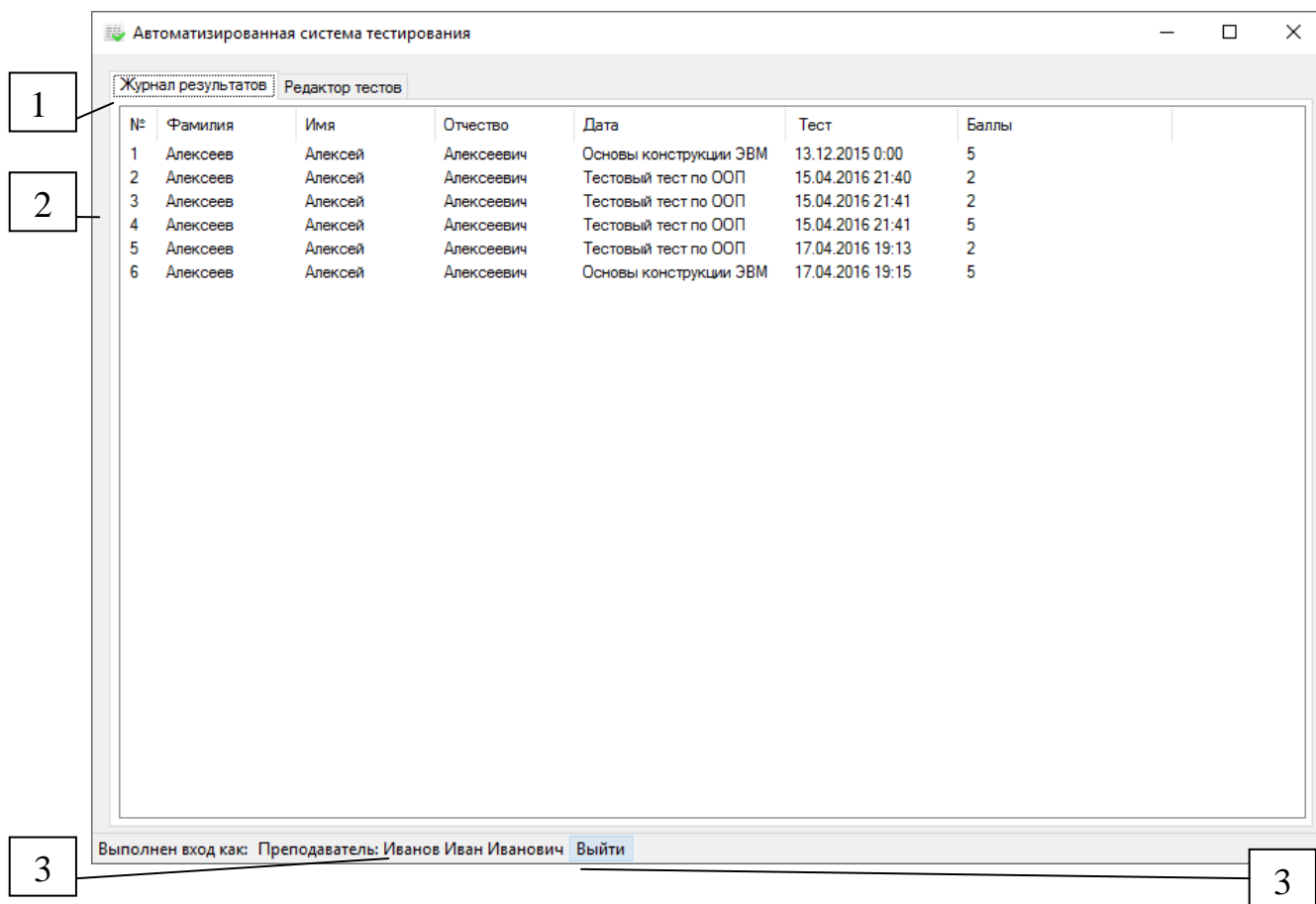
Данное окно предназначено для регистрации пользователя в системе. Возможна регистрация как преподавателя, с указанием соответствующего признака, так и студента. Впоследствии зарегистрированные пользователи выполняют вход в систему с указанием логина и пароля введенных на этапе регистрации в системе.

В таблице 15 приведены компоненты формы и их действия.

Таблица 15 – Конструкции окна входа в систему

Наименование элемента формы	Тип элемента формы	Действие пользователя	Отклик системы
«Поле ввода информации о пользователе»	Текстовое поле ввода	ввод с клавиатуры	-
«Флажок регистрации от имени преподавателя»	Флажок	одинарный щелчок левой кнопкой мыши	установка либо сброс признака регистрации от имени преподавателя
«ОК»	Кнопка одинарный щелчок левой кнопкой мыши	система проверяет введенную информацию пользователем и в случае успеха регистрирует пользователя в системе	
«Отмена»	Кнопка	одинарный щелчок левой кнопкой мыши	система отменяет регистрацию пользователя и возвращает его на экран авторизации

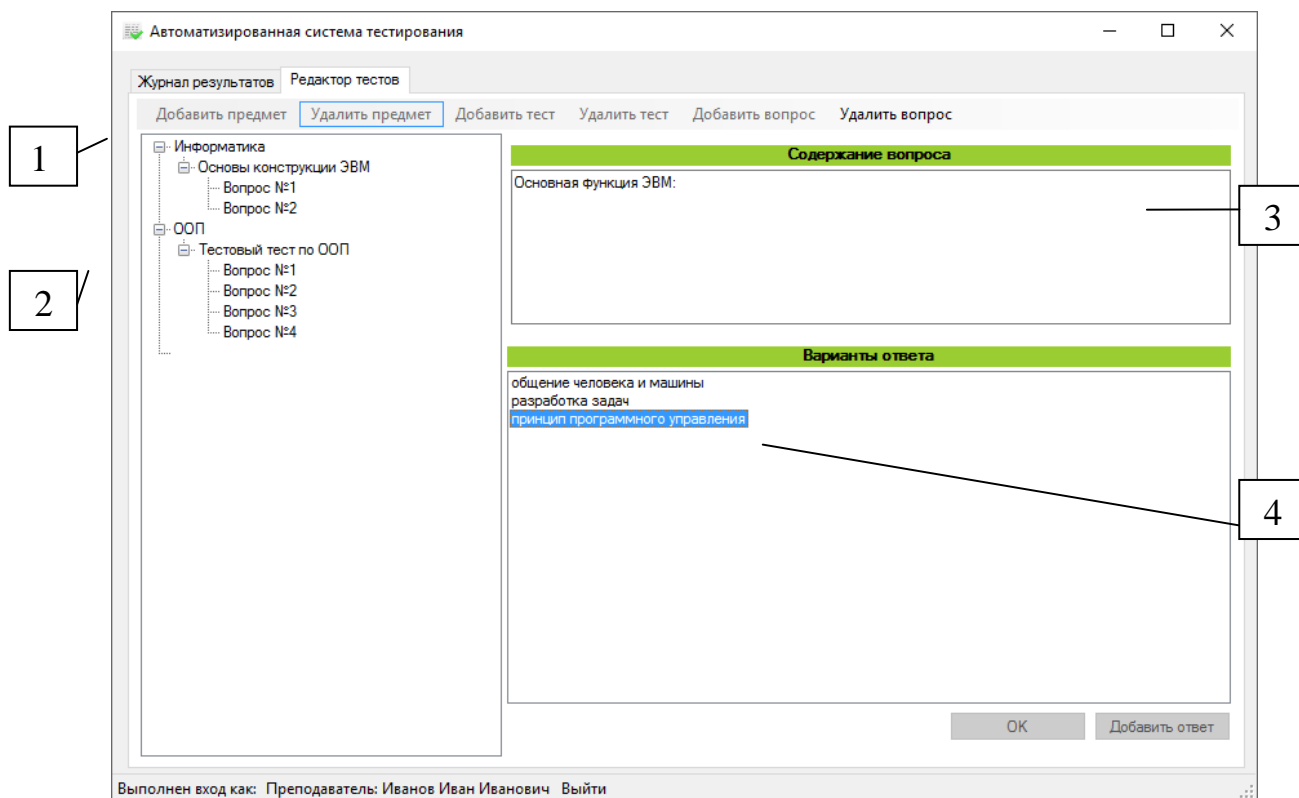
Главное окно для авторизованного преподавателя представлено на рисунке 15.



- 1 – вкладки на окне авторизованного преподавателя, компонент `tabControl`;
- 2 – таблица сводных результатов тестирования, компонент `listView`;
- 3 – строка состояний, компонент `statusStrip`;
- 4 – кнопки для применения изменений и отмены, компонент `button`.

Рисунок 15 – Главное окно авторизованного преподавателя

Данное окно является главным для преподавателя. На первую вкладку вынесены результаты тестирования знаний студентов по дисциплинам. На второй вкладке (рисунок 16) представлен редактор тестов.



- 1 – панель инструментов, компонент `toolStrip`;
- 2 – дерево тестов, компонент `treeView`;
- 3 – поле ввода содержания вопроса, компонент `textBox`;
- 4 – список возможных ответов на вопрос, компонент `listView`;
- 4 – кнопки для добавления вопроса и ответа, компонент `button`.

Рисунок 16 – Окно редактирования тестов

Далее в таблице 16 приведены компоненты формы с описанием.

Таблица 16 – Конструкции окна правки информации о сборке

Наименование элемента формы	Тип элемента формы	Действие пользователя	Отклик системы
«Добавить предмет»	Кнопка панели инструментов	одинарный щелчок левой кнопкой мыши	Система добавляет новый предмет в дерево и запрашивает ввод его наименования
«Удалить предмет»	Кнопка панели инструментов	одинарный щелчок левой кнопкой мыши	Система удаляет выбранный предмет и все его тесты

Продолжение таблицы 16

Наименование элемента формы	Тип элемента формы	Действие пользователя	Отклик системы
«Удалить тест»	Кнопка панели инструментов	одинарный щелчок левой кнопкой мыши	Система удаляет выбранный тест и все его вопросы
«Добавить вопрос»	Кнопка панели инструментов	одинарный щелчок левой кнопкой мыши	Система добавляет вопрос, как узел выбранного теста
«Удалить вопрос»	Кнопка панели инструментов	одинарный щелчок левой кнопкой мыши	Система удаляет выбранный вопрос и все его ответы
«ОК»	Кнопка	одинарный щелчок левой кнопкой мыши	Подтверждение добавления очередного вопроса.
«Отмена»	Кнопка	одинарный щелчок левой кнопкой мыши	Отмена редактирования тестов.

Спроектированный графический пользовательский интерфейс является удобным и интуитивно понятным.

2.1.4 Проектирование программных средств

В соответствии с моделью предметной области были спроектированы следующие классы, представленные на диаграмме классов (рисунок 17).



Рисунок 17 – Диаграмма классов

Здесь представлены: 1 интерфейсный класс и 10 класса сущностей.
Краткое описание классов приведено ниже.

Интерфейсные классы:

– TestSession – главное окно приложение, содержащее внутри себя функции: авторизации, регистрации новых пользователей, редактирования.

Классы – сущности:

- Question – базовый класс для вопроса теста;
- NewQuestion – класс для нового вопроса, на этапе редактирования;
- Testing – класс для тестирования, получения результатов и сохранения их в базу данных;
- Logon – класс, для работы с пользователями, позволяет регистрировать новых пользователей и проводить авторизацию;
- SqlGateway – класс-шлюз к базе данных;
- TestSession – класс для хранения информации о тестировании;
- User – базовый класс пользователя;
- Teacher – класс-преподавателя.

3 Реализация

3.1.2 Физическая модель и реализация базы данных системы

Физическая модель строится на основе логической модели данных и учитывает особенности целевой СУБД. В качестве целевой СУБД выбран SQL Server 2012. В ERWin можно с легкостью перейти от логической модели к физической, при этом:

- типы сущностей становятся таблицами в физической базе данных;
- атрибуты становятся колонками в физической базе данных;
- уникальные идентификаторы становятся колонками, не допускающими значение NULL. В физической базе данных они называются первичными ключами (primary keys);
- отношения моделируются в виде внешних ключей (foreign keys).

Одним из важнейших качеств ERWin Data Modeler является возможность проводить как прямое проектирование, так и обратное.

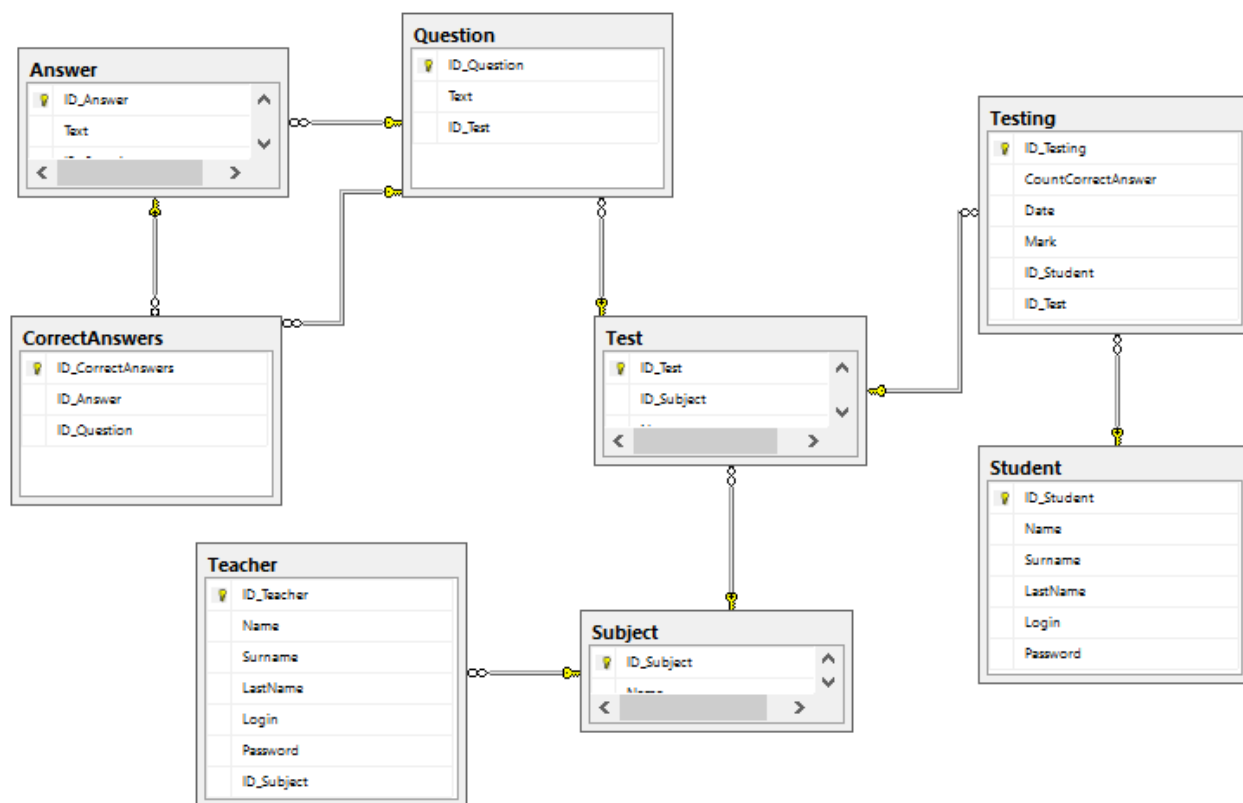


Рисунок 18 – Физическая модель системы

Скрипт DDL сгенерированный ERWin (приложение Г) был выполнен в SQL Server 2012. В результате была получена схема базы данных, представленная на рисунке 18.

3.1.1 Серверная часть приложения системы

В ходе разработки программных средств поддержки непрерывной интеграции было принято решение перенести логику приложения для работы с базой данных на сервер, поэтому клиентское приложение не отправляет SQL-запросы на сервер для получения результатов, а вызывает хранимые процедуры. Данное решение обусловлено следующими преимуществами хранимых процедур по сравнению с запросами.

При обработке даже небольших объемов данных во внешнем приложении тратится дополнительное время на передачу по сети и преобразование данных в нужный нам формат. Также происходит значительная нагрузка на сеть в виду того, что запрос необходимо сформировать на клиентской стороне, затем отправить на сервер базы данных, и получить результат выполнения запроса назад в клиентское приложение [11].

С ростом и эволюцией программной системы схема данных может и должна меняться. Хорошо спроектированный программный интерфейс на хранимых процедурах позволит менять схему данных, не изменяя код внешних приложений.

При использовании SQL со стороны клиентской программы клиентская программа передает СУБД SQL команды в виде строк, предварительно формируемых в коде. При использовании хранимых процедур SQL код на стороне приложения обычно статический и выглядит как простой вызов хранимой процедуры с последующей передачей параметров.

Реализуя логику работы с данными в хранимом слое мы получаем привычную иерархическую модель повторного использования SQL кода.

Упрощается отладка, как клиентского приложения, так и серверной части базы данных, в связи с тем, что мы избегаем смещения кода клиентского приложения с кодом работы с базой данных. Семантическая и синтаксическая проверка SQL производится на этапе компиляции.

3.1.1.1 Хранимые процедуры

Для базы данных автоматизированной системы тестирования знаний были реализованы следующие хранимые процедуры:

Листинг 1 - Получить информацию о преподавателе

```
Create PROC GetTeacherInfo
@Login VARCHAR(20),
@Password VARCHAR(20)
AS
SELECT Teacher.ID_Teacher, Teacher.Name, Surname, LastName, Subject.Name FROM Teacher inner join Subject on Subject.ID_Subject = Teacher.ID_Subject WHERE Login = @Login AND Password = @Password
```

Листинг 2 - Получить информацию о студенте

```
Create PROC GetStudentInfo
@Login VARCHAR(20),
@Password VARCHAR(20)
AS
SELECT ID_Student, Name, Surname, LastName FROM Student WHERE Login = @Login AND Password = @Password
```

Листинг 3 - Добавить пользователя

```
CREATE PROC AddUser
@Name VARCHAR(20),
@Surname VARCHAR(40),
@LastName VARCHAR(20),
@Login VARCHAR(20),
@Password VARCHAR(20),
@IsTeacher BIT,
@Subject VARCHAR(50)
AS
if(@IsTeacher = 1)
    INSERT INTO Teacher VALUES (@Name,@Surname, @LastName, @Login, @Password, (SELECT ID_Subject From Subject WHERE Name = @Subject));
else
    INSERT INTO Student VALUES (@Name,@Surname, @LastName, @Login, @Password)
```

Листинг 4 - Получить список предметов

```
Create proc GetSubjects
AS
Select Name From Subject
```

Листинг 5 - Получить список вопросов

```
Create proc GetQuestions
@IdTest INT
AS
SELECT ID_Question, Text From Question WHERE ID_Test = @IdTest
```

Листинг 6 - Получить список ответов

```
Create proc GetAnswers
@IdQuestion INT
AS
SELECT ID_Answer, Text From Answer WHERE ID_Question = @IdQuestion
```

Листинг 7 - Получить список тестов

```
Create proc GetTests
@Subject Varchar(30)
AS
SELECT Name From Test WHERE ID_Subject = (SELECT ID_Subject From Subject
where Name = @Subject)
```

Листинг 8 - Получить номер теста

```
Create proc GetTestId
@TestName Varchar(30)
AS
SELECT Id_Test From Test where Name = @TestName
```

Листинг 9 - Получить правильный ответ

```
Create proc GetCorrectAnswer
@IdQuestion int
AS
SELECT ID_Answer From CorrectAnswers where ID_Question = @IdQuestion
```

Листинг 10 - Добавить предмет

```
Create proc AddSubject
@Name VARCHAR(50)
AS
INSERT INTO Subject VALUES (@Name)
```

Листинг 11 - Получить номер предмета

```
Create proc GetSubjectId
@Name VARCHAR(50)
AS
Select Id_Subject from Subject where Name = @Name
```

Листинг 12 - Сохранить информацию о тестирование

```
CREATE PROC ReportTestSession
@CorrectCount INT,
@Date DATETIME,
@Mark INT,
@IdUser INT,
@IdTest INT
```

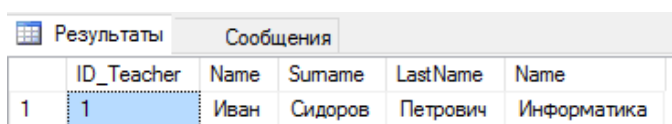
Продолжение листинга 12

```
AS  
INSERT INTO Testing VALUES (@CorrectCount,@Date, @Mark, @IdUser, @IdTest)
```

END

3.1.2 Результаты работы серверной части

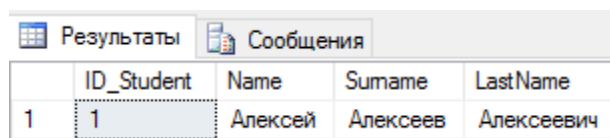
Результат выполнения хранимой процедуры GetTeacherInfo Login = 'test', Password = 'test' представлен на рисунке 19.



	ID_Teacher	Name	Surname	LastName	Name
1	1	Иван	Сидоров	Петрович	Информатика

Рисунок 19 – Результат хранимой процедуры GetTeacherInfo

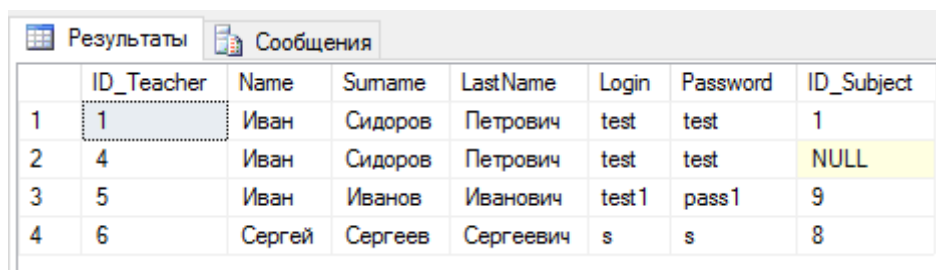
Результат выполнения хранимой процедуры GetStudentInfo с параметрами Login = '123', Password = '123' представлен на рисунке 20.



	ID_Student	Name	Surname	LastName
1	1	Алексей	Алексеев	Алексеевич

Рисунок 20 – Результат хранимой процедуры GetStudentInfo

Результат выполнения хранимой процедуры AddUser с параметрами Name = 'Сергей', Surname = 'Сергеев', LastName = 'Сергеевич', Login = 's', Password = 's', IsTeacher = 1, Subject = 'ООП' представлен на рисунке 21.



	ID_Teacher	Name	Surname	LastName	Login	Password	ID_Subject
1	1	Иван	Сидоров	Петрович	test	test	1
2	4	Иван	Сидоров	Петрович	test	test	NULL
3	5	Иван	Иванов	Иванович	test1	pass1	9
4	6	Сергей	Сергеев	Сергеевич	s	s	8

Рисунок 21 – Результат хранимой процедуры AddUser

Результат выполнения хранимой процедуры GetSubjects представлен на рисунке 22.

Результаты		Сообщения	
	Name		
1	Информатика		
2	ООП		

Рисунок 22 – Результат хранимой процедуры GetSubjects

Результат выполнения хранимой процедуры GetQuestions с параметром ID_Test = 1 представлен на рисунке 23.

Результаты			Сообщения		
	ID_Question	Text			
1	1	Основная функция ЭВМ:			
2	2	Микропроцессор предназначен для:			

Рисунок 23 – Результат хранимой процедуры GetQuestions

Результат выполнения хранимой процедуры ReportTestSession с параметрами CorrectCount = 5, Date = '04-05-2015', Mark = 5, IdUser = 1, IdTest = 1 представлен на рисунке 24.

Результаты							Сообщения						
	ID_Testing	CountCorrectAnswer	Date	Mark	ID_Student	ID_Test							
1	34	5	2015-12-13 00:00:00.000	5	1	1							
2	35	1	2016-04-15 21:40:50.253	2	1	8							
3	36	1	2016-04-15 21:41:23.407	2	1	8							
4	37	2	2016-04-15 21:41:28.360	5	1	8							
5	38	3	2016-04-17 19:13:59.830	2	1	8							
6	39	2	2016-04-17 19:15:02.590	5	1	1							
7	40	1	2016-05-03 15:46:53.597	2	1	8							
8	41	2	2016-05-03 15:47:18.153	5	1	1							
9	42	5	2016-05-04 00:00:00.000	5	1	1							

Рисунок 24 – Результат хранимой процедуры ReportTestSession

Результаты выполнения хранимых процедур соответствуют ожидаемым результатам, вследствие чего можно сделать вывод, что хранимые процедуры работают корректно.

3.1.3 Модель реализации клиентского приложения

Модель реализации описывает, как реализуются в виде компонентов элементы модели проектирования [12].

В унифицированном процессе разработки для отображения решений реализации чаще всего используются диаграммы компонентов. Диаграмма компонентов для решения (рисунок 25).

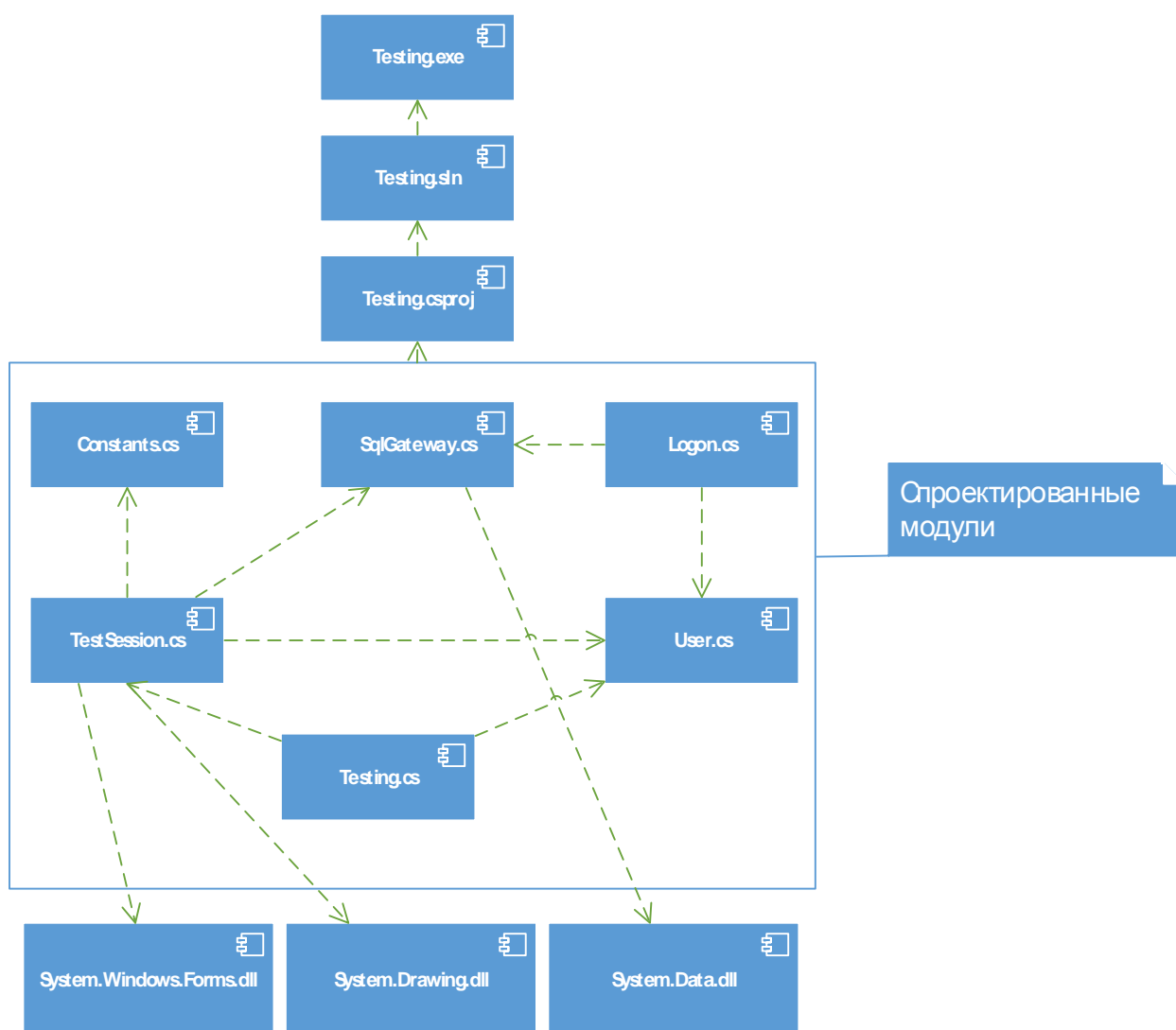


Рисунок 25 – Диаграмма компонентов для решения

Ниже (таблица 22 – 26) приведены сокращенные спецификации основных классов.

Таблица 22 – Спецификация класса SqlGateway

<i>Класс SqlGateway</i>	
Наименование	Назначение
<i>SqlGateway</i>	Класс посредник между клиентским приложением и базой данных
Члены	
SqlConnection connection	Подключение к базе данных
Методы	
User GetUserInfo()	Получить информацию о пользователе
Int AddNewUser()	Метод проверки корректности исходных данных
Int GetSubjectId()	Выполнить команду добавления или редактирования
Void AddSubject()	Добавить предмет
Void AddTest()	Добавить тест
List<string> GetSubjects()	Получить список предметов
List<string> GetTests()	Получить список тестов по предмету
Int GetTestId()	Получить номер теста
Dictionary<int, string> GetAnswers() Dictionary<int, string> GetQuestions()	Получить словарь ответов
Int GetCorrectAnswer()	Получить номер правильного ответа
Void ReportTestSession()	Сохранить результаты тестирования в БД
Void RemoveSubject() List<TestSession> GetTestSessions()	Удалить предмет
Void RemoveQuestion()	Удалить вопрос
Void AddQuestion()	Добавить вопрос
Void AddAnswer()	Добавить ответ
Void GetQuestionId()	Получить номер вопроса
Void AddCorrectAnswer() Void GetAnswerId()	Добавить правильный ответ

Таблица 23 – Спецификация класса Logon

<i>Класс Logon</i>	
Наименование	Назначение
<i>Logon</i>	Класс для работы с пользователями системы
Члены	
SqlGateway gateway	Ссылка на шлюз к базе данных
Методы	
User CheckUser()	Получить информацию о пользователе

Продолжение таблицы 23

Методы	
ErrorState AddNewUser()	Метод проверки корректности исходных данных

Таблица 24 – Спецификация класса User

Класс User	
Наименование	Назначение
User	Класс для хранения информации о пользователе
Члены	
int Id	Идентификатор пользователя
string Login	Имя пользователя
string Password	Пароль
string Name	Имя
string Surname	Фамилия
string LastName	Отчество

Таблица 25 – Спецификация класса Testing

Класс Testing	
Наименование	Назначение
Testing	Класс для хранения информации о пользователе
Члены	
SqlGateway sqlGateway	Ссылка на класс взаимодействия с БД
int idTest	Номер теста
int idUser	Номер пользователя
List<InitializedQuestion> Questions	Список вопросов
Методы	
Void GetCorrectAnswer()Сохранить результаты тестирования	Получить номер правильного ответа
Void ReportTestSession()	

Продолжение таблицы 25

Методы	
ErrorState AddNewUser()	Метод проверки корректности исходных данных
Void GetAnswers()	Получить ответы на вопрос
Void GetQuestions()	Получить вопросы теста
Tuple<int,int> GetResults()	Получить результаты тестирования
Int GetMark()	Рассчитать оценку за тест

Таблица 26 – Спецификация класса TestSession

Класс TestSession	
Наименование	Назначение
<i>TestSession</i>	Класс для хранения информации о сессии тестирования
Члены	
string Name	Имя
string Surname	Фамилия
DateTime Date Отчество	Дата тестирования
string LastName	
string TestName	Имя теста
int Mark	Оценка за тест

Одним из способов моделирования статического вида системы с точки зрения развертывания является диаграмма развертывания [13]. Диаграмма развертывания представлена на рисунке 26.

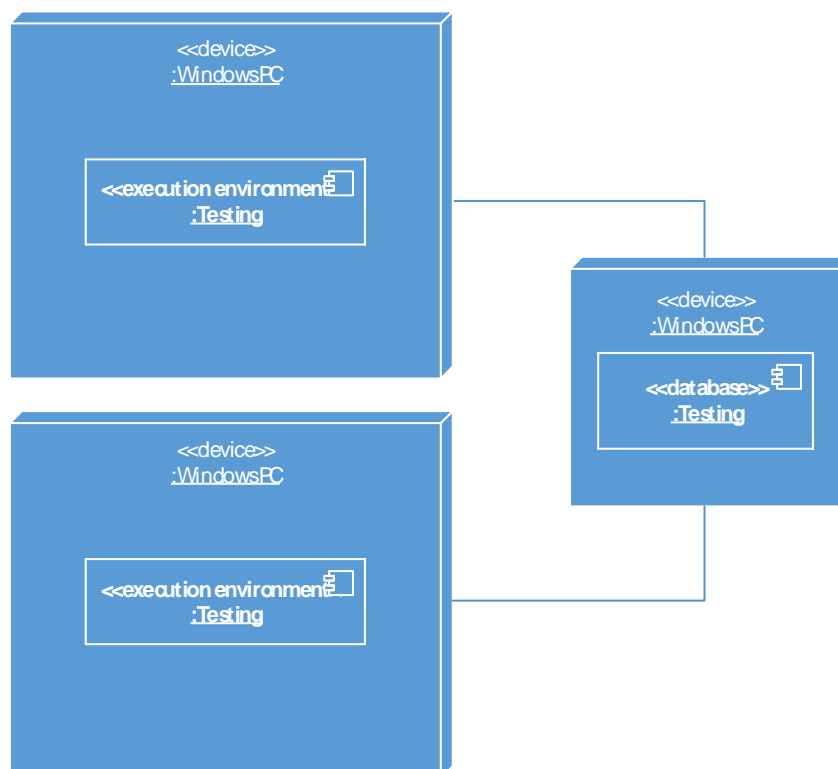


Рисунок 26 – Диаграмма развертывания

Из рисунка 26 видно, что клиентское приложение разворачивается на рабочих станциях преподавателя и студентов, а база данных разворачивается на отдельном сервере.

3.2 Контроль качества программного обеспечения

Тестирование – этап разработки программного обеспечения, имеющий целью выявить ситуации, в которых поведение программы не соответствует спецификации [14]. В данной выпускной квалификационной работы были выбраны две методологии тестирования: функциональное и модульное.

3.2.1 Функциональное тестирование

Функциональное тестирование предполагает составление плана тестирования, который строится на основе вариантов использования и включает в себя описание тестов и их результаты.

Функциональное тестирование позволяет оценить работоспособность каждой из функций разработанного программного обеспечения, не заглядывая в особенности реализации, то есть фактически производится тестирование методом черного ящика.

План тестирования приведен в таблице 27.

Таблица 27 – План тестирования

№	Вариант использования	Тест	Результаты
1		Авторизация преподавателя имя пользователя: test пароль: test	Тест выполнен (рисунок Б.1)
2	Вход в программу	Авторизация Несуществующего пользователя Имя пользователя: 1 пароль: 1	Тест не выполнен (рисунок Б.2)
3		Авторизация студента Имя пользователя: 123 пароль: 123	Тест выполнен (рисунок Б.3)
4	Пройти тест	Выполнить тест Предмет: Информатика Тест: Основы конструкции ЭВМ	Тест выполнен (рисунок Б.5, рисунок Б.6)
5	Ознакомиться с результатами	Завершить выполнение теста студентом.	Тест выполнен (рисунок Б.7)
6		Выполнить авторизацию преподавателя и ознакомиться с общими результатами тестирования знаний.	Тест выполнен (рисунок Б.8)
7	Добавить предмет	Добавить предмет: Название предмета: СУБД	Тест выполнен (рисунок Б.9)
8	Удалить предмет	Удалить предмет: Название предмета: СУБД	Тест выполнен (рисунок Б.10)
9	Добавить тест	Добавить тест: Наименование теста: Основы СУБД	Тест выполнен (рисунок Б.11)

Продолжение таблицы 27

№	Вариант использования	Тест	Результаты
10	Добавить вопрос	Добавить вопрос: Содержание вопроса: Таблица СУБД содержит Варианты ответа: Отсутствуют Правильный ответ: Не указан	Тест не выполнен (рисунок Б.12)
11		Добавить вопрос: Содержание вопроса: Таблица СУБД содержит Варианты ответа: Информацию о совокупности однотипных объектов Правильный ответ: Не указан	Тест не выполнен (рисунок Б.14)
12		Добавить вопрос: Содержание вопроса: Таблица СУБД содержит Варианты ответа: Информацию о совокупности однотипных объектов; Правильный ответ: Информацию о совокупности однотипных объектов;	Тест выполнен (рисунок Б.15, рисунок Б.16)
13	Удалить вопрос	Удалить вопрос: Содержание вопроса: Таблица СУБД содержит	Тест выполнен (рисунок Б.17)
14	Удалить тест	Удалить вопрос: Название теста: Основы СУБД	Тест выполнен (рисунок Б.18)
15	Регистрация студента	Регистрация нового студента: Фамилия: Валерьев Имя: Егор Отчество: Александрович Логин: testLogin Пароль: testPass Подтверждение пароля: testPass	Тест выполнен (рисунок Б.19)

На основе анализа результатов тестирования можно сделать вывод, что разработанное программное обеспечение поддержки непрерывной интеграции работает корректно.

3.2.2 Модульное тестирование

Модульное или unit-тестирование позволяет провести более глубокое исследование и анализ работоспособности отдельных модулей программного обеспечения. В качестве каркаса автономного тестирования был выбран MS Test – встроенный каркас в IDE Visual Studio.

Написание модульных тестов же в отличие от функционального тестирования более похоже на тестирование методом белого ящика, когда детали реализации модулей нам известны [15].

Рассмотрим модульное тестирование на примере класса Logon, отвечающих за получение информации о пользователях и регистрации новых пользователей соответственно. Для выполнения автономного тестирования был создан и добавлен в решение тестовый проект, содержащий класс LogonTest для тестируемого класса. Результаты автономного тестирования представлены на рисунке 27. Исходный код тестовых классов приведен в приложении В.

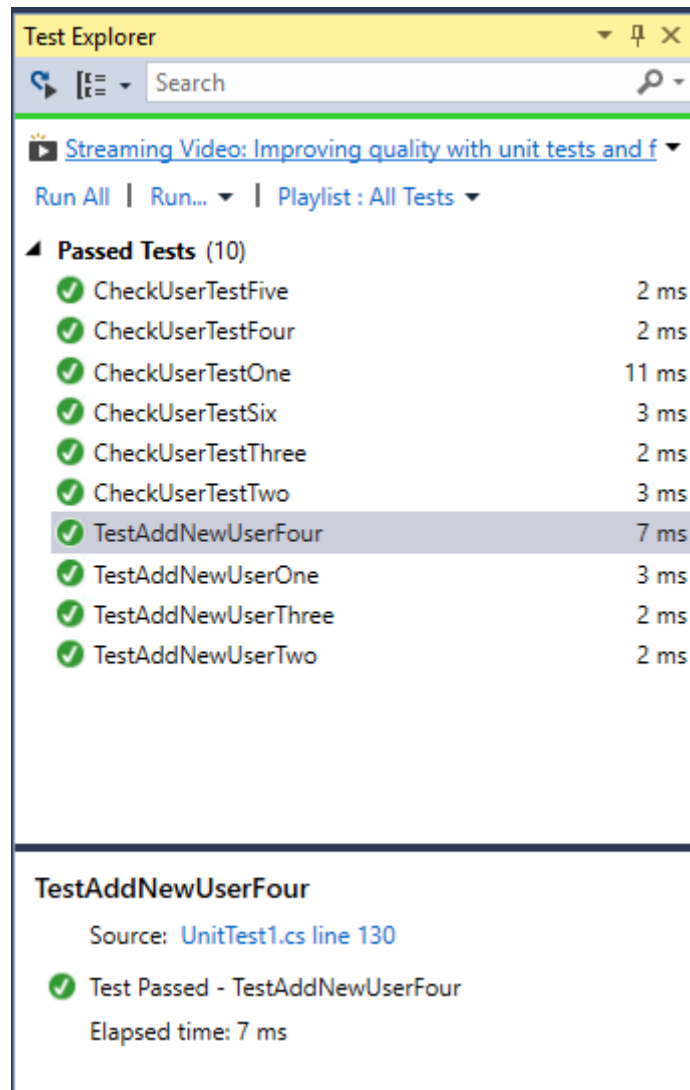


Рисунок 27 – Результаты модульных тестов

На данном рисунке в верхней части – перечень тестовых методов, статус выполнения и время их выполнения. 10 из 10 тестов выполнены успешно. Общее тестирование длилось менее секунды.

Встроенные средства Visual Studio позволяют оценить процент покрытия блоков исходного кода тестами и сгенерировать соответствующий отчет. Анализ покрытия исходного кода представлен на рисунке 28.

Code Coverage Results				
Alxpash_ALXPASH-PC 2015-04-17 11_34_07.				
Hierarchy	Not Covered (Blocks)	Not Covered (% Blocks)	Covered (Blocks)	Covered (% Blocks)
▾ distributer.exe	3582	95,44%	171	4,56%
▾ { } DistributerGuiDo2	3582	95,44%	171	4,56%
▸ Constants	0	0,00%	10	100,00%
▾ InformationGetter	0	0,00%	39	100,00%
GetIndexOfSize(...)	0	0,00%	7	100,00%
GetMd5Hash(strin...)	0	0,00%	13	100,00%
GetUserGroup()	0	0,00%	16	100,00%
InformationGetter...	0	0,00%	3	100,00%
▾ Registrator	32	24,06%	101	75,94%
InitializeCompone...	0	0,00%	92	100,00%
Registrator(Distrib...	0	0,00%	9	100,00%
Abort(object, Syst...	2	100,00%	0	0,00%
AddUser(object, S...	26	100,00%	0	0,00%
RegistratorClose(o...	4	100,00%	0	0,00%
▾ RegistrationTransaction	15	41,67%	21	58,33%
RegistrationTrans...	0	0,00%	3	100,00%
Validate()	0	0,00%	18	100,00%
Execute()	4	100,00%	0	0,00%
InsertUserInDataB...	11	100,00%	0	0,00%

Рисунок 28 – Результаты покрытия кода

На рисунке 23 представлены результаты, как в процентном соотношении, так и в блочном. Исходя из результатов, представленных на рисунке выше можно сделать вывод о том, что покрытие рассматриваемого класса находится на достаточно высоком уровне, что свидетельствует о надежности исходного кода.

3.2.3 Метрики кода

Метрика программного обеспечения (англ. software metric) – это мера, позволяющая получить численное значение некоторого свойства программного обеспечения или его спецификаций.

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

$$MI = \text{MAX}(0, (171 - 5.2 * \ln(HV) - 0.23 * CC - 16.2 * \ln(LoC)) * 100 / 171),$$

где HV – Halstead Volume, вычислительная сложность;

CC – Cyclomatic Complexity. показывает структурную сложность кода, т.е. количество различных ветвей в коде;

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 и длину программы, а затем вычислить на их основе HV .

Словарь программы:

$$\eta = \eta_1 + \eta_2 ,$$

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

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

Длина программы:

$$N = N_1 + N_2 ,$$

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

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

Вычислительная сложность:

$$HV = N \log_2 \eta ,$$

Для определения структурной сложности программы сначала следует построить управляющий граф программы. Затем на его основе определить СС по следующей формуле:

$$CC = e - v + 2 \times p ,$$

где e – число дуг ориентированного графа программы;

v – число вершин ориентированного графа программы;

p – число компонент связности ориентированного графа программы.

Структурную сложность программы можно рассчитать и таким образом:

СС = количество операторов if + количество операторов цикла + 1.

Если полученное значение СС больше 10, то это свидетельствует о структурной сложности программы, что затрудняет ее функциональное тестирование (методом «белого ящика») и требует упрощения программы.

Visual Studio позволяет вычислять метрики автоматически. Произведем вычисления для разрабатываемого программного обеспечения. Метрики исходного кода представлены на рисунке 29.

Hierarchy	Maintainability Index	Cyclomatic Complexity	Depth of Inheritance	Class Coupling	Lines of Code
Testing (Debug)	84	226	7	92	1 571
Testing.Forms	52	104	7	83	1 154
Testing.Session	52	104	7	83	1 154
Testing.Model	82	74	2	14	345
SqlGateway	56	47	1	12	314
Logon	73	11	1	3	13
Teacher	91	2	2	1	3
User	92	14	1	0	15
ErrorState	100	0	1	0	0
Testing	89	48	2	13	72
Testing	69	16	1	9	38
Program	81	1	1	3	3
TestSessions	93	13	1	1	13
Question	94	7	1	0	7
NewQuestion	94	7	2	2	7
InitializedQuestion	95	3	2	2	3
Constants	100	1	1	0	1

Рисунок 29 – Метрики исходного кода проекта

Проанализируем полученные результаты. Комплексный показатель качества кода Maintainability Index может принимать значения от 0 до 100 и показывает относительную сложность поддержки кода. В данном проекте комплексный показатель находится на достаточно высоком уровне и лишь в некоторых классах опускается ниже 60%, что свидетельствует об относительной простоте поддержки исходного кода. Структурная сложность кода Cyclomatic Complexity, в некоторых классах достаточно высока, что свидетельствует о том, что их необходимо покрывать большим количеством тестов. Данные классы требуют упрощения. Depth of Inheritance – глубина наследования. В целом по проекту лишь классы-формы обладают глубиной наследования > 2 , в остальном данный показатель не велик. Class Coupling – показывает степень зависимости классов друг с другом. Чем больше данный показатель, тем сложнее поддержи-

вать данные классы. Как видим из результатов, классы форм получились несколько перегруженными.

Lines of Code – показывает количество строк кода, число строк кода лишь у главной формы является высоким, ввиду обширного функционала, в остальном классы получились достаточно компактными.

3.3 Планирование проекта и оценка бюджета

Управление проектами – область деятельности, в ходе которой определяются и достигаются цели проекта при балансировании между объемом работ, ресурсами (такими как люди, деньги, труд, материалы, энергия, пространство и др.), временем, качеством и рисками. Ключевым фактором успеха проектного управления является наличие четкого заранее определенного плана.

3.3.1 Планирование проекта

В ходе планирования проекта было выделено 18 задач. Перечень задач проекта, их виды, связи, длительность и предшественники приведены в таблице 28.

Таблица 28 – Задачи проекта

№	Название	Вид Задачи	Предшественники	Длительность
1	Начало реализации проекта	Веха		-
2	Анализ требований	Фаза		-
3	Анализ функциональных требований	Задача	1	1
4	Выделение бизнес-процессов предметной области	Задача	3	1
5	Анализ требований завершен	Веха	4	-
6	Проектирование	Фаза		-
7	Концептуальное проектирование базы данных	Задача	5	1
8	Логическое проектирование базы данных	Задача	7	1

Продолжение таблицы 28

№	Название	Вид Задачи	Предшественники	Длительность
9	Физическое проектирование базы данных	Задача	8	1
10	Проектирование пользовательского интерфейса	Задача	5	5
11	Проектирование интерфейса взаимодействия с базой данных	Задача	5	2
12	Проектирование завершено	Веха	9, 10, 11	-
13	Реализация	Фаза		-
14	Реализация интерфейса взаимодействия с базой данных	Задача	12	2
15	Реализация серверной части	Задача	8	1
16	Реализация завершена	Веха	14, 15	-
17	Отладка и тестирование	Фаза		-
18	Отладка клиентского приложения	Задача	14	3
19	Отладка серверной части	Задача	15	2
20	Тестирование серверной части	Задача	19	1
21	Автономное тестирование	Задача	18	1
22	Интеграционное тестирование	Задача	20, 21	3
23	Рефакторинг	Задача	21	5
24	Отладка и тестирование завершены	Веха	22, 23	-
25	Завершение проекта	Фаза		-
26	Составление программной документации	Задача	24	4
27	Сдача в эксплуатацию	Задача	26	1
28	Проект завершен	Веха	27	-
29	Конец проекта	Веха	28	-

Создадим проект в MS Project, установим дату начала проекта 01.09.15. Введем выделенные задачи в созданный проект, также установим их длительность и предшественников.

Полученная диаграмма Ганта приведена на рисунке 30.

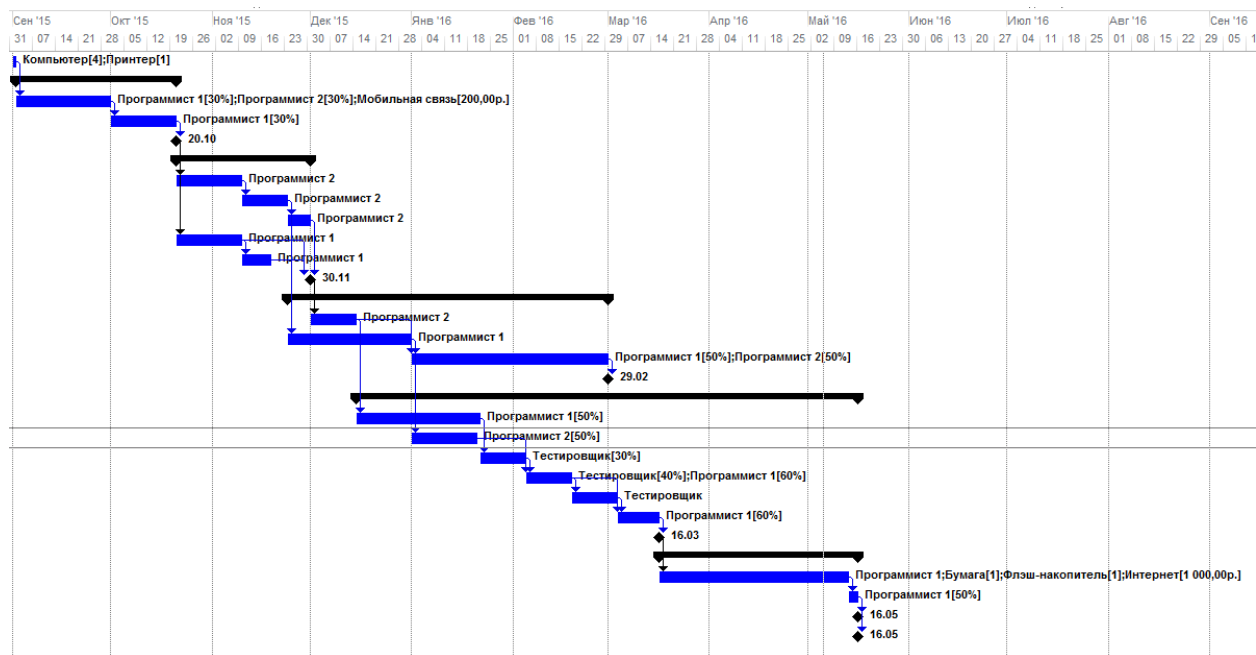


Рисунок 30 – Диаграмма Ганта

Определим необходимые ресурсы для выполнения каждой из задач данного проекта. Ресурсы и затраты приведены в таблице 29.

Таблица 29 – Ресурсы проекта

Название ресурса	Тип	Затраты			Затраты на исп.
		Таблица норм	Станд. ставка	Ставка сверхур.	
Программист 1	Т	А	30000 р./мес	350р./ч	—
Программист 2	Т	А	20000 р./мес	250р./ч	—
Тестировщик	Т	А	15000 р./мес	150р./ч	—
Компьютер	М	А	15000		—
Принтер	М	А	3000		—
Бумага	М	А	200		—
Флэш-накопитель	М	А	500		—
Интернет	З				—
Мобильная связь	З				—

Распределение ресурсов по задачам приведено в таблице 30.

Таблица 30 – Задачи проекта

Название	Ресурс	Единицы (затраты)	Таблица норм затрат
Начало проекта	Компьютер, Принтер	4 1	A A
Анализ функциональных требований	Программист 1, Программист 2 Мобильная связь	30 30	A A 200
Выделение бизнес-процессов предметной области	Программист 1, Программист 2	30 30	A A
Концептуальное проектирование базы данных	Программист 2	100	A
Логическое проектирование базы данных	Программист 2	100	A
Физическое проектирование базы данных	Программист 2	100	A
Проектирование пользовательского интерфейса	Программист 1	100	A
Проектирование интерфейса взаимодействия с базой данных	Программист 1	100	A
Реализация интерфейса взаимодействия с базой данных	Программист 2	100	A
Реализация серверной части	Программист 2	100	A
Отладка клиентского приложения	Программист 1	50	A
Отладка серверной части	Программист 2	100	A
Тестирование серверной части	Программист 2, Тестировщик	70 30	A A
Автономное тестирование	Тестировщик, Программист 1	40 60	A A
Интеграционное тестирование	Тестировщик	100	A
Рефакторинг	Программист 1	60	A
Составление программной документации	Программист 1, Программист 2, Бумага Флеш-накопитель Интернет	100 100 1 1	A A A A 1000
Сдача в эксплуатацию	Программист 1	50	A

Введем ресурсы в наш проект. Результаты приведены на рисунке 31.

Название ресурса	Тип	Единицы	Краткое	Группа	Макс.	Стандартная	Ставка	Затраты на	Начисление	Базовый
Программист 1	Трудовой		П		100%	30 000,00р./мес	350,00р./ч	0,00р.	Пропорциональн	Стандартный
Программист 2	Трудовой		П		100%	20 000,00р./мес	250,00р./ч	0,00р.	Пропорциональное	Стандартный
Тестировщик	Трудовой		Т		100%	15 000,00р./мес	150,00р./ч	0,00р.	Пропорциональное	Стандартный
Компьютер	Материальный		К			15 000,00р.		0,00р.	Пропорциональное	
Принтер	Материальный		П			3 000,00р.		0,00р.	Пропорциональное	
Бумага	Материальный		Б			200,00р.		0,00р.	Пропорциональное	
Флэш-накопитель	Материальный		Ф			500,00р.		0,00р.	Пропорциональное	
Интернет	Затраты		И						Пропорциональное	
Мобильная связь	Затраты		М						Пропорциональное	

Рисунок 31 – Ресурсы проекта

Закрепим за задачами соответствующие ресурсы. Результаты выполнения приведены на рисунке 32.

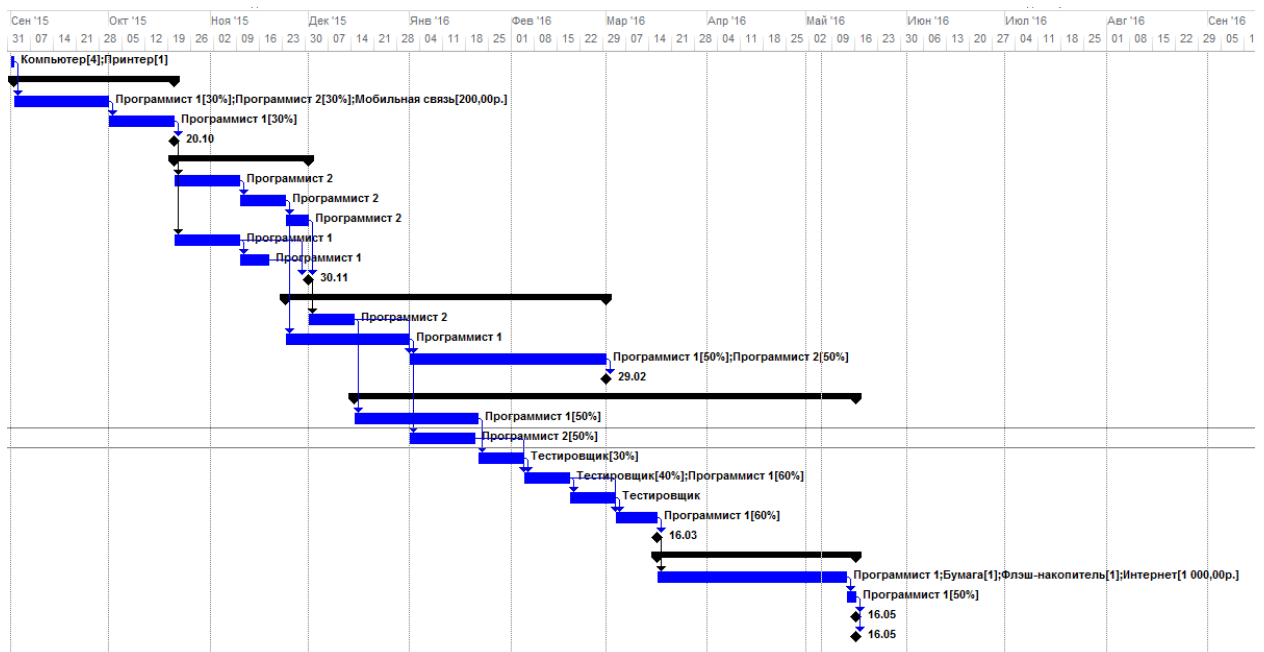


Рисунок 32 – Диаграмма Ганта

Сетевой график данного проекта представлен на рисунке 33. Красным цветом на рисунке 33 выделен критический путь.

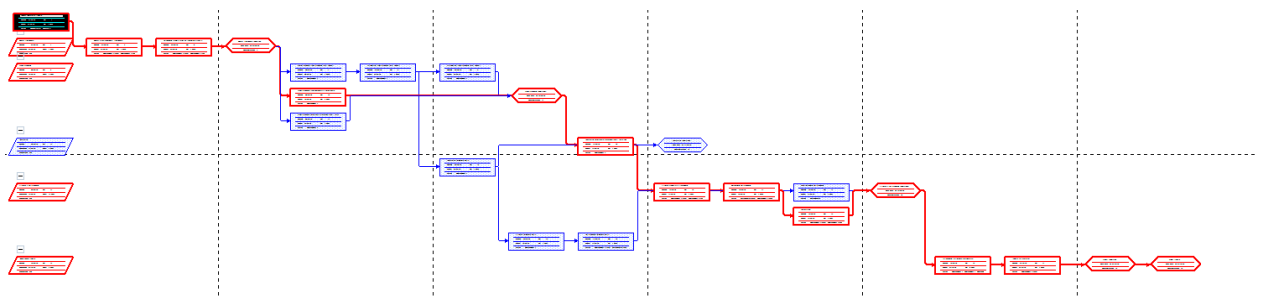


Рисунок 33 – Сетевой график

Графики загрузки трудовых ресурсов программиста 1, программиста 2 и тестировщика представлены на рисунках 34 – 36 соответственно.

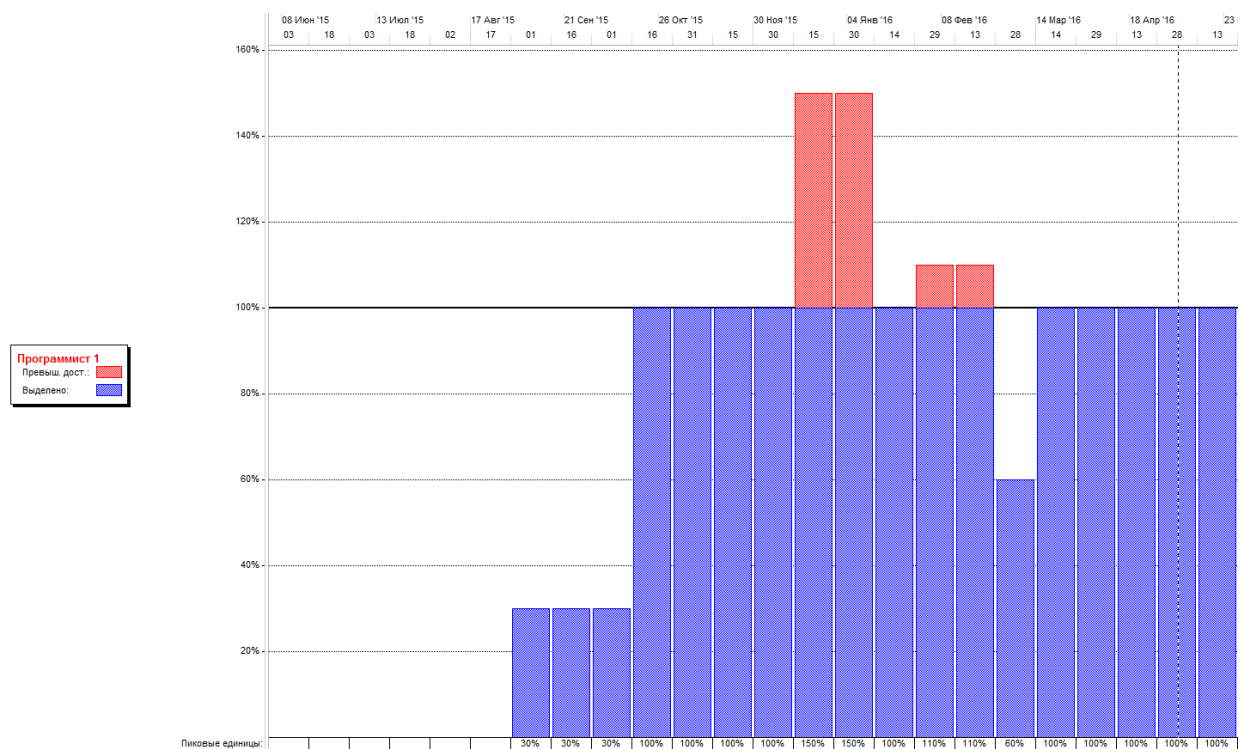


Рисунок 34 – График загрузки программиста 1

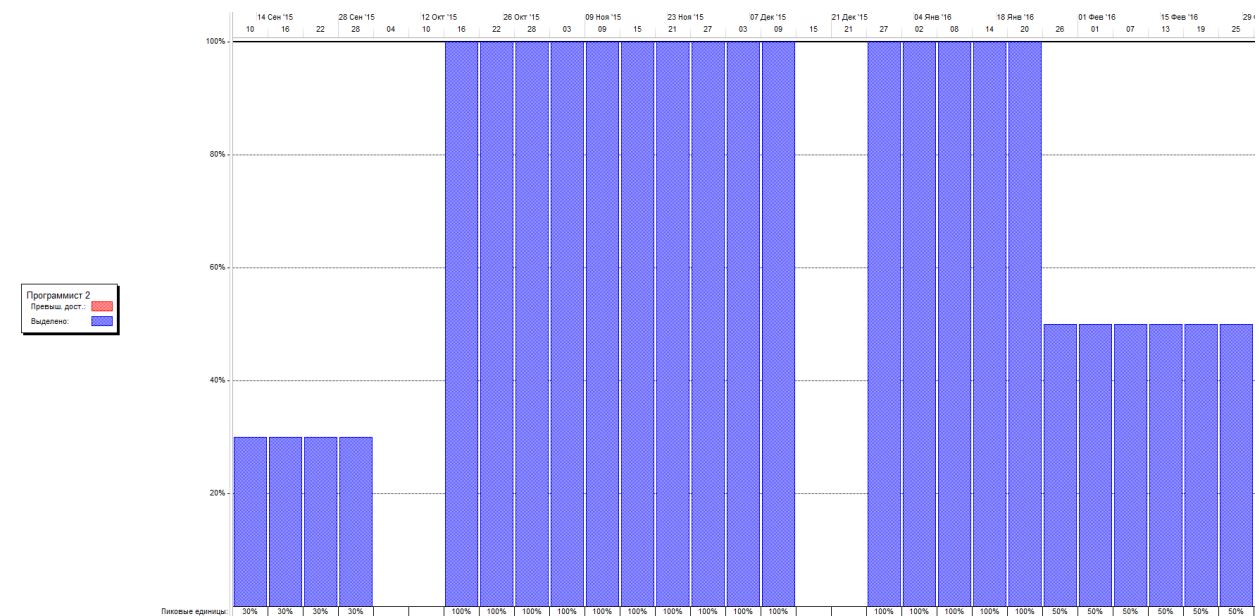


Рисунок 35 – График загрузки программиста 2

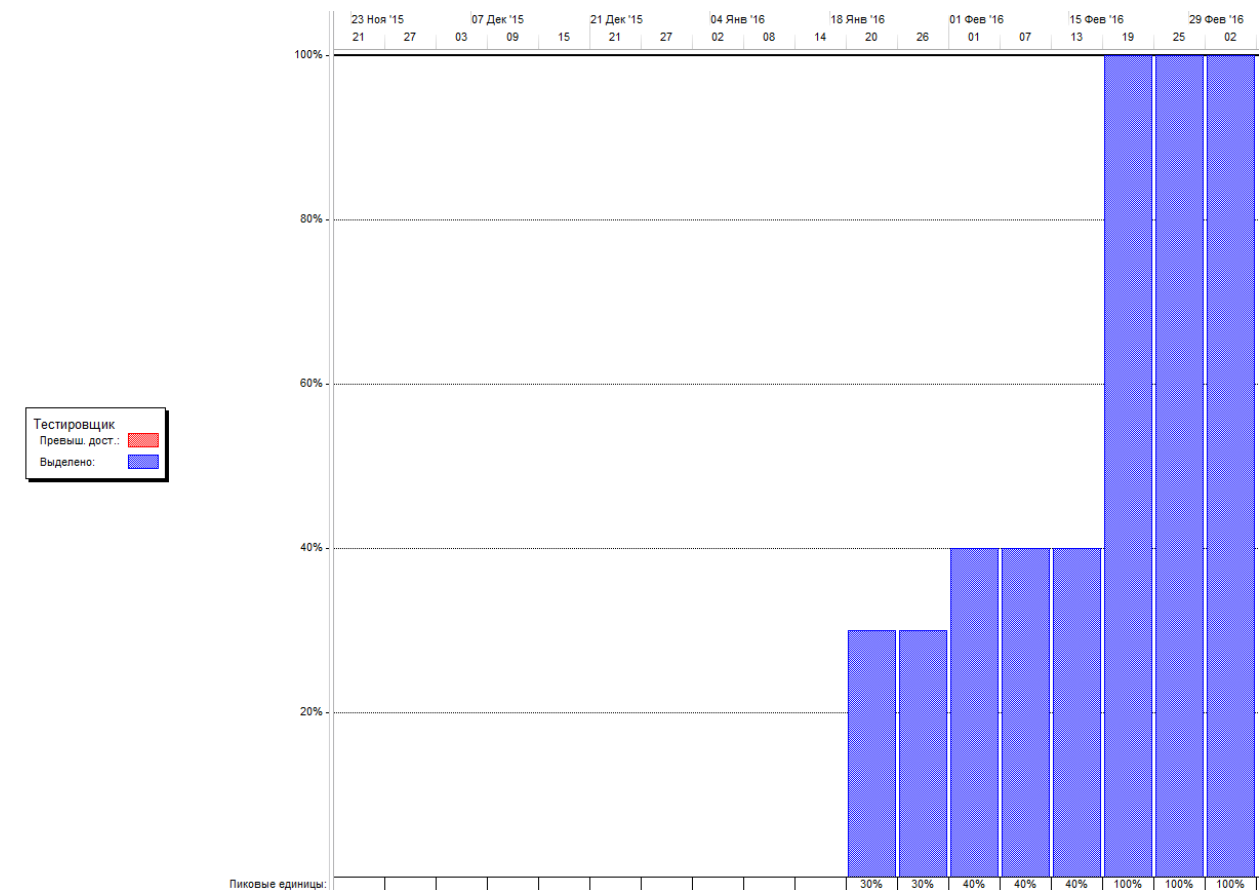
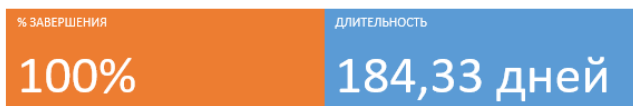


Рисунок 36 – График загрузки тестировщика

Сводка по проекту представлена на рисунке 37. На полученной сводке указана дата начала и окончания работ, длительность в днях, процент выполнения на данном этапе, а также процент завершения по каждой из задач верхнего уровня [16, 17].

ОБЗОР ПРОЕКТА

ВТ 01.09.15 - ПН 16.05.16

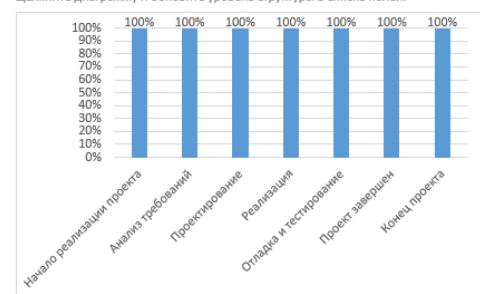


ВСЕХ С НАСТУПИВШИМ СРОКОМ
Приближающиеся веки

Название	Окончание
----------	-----------

% ЗАВЕРШЕНИЯ

Состояние всех задач верхнего уровня. Чтобы просмотреть состояние вложенных задач, щелкните диаграмму и обновите уровень структуры в списке полей.



ЗАДАЧИ С ЗАДЕРЖКОЙ
Просроченные задачи

Название	Начало	Окончание	Длительность	% завершения	Названия ресурсов
----------	--------	-----------	--------------	--------------	-------------------

Рисунок 37 – Сводка по проекту

При помощи Microsoft Project получим отчеты по затратам на данный программный продукт (рисунок 38).

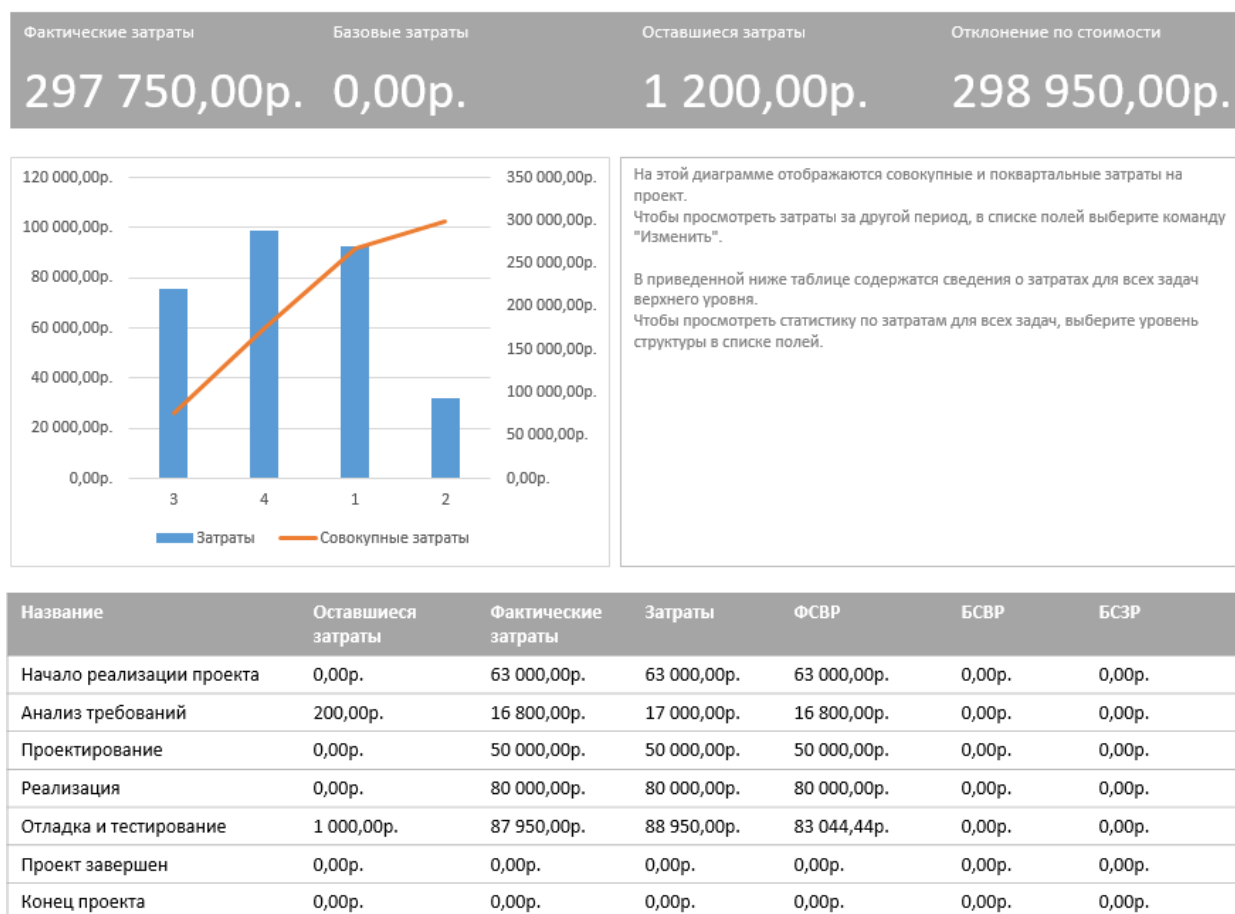


Рисунок 38 – Отчет по затратам

На отчетах видно, что длительность выполнения проекта составила 184 дня, а его бюджет составляет 298 950 р.

Заключение

При выполнении выпускной квалификационной работы была разработана автоматизированная система тестирования знаний, состоящая из клиентского приложения и базы данных.

Было проведено концептуальное, логическое и физическое проектирование реляционной базы данных, проектирование графического интерфейса. База данных была реализована в СУБД MS SQL-Server 2012.

Клиентское приложение реализовано с помощью языка C# в интегрированной среде разработки Visual Studio 2013. Серверная часть приложения: хранимые процедуры и триггеры реализованы с помощью языка SQL.

В ходе выполнения работы также было выполнено планирование проекта. Длительность выполнения проекта равна 184 дня, а бюджет равен 298 950 р. Оценка сроков и бюджета говорит о том, что данная разработка экономически выгодна.

В ходе работы было проведено функциональное и модульное тестирование. На основе анализа результатов тестирования можно сделать вывод, что разработанное программное обеспечение поддержки непрерывной интеграции работает корректно.

Таким образом, в результате выполнения выпускной квалификационной работы было разработано программное обеспечение, отвечающее всем требованиям технического задания.

Список использованных источников

1. Нуждин В.Н., Кадамцева Г.Г. Стратегическое управление качеством образования: Учеб. пособие. / В.Н. Нуждин – Иваново, 2003. – 88 с.
2. Афанасьев В.В., Тыщенко О.Б., Афанасьева И.В. Оценка уровня усвоения знаний с применением компьютерной техники: Тез. докл. I Всерос. науч.-техн. конф. «Компьютерные технологии в науке, проектировании и производстве», часть V. / В.В. Афанасьев – Нижний Новгород, 1999. – 15 с.
3. Арлоу, Д. UML 2 и Унифицированный процесс. Второе издание / Д. Арлоу, А. Нейштадт. – СПб.: Питер, 2006. – 624 с.
4. Рамбо, Дж. UML 2.0. Объектно-ориентированное моделирование и разработка. / Дж. Рамбо, М. Блаха. – СПб.:Питер, 2007. – 544 с.
5. Розенберг, Д. Применение объектного моделирования с использованием UML и анализ прецедентов. / Д. Розенберг – М.: Книга по требованию, 2002. – 159 с.
6. Романов, А.А. Введение в Rational Unified Process / А.А. Романов – Вильямс, 2002. – 240 с.
7. Конноли, Т. Базы данных: Проектирование, реализация и сопровождение. Теория и практика. Второе издание / Т. Конноли, К. Бегг, А. Страчан. – М.: Вильямс, 2000. – 1139 с.
8. Сеппа, Д. Программирование на MS ADO.NET 2.0. Мастер класс. / Д. Сеппа – СПб.: Питер, 2007. – 638 с.
9. Казакова, И.А. Основы языка Transact SQL : учеб. пособие / И. А. Казакова. – Пенза : Издательство ПГУ, 2010. – 164 с.
10. Ошероув, Р. Искусство автономного тестирования с примерами на C#. / Р. Ошероув – М.: ДМК Пресс, 2014. – 360 с.
11. Зиборов, В. Visual C# 2010 на примерах. / В. Зиборов – СПб. БХВ-Петербург, 2011. – 480 с.

12. Макконнелл, С. Совершенный код. Мастер-класс. / С. Макконнелл – М.: Русская редакция, 2013. – 893 с.
13. Рихтер, Дж. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#. Четвертое издание / Дж. Рихтер – СПб.: Питер, 2013. – 656 с.
14. Скит, Д. C#: программирование для профессионалов, Третье издание / Д. Скит – М.: Вильямс, 2014. – 608 с.
15. Гамма, Э. Приемы объектно-ориентированного проектирования. Паттерны проектирования. / Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес – СПб.: Питер, 2001. – 344 с.
16. Куперштейн, В. И. Microsoft Project 2010 в управлении проектами. / В. И. Куперштейн, А.В. Цветкова – СПб.: БХВ-Петербург, 2011. – 416 с.
17. Шкрыль, А.А. MS Project 2007: современное управление проектами. / А.А. Шкрыль – СПб.: БХВ-Петербург, 2007. – 256 с.

ЛИСТИНГ ПРОГРАММЫ

Приложение А

(обязательное)

A.1 Листинг на языке C# (Logon.cs)

```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Testing.Model
{
    //статусы выполнения операций
    public enum ErrorState
    {
        Success,
        SubjectIsEmpty,
        PasswordNotCorrectly,
        UserExists,
        FieldsIsEmpty
    }
    /// <summary>
    /// Класс для входа и регистрации пользователей
    /// </summary>
    public class Logon
    {
        //ссылка на шлюз с функциями вызывающими ХП в БД
        private SqlGateway gateway;

        public Logon(SqlGateway gateway)
        {
            this.gateway = gateway;
        }
        /// <summary>
        /// Получить информацию о пользователе по логину и паролю
        /// </summary>
        /// <param name="login">Логин</param>
        /// <param name="password">Пароль</param>
        /// <param name="isTeacher">Признак того, что идёт проверка среди преподавателей</param>
        /// <returns>Ссылку на класс пользователя, если он был найден, в противном случае null</returns>
        public User CheckUser(string login, string password, bool isTeacher)
        {
            return gateway.GetUserInfo(isTeacher, login, password);
        }
        /// <summary>
        /// Регистрация нового пользователя
        /// </summary>
        /// <param name="name">Имя</param>
        /// <param name="surname">Фамилия</param>
        /// <param name="lastname">Отчество</param>
        /// <param name="login">Логин</param>
        /// <param name="pass">Пароль</param>
        /// <param name="pass2">Подтверждение</param>
        /// <param name="isTeacher">Если регистрируется как преподаватель, то этот сюда передаём True</param>
        /// <param name="subject">Если регистрируется преподаватель, то здесь название предмета</param>
        /// <returns>Статус выполнения операции, если успешно - Success, в противном случае один из статусов ошибки</returns>
    }
}
```

```

        public ErrorState AddNewUser(string name, string surname, string
lastname, string login, string pass, string pass2, bool isTeacher)
        {
            //все поля должны быть заполнены
            if (string.IsNullOrEmpty(name) ||
                string.IsNullOrEmpty(surname) ||
                string.IsNullOrEmpty(lastname) ||
                string.IsNullOrEmpty(login) ||
                string.IsNullOrEmpty(pass) ||
                string.IsNullOrEmpty(pass2))
                return ErrorState.FieldsIsEmpty;

            //введенные пароли должны совпадать
            if (!pass.Equals(pass2))
                return ErrorState.PasswordNotCorrectly;
            //пользователь с таким логином и паролем должен отсутствовать
            if (CheckUser(login, pass, isTeacher) != null)
                return ErrorState.UserIsExists;

            //если все условия выполнены - регистрируем нового пользова-
теля
            gateway.AddNewUser(name, surname, lastname, login,
pass,isTeacher);
            return ErrorState.Success;
        }
    }
}

```

A.2 Листинг на языке C# (SqlGateway.cs)

```

using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Testing.Forms;

namespace Testing.Model
{
    /// <summary>
    /// Класс-sql шлюз
    /// </summary>
    public class SqlGateway
    {
        //подключение к SQL серверу
        private readonly SqlConnection connection;
        /// <summary>
        /// Конструктор шлюза SQL
        /// </summary>
        /// <param name="connection"></param>
        public SqlGateway(SqlConnection connection)
        {
            this.connection = connection;
        }
        /// <summary>
        /// Получить информацию о пользователе
        /// </summary>
        /// <param name="isTeacher">Признак того, что идёт проверка среди
преподавателей</param>
        /// <param name="login">Логин</param>
        /// <param name="password">Пароль</param>
    }
}

```



```

        /// <returns>Если информация была получена из БД, то возвращается
        проинициализированный объект пользователя</returns>
        public User GetUserInfo(bool isTeacher, string login, string pass-
word)
        {
            SqlCommand sqlComm;
            sqlComm = isTeacher ? new SqlCommand("GetTeacherInfo", connec-
tion) :
                new SqlCommand("GetStudentInfo", connection);

            sqlComm.CommandType = System.Data.CommandType.StoredProcedure;

            SqlParameter param = new SqlParameter();
            param.ParameterName = "@Login";
            param.SqlDbType = System.Data.SqlDbType.VarChar;
            param.Value = login;

            sqlComm.Parameters.Add(param);

            param = new SqlParameter();
            param.ParameterName = "@Password";
            param.SqlDbType = System.Data.SqlDbType.VarChar;
            param.Value = password;

            sqlComm.Parameters.Add(param);

            //выполняем запрос
            using (SqlDataReader sqlDReader = sqlComm.ExecuteReader())
            {
                //если данные были получены, то создаём объект класса
                преподаватель/студент
                if (sqlDReader.Read())
                {
                    if (isTeacher)
                    {
                        Teacher teacher = new Teacher();
                        teacher.Id = sqlDReader.GetInt32(0);
                        teacher.Name = sqlDReader.GetString(1);
                        teacher.Surname = sqlDReader.GetString(2);
                        teacher.LastName = sqlDReader.GetString(3);

                        return teacher;
                    }
                    User user = new User();
                    user.Id = sqlDReader.GetInt32(0);
                    user.Name = sqlDReader.GetString(1);
                    user.Surname = sqlDReader.GetString(2);
                    user.LastName = sqlDReader.GetString(3);

                    return user;
                }
                return null;
            }
        }
        /// <summary>
        /// Регистрировать нового пользователя
        /// </summary>
        /// <param name="name"></param>
        /// <param name="surname"></param>
        /// <param name="lastname"></param>
        /// <param name="login"></param>

```

```

/// <param name="password"></param>
/// <param name="isTeacher"></param>
/// <param name="subject"></param>
/// <returns></returns>
public int AddNewUser(string name, string surname, string lastname,
string login, string password, bool isTeacher = false, string subject = "")
{
    if (GetSubjectId(subject) == -1)
    {
        AddSubject(subject);
    }

    SqlCommand sqlComm = new SqlCommand("AddUser", connection);

    sqlComm.CommandType = System.Data.CommandType.StoredProcedure;

    SqlParameter param = new SqlParameter();
    param.ParameterName = "@Name";
    param.SqlDbType = System.Data.SqlDbType.VarChar;
    param.Value = name;

    sqlComm.Parameters.Add(param);

    param = new SqlParameter();
    param.ParameterName = "@Surname";
    param.SqlDbType = System.Data.SqlDbType.VarChar;
    param.Value = surname;

    sqlComm.Parameters.Add(param);

    param = new SqlParameter();
    param.ParameterName = "@LastName";
    param.SqlDbType = System.Data.SqlDbType.VarChar;
    param.Value = lastname;

    sqlComm.Parameters.Add(param);

    param = new SqlParameter();
    param.ParameterName = "@Login";
    param.SqlDbType = System.Data.SqlDbType.VarChar;
    param.Value = login;

    sqlComm.Parameters.Add(param);

    param = new SqlParameter();
    param.ParameterName = "@Password";
    param.SqlDbType = System.Data.SqlDbType.VarChar;
    param.Value = password;

    sqlComm.Parameters.Add(param);

    param = new SqlParameter();
    param.ParameterName = "@IsTeacher";
    param.SqlDbType = System.Data.SqlDbType.Bit;
    param.Value = isTeacher;

    sqlComm.Parameters.Add(param);

    param = new SqlParameter();
    param.ParameterName = "@Subject";
    param.SqlDbType = System.Data.SqlDbType.VarChar;

```

```

        param.Value = subject;

        sqlComm.Parameters.Add(param);

        return sqlComm.ExecuteNonQuery();
    }
    /// <summary>
    /// Получить идентификатор предмета
    /// </summary>
    /// <param name="subject"></param>
    /// <returns>ID предмета если он существует, в противном случае -
1</returns>
    public int GetSubjectId(string subject)
    {
        SqlCommand sqlComm = new SqlCommand("GetSubjectId", connec-
tion);

        sqlComm.CommandType = System.Data.CommandType.StoredProcedure;

        SqlParameter param = new SqlParameter();
        param.ParameterName = "@Name";
        param.SqlDbType = System.Data.SqlDbType.VarChar;
        param.Value = subject;

        sqlComm.Parameters.Add(param);
        using (SqlDataReader sqlDReader = sqlComm.ExecuteReader())
        {
            int result = -1;

            while (sqlDReader.Read())
            {
                result = sqlDReader.GetInt32(0);
            }
            return result;
        }
    }
    /// <summary>
    /// Добавить предмет
    /// </summary>
    /// <param name="subject"></param>
    public void AddSubject(string subject)
    {
        SqlCommand sqlComm = new SqlCommand("AddSubject", connection);

        sqlComm.CommandType = System.Data.CommandType.StoredProcedure;

        SqlParameter param = new SqlParameter();
        param.ParameterName = "@Name";
        param.SqlDbType = System.Data.SqlDbType.VarChar;
        param.Value = subject;

        sqlComm.Parameters.Add(param);

        sqlComm.ExecuteNonQuery();
    }

    /// <summary>
    /// Добавить предмет
    /// </summary>
    /// <param name="subject"></param>

```

```

public void AddTest(int idSubject, string test)
{
    SqlCommand sqlComm = new SqlCommand("AddTest", connection);

    sqlComm.CommandType = System.Data.CommandType.StoredProcedure;

    SqlParameter param = new SqlParameter();
    param.ParameterName = "@IdSubject";
    param.SqlDbType = System.Data.SqlDbType.Int;
    param.Value = idSubject;
    sqlComm.Parameters.Add(param);

    param = new SqlParameter();
    param.ParameterName = "@Name";
    param.SqlDbType = System.Data.SqlDbType.VarChar;
    param.Value = test;

    sqlComm.Parameters.Add(param);

    sqlComm.ExecuteNonQuery();
}

/// <summary>
/// Получить список предметов
/// </summary>
/// <returns></returns>
public List<string> GetSubjects()
{
    SqlCommand sqlComm = new SqlCommand("GetSubjects", connection);

    sqlComm.CommandType = System.Data.CommandType.StoredProcedure;

    using (SqlDataReader sqlDReader = sqlComm.ExecuteReader())
    {
        List<string> subjects = new List<string>();

        while (sqlDReader.Read())
        {
            subjects.Add(sqlDReader.GetString(0));
        }

        return subjects;
    }
}

/// <summary>
/// Получить список тестов
/// </summary>
/// <param name="subjectName"></param>
/// <returns></returns>
public List<string> GetTests(string subjectName)
{
    SqlCommand sqlComm = new SqlCommand("GetTests", connection);

    sqlComm.CommandType = System.Data.CommandType.StoredProcedure;

    SqlParameter param = new SqlParameter();
    param.ParameterName = "@Subject";
    param.SqlDbType = System.Data.SqlDbType.VarChar;
    param.Value = subjectName;

```

```

        sqlComm.Parameters.Add(param);
        using (SqlDataReader sqlDReader = sqlComm.ExecuteReader())
        {
            List<string> tests = new List<string>();

            while (sqlDReader.Read())
            {
                tests.Add(sqlDReader.GetString(0));
            }

            return tests;
        }
    }
    /// <summary>
    /// Получить ID-теста по названию
    /// </summary>
    /// <param name="testName">Название теста</param>
    /// <returns>Id в случае успеха, в противном случае -1</returns>
    public int GetTestId(string testName)
    {
        SqlCommand sqlComm = new SqlCommand("GetTestId", connection);

        sqlComm.CommandType = System.Data.CommandType.StoredProcedure;

        SqlParameter param = new SqlParameter();
        param.ParameterName = "@TestName";
        param.SqlDbType = System.Data.SqlDbType.VarChar;
        param.Value = testName;

        sqlComm.Parameters.Add(param);
        using (SqlDataReader sqlDReader = sqlComm.ExecuteReader())
        {
            int result = -1;

            while (sqlDReader.Read())
            {
                result = sqlDReader.GetInt32(0);
            }

            return result;
        }
    }
    /// <summary>
    /// Получить словарь пар Id вопроса и его содержание
    /// </summary>
    /// <param name="idTest">Id теста</param>
    /// <returns>Словарь пар значений id, вопрос</returns>
    public Dictionary<int, string> GetQuestions(int idTest)
    {
        SqlCommand sqlComm = new SqlCommand("GetQuestions", connec-
tion);

        sqlComm.CommandType = System.Data.CommandType.StoredProcedure;

        SqlParameter param = new SqlParameter();
        param.ParameterName = "@IdTest";
        param.SqlDbType = System.Data.SqlDbType.Int;
        param.Value = idTest;

        sqlComm.Parameters.Add(param);

```

```

        using (SqlDataReader sqlDReader = sqlComm.ExecuteReader())
        {
            Dictionary<int, string> questions = new Dictionary<int,
string>();

            while (sqlDReader.Read())
            {
                questions.Add(sqlDReader.GetInt32(0), sqlDRead-
er.GetString(1));
            }

            return questions;
        }
    }
    /// <summary>
    /// Получить словарь ответов id, содержание ответа
    /// </summary>
    /// <param name="idQuestion">Id вопроса</param>
    /// <returns>Словарь пар значений id, содержание ответа</returns>
    public Dictionary<int, string> GetAnswers(int idQuestion)
    {
        SqlCommand sqlComm = new SqlCommand("GetAnswers", connection);

        sqlComm.CommandType = System.Data.CommandType.StoredProcedure;
        SqlParameter param = new SqlParameter();
        param.ParameterName = "@IdQuestion";
        param.SqlDbType = System.Data.SqlDbType.Int;
        param.Value = idQuestion;

        sqlComm.Parameters.Add(param);

        using (SqlDataReader sqlDReader = sqlComm.ExecuteReader())
        {
            Dictionary<int, string> answers = new Dictionary<int,
string>();

            while (sqlDReader.Read())
            {
                answers.Add(sqlDReader.GetInt32(0), sqlDRead-
er.GetString(1));
            }

            return answers;
        }
    }
    /// <summary>
    /// Получить ID правильного ответа по Id вопроса
    /// </summary>
    /// <param name="idQuestion">Id вопроса</param>
    /// <returns>Id правильного ответа</returns>
    public int GetCorrectAnswer(int idQuestion)
    {
        SqlCommand sqlComm = new SqlCommand("GetCorrectAnswer", con-
nection);

        sqlComm.CommandType = System.Data.CommandType.StoredProcedure;
        SqlParameter param = new SqlParameter();
        param.ParameterName = "@IdQuestion";
        param.SqlDbType = System.Data.SqlDbType.Int;
        param.Value = idQuestion;
    }

```

```

        sqlComm.Parameters.Add(param);

        using (SqlDataReader sqlDReader = sqlComm.ExecuteReader())
        {
            int result = -1;

            while (sqlDReader.Read())
            {
                result = sqlDReader.GetInt32(0);
            }

            return result;
        }
    }
    /// <summary>
    /// Сохранить результаты прохождения тестирования
    /// </summary>
    /// <param name="correctCount">Количество правильных ответов</param>
    /// <param name="mark">Оценка</param>
    /// <param name="idUser">Номер студента</param>
    /// <param name="idTest">Номер теста</param>
    public void ReportTestSession(int correctCount, int mark, int
idUser, int idTest)
    {
        SqlCommand sqlComm = new SqlCommand("ReportTestSession", con-
nection);

        sqlComm.CommandType = System.Data.CommandType.StoredProcedure;

        SqlParameter param = new SqlParameter();
        param.ParameterName = "@CorrectCount";
        param.SqlDbType = System.Data.SqlDbType.Int;
        param.Value = correctCount;

        sqlComm.Parameters.Add(param);

        param = new SqlParameter();
        param.ParameterName = "@Date";
        param.SqlDbType = System.Data.SqlDbType.DateTime;
        param.Value = DateTime.Now;

        sqlComm.Parameters.Add(param);

        param = new SqlParameter();
        param.ParameterName = "@Mark";
        param.SqlDbType = System.Data.SqlDbType.Int;
        param.Value = mark;

        sqlComm.Parameters.Add(param);

        param = new SqlParameter();
        param.ParameterName = "@IdUser";
        param.SqlDbType = System.Data.SqlDbType.VarChar;
        param.Value = idUser;

        sqlComm.Parameters.Add(param);

        param = new SqlParameter();
        param.ParameterName = "@IdTest";
        param.SqlDbType = System.Data.SqlDbType.VarChar;
        param.Value = idTest;
    }

```

```

        sqlComm.Parameters.Add(param);

        sqlComm.ExecuteNonQuery();
    }
    /// <summary>
    /// Получить результаты тестирования
    /// </summary>
    /// <returns>Список результатов</returns>
    public List<TestSessions> GetTestSessions()
    {
        List<TestSessions> testSessions = new List<TestSessions>();
        SqlCommand sqlComm = new SqlCommand("GetTestSessions", connec-
tion);

        using (var sqlDReader = sqlComm.ExecuteReader())
        {
            while (sqlDReader.Read())
            {
                TestSessions tSession = new TestSessions();

                tSession.Surname = sqlDReader.GetString(0);
                tSession.Name = sqlDReader.GetString(1);
                tSession.LastName = sqlDReader.GetString(2);
                tSession.Date = sqlDReader.GetDateTime(3);
                tSession.TestName = sqlDReader.GetString(4);
                tSession.Mark = sqlDReader.GetInt32(5);

                testSessions.Add(tSession);
            }
        }

        return testSessions;
    }
    /// <summary>
    /// Удалить предмет
    /// </summary>
    /// <param name="idSubject">Номер предмета</param>
    public void RemoveSubject(int idSubject)
    {
        SqlCommand sqlComm = new SqlCommand("DeleteSubject", connec-
tion);

        sqlComm.CommandType = System.Data.CommandType.StoredProcedure;
        SqlParameter param = new SqlParameter();
        param.ParameterName = "@IdSubject";
        param.SqlDbType = System.Data.SqlDbType.Int;
        param.Value = idSubject;

        sqlComm.Parameters.Add(param);

        sqlComm.ExecuteNonQuery();
    }
    /// <summary>
    /// Удалить тест
    /// </summary>
    /// <param name="idTest">Номер теста</param>
    public void RemoveTest(int idTest)
    {
        SqlCommand sqlComm = new SqlCommand("DeleteTest", connection);

```



```

        sqlCommand.CommandType = System.Data.CommandType.StoredProcedure;
        SqlParameter param = new SqlParameter();
        param.ParameterName = "@IdTest";
        param.SqlDbType = System.Data.SqlDbType.Int;
        param.Value = idTest;

        sqlCommand.Parameters.Add(param);

        sqlCommand.ExecuteNonQuery();
    }
    /// <summary>
    /// Удалить вопрос
    /// </summary>
    /// <param name="idQuestion">Номер вопроса</param>
    public void RemoveQuestion(int idQuestion)
    {
        SqlCommand sqlCommand = new SqlCommand("DeleteQuestion", connec-
tion);

        sqlCommand.CommandType = System.Data.CommandType.StoredProcedure;
        SqlParameter param = new SqlParameter();
        param.ParameterName = "@IdQuestion";
        param.SqlDbType = System.Data.SqlDbType.Int;
        param.Value = idQuestion;

        sqlCommand.Parameters.Add(param);

        sqlCommand.ExecuteNonQuery();
    }
    /// <summary>
    /// ДОБАВИТЬ вопрос в базу данных
    /// </summary>
    /// <param name="idTest">Номер теста</param>
    /// <param name="text">Текст вопроса</param>
    public void AddQuestion(int idTest, string text)
    {
        SqlCommand sqlCommand = new SqlCommand("AddQuestion", connec-
tion);

        sqlCommand.CommandType = System.Data.CommandType.StoredProcedure;
        SqlParameter param = new SqlParameter();
        param.ParameterName = "@IdTest";
        param.SqlDbType = System.Data.SqlDbType.Int;
        param.Value = idTest;

        sqlCommand.Parameters.Add(param);

        param = new SqlParameter();
        param.ParameterName = "@Text";
        param.SqlDbType = System.Data.SqlDbType.VarChar;
        param.Value = text;

        sqlCommand.Parameters.Add(param);

        sqlCommand.ExecuteNonQuery();
    }
    /// <summary>
    /// Получить номер вопроса по тексту
    /// </summary>
    /// <param name="text">Текст вопроса</param>
    /// <returns>Номер вопроса в случае успеха, иначе -1</returns>

```

```

public int GetQuestionId(string text)
{
    SqlCommand sqlComm = new SqlCommand("GetQuestionId", connec-
tion);

    sqlComm.CommandType = System.Data.CommandType.StoredProcedure;
    SqlParameter param = new SqlParameter();
    param.ParameterName = "@Text";
    param.SqlDbType = System.Data.SqlDbType.VarChar;
    param.Value = text;

    sqlComm.Parameters.Add(param);

    int idQuestion = -1;

    using (var sqlReader = sqlComm.ExecuteReader())
    {
        if (sqlReader.Read())
            idQuestion = sqlReader.GetInt32(0);
    }

    return idQuestion;
}
/// <summary>
/// Добавить ответ на вопрос
/// </summary>
/// <param name="answer">Текст ответа</param>
/// <param name="idQuestion">Номер вопроса</param>
public void AddAnswer(string answer, int idQuestion)
{
    SqlCommand sqlComm = new SqlCommand("AddAnswer", connection);

    sqlComm.CommandType = System.Data.CommandType.StoredProcedure;

    SqlParameter param = new SqlParameter();
    param.ParameterName = "@Text";
    param.SqlDbType = System.Data.SqlDbType.VarChar;
    param.Value = answer;

    sqlComm.Parameters.Add(param);

    param = new SqlParameter();
    param.ParameterName = "@IdQuestion";
    param.SqlDbType = System.Data.SqlDbType.Int;
    param.Value = idQuestion;

    sqlComm.Parameters.Add(param);

    sqlComm.ExecuteNonQuery();
}
/// <summary>
/// Получить номер ответа по тексту
/// </summary>
/// <param name="answerText">Текст ответа</param>
/// <returns>Номер вопроса в случае успеха, иначе -1</returns>
public int GetAnswerId(string answerText)
{
    SqlCommand sqlComm = new SqlCommand("GetAnswerId", connec-
tion);

```

```

        sqlComm.CommandType = System.Data.CommandType.StoredProcedure;
        SqlParameter param = new SqlParameter();
        param.ParameterName = "@Text";
        param.SqlDbType = System.Data.SqlDbType.VarChar;
        param.Value = answerText;

        sqlComm.Parameters.Add(param);

        int idAnswer = -1;

        using (var sqlReader = sqlComm.ExecuteReader())
        {
            if (sqlReader.Read())
                idAnswer = sqlReader.GetInt32(0);
        }

        return idAnswer;
    }
    /// <summary>
    /// Добавить правильные ответ в базу данных
    /// </summary>
    /// <param name="idQuestion">Номер вопроса</param>
    /// <param name="correctAnswer">Номер правильного ответа</param>
    public void AddCorrectAnswer(int idQuestion, int correctAnswer)
    {
        SqlCommand sqlComm = new SqlCommand("AddCorrectAnswer", con-
nection);

        sqlComm.CommandType = System.Data.CommandType.StoredProcedure;

        SqlParameter param = new SqlParameter();
        param.ParameterName = "@IdAnswer";
        param.SqlDbType = System.Data.SqlDbType.Int;
        param.Value = correctAnswer;

        sqlComm.Parameters.Add(param);

        param = new SqlParameter();
        param.ParameterName = "@IdQuestion";
        param.SqlDbType = System.Data.SqlDbType.Int;
        param.Value = idQuestion;

        sqlComm.Parameters.Add(param);

        sqlComm.ExecuteNonQuery();
    }
}

```

A.3 Листинг на языке C# (User.cs)

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Testing.Model
{
    /// <summary>
    /// Класс пользователя (обычный студент, умеет проходить тесты и получать
    результаты собственного тестирования)
    /// </summary>

```

```

public class User
{
    public int Id { get; set; }
    public string Login { get; set; }

    public string Password { get; set; }

    public string Name { get; set; }

    public string Surname { get; set; }

    public string LastName { get; set; }

    public override string ToString()
    {
        return String.Format("{0} {1} {2}", Surname, Name, LastName);
    }
}
/// <summary>
/// Класс преподавателя (умеет добавлять тесты, получать результаты тести-
рования)
/// </summary>
public class Teacher: User
{
    public override string ToString()
    {
        return String.Format("Преподаватель: {0} {1} {2}", Surname,
Name, LastName);
    }
}
}

```

A.4 Листинг на языке C# (Testing.cs)

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Testing.Model;

namespace Testing
{
    /// <summary>
    /// Класс тестирования
    /// </summary>
    public class Testing
    {
        //ссылка на класс-шлюз между программой и бд
        private SqlGateway sqlGateway;
        //номер теста
        private int idTest;
        //номер студента
        private int idUser;
        //список вопросов теста
        public List<InitializedQuestion> Questions;
        public Testing(int idTest, SqlGateway _sqlGateway, int idUser)
        {
            this.sqlGateway = _sqlGateway;
            this.idTest = idTest;
            this.idUser = idUser;
            Questions = new List<InitializedQuestion>();
        }
    }
}

```

```

        GetQuestions();
        GetAnswers();
        GetCorrectAnswer();
    }
    /// <summary>
    /// Сохраняем в базу результаты тестирования
    /// </summary>
    /// <param name="correctCount">Количество правильных ответов</param>
    /// <param name="mark">Оценка</param>
    public void ReportTestSession(int correctCount, int mark)
    {
        sqlGateway.ReportTestSession(correctCount, mark, idUser,
idTest);
    }

    /// <summary>
    /// Получить правильные ответы
    /// </summary>
    private void GetCorrectAnswer()
    {
        for (int i = 0; i < Questions.Count; i++)
        {
            var answer = sql-
Gateway.GetCorrectAnswer(Questions[i].IdQuestion);

            Questions[i].CorrectAnswer = answer;
        }
    }
    /// <summary>
    /// Получить список ответов
    /// </summary>
    private void GetAnswers()
    {
        for (int i = 0; i < Questions.Count; i++)
        {
            var answers = sql-
Gateway.GetAnswers(Questions[i].IdQuestion);

            Questions[i].Answers = answers;
        }
    }
    /// <summary>
    /// Получить список вопросов
    /// </summary>
    private void GetQuestions()
    {
        var questions = sqlGateway.GetQuestions(idTest);

        foreach (var question in questions)
        {
            Questions.Add(new InitializedQuestion() { IdQuestion =
question.Key, Text = question.Value });
        }
    }
    /// <summary>
    /// Получить результаты тестирования
    /// </summary>
    /// <param name="answers">Список ответов на вопросы</param>
    /// <returns>Пару значений (количество правильных ответов, общая
оценка в баллах от 2 до 5)</returns>
    public Tuple<int, int> GetResults(List<int> answers)

```

```

{
    int correctCount = 0;

    float percent = 0.0f;
    //получаем количество правильных ответов
    for (int i = 0; i < Questions.Count; i++)
    {
        if (Questions[i].CorrectAnswer == answers[i])
            correctCount++;
    }
    //если таковых не обнаружено, возвращаем пару пустых значений
    if(correctCount == 0)
        return new Tuple<int, int>(correctCount, (int)percent);

    //рассчитываем процентп равильных ответов и оценку по нему
    percent = ((float)(correctCount / Questions.Count)) * 100;

    int mark = GetMark((int)percent);

    ReportTestSession(correctCount, mark);

    return new Tuple<int, int>(correctCount, mark);
}
/// <summary>
/// Получить оценку
/// </summary>
/// <param name="percent">Процент правильных ответов</param>
/// <returns>Оценка за тест</returns>
private int GetMark(int percent)
{
    if (percent >= 85)
        return 5;

    if (percent >= 75)
        return 4;

    if (percent >= 65)
        return 3;

    return 2;
}
}
/// <summary>
/// Класс для вопроса теста
/// </summary>
public class Question
{
    /// <summary>
    /// Номер вопроса в БД
    /// </summary>
    public int IdQuestion { get; set; }
    /// <summary>
    /// Номер правильного ответа
    /// </summary>
    public int CorrectAnswer { get; set; }

    /// <summary>
    /// Содержание вопроса
    /// </summary>
    public string Text { get; set; }
}

```

```

/// <summary>
/// Класс вопроса, который уже есть в БД
/// </summary>
public class InitializedQuestion : Question
{
    /// <summary>
    /// Словарь ответов
    /// </summary>
    public Dictionary<int, string> Answers { get; set; }
}
/// <summary>
/// Класс нового вопроса, которого ещё нет в БД
/// </summary>
public class NewQuestion: Question
{
    /// <summary>
    /// Номер теста
    /// </summary>
    public int IdTest { get; set; }

    public int CorrectAnswerIndex { get; set; }

    public List<string> Answers { get; set; }
}
/// <summary>
/// Класс для сессии тестирования
/// </summary>
public class TestSessions
{
    public string Surname { get; set; }

    public string Name { get; set; }

    public string LastName { get; set; }

    public DateTime Date { get; set; }

    public string TestName { get; set; }

    public int Mark { get; set; }
}
}

```

РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

Приложение Б

(обязательное)

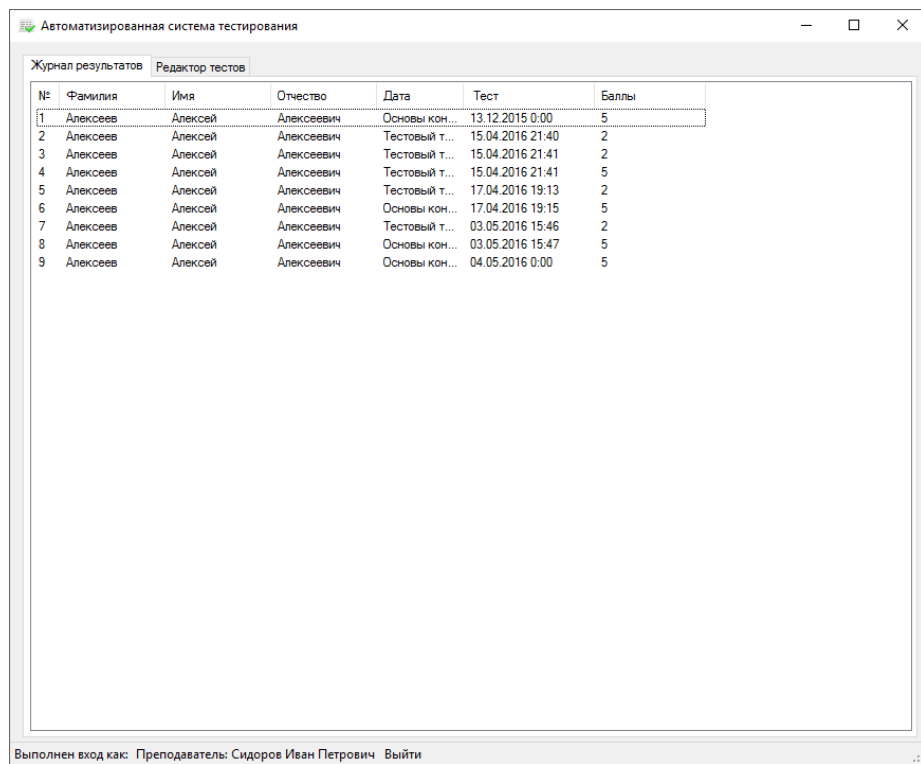


Рисунок Б.1 – Главное окно программы

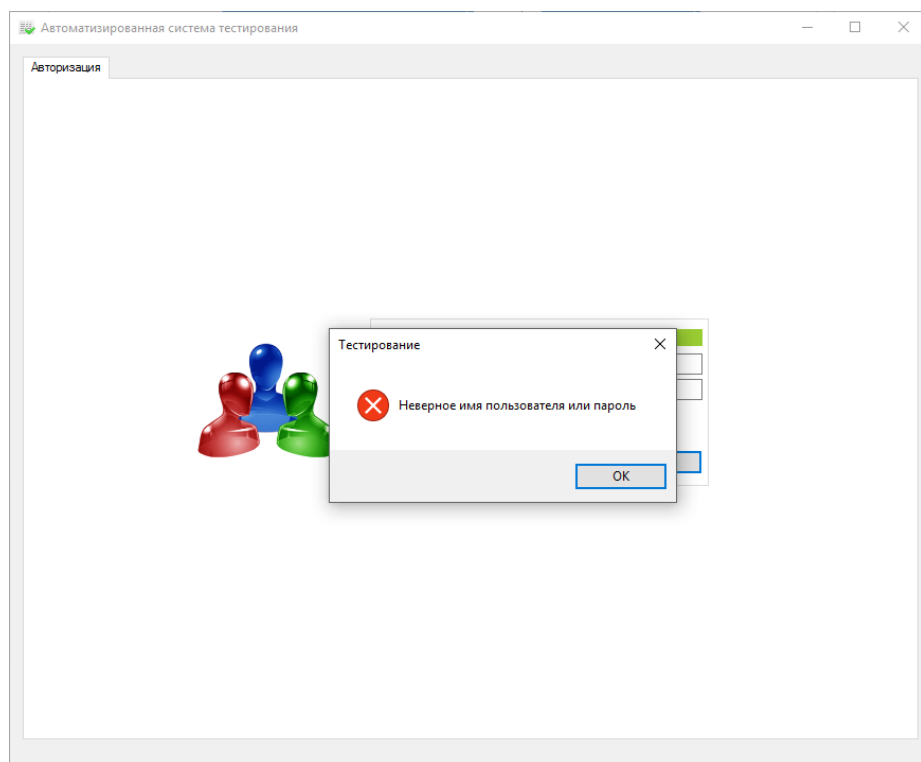


Рисунок Б.2 – Авторизация несуществующего пользователя

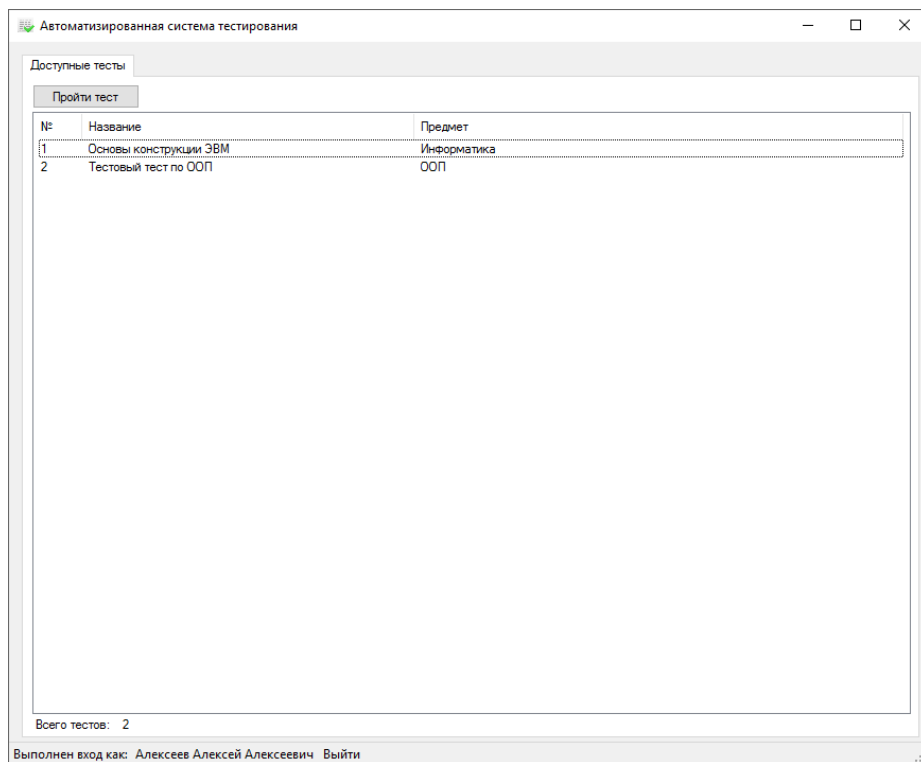


Рисунок Б.3 – Авторизация студента

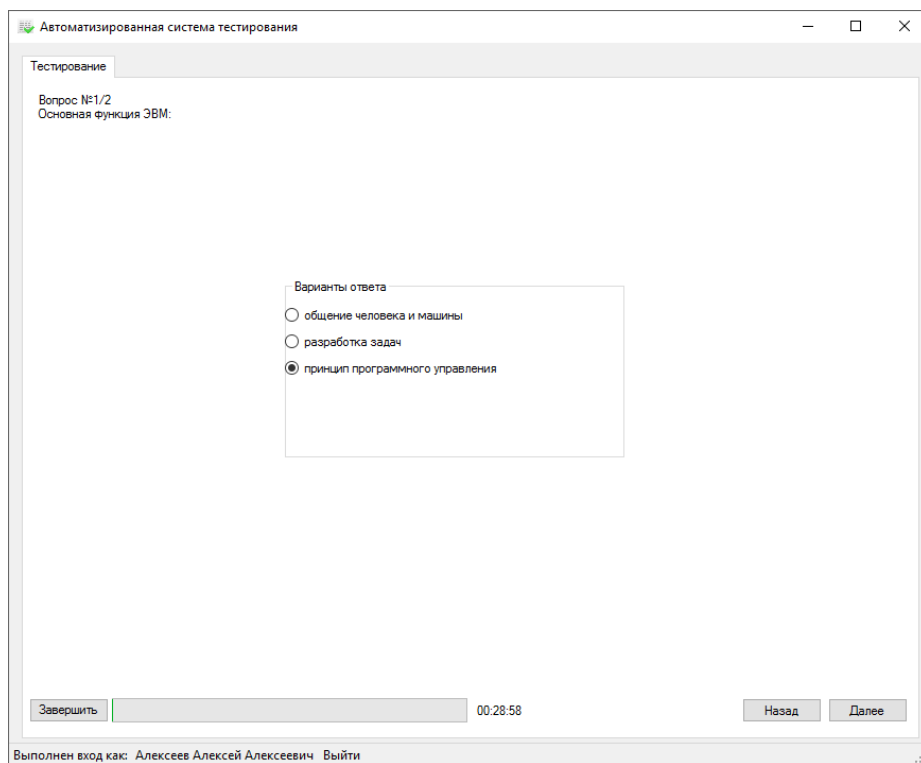


Рисунок Б.4 – Процесс тестирования

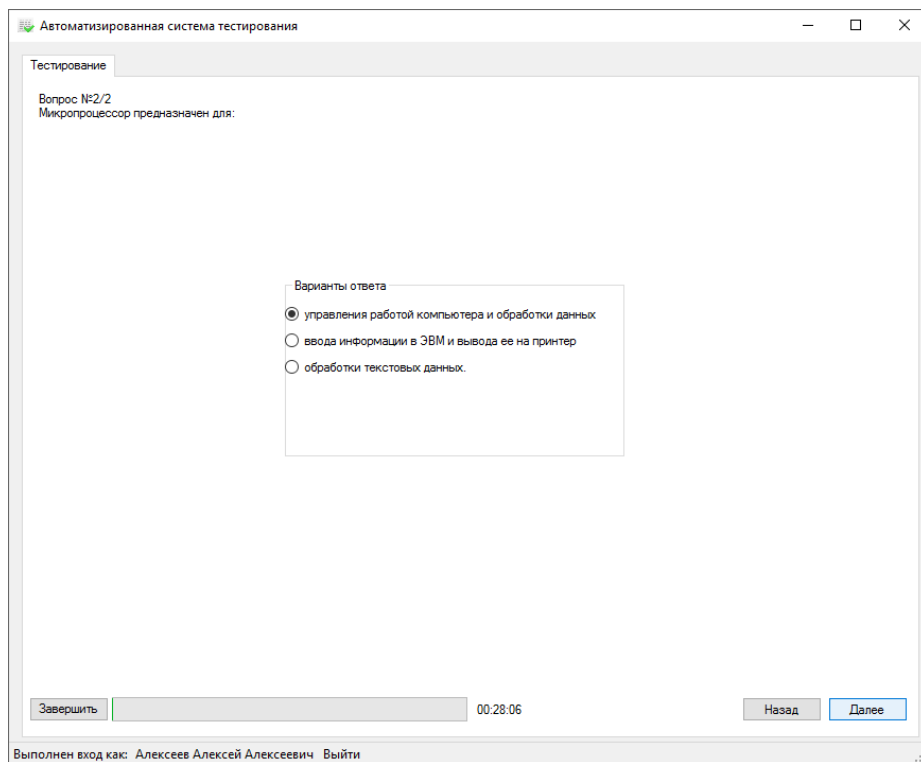


Рисунок Б.5 – Процесс тестирования

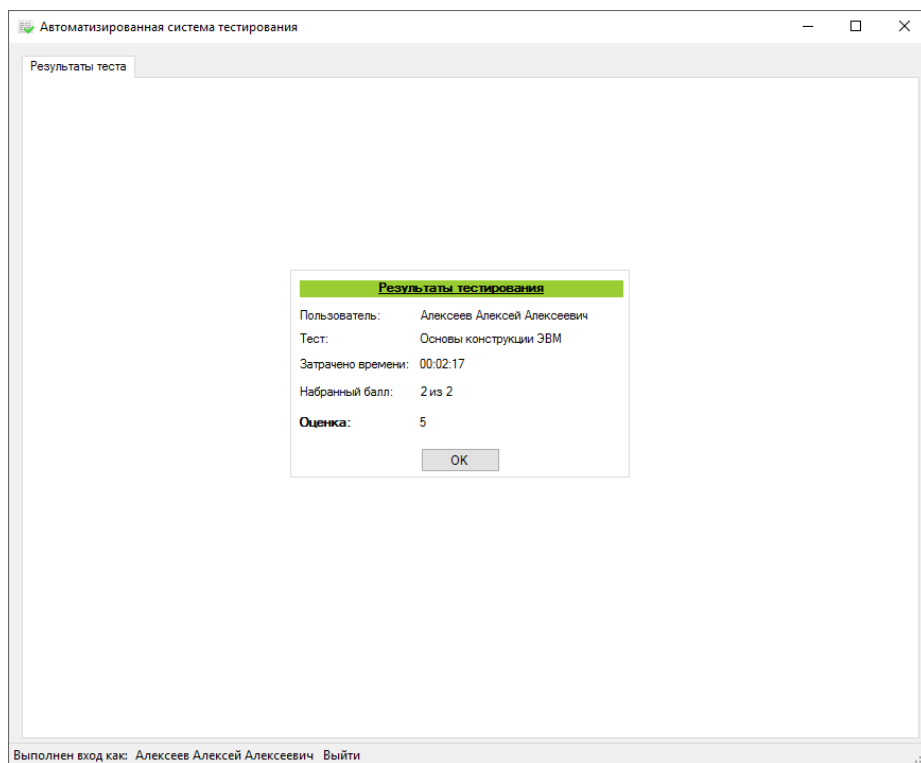


Рисунок Б.6 – Результаты текущего тестирования

Автоматизированная система тестирования

Журнал результатов Редактор тестов

№	Фамилия	Имя	Отчество	Дата	Тест	Баллы
1	Алексеев	Алексей	Алексеевич	Основы кон...	13.12.2015 0:00	5
2	Алексеев	Алексей	Алексеевич	Тестовый т...	15.04.2016 21:40	2
3	Алексеев	Алексей	Алексеевич	Тестовый т...	15.04.2016 21:41	2
4	Алексеев	Алексей	Алексеевич	Тестовый т...	15.04.2016 21:41	5
5	Алексеев	Алексей	Алексеевич	Тестовый т...	17.04.2016 19:13	2
6	Алексеев	Алексей	Алексеевич	Основы кон...	17.04.2016 19:15	5
7	Алексеев	Алексей	Алексеевич	Тестовый т...	03.05.2016 15:46	2
8	Алексеев	Алексей	Алексеевич	Основы кон...	03.05.2016 15:47	5
9	Алексеев	Алексей	Алексеевич	Основы кон...	04.05.2016 0:00	5
10	Алексеев	Алексей	Алексеевич	Основы кон...	05.05.2016 8:07	5

Выполнен вход как: Преподаватель: Сидоров Иван Петрович Выйти

Рисунок Б.7 – Сводные результаты тестирования знаний студентов

Автоматизированная система тестирования

Журнал результатов Редактор тестов

Добавить предмет Удалить предмет Добавить тест Удалить тест Добавить вопрос Удалить вопрос

Информатика
ООП
СУБД

Содержание вопроса

Варианты ответа

OK Добавить ответ

Выполнен вход как: Преподаватель: Сидоров Иван Петрович Выйти

Рисунок Б.8 – Результат добавления предмета

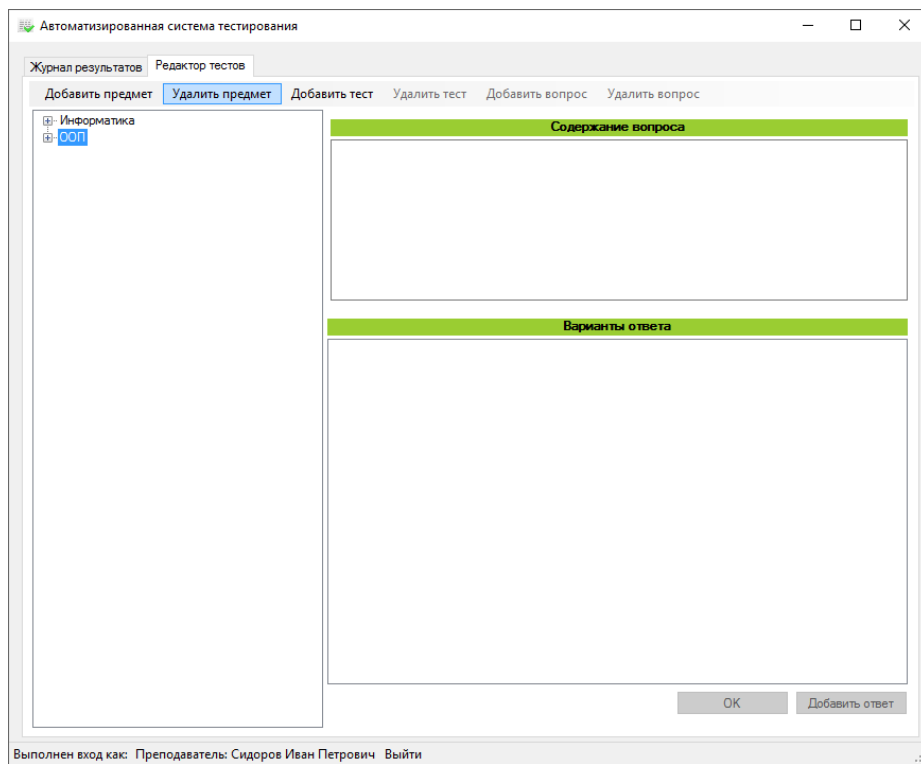


Рисунок Б.9 – Удаление предмета

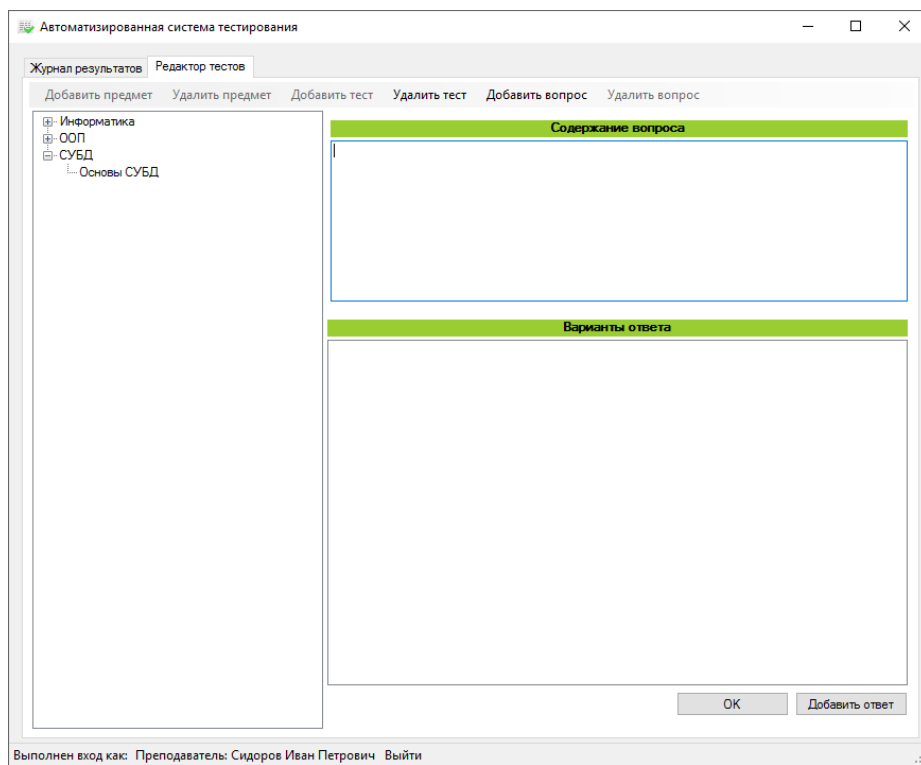


Рисунок Б.10 – Добавление теста

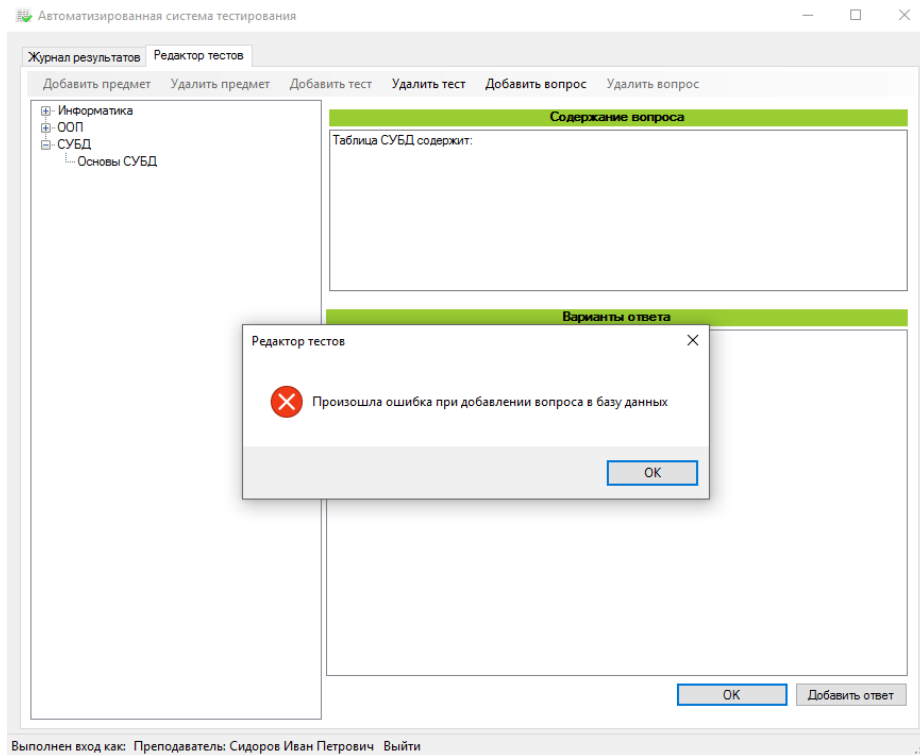


Рисунок Б.11 – Добавление вопроса без ответов

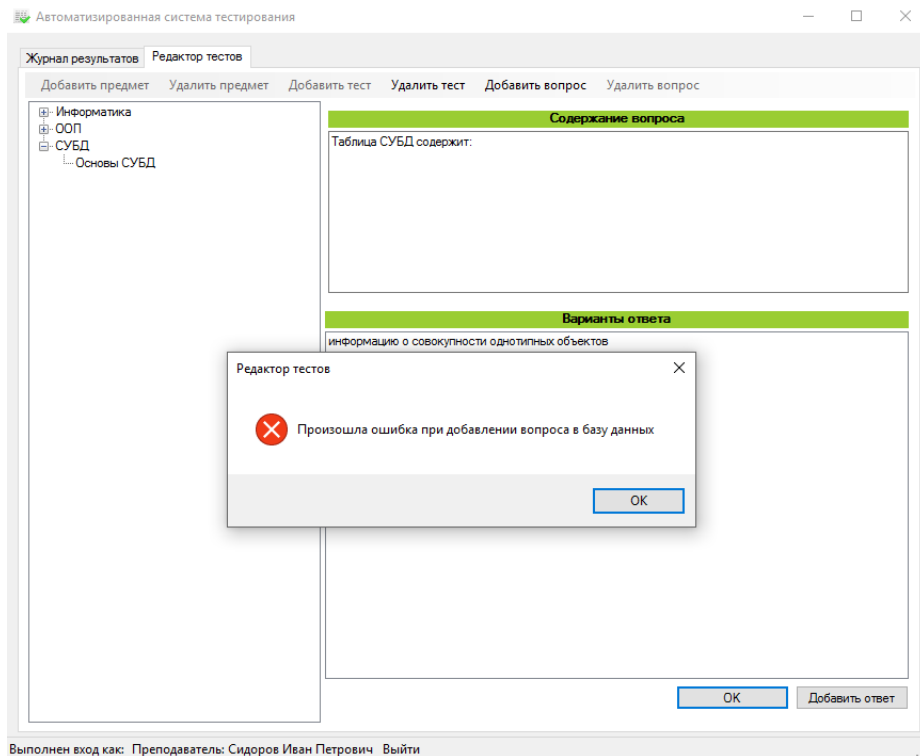


Рисунок Б.12 – Добавление вопроса без указания правильного ответа

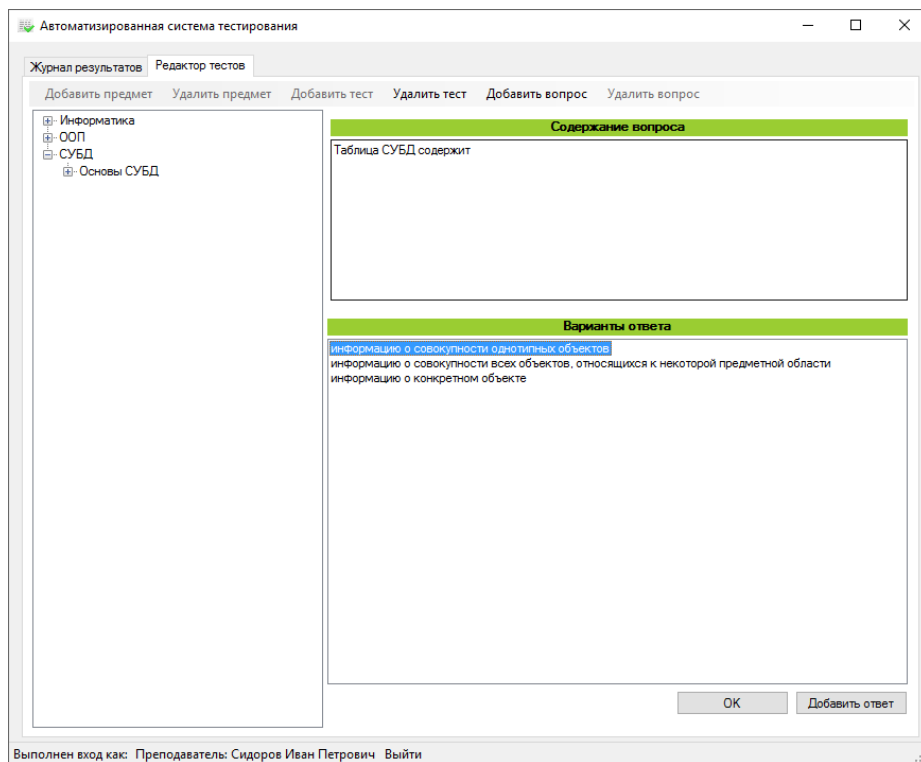


Рисунок Б.13 – Добавление вопроса с указанием правильного ответа

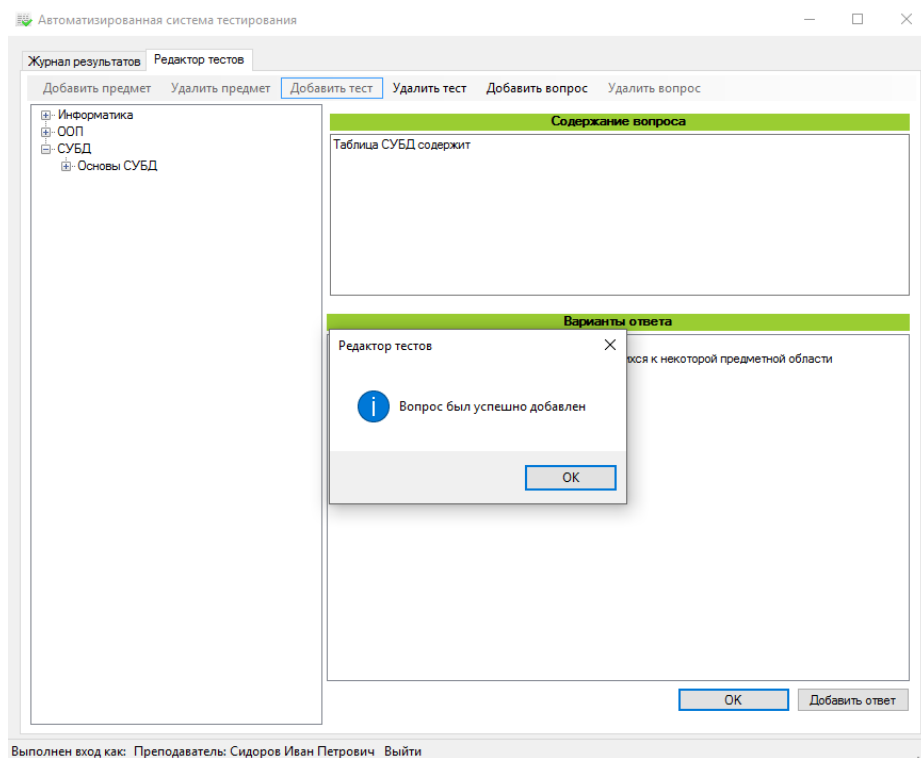


Рисунок Б.14 – Успешное добавление вопроса

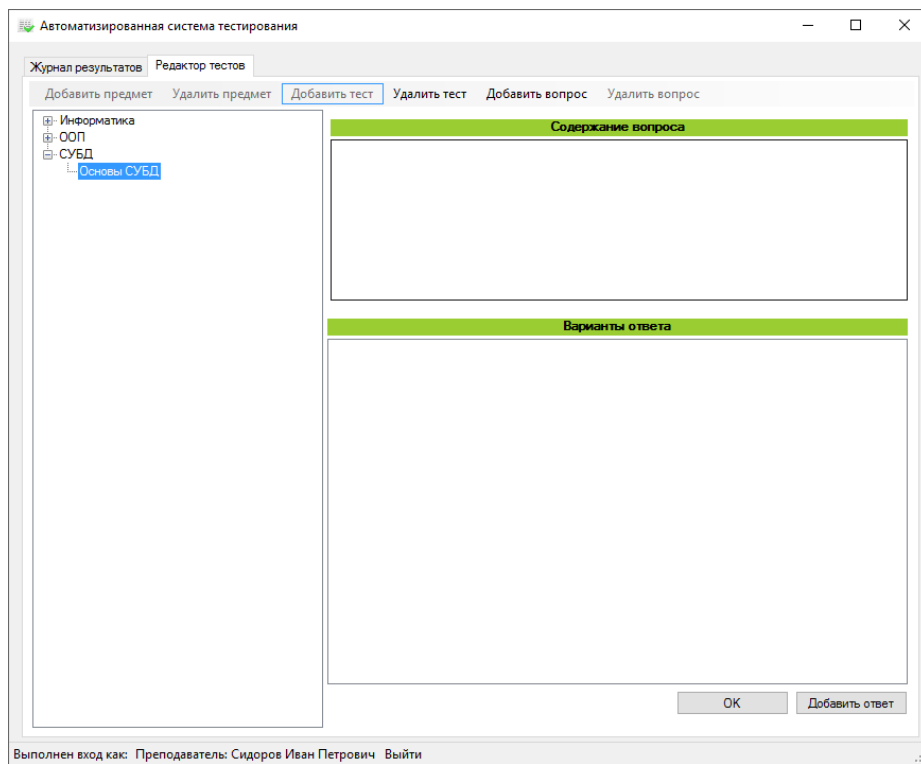


Рисунок Б.15 – Результат удаления вопроса

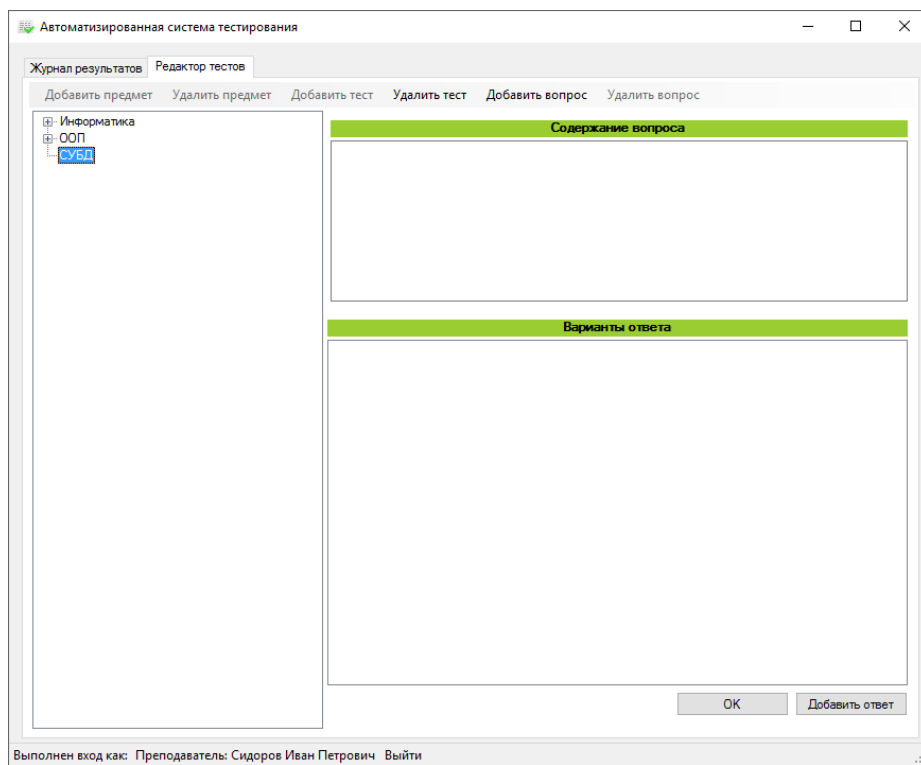


Рисунок Б.16 – Результат удаления теста

Автоматизированная система тестирования

Регистрация

Регистрация нового пользователя

Фамилия:

Имя:

Отчество:

☐ Преподаватель

Логин:


Пароль:

Повторите пароль:

Рисунок Б.17 – Окно регистрации

Автоматизированная система тестирования

Авторизация



Авторизация пользователя

Логин:

Пароль:

☐ Войти, как преподаватель

Вы впервые здесь? [Зарегистрируйтесь](#)

Рисунок Б.18 – Регистрация успешна

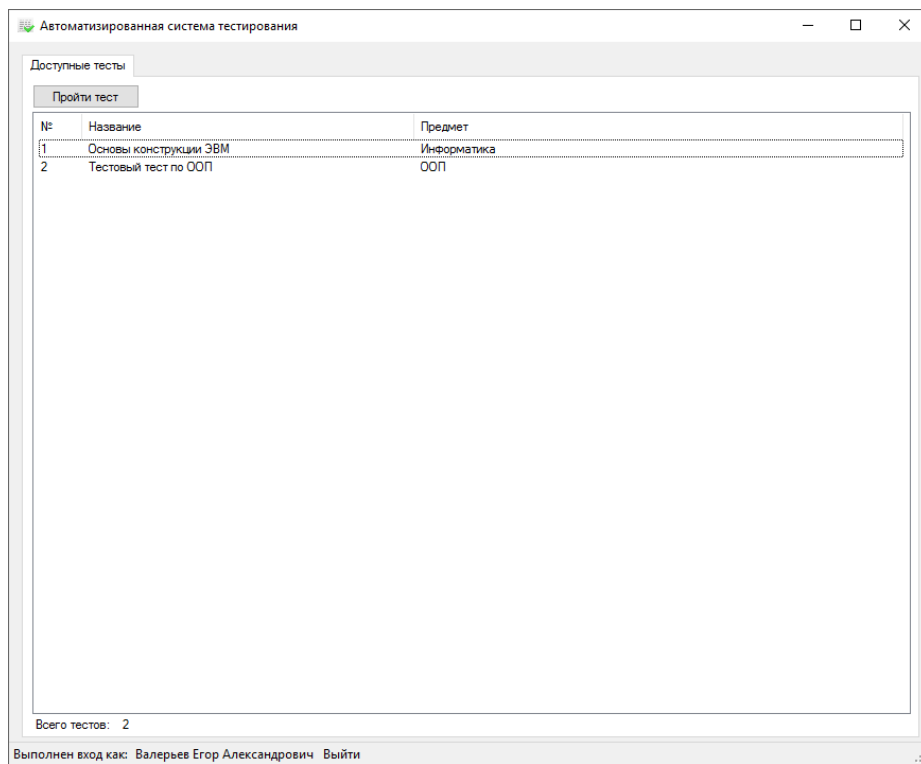


Рисунок Б.19 – Вход нового пользователя

ЛИСТИНГ ТЕСТОВОГО ПРОЕКТА

Приложение В

(обязательное)

В.1 Листинг на языке C# (Файл LogonTest.cs)

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Testing;
using Testing.Model;
using System.Data.SqlClient;

namespace UnitTestProject1
{
    [TestClass]
    public class LogonTests
    {
        public SqlGateway PrepareGateway()
        {
            SqlConnection globalConnection = new SqlConnection(Constants.ConnectionString);
            globalConnection.Open();

            return new SqlGateway(globalConnection);
        }

        [TestMethod]
        public void CheckUserTestOne()
        {
            var gateway = PrepareGateway();

            Logon logon = new Logon(gateway);

            var result = logon.CheckUser(String.Empty, String.Empty,
true);

            Assert.AreEqual(null, result);
        }

        [TestMethod]
        public void CheckUserTestTwo()
        {
            var gateway = PrepareGateway();

            Logon logon = new Logon(gateway);

            var result = logon.CheckUser(String.Empty, String.Empty,
false);

            Assert.AreEqual(null, result);
        }

        [TestMethod]
        public void CheckUserTestThree()
        {
            var gateway = PrepareGateway();

            Logon logon = new Logon(gateway);

            var result = logon.CheckUser("123", "123", true);

            Assert.IsNull(result);
        }

        [TestMethod]
        public void CheckUserTestFour()
        {

```

```

        var gateway = PrepareGateway();

        Logon logon = new Logon(gateway);

        var result = logon.CheckUser("123", "123", false);

        Assert.IsNotNull(result);
    }

    [TestMethod]
    public void CheckUserTestFive()
    {
        var gateway = PrepareGateway();

        Logon logon = new Logon(gateway);

        var result = logon.CheckUser("test", "test", true);

        Assert.IsNotNull(result);
    }

    [TestMethod]
    public void CheckUserTestSix()
    {
        var gateway = PrepareGateway();

        Logon logon = new Logon(gateway);

        var result = logon.CheckUser("test", "test", false);

        Assert.IsNull(result);
    }

    [TestMethod]
    public void TestAddNewUserOne()
    {
        var gateway = PrepareGateway();

        Logon logon = new Logon(gateway);

        var result = logon.AddNewUser(String.Empty, String.Empty, String.Empty, String.Empty, String.Empty, String.Empty, false);

        Assert.AreEqual(ErrorState.FieldsIsEmpty, result);
    }

    [TestMethod]
    public void TestAddNewUserTwo()
    {
        var gateway = PrepareGateway();

        Logon logon = new Logon(gateway);

        var result = logon.AddNewUser("Василий", "Васильев", "Витальевич", "test", "test", "test1", true);

        Assert.AreEqual(ErrorState.PasswordNotCorrectly, result);
    }

```

```
[TestMethod]
public void TestAddNewUserThree()
{
    var gateway = PrepareGateway();

    Logon logon = new Logon(gateway);

    var result = logon.AddNewUser("Василий", "Васильев", "Виталье-
вич", "test", "test", "test", true);

    Assert.AreEqual(ErrorState.UserIsExists, result);
}

[TestMethod]
public void TestAddNewUserFour()
{
    var gateway = PrepareGateway();

    Logon logon = new Logon(gateway);

    var result = logon.AddNewUser("Василий", "Васильев", "Виталье-
вич", "test2", "test2", "test2", true);

    Assert.AreEqual(ErrorState.Success, result);
}
}
```

ЛИСТИНГ СКРИПТА DDL

Приложение Г

(обязательное)

Г.1 Листинг скрипта создания таблиц базы данных на языке SQL

```
CREATE TABLE Answer
(
    ID_Answer          int IDENTITY ( 1,1 ) ,
    Text               varchar(250)  NOT NULL ,
    ID_Question        int   NOT NULL
)
go
```

```
ALTER TABLE Answer
    ADD  PRIMARY KEY (ID_Answer ASC)
go
```

```
CREATE TABLE CorrectAnswers
(
    ID_CorrectAnswers  int IDENTITY ( 1,1 ) ,
    ID_Answer          int   NOT NULL ,
    ID_Question        int   NOT NULL
)
go
```

```
ALTER TABLE CorrectAnswers
    ADD  PRIMARY KEY (ID_CorrectAnswers ASC)
go
```

```
CREATE TABLE Question
(
    ID_Question        int IDENTITY ( 1,1 ) ,
    Text               varchar(255)  NOT NULL ,
    ID_Test            int   NOT NULL
)
go
```

```
ALTER TABLE Question
    ADD  PRIMARY KEY (ID_Question ASC)
go
```

```
CREATE TABLE Student
(
    ID_Student         int IDENTITY ( 1,1 ) ,
    Name               varchar(20)   NOT NULL ,
    Surname            varchar(40)   NOT NULL ,
    LastName           varchar(20)   NOT NULL ,
    Login              varchar(20)   NOT NULL ,
    Password           varchar(20)   NOT NULL
)
go
```



```
ALTER TABLE Student
    ADD PRIMARY KEY (ID_Student ASC)
go
```

```
CREATE TABLE Subject
(
    ID_Subject          int IDENTITY ( 1,1 ) ,
    Name                varchar(50)  NOT NULL
)
go
```

```
ALTER TABLE Subject
    ADD PRIMARY KEY (ID_Subject ASC)
go
```

```
CREATE TABLE Teacher
(
    ID_Teacher          int IDENTITY ( 1,1 ) ,
    Name                varchar(20)  NOT NULL ,
    Surname              varchar(40)  NOT NULL ,
    LastName             varchar(20)  NOT NULL ,
    Login                varchar(20)  NOT NULL ,
    Password              varchar(20)  NOT NULL ,
    ID_Subject           int    NULL
)
go
```

```
ALTER TABLE Teacher
    ADD PRIMARY KEY (ID_Teacher ASC)
go
```

```
CREATE TABLE Test
(
    ID_Test              int IDENTITY ( 1,1 ) ,
    ID_Subject           int    NOT NULL ,
    Name                 varchar(30)  NOT NULL
)
go
```

```
ALTER TABLE Test
    ADD PRIMARY KEY (ID_Test ASC)
go
```

```
CREATE TABLE Testing
```

```
(
    ID_Testing          int IDENTITY ( 1,1 ) ,
    CountCorrectAnswer  int NOT NULL ,
    Date                datetime NOT NULL ,
    Mark                int NOT NULL ,
    ID_Student          int NOT NULL ,
    ID_Test             int NOT NULL
)
go
```

```
ALTER TABLE Testing
    ADD PRIMARY KEY (ID_Testing ASC)
go
```

```
ALTER TABLE Answer
    ADD FOREIGN KEY (ID_Question) REFERENCES Question(ID_Question)
        ON DELETE CASCADE
        ON UPDATE CASCADE
go
```

```
ALTER TABLE CorrectAnswers
    ADD FOREIGN KEY (ID_Answer) REFERENCES Answer(ID_Answer)
        ON DELETE CASCADE
        ON UPDATE CASCADE
go
```

```
ALTER TABLE CorrectAnswers
    ADD FOREIGN KEY (ID_Question) REFERENCES Question(ID_Question)
        ON DELETE CASCADE
        ON UPDATE CASCADE
go
```

```
ALTER TABLE Question
    ADD FOREIGN KEY (ID_Test) REFERENCES Test(ID_Test)
        ON DELETE CASCADE
        ON UPDATE CASCADE
go
```

```
ALTER TABLE Teacher
    ADD FOREIGN KEY (ID_Subject) REFERENCES Subject(ID_Subject)
        ON DELETE CASCADE
        ON UPDATE CASCADE
go
```

```
ALTER TABLE Test
```

```
ADD FOREIGN KEY (ID_Subject) REFERENCES Subject(ID_Subject)
ON DELETE CASCADE
ON UPDATE CASCADE
go
```

```
ALTER TABLE Testing
ADD FOREIGN KEY (ID_Student) REFERENCES Student(ID_Student)
ON DELETE CASCADE
ON UPDATE CASCADE
go
```

```
ALTER TABLE Testing
ADD FOREIGN KEY (ID_Test) REFERENCES Test(ID_Test)
ON DELETE CASCADE
ON UPDATE CASCADE
go
```