МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»
Институт математики, физики и информационных технологий

(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование кафедры)

09.03.03 Прикладная информатика
(код и наименование направления подготовки, специальности)

Прикладная информатика в социальной сфере

(направленность (профиль)/специализация)

БАКАЛАВРСКАЯ РАБОТА

на тему «Разработка системы тестирования с элементами геймификации для оценки уровня знаний школьников»

Студент Е.Е. Ульянина _____
(И.О. Фамилия) (личная подпись)

Руководитель Е.А. Ерофеева _____
(И.О. Фамилия) (личная подпись)

Консультанты И.Ю. Усатова _____
(И.О. Фамилия) (личная подпись)

Допустить к защите

Заведующий кафедрой к.т.н., доцент, А.В. Очеповский _____
(ученая степень, звание, И.О. Фамилия) (личная подпись)

«_____» _____ 20___ г.

Тольятти 2018

АННОТАЦИЯ

Тема: «Разработка системы тестирования с элементами геймификации для оценки уровня знаний школьников»

Ключевые слова: АВТОМАТИЗИРОВАННАЯ ИНФОРМАЦИОННАЯ СИСТЕМА, СИСТЕМА ТЕСТИРОВАНИЯ, ЛОГИЧЕСКАЯ МОДЕЛЬ, ФИЗИЧЕСКАЯ МОДЕЛЬ, БАЗА ДАННЫХ.

Целью выпускной квалификационной работы является разработка автоматизированной информационной системы тестирования с элементами геймификации для школьников.

Объектом исследования является система тестирования с элементами геймификации.

Предмет исследования - процесс автоматизации проведения тестирования учащихся.

В аналитической части произведен анализ организации «КАК ЕСТЬ», на основе структурного подхода разработана концептуальная модель «КАК ДОЛЖНО БЫТЬ». В качестве языка программирования выбран язык Java, средства разработки – NetBeans 8.2, база данных PostgreSQL и среда управления базой данных – pgAdmin III.На стадии логического проектирования на основе объектно –ориентированного подхода разработана логическая модель. С помощью методологии IDEF1X разработана физическая модель. Структура выпускной квалификационной работы представлена введением, тремя главами, заключением, списком литературы.

Во введении определены актуальность данной темы, цели и задачи, а также объект и предмет исследования. В основном разделе работы проведен анализ бизнес-процессов деятельности отдела компании для определения задач. В заключении, сделаны выводы о проделанной работе и подведен итог.

Результатом выполнения ВКР является разработанная система тестирования с элементами геймификации для оценки уровня знаний школьников. Работа включает: 98 страниц с приложениями, 32 рисунка, 6 таблиц, 31 источник.

ABSTRACT

This diploma paper deals with the development of testing system for schoolchildren.

The aim of the work is to develop testing system with elements of gaming in order to assess the level of knowledge of schoolchildren.

The object of the graduation work is the development of the testing system with elements of gaming in order to assess the level of knowledge of schoolchildren.

The subject of the graduation work is testing system with elements of gaming.

Computer games have much in common with gaming, the main principle of which is to ensure continuous feedback from the user.

Gaming is the application of gaming approaches for non-gaming processes in order to increase the involvement of participants in the solution of applied problems.

The special part of the project gives details on how to use game mechanics in training. It explains how to get awarding points for the successfully done tasks. The presence of competitive element is described the use of ratings allows schoolchildren to organize training on a higher level, pushing to new successes arguing intensely studying.

In conclusion we would like to stress that the developed testing system was based on the language of programming Java 1.8, also a free software development environment NetBeans 8.2, the PostgreSQL database, and the database management environment is pgAdmin III.

The work is of interest for narrow circle of readers, namely for schoolchildren who undergo routine testing without elements of the game.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В современном мире для достижения лучших результатов необходимо автоматизировать процесс работы путем внедрения электронно-вычислительных средств. В наши дни мы имеем огромный выбор различных технологий и программных средств для достижения поставленных целей в области автоматизации. Развитие компьютерных сетей и веб-приложений делают образование доступнее, а использование интерактивных сервисов позволяет осуществлять это интереснее и познавательнее, повышая мотивацию к обучению.

Актуальность бакалаврской работы обусловлена привлечением и повышением внимания школьников для улучшения их мотивации при похождении тестирования в школе.

Данная система тестирования должна заинтересовать учащихся, что будет способствовать закреплению и повышению уровня знаний.

Целью автоматизации проведения тестирования учащихся является:

- завлечение учащихся в ход проведения тестирования путем введения элементов геймификации с целью закрепления и повышения знаний;
- улучшение показателей обработки информации полученной путем проведения тестирования, повышение степени достоверности информации путем удаление человеческого фактора в ход проверки результатов тестирования.

Целью ВКР является разработка автоматизированной информационной системы тестирования с элементами геймификации для школьников.

Объектом исследования в данной работе является система тестирования с элементами геймификации.

Предмет исследования в данной работе - процесс автоматизации проведения тестирования учащихся.

Задачи бакалаврской работы:

- произвести анализ и сделать общую характеристику предметной области и объекта автоматизации;

- обосновать необходимость автоматизации процесса тестирования учащихся;
- выбрать технические и программные средства для реализации АИС;
- спроектировать структуру БД;
- выполнить проектирование интерфейса системы тестирования;
- выполнить программную реализацию информационной системы.

Пояснительная записка состоит из трех глав. В первой главе раскрываются все аспекты функционального моделирования предметно области. Описана технико-экономическая характеристика предметной области, выбрана технология концептуального моделирования, разработана и проанализирована модель бизнес-процесса «КАК ЕСТЬ». Также проведен анализ и сделано обоснование для введения автоматизации в процесс тестирования учащихся. На основании требования к данной разработке исследован рынок на предмет аналогов в данной предметной области. И как итог, была разработана модель бизнес-процесса «КАК ДОЛЖНО БЫТЬ».

Во второй главе произведено логическое проектирование АИС. Выбрана технология логического моделирования, приведено описание модели, разработаны необходимые диаграммы. Далее выбрана технология проектирования БД АИС и разработана концептуальная модель данных. Как итог, была разработана логическая модель данных АИС, а также сформированы требования к аппаратно-программному обеспечению АИС.

Третья глава посвящена практической части проекта. Была выбрана архитектура автоматизированной системы, выбрана технология разработки ПО. Обоснован выбор и выбрана сама система управления базой данных, которой стала PostgreSQL 9.6. Разработано программное обеспечение средствами языка программирования Java и среды разработки NetBeans. Приведено описание модулей проекта и функциональности автоматизированной системы. После чего было проведено модульное и системное тестирования всей системы.

Глава 1 Функциональное моделирование предметной области

1.1 Технико-экономическая характеристика предметной области

Объектом исследования является деятельность среднеобразовательного учебного заведения. А именно, это процесс проведения тестирования учащихся. Предмет исследования – процесс автоматизации проведения тестирования учащихся.

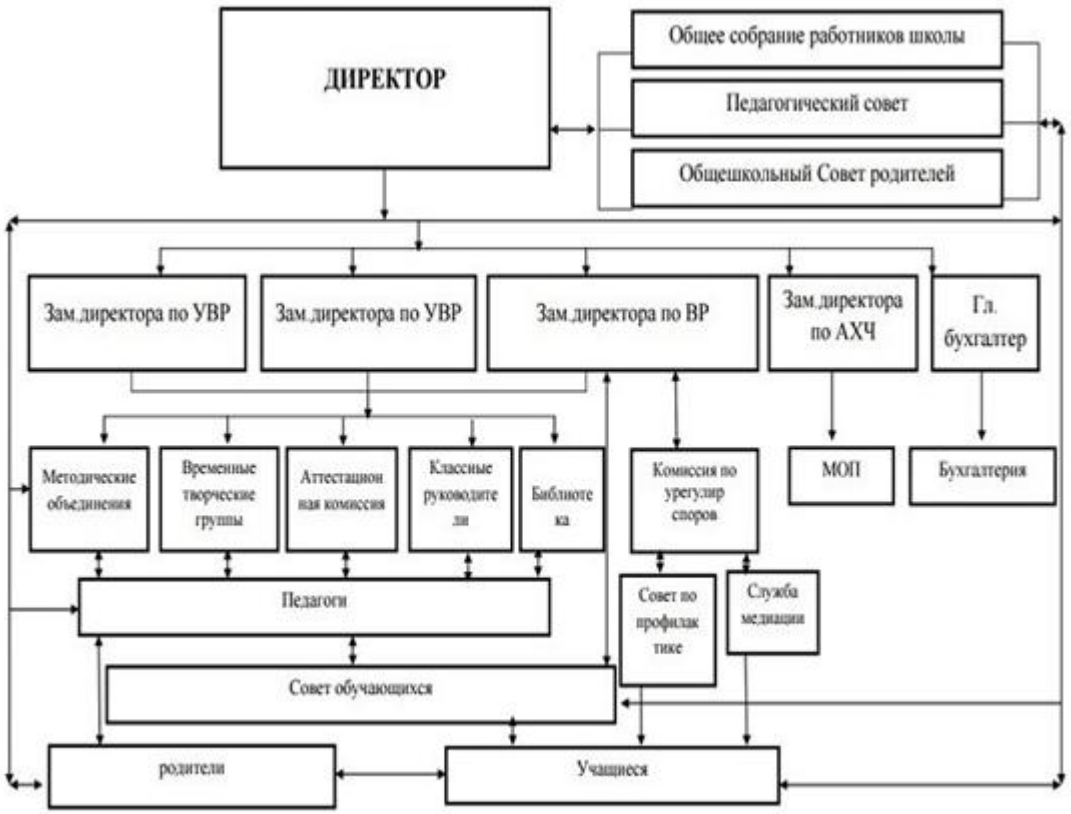

Рисунок 1.1 - Организационная структурная схема учебного заведения

В организационной структуре можно выделить несколько подразделений – это взаимодействие учебно-воспитательное подразделение, воспитательное и административно-хозяйственная часть.

Целью работы является процесс автоматизации путем введения автоматизированной информационной системы (АИС) для проведения тестирования учащихся по всем дисциплинам с элементами геймификации.

Со стороны преподавателя остается лишь контроль и анализ результатов тестирования, в виду чего значительно уменьшаются временные затраты на проведение и проверку тестов, а со стороны учащихся – станут более заинтересованными в проведении такого рода тестирования.

1.2 Концептуальное моделирование предметной области

1.2.1 Выбор технологии концептуального моделирования предметной области

Методология IDEF0 является эффективным средством анализа, проектирования, а также представления деловых процессов. С помощью нее можно сформировать целостную картина деятельности предприятия – от моделей организации работы в маленьких отделах до сложных иерархических структур. Модели дают основу для осмысления бизнес-процессов и оценки влияния тех или иных событий, а также описывают взаимодействие процессов и потоков информации в организации. Неэффективная или избыточная деятельность может быть легко выявлена и, следовательно, усовершенствована, изменена или устранена в соответствии с общими целями организации [9]. На всех стадиях проектирования АИС в данной задаче целесообразно будет использовать технологию, основанную на объектно-ориентированном подходе. Для того, чтобы создавать диаграммы, использовались возможности программы – Ramus. Программа является кроссплатформенной системой моделирования, а также анализа бизнес-процессов.

1.2.3 Разработка и анализ модели бизнес-процесса «КАК ЕСТЬ»

Функциональная модель бизнес-процесса «КАК ЕСТЬ» представлена и составлена в виде контекстной диаграммы с помощью методологии IDEF0, приведенная на рисунке 1.2. Процесс тестирования «КАК ЕСТЬ» описывается в диаграмме IDEF0, на входе здесь находится бланк тестирования, который раздается ученикамдля заполнения. Управляющими механизмами при этом

является нормативно-правовые акты Минобрнауки и руководство по проведению тестирования. Механизмы существующей системы тестирования включают в себя ученика и учителя. На выходе из функционального блока, который описывает систему тестирования, находятся результаты тестирования и отчет о тестировании, которые используются для составления рейтингов и поощрения лучших учеников.

Рисунок 1.2 - Контекстная диаграмма «КАК ЕСТЬ» процесса «Тестирование учеников»

После моделирования контекстной диаграммы необходимо провести ее декомпозицию. На рисунке 1.3 изображена модель декомпозиции бизнес-процессов в системе тестирования «КАК ЕСТЬ».

Весь процесс тестирования начинается с составления вопросов для тестирования, управляющим элементов при этом являются нормативно-правовые акты Минобрнауки, на выходе из данного блока находятся вопросы для тестов, из которых формируются тесты, при этом также учитываются нормы актов Минобрнауки.

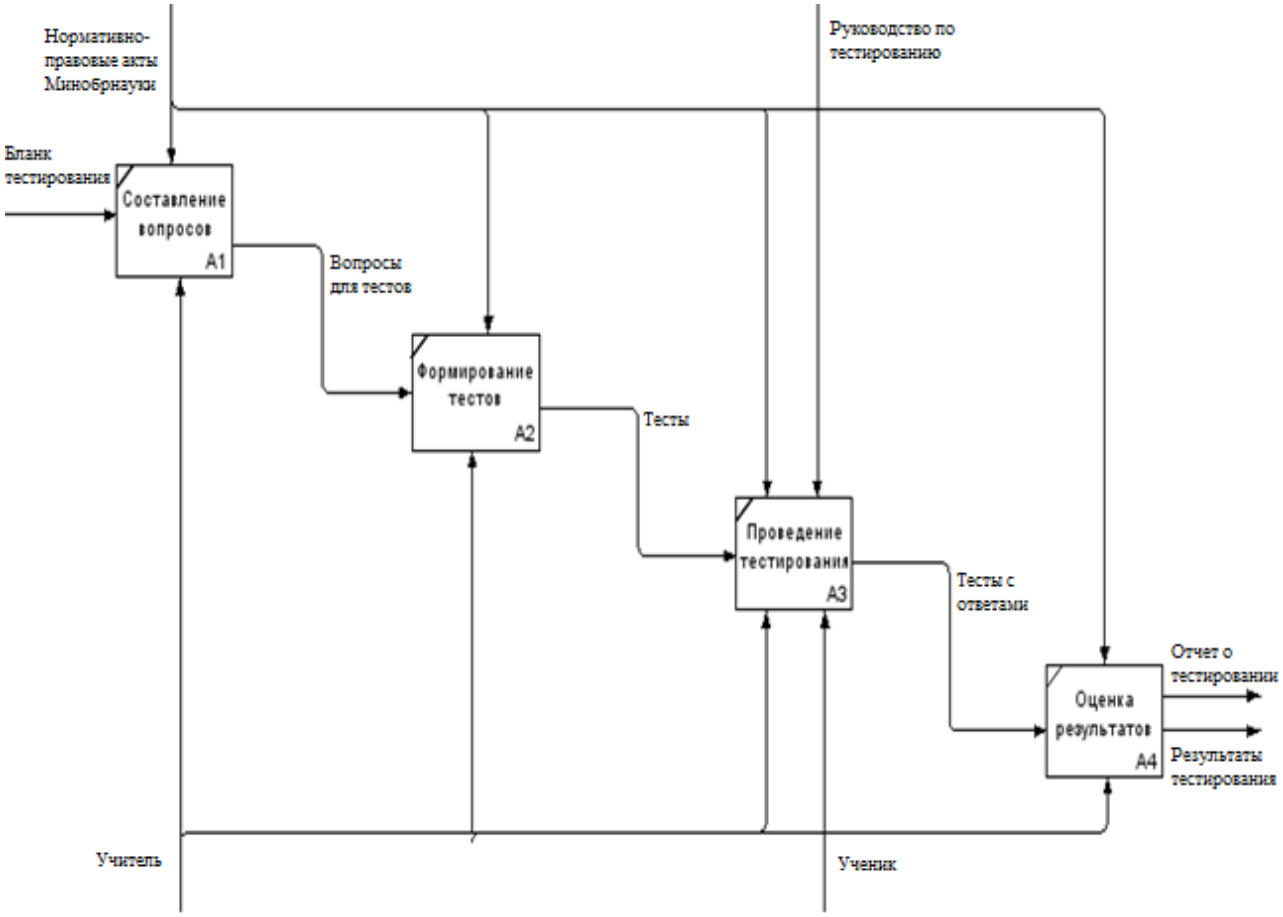

Рисунок 1.3 – Декомпозиция процесса «Тестирование учеников»

Процесс проведения тестов управляется нормативно-правовыми актами Минобрнауки и руководством по тестированию, на выходе из этого процесса находятся тесты с ответами, по результатам которых осуществляется оценка результатов, на основании которой формируется отчет о тестировании и выводится результат тестирования, для определения лучших учеников, формирования рейтинга.

Представленные на рис. 1.2 и 1.3 процессы описывают деятельность учителя (администратора системы тестирования). На рис. 1.4 показана модель бизнес-процессов для учеников.

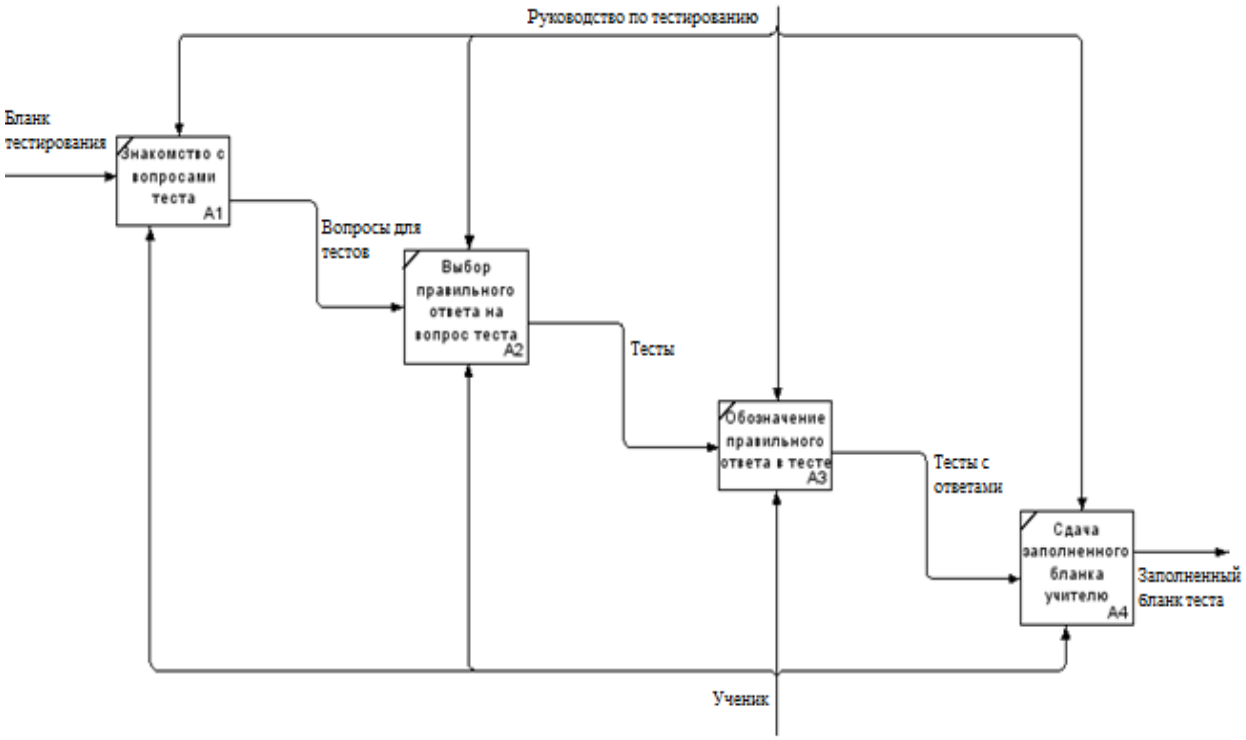

Рисунок 1.4 – Декомпозиция процесса «Проведение тестирования»

На рисунке 1.4 представлена декомпозиция процесса тестирования ученика, и включает знакомство ученика с вопросами, выбор правильного ответа, обозначение правильного ответа в анкете, сдачу заполненной формы учителю. Управляющим элементом при этом является руководство пользователя. На выходе процесса находится заполненный учеником бланк тестирования.

1.2.3 Обоснование необходимости автоматизированного варианта решения и формирование требований к проектируемой информационной системе

Как уже упоминалось выше, для реализации механизмов управления существующим бизнес-процессом не используются никакая автоматизированная система управления. Можно с уверенностью сказать, что для решения задач по формированию отчетов о результатах тестирования возникает потребность в обработке информации, полученной вследствие прохождения тестирования. На текущий момент времени вся полученная

информация хранится в бумажном виде. Для обработки этой информации требуется затратить значительное время. Также во время обработки информации не исключается занесения ошибка в виду человеческого фактора. От всего этого можно уйти путем внедрения автоматизации этого процесса.

Благодаря автоматизированному управлению системой тестирования учащихся можно увеличить эффективность обучения во многих направлениях:

- снижение трудоёмкости обработки информации;
- высокая достоверность результатов тестирования;
- совершенство сбора и регистрации исходной информации;
- легкость в управлении процессом;
- более высокая заинтересованность учащихся в образовательном процессе (имеется в виду – сам факт проведения тестирования с использованием ЭВМ, а также функция геймификации).

Также приведем критерии анализа программ для тестирования:

Для пользователя:

- регистрация;
- вход;
- выбор теста;
- результаты;

Для учителя:

- вход;
- смена пароля;
- список вопросов;
- скрытие вопроса;
- удаление вопроса;
- добавление вопроса;
- добавление категории;
- удаление категории;
- результаты;
- настройки количества отображаемых вопросов;

- настройки количества времени на тест;

- начисление баллов.

Не один из приведенных выше факторов не несет в себе отрицательного (негативного) эффекта после ввода автоматизированный системы тестирования учащихся. Поэтому целесообразность введения данной системы в средней общеобразовательной школе оправдана.

1.3 Анализ существующих разработок на предмет соответствия сформулированным требованиям

1.3.1 Определение критериев анализа

Существует множество критериев, по которым можно провести анализ. Для такого учреждения как средняя школа, имеет смысл использовать программное обеспечение, распространяемое на бесплатной основе. Чтобы внести что-то интересное в ПО, необходимо найти такое, чтобы было чем увлечь учащихся в ходе работы с приложением. К таким критериям можно отнести – простоту использования программного продукта.

Выделим самые важные критерии анализа программного обеспечения:

- стоимость ПО;

- простота реализации интерфейса;

- геймификация.

1.3.2 Сравнительная характеристика существующих разработок

В наше время в мире есть много IT-компаний, которые занимаются разработкой в сфере проведения тестирования.

Каждый предлагаемый продукт имеет свои плюсы и минусы. Познакомимся с несколькими из такого рода программ и прибегнем к их оценке по выделенным критериям. Чтобы сравнить проектируемую систему с другими программами необходимо провести анализ аналогичных средств. В таблице 1.1 приведена сравнительная характеристика функций известных ИТ-решение системы тестирования.

MyTestXPro – это система программ для создания и проведения компьютерного тестирования, сбора и анализа их результатов [22]. Программа MyTestX позволяет проводить не только тестирование, но и экзамены в школах, высших учебных заведениях. Также в данном программном обеспечении есть функции проведения аттестаций и сертификаций сотрудников предприятий и организаций. Можно смело сказать, что MyTestX это – программный комплекс (тестирование, аттестация, сертификация, редактор тестов и журнал результатов) от создания тестирования и его проведения, сбора и анализа результатов, до получения результатов по заданной шкале. Данная программа имеет удобный и понятный интерфейс, не требуется много времени для ее освоения.

tTester – с помощью этой программы возможны тестирование студентов, школьников, аттестация персонала, тестирование кандидатов при приеме на работу. tTester входит в состав SunRav TestOfficePro, программы для создания тестов, составления отчетов по результатам тестирования. Программа проста в эксплуатации, легко осваивается пользователями с любым уровнем компьютерной подготовки. Используя профессиональный тестовый материал, можно быть уверенными в получении достоверной информации о степени знаний и навыков школьников [24].

Основные возможности:

- регистрация пользователей;

- запуск внешних программ;

- командная строка (настраивать ярлыки, использовать ссылки из различных документов и т.д.);

- информация по ходу тестирования;

- процесс тестирования (запрет выхода из программы, отключение доступа к рабочему столу во время тестирования, настройка программы на закрытия после окончания тестов);

- безопасность (для изменения параметров программы нужно знать пароль);

- многоязычный интерфейс.

Таблица 1.1 – Сравнительный анализ программ для тестирования

Критерии анализа	Система				
	MyTest XPro	Тест-контроль	tTester	Мастер-Тест	Let'stest
1	2	3	4	5	6
Для пользователя					
Регистрация	+	+	+	+	+
Вход	+	+	+	+	+
Результаты	+	+	+	+	+
Для учителя (администратора)					
Вход	+	+	+	+	+
Смена пароля	+	+	+	+	+
Список вопросов	+	+	+	+	+
Скрытие вопроса	+	+	+	+	+
Удаление вопроса	+	+	+	+	+
Добавление вопроса	+	+	+	+	+
Добавление категории	+	+	+	+	+
Удаление категории	+	+	+	+	+
Результаты	+	+	+	+	+
Настройки количества отображаемых вопросов	-	-	-	-	-
Настройки количества времени на тест	-	-	-	-	-
Начисление баллов	-		-	-	+

Исходя из описания этих программ видно, что в данном сегменте рынка много подходящего ПО для поставленной задачи. Но, нужно заметить, что данное ПО является платным, что не удовлетворяет одному из поставленных критериев. Также стоит отметить, что одним из критериев было внедрение геймификации для заинтересованности учащихся, который отсутствует в этих системах. Не стоит забывать о том, что все программное обеспечение, которое приобретается у сторонних разработчиков идет без исходного кода, вследствие чего нет возможности внести какие-либо изменения или дополнения.

1.4 Постановка задачи на разработку автоматизированной информационной системы тестирования с элементами геймификации для школьников

Постановка задачи базируется на изучении и анализе функционировании объекта автоматизации. Постановка задачи является основной частью для дальнейшей разработки технического, программного и информационного обеспечения автоматизированной информационной системы.

Целью автоматизации можно назвать завлечение учащихся в ход проведения тестирования путем введения элементов геймификации, улучшение показателей обработки информации, полученной путем проведения тестирования, а также повышение степени достоверности информации путем удаление человеческого фактора в ход проверки результатов тестирования.

Назначение разработанного ПО тестирования учеников:

1. Удобная подготовки вопросов для тестирования.

2. Сохранение данных по системе тестирования в базе данных.

3. Сохранение данных пользователей.

4. Автоматизация проведения тестирования.

5. Завлечение учащихся путем внедрения такой функции как – геймификация (организация своего рода соревнования).

6. Сохранение данных о результатах тестирования.

7. Автоматизированный подсчет результатов тестирования.

В рамках данном проекте ставятся такие задачи:

В рамках данного проекта ставятся такие задачи:

- произвести анализ и сделать общую характеристику предметной области и объекта автоматизации;

- обосновать необходимость автоматизации;

- выбрать технические и программные средства;

- минимизация расходов на разработку системы;

- спроектировать структуру БД;

- выполнить проектирование интерфейса;

- выполнить программную реализацию информационной системы.

Так как одной из задач стоит минимизация расходов на разработку автоматизированной информационной системы, то необходимо понимать, что требования к архитектуре и реализации АИС должны сводиться к выбору технологий, находящихся в свободном доступе. Положительным моментом проектирования данной системы является то, что ее можно впоследствии можно будет внедрить не только в одном учебном заведении, но и в других учебных заведениях (среднетехнических, высших учебных заведениях).

1.5 Разработка модели бизнес-процесса «КАК ДОЛЖНО БЫТЬ»

В ходе проведенного анализа в предыдущих разделах необходимо построить модель «КАК ДОЛЖНО БЫТЬ». Для совершенствования системы тестирования необходимо разработать новую модель бизнес-процессов.

На рис. 1.6 и 1.7 показаны обновленные модели бизнес-процессов системы тестирования, которые показывают внедрение системы тестирования учеников.

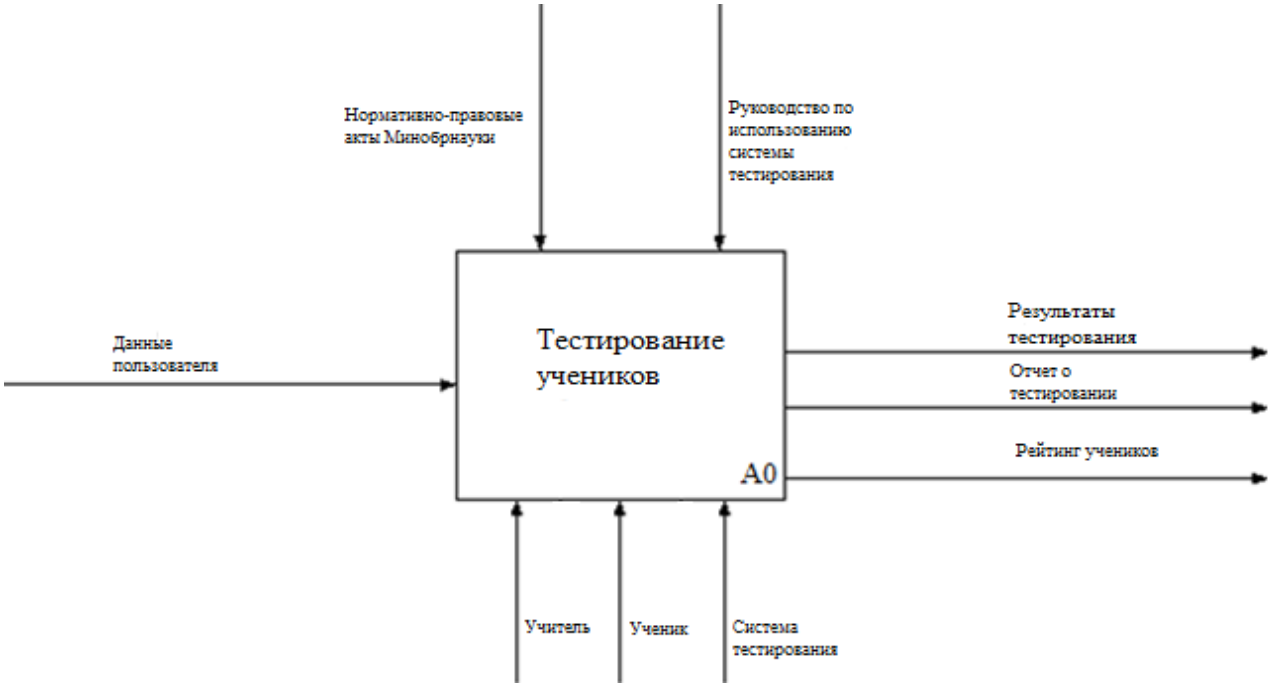

Рисунок 1.6 - Контекстная диаграмма «КАК ДОЛЖНО БЫТЬ» процесса

«Тестирование учеников»

Новая бизнес-модель определяет использование автоматизированной информационной системы, которая предназначена для автоматизации процесса тестирования, от создания тестов, до проведения самого тестирования и оценки его результатов.

С помощью системы тестирования учителю предоставляется автоматизированная возможность создавать вопросы для тестирования, которые сохраняются в базу данных, после этого, на базе созданных вопросов уже формируются тесты.

Проведение тестов определяет использование персональных компьютеров, на которых ученики могут получить доступ к бланкам тестов и пройти тестирование.

После этого происходит оценка результатов и формирование рейтингов.

По оценке результатов, создается отчет о тестировании и отчет с результатами тестирования.

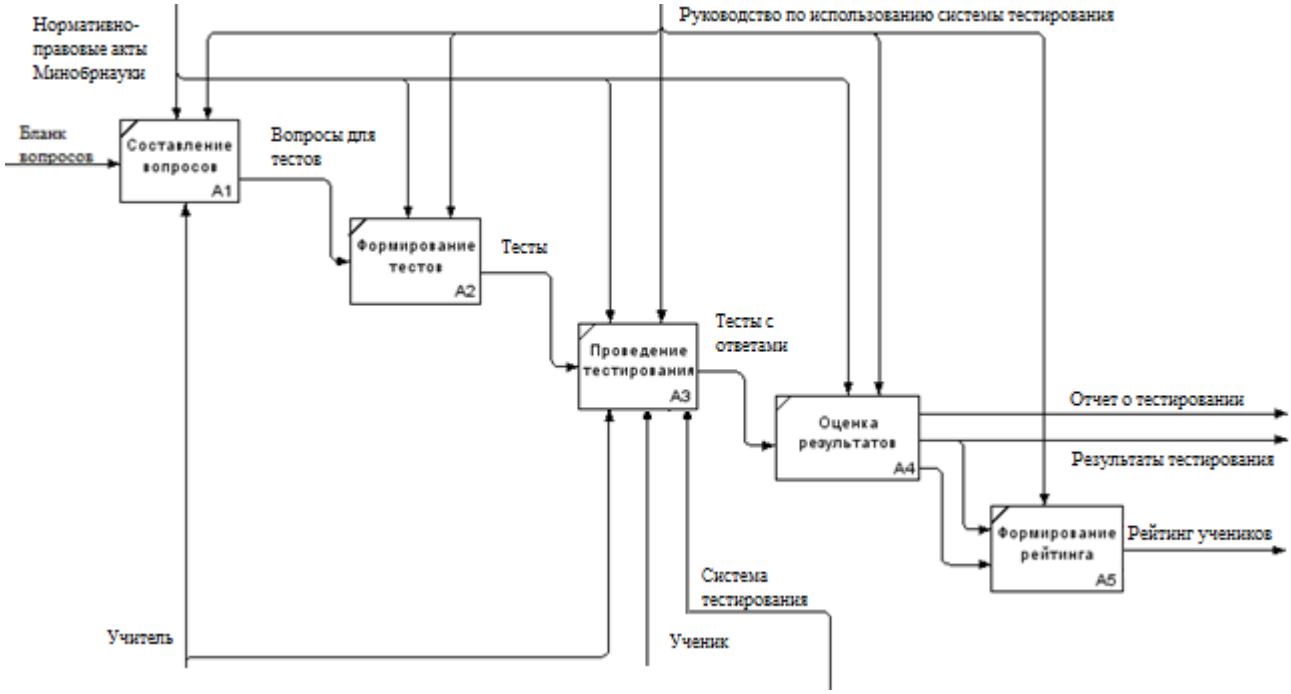

Рисунок 1.7 - Декомпозиции процесса «Тестирование учеников»

Представленные на рис. 1.6 и 1.7 процессы описывают деятельность учителя (администратора) обновленной модели системы тестирования.

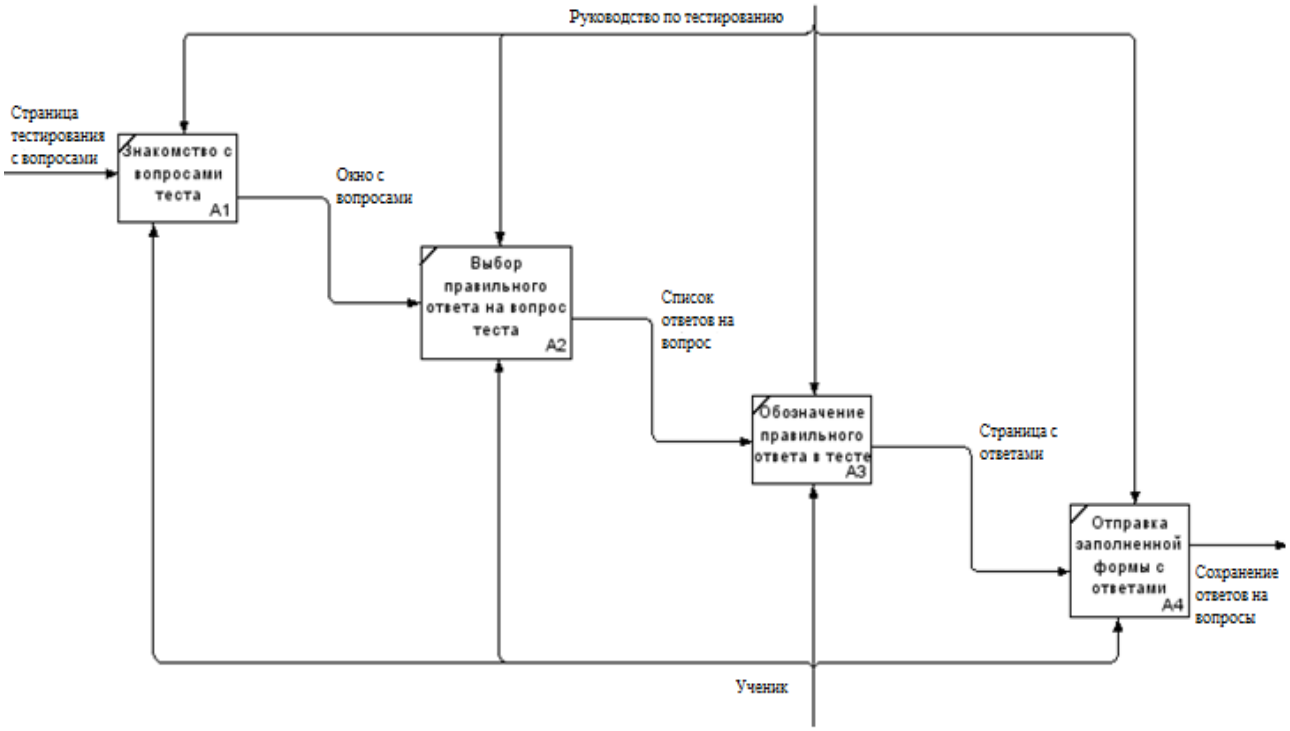

Рисунок 1.8 - Декомпозиции процесса «Проведение тестирования»

Рис. 1.8 показывает декомпозицию процесса тестирования ученика, и включает знакомство ученика с вопросами, выбор правильного ответа, обозначение правильного ответа в анкете, отправку заполненной формы с ответами. Управляющим элементом при этом является руководство пользователя.

Выводы по главе 1

Анализ реального использующего бизнес-процесса показал низкую производительность работы с документами. А именно, вся работа, которую предстоит автоматизировать происходит в ручном режиме, который может потянуть за собой ошибки ввиду человеческого фактора. Внедрение автоматизации в данный процесс внесет существенный вклад так как существенно увеличится скорость обработки данных с минимальными рисками на внесение ошибки в итоговые результаты.

Глава 2. Логическое проектирование автоматизированной информационной системы тестирования с элементами геймификации для школьников

2.1 Выбор технологии логического моделирования автоматизированной информационной системы тестирования с элементами геймификации для школьников

Логическое проектирование автоматизированной информационной системы – это процесс конструирования общей информационной модели компании на основе отдельных моделей данных пользователей, которая является независимой от особенностей реально используемой СУБД и других физических условий.

Построение логической модели данных отражает представление отдельного пользователя о предметной области приложения, и включает в себя проверку полученной модели с помощью методов нормализации. Также построение логической модели дает наглядный пример для последующего физического проектирования БД.

Для автоматизированной поддержки всех этапов разработки АИС используются CASE-средства (CASE - Computer Aided System/Software Engineering) [18]. Применение CASE-средств целесообразно в проектах любой сложности. Но в основном применения автоматизированных средств определяется возможностью концентрации сложности на начальных этапах разработки при относительно невысокой сложности и трудоемкости последующих этапов. В больших проектах благодаря применению CASE-средств можно свести уровень системных ошибок к минимальному значению, что способствует снижению общего объема разработки, как итог, уменьшаются затрат на разработку проекта.

Обобщим вышесказанное и приведем преимущества использования CASE-средств при разработке информационных систем:

- сокращение сроков и затрат на разработку ИС за счет автоматизации операций проектирования;

- повышение качества самого проекта как следствие задействования современных подходом и методов проектирования;

- обеспечение согласованности и полноты документации проекта;

- есть возможность использовать существующие наработки в других проектах.

В данный момент одним из популярных CASE-средств для проектирования являются ERwin, а также RationalRose.

ERwin – CASE-средство для проектирования баз данных, которое позволяет создавать, документировать и сопровождать базы данных, хранилища и витрины данных [20]. Данное средство имеет большой и удобный функционал для проектирования, что позволило хорошо зарекомендовать себя.

RationalRose представляет собой CASE средство проектирования и разработки информационных систем и программного обеспечения для управления предприятиями. Как и другие CASE средства (ARIS, BPwin, ERwin) его можно применять для анализа и моделирования бизнес процессов. Первая версия этого продукта была выпущена компанией Rational Software. В дальнейшем Rational Rose был куплен IBM. Принципиальное отличие Rational Rose от других средств заключается в объектно-ориентированном подходе. Графические модели, создаваемые с помощью этого средства, основаны на объектно-ориентированных принципах и языке UML (Unified Modeling Language). Инструменты моделирования Rational Rose позволяют разработчикам создавать целостную архитектуру процессов предприятия, сохраняя все взаимосвязи и управляющие воздействия между различными уровнями иерархии [23].

2.2 Логическая модель автоматизированной информационной системы тестирования с элементами геймификации для школьников и ее описание.

Необходимо разработать различные диаграммы, которые впоследствии станут основой создания логической модели автоматизированной информационной системы.

UML (англ. Unified Modeling Language - унифицированный язык моделирования) - язык графического описания для объектного моделирования в области разработки программного обеспечения, моделирования бизнес-процессов, системного проектирования и отображения организационных структур [25].

Язык UML является графическим языком. Данный стандарт применяется для создания абстрактной модели разрабатываемой системы, которую называют UML-моделью. Этот язык моделирования не относится к языкам программирования, но на основании разработанных моделей при помощи UML существует возможность генерации кода. Рассмотрим диаграммы, которыми в последствии воспользуемся для создания логической модели.

Диаграмма прецедентов (диаграмма вариантов использования) в UML - диаграмма, отражающая отношения между актёрами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне [16].

На рисунке 2.1 приведена диаграмма вариантов использования. Данная диаграмма позволяет описать работу системы на концептуальном уровне. На ней изображена связь актеров (в частности – преподаватели и учащиеся) и их отношения с прецедентами.

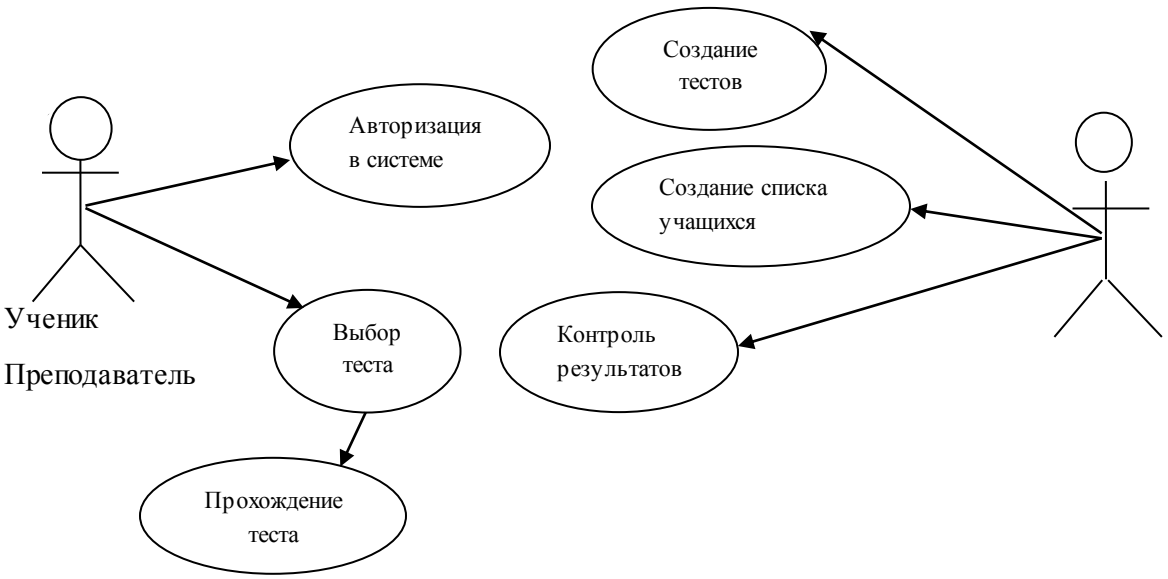

Рисунок 2.1 – Диаграмма вариантов использования

Диаграмма последовательности (англ. sequence diagram) - диаграмма, на которой для некоторого набора объектов на единой временной оси показан жизненный цикл какого-либо определённого объекта (создание-деятельность-уничтожение некой сущности) и взаимодействие актёров (действующих лиц) ИС в рамках какого-либо определённого прецедента (отправка запросов и получение ответов). Используется в языке UML.

На диаграмме последовательности объекты располагаются слева направо [15].

На рисунках 2.2 - 2.3 показаны диаграммы последовательности для учащихся и преподавателей соответственно. Данные диаграммы отображают последовательность действий при работе с программой. Ученик должен зарегистрироваться в системе, выбрать предложенный тест, затем пройти его и посмотреть на результаты его прохождения. Преподаватель должен войти в систему под своими данными, далее он может посмотреть результаты тестирования либо создать очередной тест.

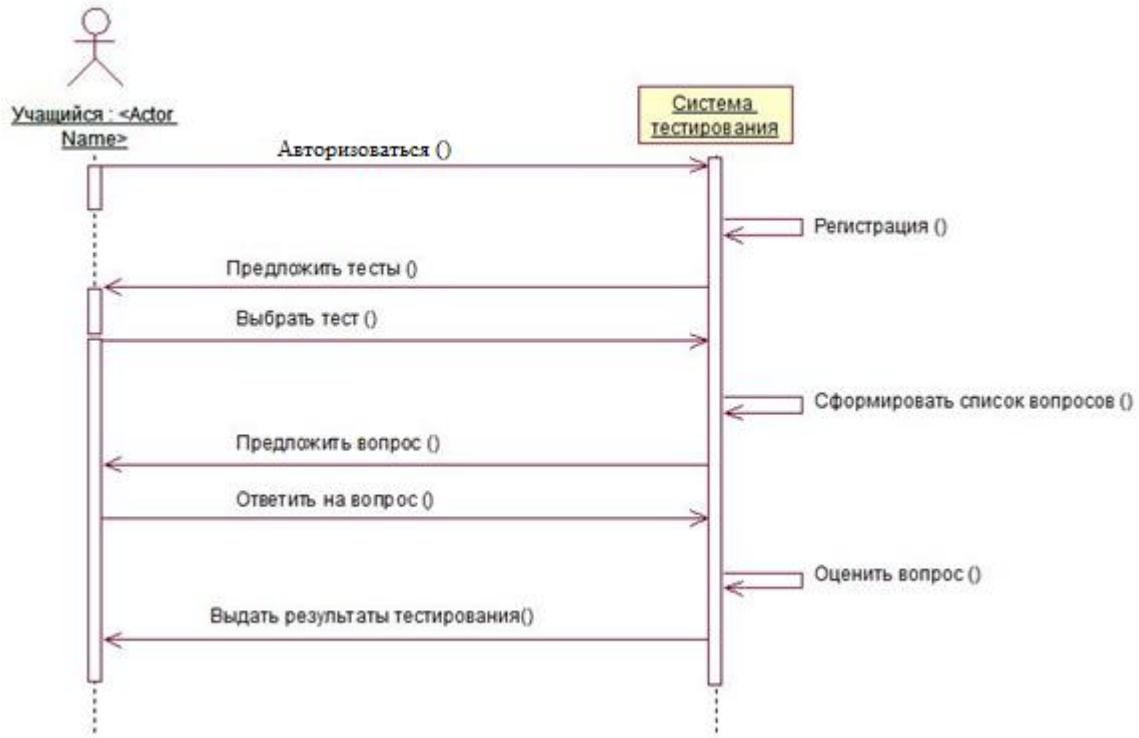

Рисунок 2.2 – Диаграмма последовательности для учащихся

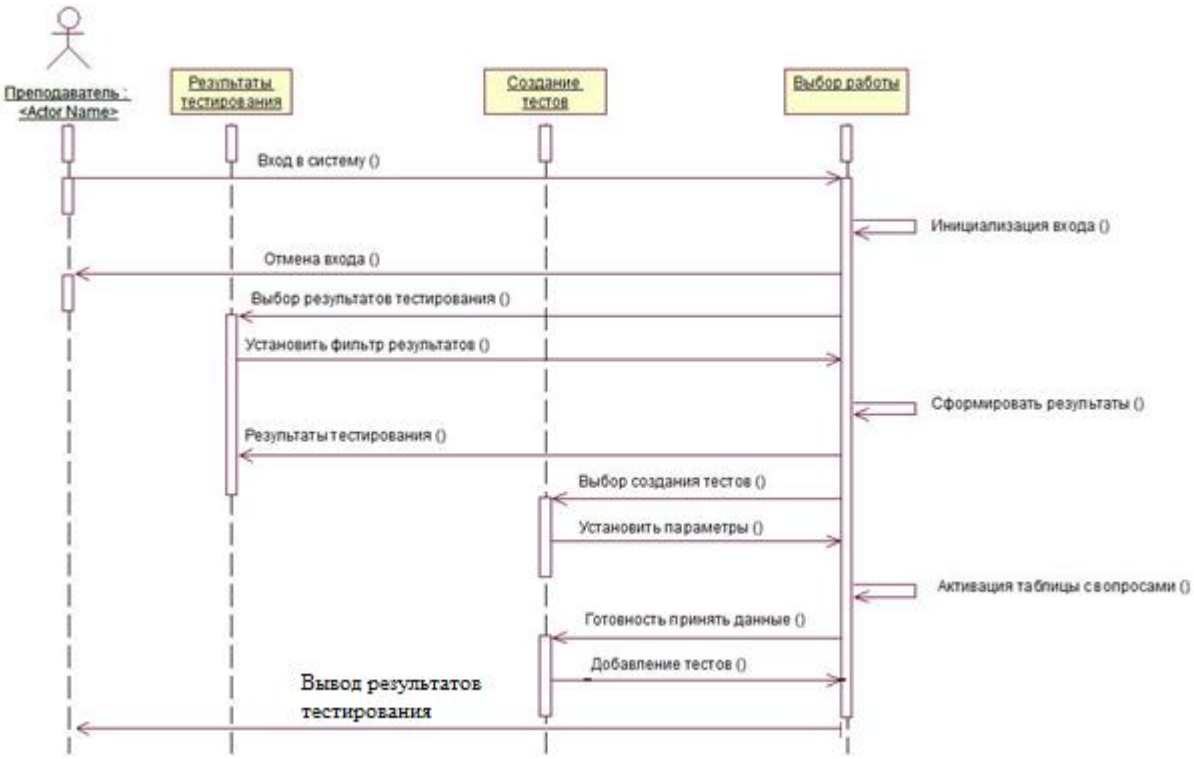

Рисунок 2.3 – Диаграмма последовательности для преподавателей

Диаграмма классов (англ. Static Structure diagram) - диаграмма, демонстрирующая классы системы, их атрибуты, методы и взаимосвязи между ними [14]. Входит в UML.

Существует два вида диаграмм: статический и аналитический.

Первый из них направлен на рассмотрение логической взаимосвязи классов между собой.

Второй служит для рассмотрения общего вида и взаимосвязи классов, которые входят в рассматриваемую систему [7].

Диаграмма, которая описываем модель предметной области, присуща концептуальной точке зрения.

На рисунке 2.4 изображена диаграмма классов информационной системы. Данная диаграмма показывает связь между классами (объектами) базы данных.

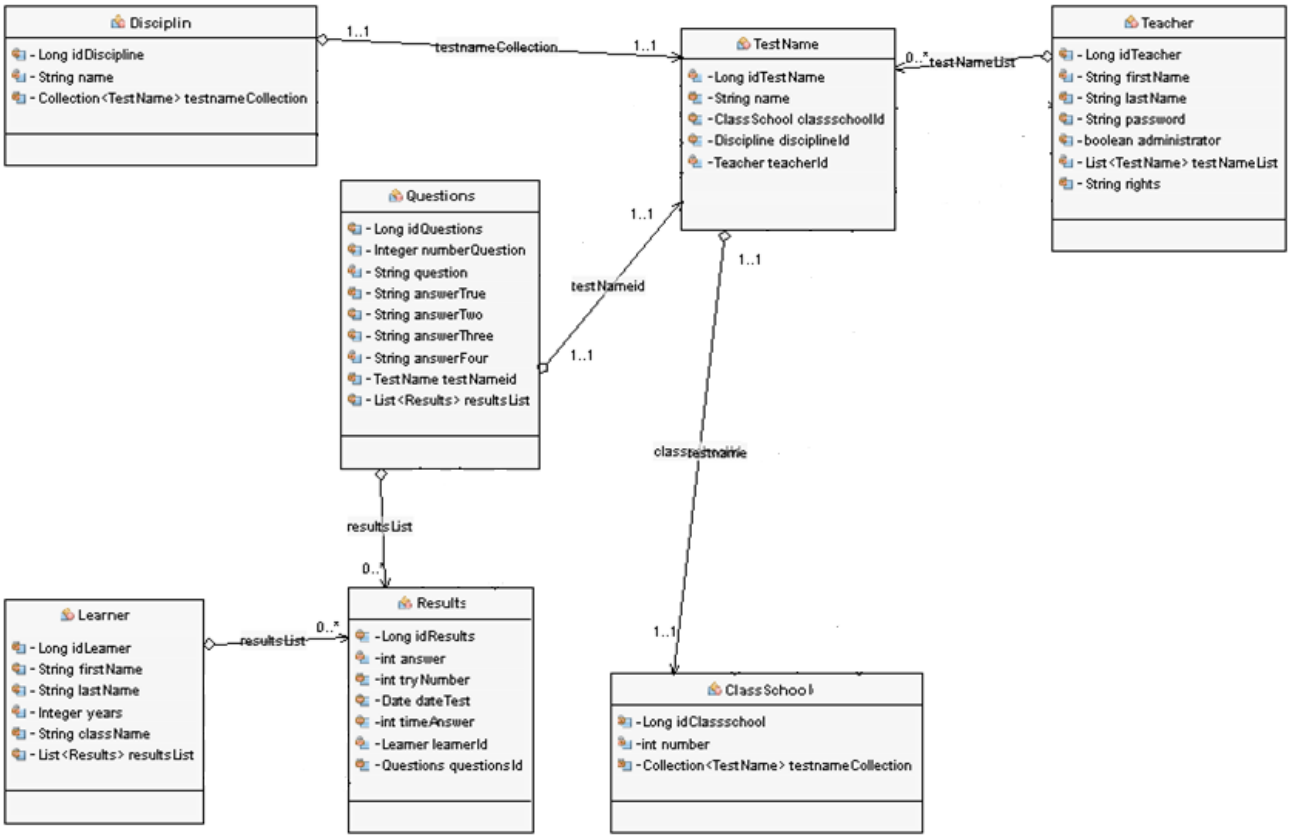

Рисунок 2.4 – Диаграмма классов системы тестирования

Руководствуясь приведенными диаграммами, будет строиться логическая модель информационной системы.

2.3 Информационное обеспечение автоматизированной информационной системы тестирования с элементами геймификации для школьников

2.3.1 Используемые классификаторы и системы кодирования

В данном разрабатываемом проекте будет использоваться порядковая система кодирования, в которой значения присваиваются последовательно по мере поступления новых объектов.

В составе информационного обеспечения рассматриваемого объема задач выделены следующие классификаторы:

- классификатор дисциплин;
- классификатор классов;
- классификатор названий тестов;

- классификатор вопросов;

- классификатор результатов;

- классификатор учащихся;

- классификатор преподавателей.

Характеристики классификаторов представлены в таблице 2.

Таблица 2.1 - Характеристики классификаторов

Наименование классификатора	Значность кода	Система кодирования	Система классификации	Вид классификатора
Дисциплины	-	Порядковая	Иерархическая	Общесистемный
Классы	-	Порядковая	Иерархическая	Общесистемный
Название тестов	-	Порядковая	Иерархическая	Общесистемный
Вопросы	-	Порядковая	Иерархическая	Общесистемный
Результаты	-	Порядковая	Иерархическая	Общесистемный
Учащиеся	-	Порядковая	Иерархическая	Общесистемный
Преподаватели	-	Порядковая	Иерархическая	Общесистемный

2.3.2 Характеристика нормативно-справочной и входной/выходной оперативной информации.

Для разработки таблиц необходимо изначально продумать их структуру. В правильно разработанных таблицах не должны присутствовать повторения полей. Таблица должна содержать всю необходимую информацию, но при этом не засорять таблицу ненужными данными. При необходимости всегда можно ввести в таблицу дополнительное поле. При использовании больших БД часто целесообразно разбивать одну большую таблицу на несколько маленьких взаимосвязанных таблиц.

В данной информационной системе можно сказать, что входная информация представляет собой учащегося, который будет проходить тестирование.

Входная информация необходима для обработки данных, она также представлена в виде таблиц, поля в которых имеют соответствующий шифр, тип и размер, а также ключевое поле-идентификатор.

В данном проекте автоматизированной информационной системы в качестве результирующей информации представляет таблица, созданная в

экранной форме по запросу к базе данных с фильтром по поиску, которая будет включать в себя имя, фамилия и класс учащегося, время, затраченное на проведение тестов, а также его результаты. Также после проведения тестирования учащийся может посмотреть результаты тестирования всего класса, которые отобразятся в отдельной экранной форме.

2.4 Проектирование БД автоматизированной информационной системы тестирования с элементами геймификации для школьников

2.4.1 Выбор технологии проектирования БД АИС

Выбор средства проектирования происходит по общей модели, изображенной на рисунке 2.5.

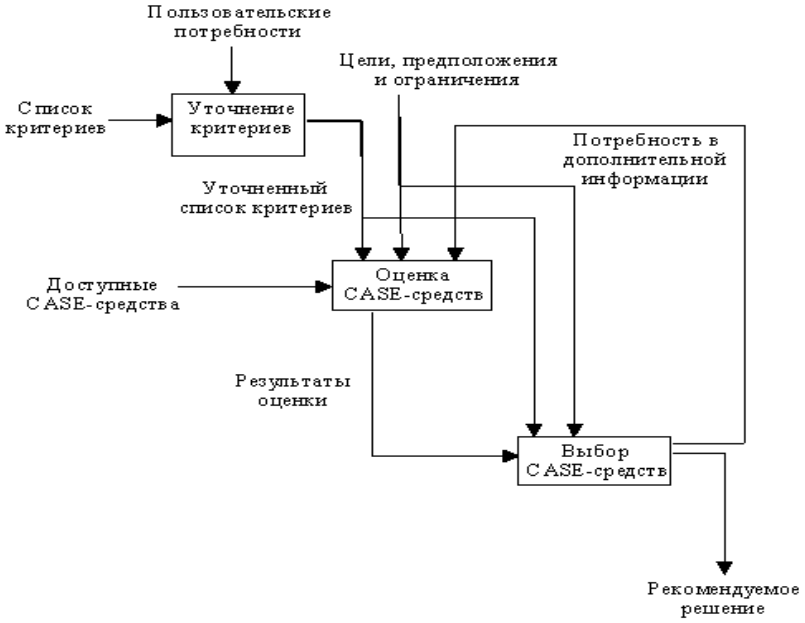

Рисунок 2.5 - Модель процесса оценки и выбора

Исходя из модели, оценка и выбор могут выполняться как независимо друг от друга, так и вместе. Для каждого из этих процессов требуется применение определенных критериев. Сам процесс оценки и выбора может преследовать несколько целей, включая одну или более из следующих:

- оценка нескольких CASE-средств и выбор одного или более из них;
- оценка одного или более CASE-средств и сохранение результатов для последующего использования;

- выбор одного или более CASE-средств с использованием результатов предыдущих оценок.

Как видно из рисунка, входной информацией для процесса оценки является:

- определение пользовательских потребностей;
- цели и ограничения проекта;
- данные о доступных CASE-средствах;
- список критериев, используемых в процессе оценки.

В качестве средства проектирования баз данных была выбрана система PowerDesigner. Эта система имеет большой набор инструментов для разработки баз данных, а также фактически имеет все необходимые инструменты для сопровождения полного жизненного цикла ИС. Основные определяющие факторы, в пользу выбора данного продукта:

- простота создания объектов и их связей, назначения типов данных;
- генерация SQL - скриптов создания БД;
- мощная графическая составляющая для проектирования;

Данный продукт отвечает всем требованиям для разработки бизнес-процессов и проектирования ИС, по тому будет хорошим выбором для данной работы.

2.4.2 Разработка концептуальной модели данных автоматизированной информационной системы тестирования с элементами геймификации для школьников

Главным этапом для разработки концептуальной модели, прежде всего, является описание объектов, а также их атрибутов. Объект – это сущность, обладающая определённым состоянием и поведением, имеющая заданные значения свойств и операций над ними. Атрибутом можно назвать поименованную характеристику объекта, с помощью которой моделируются его свойства.

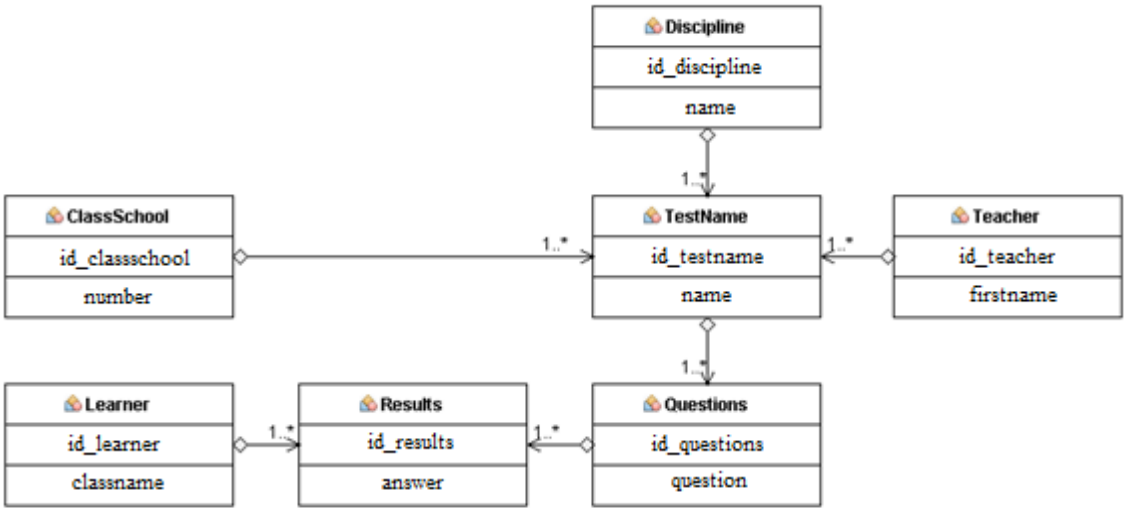

Рисунок 2.6 – Концептуальная модель АИС тестирования учащихся

На рисунке 2.6 показана концептуальная модель сущностей. Также на данной модели показаны связи между сущностями.

2.4.3 Разработка логической модели данных АИС.

На основании вышеизложенного описания и проработки информации построим логическую модель информационной системы.

На рисунке 2.7 изображена логическая модель АИС.

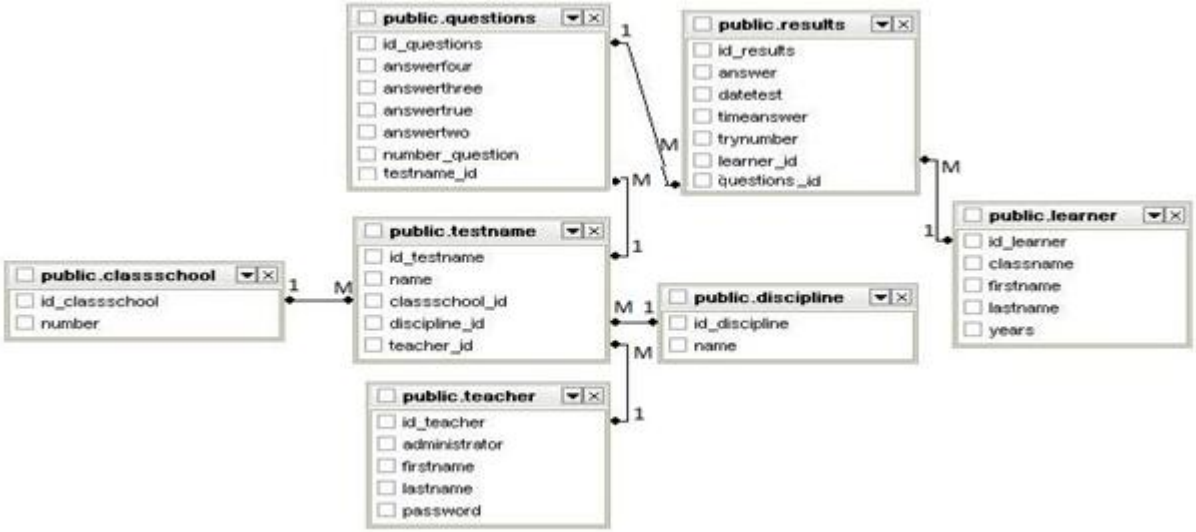

Рисунок 2.7 – Логическая модель АИС

Данная логическая модель АИС представляет собой отношение между сущностями. Для логической модели составим таблицу с описанием всех сущностей логической модели данных.

Таблица 2.3 – Сущности логической модели данных

Название сущности	Описание сущности
classSchool (номер класса)	Сущность содержит в себе информацию о номере класса
learner (учащиеся)	Сущность содержит информацию об учащихся проходивших тестирование
discipline (дисциплины)	Сущность содержит в себе информацию о названиях дисциплин
teacher (преподаватели)	Сущность содержит информацию о преподавателях
testname (название тестов)	Сущность содержит информацию о названиях тестов
questions (вопросы)	Сущность содержит в себе перечень вопросов для тестов
results (результаты)	Сущность содержит информацию в себе информацию о результатах тестирования

Благодаря смоделированной логической модели данным можно приступить к построению физической модели автоматизированной информационной системы тестирования с элементами геймификации для школьников.

2.5 **Требования к аппаратно-программному обеспечению автоматизированной информационной системы тестирования с элементами геймификации для школьников**

Как уже упоминалось ранее, данную автоматизированную информационную систему необходимо будет внедрить в среднеобразовательном учебном учреждении. Поэтому, укажем минимальные системные требования к аппаратно-программному обеспечению АИС:

- Microsoft Windows XP, Microsoft Windows Server 2003, Microsoft Windows Server 2008, Microsoft Windows Server 2008 R2, Microsoft Windows Server 2012, Microsoft Windows Server 2012 R2, Microsoft Windows Vista, Microsoft Windows 7, Microsoft Windows 8/8.1, Microsoft Windows 10;

- 512 Мб ОЗУ (рекомендуется 1024 Мб или более);

- 250 Мб свободного пространства на жестком диске для установки программы;

- возможность соединения с локальным или удаленным сервером БД;

Выводы по главе 2

В данной главе была проведена большая работа по подготовке автоматизированной системы к ее физическому проектированию. А именно:

- обоснована и выбрана технология для логического проектирования;
- разработаны необходимые диаграммы;
- описаны используемые классификаторы и системы кодирования;
- спроектирована логическая модель АИС;
- обоснованы требования к аппаратно-программному обеспечению АИС.

То есть результатом данной главы является разработанная и обоснованная логическая модель автоматизированной системы тестирования с элементами геймификации для школьников.

Глава 3. Физическое проектирование автоматизированной информационной системы тестирования с элементами геймификации для школьников

3.1 Выбор архитектуры автоматизированной информационной системы тестирования с элементами геймификации для школьников

Как как мы имеем дело с информационной системой, где хранилищем информации будет представлена БД, поэтому архитектура системы должна быть клиент-серверной.

«Клиент - сервер» (англ. client–server) - вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг, называемыми серверами, и заказчиками услуг, называемыми клиентами. По своей сути клиент и сервер - это программное обеспечение. Как правило программы расположены на разных вычислительных машинах и взаимодействуют между собой через вычислительную сеть посредством сетевых протоколов, но они могут быть расположены также и на одной машине. Программы-серверы работают в режиме ожидания. Клиентские программы делают запросы на получение информации, вследствие чего программа-сервер предоставляет им свои ресурсы в виде данных.

Особенностью архитектуры клиент-сервер является использование выделенных серверов баз данных, понимающих запросы на языке структурированных запросов SQL (Structured Query Language) и выполняющих поиск, сортировку и агрегирование информации.

На рисунке 3.1 показан пример общего вида двухуровневой архитектуры. Как можно заметить, что сервер одновременно может общаться с несколькими клиентами. Информация поступает в двух направлениях – как от сервера к клиенту, так и наоборот.

Эта часть системы отвечает за взаимодействие между клиентом и сервером, то есть клиент не общается напрямую с сервером БД.

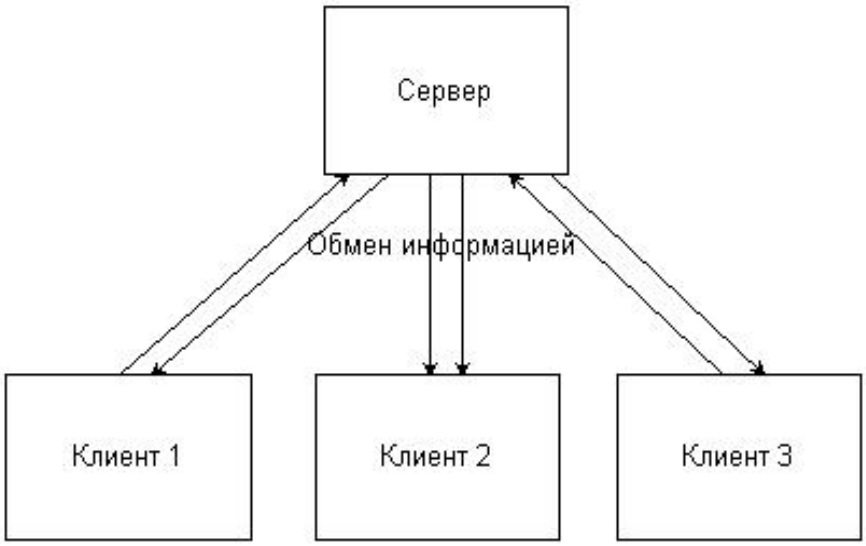

Рисунок 3.1 – Двухуровневой архитектуры АИС

Во многоуровневых системах функции обработки данных могут быть вынесены на несколько отдельных серверов. Это способствует разделению функций обработки, хранения и представления данных для более эффективного использования возможностей серверов и клиентов.

Вариант архитектуры, изображенной на рисунке 3.1, будет полностью удовлетворять требованиям, в последствии созданной, автоматизированной информационной системе тестирования. Поэтому в данном проекте выбрана двухуровневая автоматизированная информационная система «клиент-сервер».

3.2 Выбор технологии разработки программного обеспечения автоматизированной информационной системы тестирования с элементами геймификации для школьников

Сочетание различных признаков классификации методов проектирования обуславливает характер используемой технологии проектирования АИС. Технология проектирования информационных систем состоит из совокупности трех составляющих:

1 Пошаговой процедуры. Данная составляющая определяет последовательность технологических операций проектирования.

2 Критериев и правил. Эта пара составляющих используется для оценки результатов выполнения технологических операций.

3 Нотаций (текстовых и графических средств). Нотации используются для описания проектируемой системы.

Существует много CASE-средств для разработки АИС. Одним из критериев выбора технологии является использование бесплатного программного обеспечения. В подразделе 3.3 обоснован выбор самой СУБД, а именно – postgreSQL. Для этой СУБД существует бесплатное программное обеспечение pgAdmin 4.

pgAdmin 4 – ПО комплексного проектирования баз данных и управления базами данных для баз данных PostgreSQL. Данное ПО обладает достаточным функционалом для создания БД, с последующим созданием таблиц. Также есть возможность построения запросов на выборку при этом не написав ни одной строки кода.

3.3 Выбор системы управления базой данных автоматизированной информационной системы тестирования с элементами геймификации для школьников

На сегодняшний день существует огромное количество СУБД, но использована будет PostgreSQL.

PostgreSQL - свободная объектно-реляционная система СУБД. Существует в реализациях для множества UNIX-подобных платформ, включая AIX, различные BSD-системы, HP-UX, IRIX, Linux, macOS, Solaris/OpenSolaris, Tru64, QNX, а также для Microsoft Windows.

Сильными сторонами PostgreSQL считаются:

- высокопроизводительные и надёжные механизмы транзакций и репликации;

- расширяемая система встроенных языков программирования: в стандартной поставке поддерживаются PL/pgSQL, PL/Perl, PL/Python и PL/Tcl;

- дополнительно можно использовать PL/Java, PL/PHP, PL/Py, PL/R, PL/Ruby, PL/Scheme, PL/sh и PL/V8, а также имеется поддержка загрузки C-совместимых модулей;

- наследование;

- легкая расширяемость.

Также, как и MS SQL PostgreSQL работает с многочисленными типами данных, охватывающие все потребности системы, в частности будем использовать стабильную версию PostgreSQL 9.6.

3.4 Разработка физической модели данных автоматизированной информационной системы тестирования с элементами геймификации для школьников.

Физическая модель базы данных создана средствами ПО pgAdmin 4.

Данная модель имеет 7 таблиц. Составим таблицу атрибутов сущностей базы данных.

Таблица №3.2 – Атрибуты сущностей базы данных

Сущность	Атрибут	Ключ	Тип	Назначение
1	2	3	4	5
classschool (Класс в школе)	id_ classschool	Первичный	Целое число	Идентификатор
	number	Нет	Целое число	Номер класса
discipline (Дисциплины)	id_ discipline	Первичный	Целое число	Идентификатор
	name	Нет	Строка	Название дисциплины
learner (Учащиеся)	id_learner	Первичный	Целое число	Идентификатор
	classname	Нет	Строка	Название класса
	firstname	Нет	Строка	Имя
	lastname	Нет	Строка	Фамилия
	years	Нет	Целое число	Год поступления
teacher (Преподава-тель)	id_teacher	Первичный	Целое число	Идентификатор
	firstname	Нет	Строка	Имя
	lastname	Нет	Строка	Фамилия
	password	Нет	Строка	Пароль

Продолжение таблицы 3.2

1	2	3	4	5
teacher (Преподава-тель)	administrator	Нет	Логический	Права доступа
teacher (Преподава-тель)	id_teacher	Первичный	Целое число	Идентификатор
	firstname	Нет	Строка	Имя
	lastname	Нет	Строка	Фамилия
	password	Нет	Строка	Пароль
	administrator	Нет	Логический	Права доступа
testname (Название тестов)	id_testname	Первичный	Целое число	Идентификатор
	name	Нет	Строка	Название теста
	classschool_id	Внешний	Целое число	Ссылка на таблицу classschool
	discipline_id	Внешний	Целое число	Ссылка на таблицу discipline
	teacher_id	Внешний	Целое число	Ссылка на таблицу teacher
questions (Вопросы)	id_questions	Первичный	Целое число	Идентификатор
	answertrue	Нет	Строка	Ответ 1 (правильный)
	answertwo	Нет	Строка	Ответ 2
	answerthree	Нет	Строка	Ответ 3

В таблице 3.2 приведены сущности со своими атрибутами. Приведем словесное описание каждой сущности.

Classschool – сущность хранит номер класса, для которого будет формироваться тест.

Discipline – сущность хранит в себе названия учебных дисциплин, для которых будет формироваться тест.

Learner – сущность хранит в себе данные об учащихся, которые вошли в систему для прохождения тестирования.

Teacher – сущность хранит в себе данные о преподавателях. У преподавателей есть право добавлять тесты.

Testname – сущность хранит в себе названия тестов, которые зависят от номера класса и названия дисциплины. Для одного номера класса и одной и той же дисциплины не может храниться два одинаковых названия тестов (реализовано программно).

Questions – сущность хранит в себе вопросы и ответы на них. Минимально необходимо хранить два ответа, максимально – четыре. Сущность зависит от названия тестов.

Results – сущность хранит в себе результаты тестирования учащихся. Результаты хранятся по каждому ответу на вопрос. Сущность зависит от учащегося и вопросов.

На рисунке 3.2 приведена физическая модель данных. Данная модель показывает связь между сущностями. Описание каждой сущности приведено в таблице 3.2.

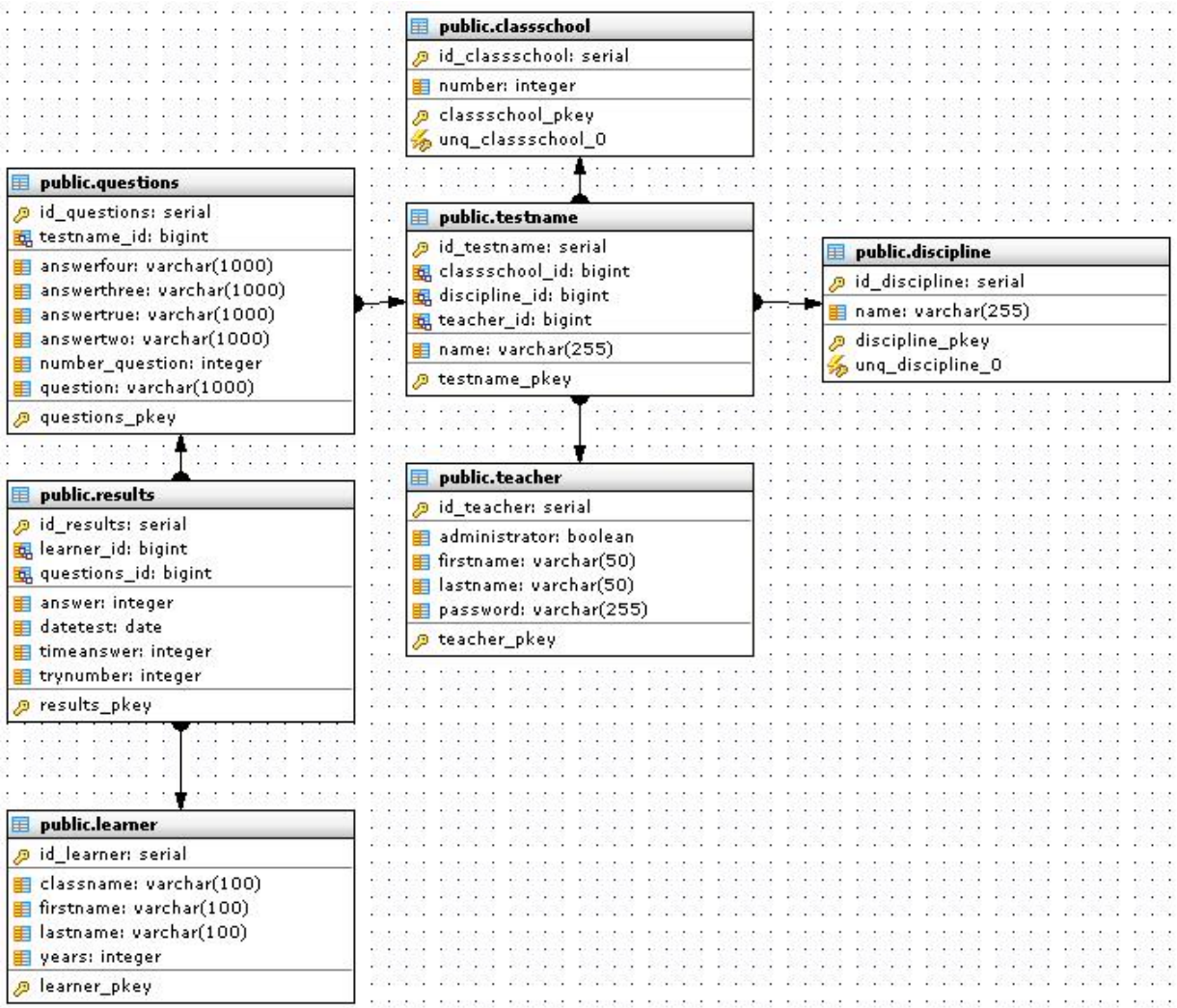

Рисунок 3.2 – Физическая модель данных

SQL-программа состоит из последовательности команд. Команда, в свою очередь, представляет собой последовательность компонентов,

оканчивающуюся точкой с запятой («;»). Конец входного потока также считается концом команды. Какие именно компоненты допустимы для конкретной команды, зависит от её синтаксиса. Компонентом команды может быть ключевое слово, идентификатор, идентификатор в кавычках, строка (или константа) или специальный символ. Компоненты обычно разделяются пробельными символами (пробел, табуляция, перевод строки), но это не требуется, если нет неоднозначности (например, когда спецсимвол оказывается рядом с компонентом другого типа) [17]. За более подробной информацией можно обратиться к документации по postgreSQL [17].

Приведем примеры нескольких запросов.

Запрос вернет всех учащихся, которые сохранены в сущности learner:

```
SELECT id_learner, classname, firstname, lastname, years
FROM public.learner;
```

Запрос вернет все поля сущности testname у которой значение classschool_id = 1:

```
SELECT id_testname, name, classschool_id, discipline_id, teacher_id
  FROM public.testname
  where classschool_id = 1;
```

3.5 Разработка программного обеспечения автоматизированной информационной системы тестирования с элементами геймификации для школьников

Перед рассмотрение взаимодействия модулей приложения целесообразно дать описания используемому языку программирования, а также среде разработки приложения.

В данном проекте используется язык программирования Java 1.8. Java - сильно типизированный объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems (в последующем приобретённой компанией Oracle). Приложения Java обычно транслируются в специальный байт-код, поэтому они могут работать на любой компьютерной архитектуре, с помощью виртуальной Java-машины. Данный язык был выбран исходя из

опыта работы с ним. Язык Java показал себе только с лучшей стороны. Имеет огромный набор библиотек, хорошую документированную поддержку. Также еще большим достоинством является независимость от операционной системы. Все что требуется для запуска приложения – это установка виртуальной машины, среды в которой будет исполняться данное приложение.

Среда разработки выбрана NetBeans 8.2. NetBeans IDE - свободная интегрированная среда разработки приложений (IDE) на языках программирования Java, Python, PHP, JavaScript, C, C++, Ада и ряда других. NetBeans Platform - платформа для разработки модульных настольных приложений. NetBeans IDE содержит все, что нужно для разработки плагинов и приложений на основе NetBeans Platform. Приложения могут динамически загружать другие модули. Любое приложение может включить модуль обновления, чтобы позволить пользователям загружать обновления для программ и модулей в работающее приложение.

Рассмотрим весь проект поэтапно. Начнем с состава пакетов. На рисунке 3.3 приведена диаграмма пакетов программного обеспечения АИС. Приведем описание и содержание каждого из них.

1 Пакет - META-INF содержит файл persistence.xml – ответственный за конфигурацию контента постоянства БД, по сути это точка входа в JPA приложение. В файле хранится информация о подключении к БД, отображаемых сущностях, а также ряд дополнительных настроек. Для того чтобы jar-файл мог производить соединение с БД, заведомо не зная ее расположения (то есть конфигурационных настроек к ней), вынесен отдельно файл config.conf в котором указываются адрес и порт к БД, имя БД, имя владельца, а также пароль к БД. Все что требуется перед запуском проекта – это настроить этот файл.

В проекте используется технология Java Persistence API (JPA). JPA - спецификация API Java EE, предоставляет возможность сохранять в удобном виде Java-объекты в базе данных. Существует много реализаций данной технологии. В текущем проекте используется фреймворк Eclipselink. Данная

технология дает возможность работать с таблицами базы данных как с объектами. Когда идет речь об отображении какого либо объекта в БД или выполнении запросов к ним, то следует использовать термин «сущность». Сущности являются собой объекты, которые не долго располагаются в памяти, но постоянно в базе данных. Можно долго описывать данную технологию, но если сказать вкратце, то данный фреймворк позволяет работать как с привычными строковыми запросами к базе, так и с объектно-ориентированными запросами.

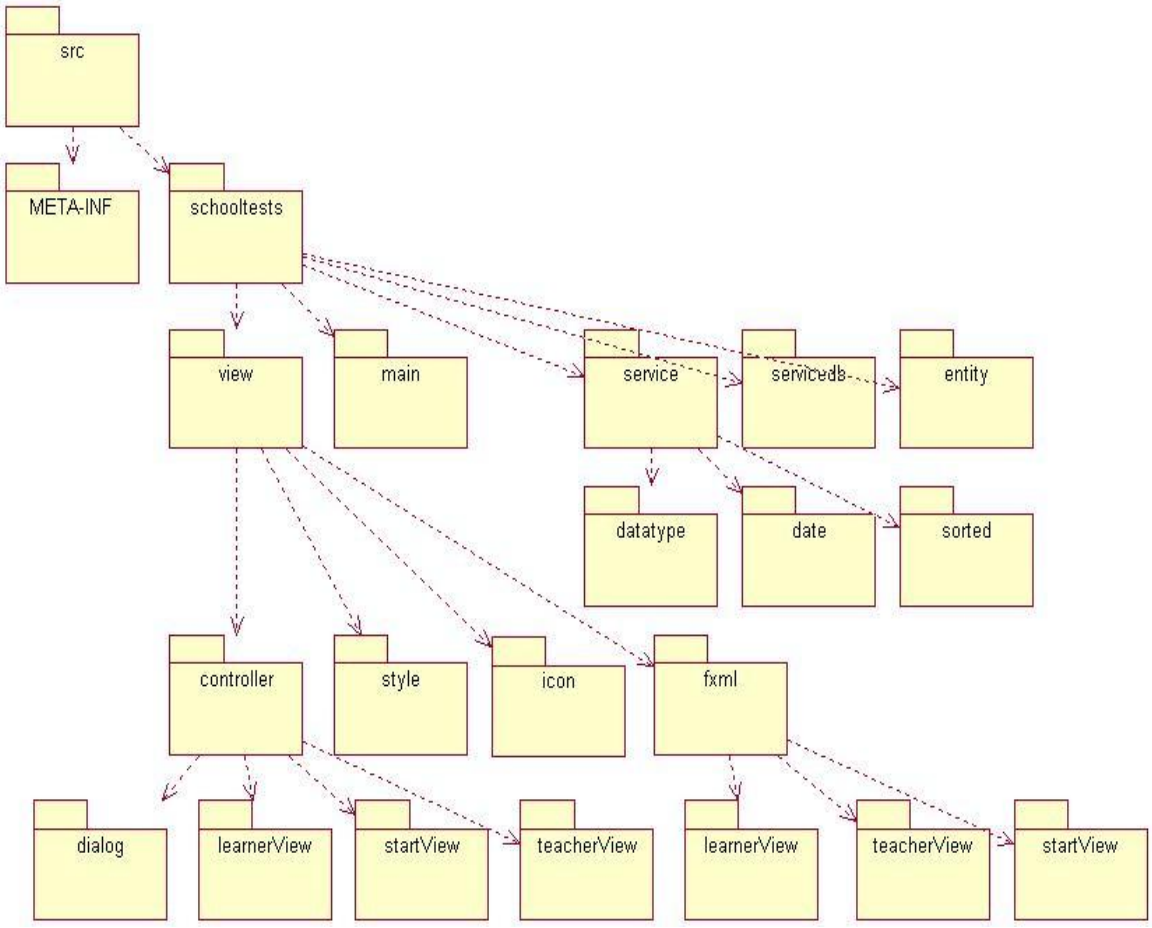

Рисунок 3.3 – Диаграмма пакетов

Весь проект построен на объектно-ориентированных запросах. Примеры таких запросов будут описаны ниже. Суть заключается в том, что запрос формируется с привычных для программиста методов, которые реализованы фреймворком. Далее все той же технологией этот запрос обрабатывается и

генерируется уже привычный для БД строковый запрос и через драйвер JDBC отправляется в БД.

JDBC (англ. Java DataBase Connectivity - соединение с базами данных на Java) - платформенно-независимый промышленный стандарт взаимодействия Java-приложений с различными СУБД, реализованный в виде пакета java.sql, входящего в состав Java SE [21].

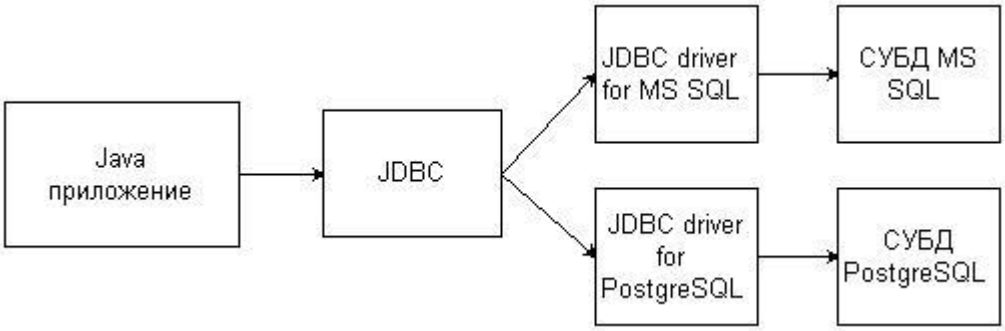

Рисунок 3.4 – Связь Java-приложения с СУБД

На рисунке 3.4 показан общий вид связи Java-приложения с базой данных посредством JDBC-драйвера. В данном проекте используется PostgreSQL_JDBC_4.2_Driver_42.1.4.jar JDBC-драйвер.

2 Пакет - schooltest содержит в себе ряд пакетов: view, main, service, servicedb, entity.

3 Пакет - entity содержит в себе все сущности (классы), описанные в базе данных: ClassSchool, Discipline, TestName, Learner, Teacher, Questions, Results. Каждая из этих сущностей называется по названию таблиц в БД.

Приведет описание одной из них. Сущность TestName – является проекцией на таблицу testname в БД. Далее приведем код сущности без его конструкторов и методов получения и установки

```
@Entity
@Table(name = "testname")
@XmlRootElement
@NamedQueries({
    @NamedQuery(name = "TestName.findAll", query = "SELECT t
FROM TestName t")
```

```
        , @NamedQuery(name = "TestName.findByIdTestName", query =
"SELECT t FROM TestName t WHERE t.idTestName = :idTestName")
        , @NamedQuery(name = "TestName.findByName", query =
"SELECT t FROM TestName t WHERE t.name = :name")})
    public class TestName implements Serializable {

        private static final long serialVersionUID = 1L;
        @Id
        @GeneratedValue(strategy = GenerationType.IDENTITY)
        @Basic(optional = false)
        @Column(name = "id_testName")
        private Long idTestName;
        @Basic(optional = false)
        @Column(name = "name", nullable = false)
        private String name;
        @JoinColumn(name = "classSchool_id", referencedColumnName
= "id_classSchool", nullable = false)
        @ManyToOne(optional = false)
        private ClassSchool classschoolId;
        @JoinColumn(name = "discipline_id", referencedColumnName
= "id_discipline", nullable = false)
        @ManyToOne(optional = false)
        private Discipline disciplineId;
        @JoinColumn(name = "teacher_id", referencedColumnName =
"id_teacher", nullable = false)
        @ManyToOne(optional = false)
        private Teacher teacherId;
```

Данный класс имеет ряд аннотаций. Код сущности непосредственно снабжается всевозможными аннотациями, описанными в javax.persistence. Приведем описание некоторых, основных из них:

- @Table – указывается имя таблицы;
- @NamedQuery – создается именованный запрос;
- @GeneratedValue – указывается стратегия автоинкрементного поля;

- @Column – описывается колонка таблицы (имя, значение null, длина строки);

- @JoinColumn – указывается связь между таблицами;

- @OneToOne, @ManyToOne – указывается отношение между таблицами (один к одному, много к одному и т.д.).

Перед запуском основного экрана приложения проверяется соответствие сущностей на их наличие в БД, если какая либо из них будет отсутствовать, то она автоматически сгенерирует нужную таблицу со всеми необходимыми колонками в базе данных. Также данные классы имеют конструкторы, методы получения и установки значений своих полей, а также переопределенные методы hashCode, equals и toString.

4 Пакет - main содержит в себе ряд классов: Main, ManConnection и ShowWindow.

Класс Main является точной входа в приложение, так как в нем создан метод main, а также метод инициализации JavaFX приложения.

Класс ShowWindow является классом, реализующим работу по смене отображаемого графического окна в приложении.

Класс ManConnection имеет статические открытые поля, а также методу установки полей. Данные поля несут в себе ряд данных активного пользователя, которые в последствие необходимы для работы.

5 Пакет - servicedb содержит ряд классов: DefaultCreateManager, Encryption, JPQLNamedQueryEntity, JPQLNativeQuery, JPQLQueryEntity, QueryCriteriaBuilder, SingleEntity, WorkViewDbRequest. Этот пакет используется для работы с базой данных.

Класс DefaultCreateManager содержит методы для открытия и закрытия соединения с базой данных.

Класс SingleEntity содержит методы для добавления, нахождения и удаления единичной сущности.

Класс JPQLQueryEntity содержит метод, реализующий динамический запрос с условием. Фактически данный метод используется для инициализации входа преподавателя.

Класс JPQLNativeQuery содержит метод для работы с родными (строковыми) запросами. Метод применяется для нахождения последнего значения автоинкрементного поля.

Класс JPQLNamedQueryEntity содержит метод для работы с именованными запросами, которые описаны в классах-сущностях.

Класс Encryption содержит метод для шифрования строки заданным алгоритмов. В проекте используется алгоритм MD5 для шифрования пароля.

Класс QueryCriteriaBuilder содержит методы для работы с простыми объектно-ориентированными запросами.

Класс WorkViewDbRequest, сервисный класс, в котором реализованы статические методы на выборку данных из БД посредством объектно-ориентированного запроса. Приведем один из таких методов:

```
      public static Teacher findTeacher(String lastName, String pass){
            EntityManager em = DefaultCreateManager.createManager();
            //Фабрика по созданию экземпляров CriteriaQuery, а также всех видов выражений запросов
            CriteriaBuilder cb = em.getCriteriaBuilder();
            CriteriaQuery<Teacher> cq = cb.createQuery(Teacher.class);
            Root<Teacher> rootTeacher = cq.from(Teacher.class);
            //---Predicate - Утверждение (условие на фильтрацию строк по заданному критерию)
            //Условие сравнения строки из поля name и заданной строки
            Predicate lastNameTeach = cb.equal(rootTeacher.get(Teacher_.lastName), lastName);
            Predicate passTeach = cb.equal(rootTeacher.get(Teacher_.password), pass);
```

```
        cq.where(lastNameTeach, passTeach);

        //Типизированный запрос
        TypedQuery<Teacher> tq = em.createQuery(cq);
        //Получение id со значение из БД
        Teacher teach = null;
        try {
            teach = tq.getSingleResult();
        } catch(Exception exc) {

        }
        DefaultCreateManager.closeManager();
        return teach;
    }
```

Метод public static Teacher findTeacher(String lastName, String pass) возвращает объект типа Teacher с условием соответствия фамилии (lastName) и пароля (pass).

Чтобы стало более понятно, покажем этот же запрос, только уже «родным» (строковым) запросом:

SELECT id_teacher, administrator, firstname, lastname, password

FROM teacher

WHERE lastname = '?' and password = '?';

Данные строки говорят, что нужно вернуть колонки id_teacher, administrator, firstname, lastname, password из таблицы teacher, с условием, что lastname = ? и password = ?.

В приложении А приведен код программы всех классов для работы с базой данных.

6 Пакет - service содержит в себе ряд пакетов: datatype, date, sorted.

7 Пакет - datetype содержит в себе класс ResultsLearner. Объекты данного класса хранят в себе выборку из БД результатов тестирования.

8 Пакет - date содержит в себе класс DateView. В данном классе реализованы методы представления даты в разных форматах.

9 Пакет - sorted содержит в себе классы SortedListQuestions, SortedListTeacher, SortedListTests. Данные классы реализуют интерфейс типа Comparator<T>. Применяются для сортировки информация в графических формах.

10 Пакет - view содержит в себе ряд пакетов: Icon, style, controller и fxml.

11 Пакет - Icon хранит иконки, необходимые для визуализации интерфейса.

12 Пакет - style содержит файл schoolTestsStyle.css в котором хранятся стили ряда элементов.

13 Пакет - controller содержит в себе ряд пакетов: startView , learnerView, teacherView и dialog.

14 Пакет - startView содержит в себе классы: StartViewController, LoginLearnerController и LoginTeacherController.

Класс StartViewController отвечает за логику работы стартового окна интерфейса.

Классы LoginLearnerController и LoginTeacherController отвечают за логику работы модальных окон входа в программу для учащихся и преподавателей соответственно.

15 Пакет - learnerView содержит в себе классы: TestSelectionController, TestRunController и LookResultsController.

Класс TestSelectionController отвечает за логику работы окна выбора теста для учащегося.

Класс TestRunController отвечает за логику работы окна прохождения тестирования.

Класс LookResultsController отвечает за логику работы окна с результатами прохождения тестирования учащимися.

16 Пакет - teacherView содержит в себе классы: ShowResultsController, AddTeacherController, AddTestsController и ShowResultsController.

Класс ShowResultsController отвечает за логику работы графического окна-меню для преподавателей.

Класс AddTeacherController отвечает за логику работы добавления, редактирования и удаления преподавателя в БД.

Класс AddTestsController отвечает за логику работы окна добавления тестов в БД.

Класс ShowResultsController отвечает за логику работы окна отображения результатов тестирования учащихся по заданному фильтру (условию).

17 Пакет - dialog содержит в себе класс Alarm.

Класс Alarm отвечает за отображение модального информационного окна инициализированного как следствие какого либо предупреждения или ошибки.

18 Пакет - fxml содержит в себе ряд пакетов: startView, learnerView и teacherView.

19 Пакет - startView содержит в себе fxml-файлы: startView.fxml, loginLearner.fxml и loginTeacher.fxml.

startView.fxml – отвечает за отображение стартового графического окна.

loginLearner.fxml – отвечает за отображение модального окна входа учащегося в программу.

loginTeacher.fxml – отвечает за отображение модального окна входа преподавателя в программу.

Пакет 20 - learnerView содержит в себе fxml-файлы: testSelection.fxml, testRun.fxml и lookResults.fxml.

testSelection.fxml – отвечает за отображение окна выбора теста для прохождения тестирования.

testRun.fxml – отвечает за отображения окна прохождения тестирования.

lookResults.fxml. – отвечает за отображения окна результатов прохождения тестирования учащимся.

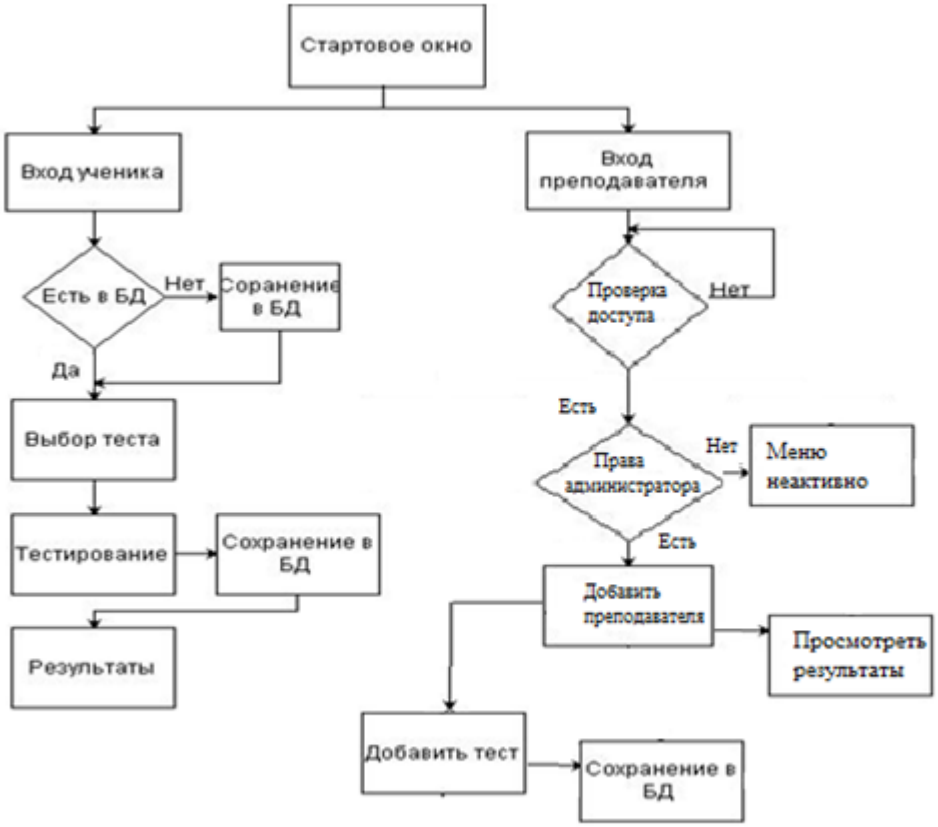

Рисунок 3.5 – Блок-схема работы программы

Чтобы лучше понять работу программы рассмотрим ее исходя из приведенной блок-схемы, изображенной на рисунке 3.5. Программное обеспечение делится на две части – для ученика и для преподавателя. Как видно из блок-схемы доступы к разным частям программы разграничены. Ученик при входе в систему автоматически регистрируется, если заходит первый раз. Далее выбирает тест, затем проходит тестирование. Данные по каждому вопросу сохраняются в БД, а в конце выводится итоговая таблица с результатами тестирования. Преподаватель при входе в систему проходит проверку на наличие данных в БД. После успешного входа происходит проверка на права доступа. Если таковыми правами данный пользователь обладает, то доступны все меню в окне, иначе меню «Добавить преподавателя» становится неактивным. Преподаватель может добавить или изменить тест и

сохранить его в БД, также есть возможность посмотреть результаты тестирования учеников при помощи установки, определенные фильтров.

3.6 Описание функциональности автоматизированной информационной системы тестирования с элементами геймификации для школьников

После запуска приложения происходит проверка соединения с базой данных. Если соединение установлено, то появляется стартовое окно, изображенное на рисунке 3.6, с активными кнопками «Ученик» и «Преподаватель», а также с надписью в правом верхнем углу «Connection: OK». В противном случае – кнопки будут неактивными, а надпись – «Connection: FAIL». Далее есть два варианта работы – для учащихся и для преподавателей.

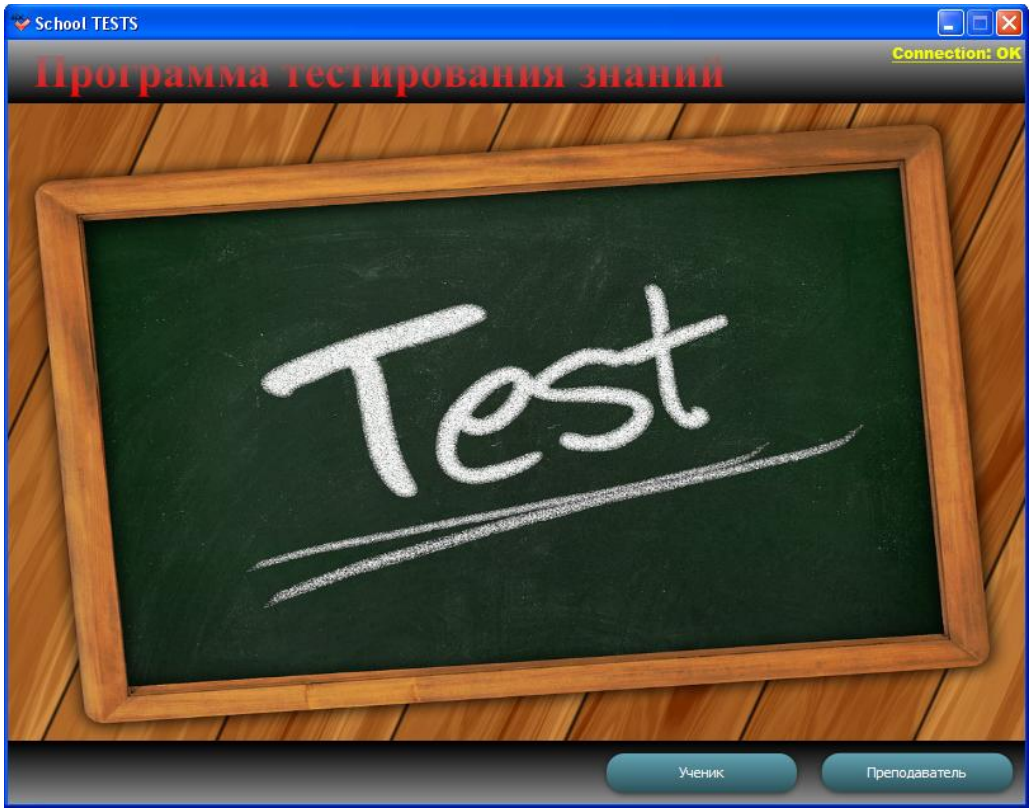

Рисунок 3.6 – Стартовое окно программы

Рассмотрим сначала вход для учащихся. По нажатию на кнопку «Ученик» открывается модальное диалоговое окно входа (рисунок 3.7).

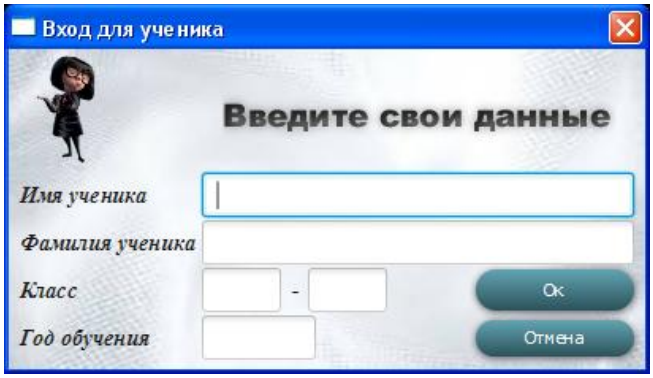

Рисунок 3.7 – Окно входа в программу для учащегося

В этом окне необходимо заполнить все поля. Если какое либо поле не заполнено, то после нажатия на кнопку «Ок» появиться диалоговое окно с сообщение-предупреждением, которое изображено на рисунке 3.8.

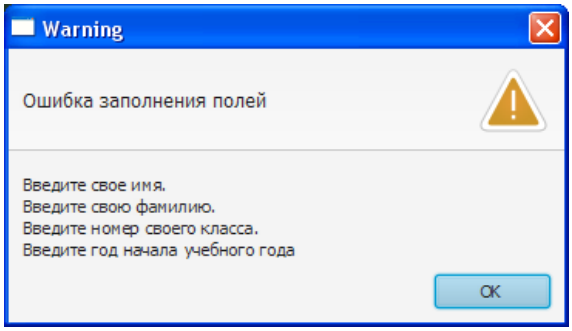

Рисунок 3.8 – Диалоговое окно с сообщением-предупреждением

После успешного ввода в программу появляется окно выбора тестов, приведенное на рисунке 3.9.

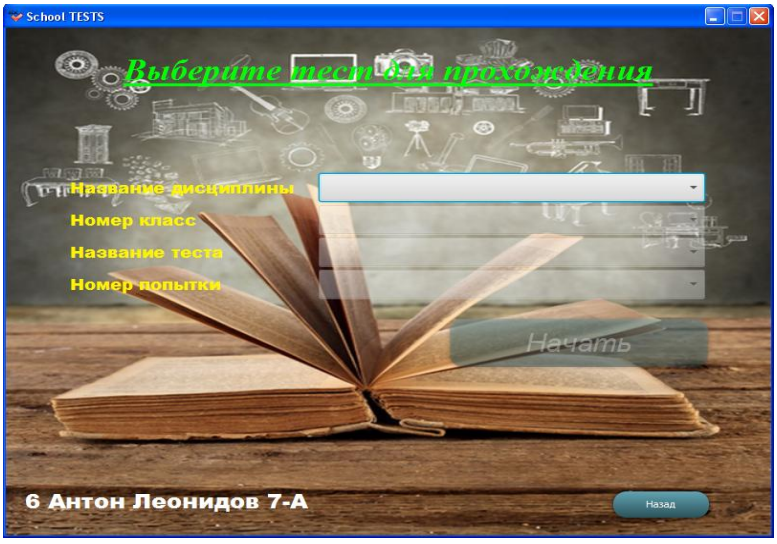

Рисунок 3.9 – Окно выбора теста

Здесь необходимо выставить фильтр для начала тестирования. А именно: выбрать дисциплину, номер класса, название теста и номер попытки. Идея реализации номера попытки заключается в том, чтобы, если необходимо, давать учащимся делать пробные попытки тестирования. А, к примеру, на попытке №5 необходимо пройти основное тестирование на оценку. Это будет зависеть от преподавателя. После выбора всех параметров фильтра активируется кнопка «Начать», которая запускает сам тест, приведенный на рисунке 3.10.

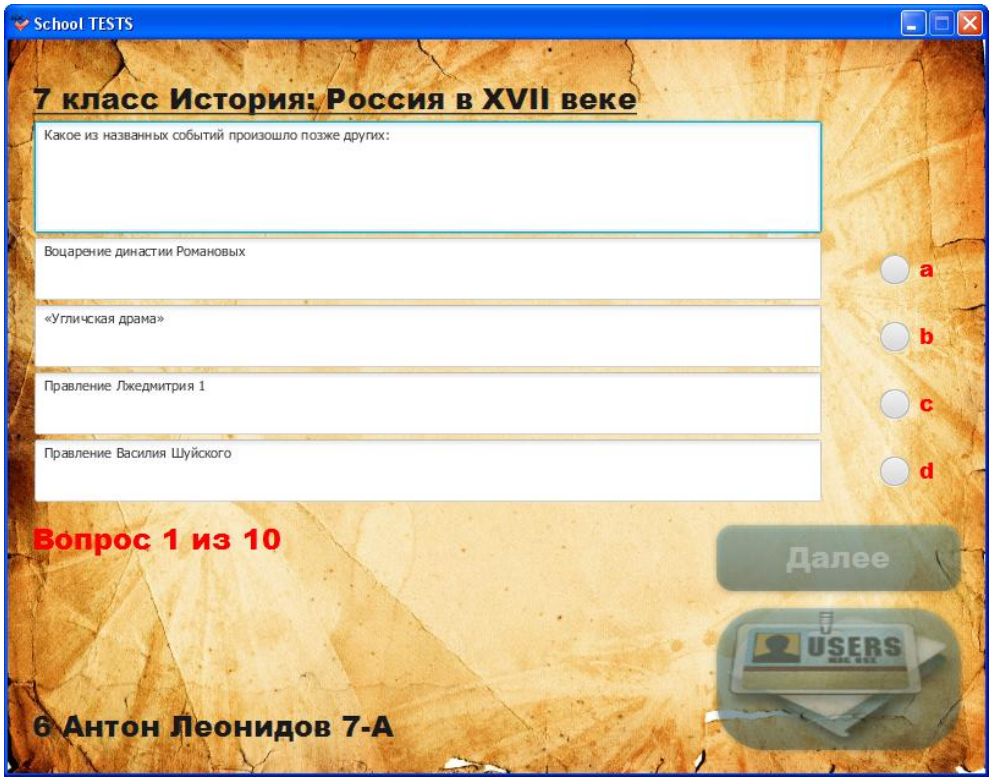

Рисунок 3.10 – Окно тестирования

На рисунке 3.10 видно, что выбрана дисциплина – История, 7 класс, название теста – Россия в XVII веке. Также виден текущий вопрос. После выбора одного из предложенных вариантов ответов активируются две кнопки – «Далее» и кнопка с иконкой «USERS».

По нажатию на кнопку «Далее» просто занесет результат на первый вопрос в БД и отобразится следующий вопрос.

По нажатию на кнопку «USERS» откроется модальное диалоговое окно изображенное на рисунке 3.11. Как видно из рисунка в окне присутствует

таблица с двумя записями. Это значит что на текущий вопрос в момент, когда Антон Леонидов нажал на кнопку «USERS» ответило 2 человека и оба они ответили не верно так как в колонке «Результат» у обоих стоит значение 0.

Можно сказать, что «изюминкой» АИС является кнопка «USERS», которая, как раз, и должна заинтересовать учащихся активно участвовать в прохождении тестирования в виду ввода в ход тестирования своего рода соревнования.

В этом же окне по нажатию на кнопку «Обновить» можно посмотреть кто еще ответил на это вопрос – таблица обновится и выведутся повторно результаты.

По нажатию на кнопку «Закрыть» окно закроется и в главном окне появится следующий вопрос.

Рисунок 3.11 – Модальное диалоговое окно с результатами на первый вопрос

После того, как учащийся ответит на все вопросы появится надпись с результатами. Данное окно приведено на рисунке 3.12.

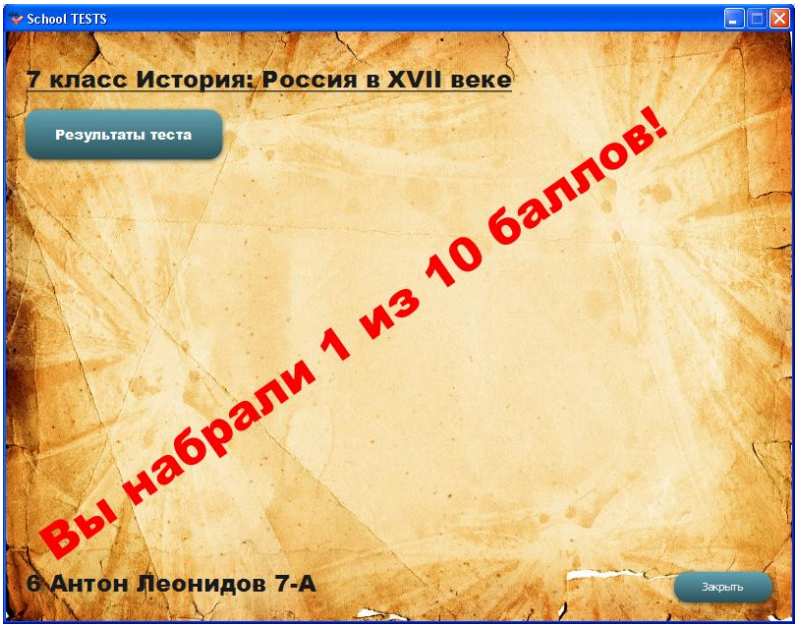

Рисунок 3.12 – Окно тестирования после прохождения теста

Как видно из рисунка 3.12 учащийся ответил только на 1 из 10 вопросов. Также есть функция посмотреть результаты всех учащихся, проходивших данный тест.

По нажатию на кнопку «Результаты теста» появится модальное диалоговое окно с общими результатами тестирования (рисунок 3.13).

Рисунок 3.13 – Результаты тестирования учащихся

Также, как и в случае, когда во время тестирования можно было посмотреть результаты на текущий вопрос, можно обновить значение для того случая, когда еще кто-то не успел завершить выполнение тестирования.

Теперь рассмотрим функционал преподавателя.

После нажатия на кнопку «Преподаватель» откроется модальное диалоговое окно входа для преподавателя (рисунок 3.14).

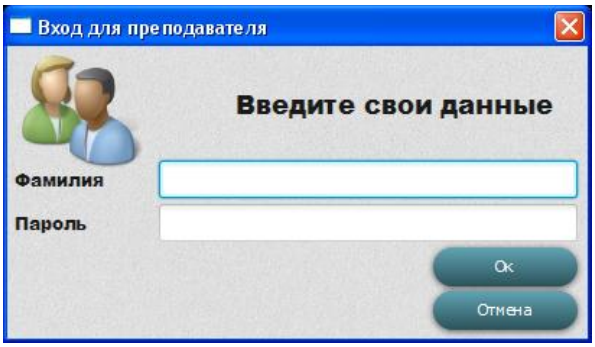

Рисунок 3.14 – Диалоговое окно входа для преподавателя

Также, как и в случае с учащимися, если не ввести все данные для входа и нажать на кнопку «Ок», то выведется окно с предупреждением.

После ввода данных и нажатия на кнопку «Ок» данные проверяются в базе и если такие данные есть, то откроется окно с выбором, показанное на рисунке 3.16, иначе появится модальное окно с сообщением, что запись не найдено и вход не произойдет.

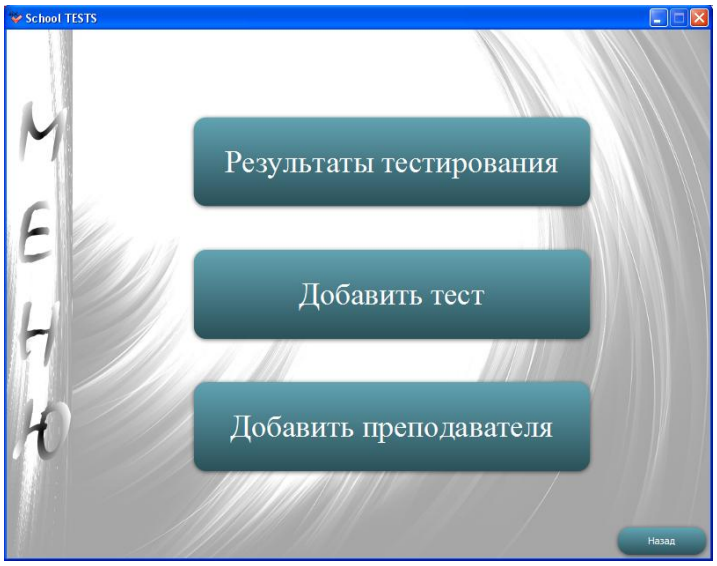

Рисунок 3.16 – Окно с выбором действий

Если у преподавателя есть права администратора, то ему также доступна кнопка-меню «Добавить преподавателя», иначе данная кнопка будет неактивна.

Рассмотрим все опции по порядку.

По нажатию на кнопку «Результаты тестирования» откроется окно, изображенное на рисунке 3.17.

В данном окне находится пустая таблица, а также фильтр-опции с помощью которых нужно установить значение для поиска результатов тестирования. Необходимо последовательно выбирать необходимые значения для того, чтобы активировались последующие. Опция даты сделана в двух вариантах – можно выбрать значение из списка значений, а можно установить вручную путем нажатия на календарь и выбрать нужную дату.

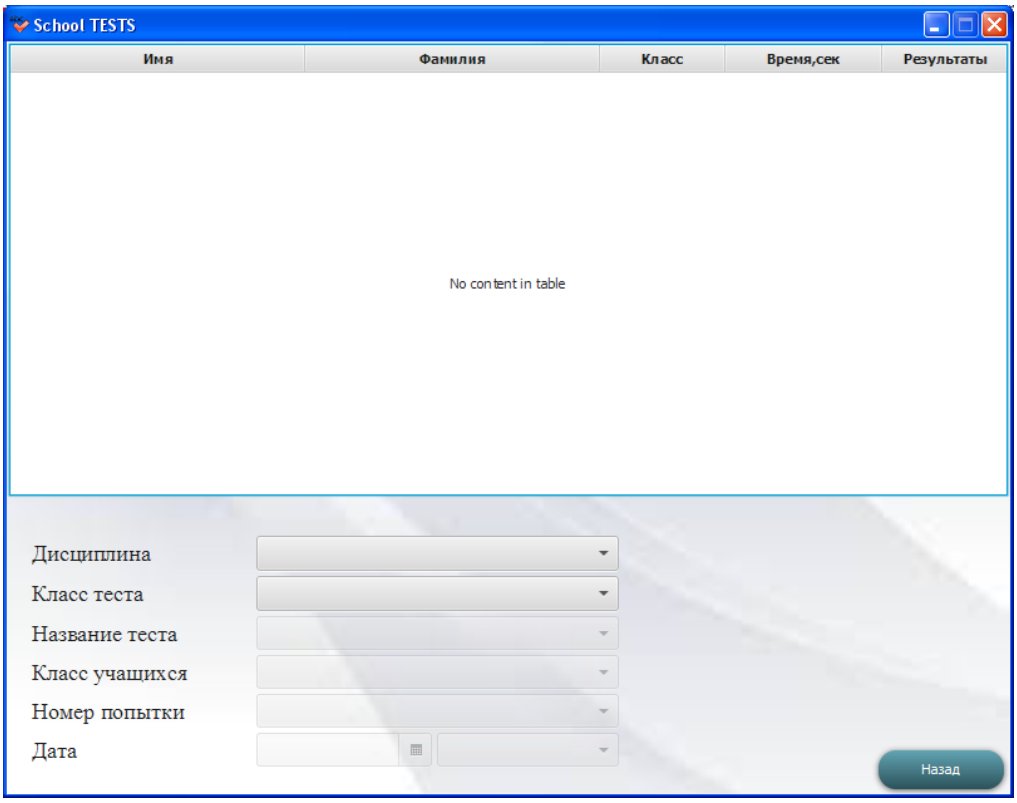

Рисунок 3.17 – Окно с результатами тестирования

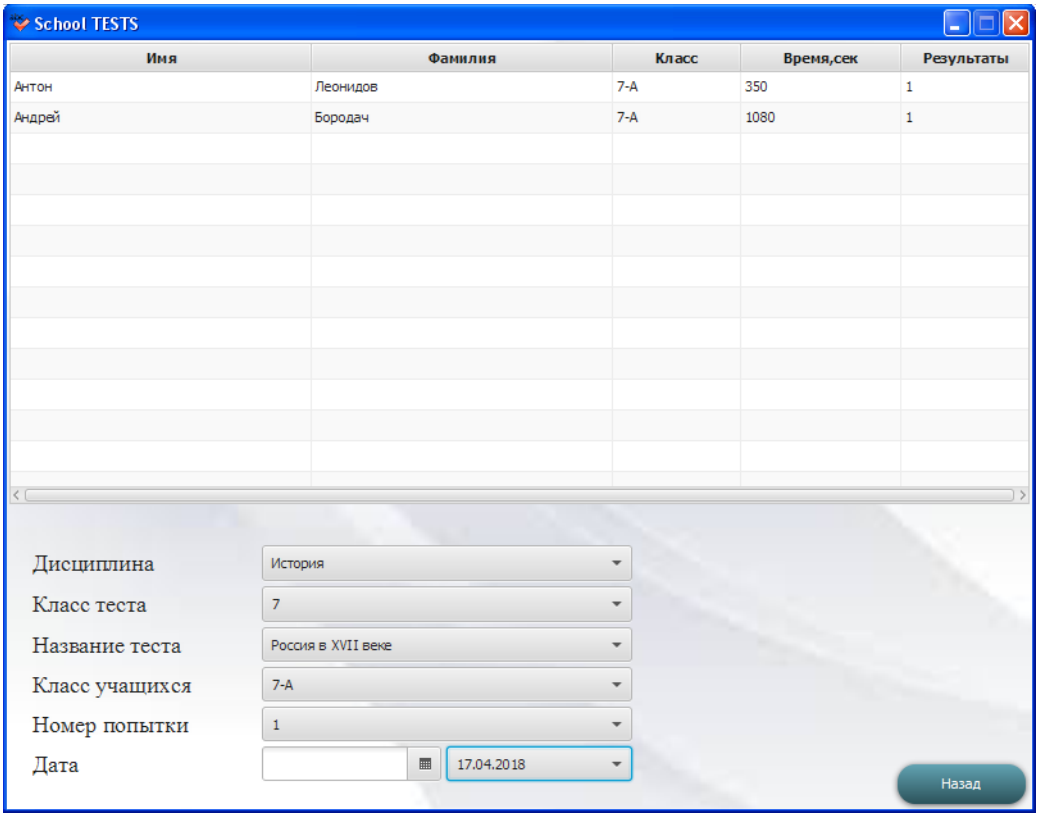

Рисунок 3.18 – Окно с результатами тестирования с установленным фильтром

На рисунке 3.18 показан пример окна с результатами тестирования и выбранным фильтром. Как видно из таблицы на рисунке 3.18 тест проходило 2 человека, и они набрали по одному баллу.

Если вернуться в основное меню для преподавателей и нажать на кнопку «Добавить тест», то откроется окно, приведенное на рисунке 3.19.

Необходимо выбрать предмет и класс, после чего активируется кнопка «Добавить» для поля названия теста. Если тест еще не создан, необходимо вписать название теста в поле и нажать на кнопку «Добавить» напротив этого поля.

Таким же образом добавляется и новый предмет, и новый класс. Также можно и удалить какой-либо предмет, класс и название. Но надо помнить, что все удаления происходят каскадно, то есть если удалить какой-либо предмет, то удалятся все названия тестов и тесты для этого предмета.

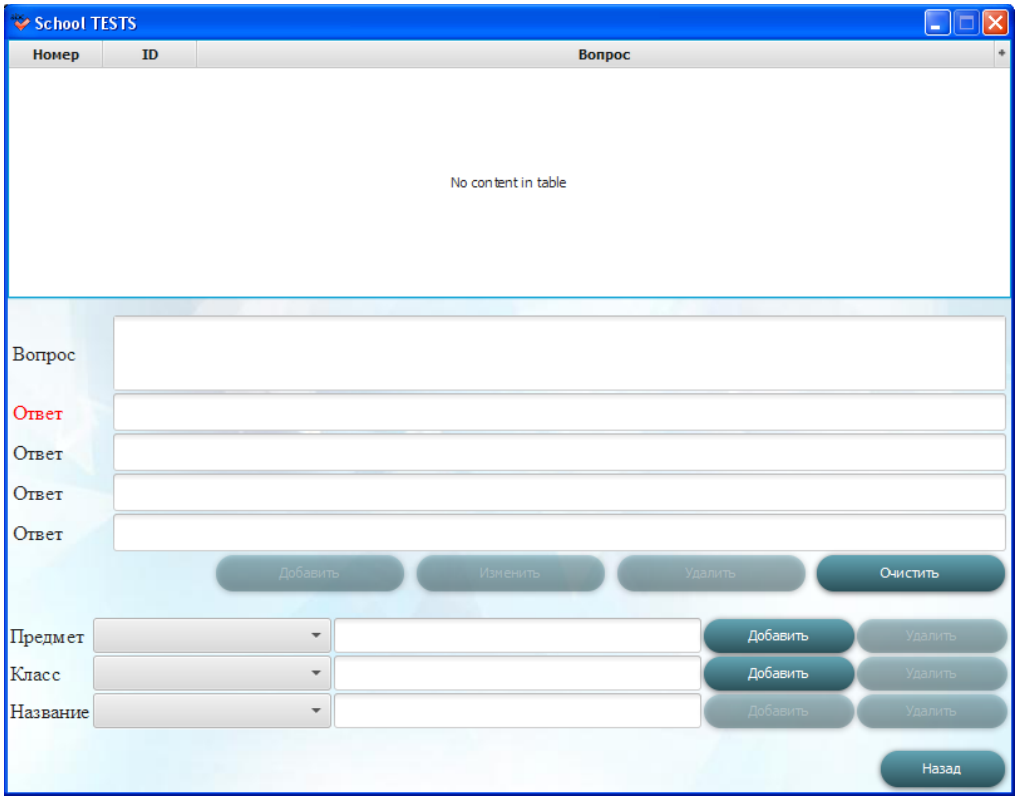

Рисунок 3.19 – Окно добавления тестов

После выбора всех трех опций активируется таблица, расположенная вверху окна. Если для данного тесте уже есть какие-либо записи (набор вопросов), то они отобразятся в таблице.

Чтобы добавить вопрос необходимо вписать в поле «Вопрос» сам вопрос, а в поля «Ответ» - варианты ответов.

В поле «Ответ» выделенное красным цветов заносится правильный ответ на поставленный вопрос. Минимально нужно внести 2 ответа, максимально 4.

По нажатию на кнопку «Добавить», расположенную под четвертым вариантом ответа, запись добавиться в БД и отобразится в таблице.

Чтобы изменить какую-либо запись необходимо выделить нужную строку в таблице, после чего все поля отобразятся в текстовых полях. Затем внести нужные изменения и нажать кнопку «Изменить».

Чтобы снова вернуться к добавлению записей, необходимо снять выделения строки в таблице путем нажатия на клавишу ESCAPE – кнопка «Изменить» деактивируется, а кнопка «Добавить» активируется.

Для удаления записи нудно выбелить строку в таблице и нажать на кнопку «Удалить».

Кнопка «Очистить» очищает все текстовые поля добавления вопроса к тестам.

Если в главном меню нажать на кнопку «Добавить преподавателя», то появится окно, изображенное на рисунке 3.20.

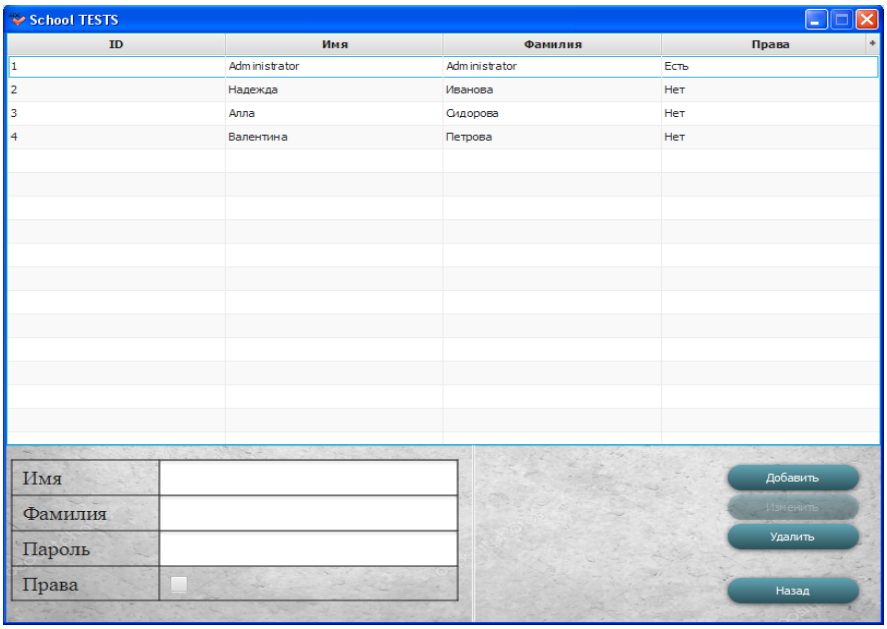

Рисунок 3.20 – Окно добавления преподавателя

К этому окну есть права только у ограниченного круга людей. Такие права устанавливаются путем установки флажка в строке «Права».

Также в данном окне в таблице показан список всех преподавателей. Также видно у кого есть права на добавление преподавателя.

Чтобы добавить преподавателя необходимо заполнить все поля, а затем нажать на кнопку «Добавить». Запись добавится в БД и отобразится в таблице.

Чтобы изменить запись необходимо в таблице выбелить нужную строку, затем внести изменения в текстовые поля и нажать «Изменить».

Чтобы снова вернуться к добавлению преподавателя необходимо снять выделение строки в таблице путем нажатия клавиши ESCAPE – кнопка «Добавить» активируется, а кнопка «Изменить» деактивируется.

Чтобы удалить преподавателя из БД необходимо выбелить запись в таблице и нажать на кнопку «Удалить».

3.7 Тестирование программного проекта

3.7.1 Выбор методов тестирования программного продукта

Тестирование – это процесс исполнения программы с целью обнаружения ошибок.

Для данного проекта выполним модульное тестирование.

Суть модульного тестирования заключается в «изолированной» проверке каждого элемента по отдельности. Такая проверка производится в отдельной закрытой среде.

Если говорить, в частности, о данном проекте, то необходимо выполнить модульное тестирование каждого метода в отдельности. Такое тестирование придает уверенности в работоспособности кода.

В стандартной библиотеке Java поставляется фреймворк для модульного тестирования программного обеспечения. В частности будет использоваться библиотека JUnit 4.12.

Данный проект основан на работе программного обеспечения с базой данных. Все добавления, изменения, удаления какой либо информации из программного обеспечения влекут за собой и удаление ее из БД. Поэтому нужно быть уверенными в правильно написанных методах-запросах в БД для корректной работы с документами. Следовательно, необходимо полностью покрыть тестами классы, использующиеся для работы с БД.

Также необходимо провести системное тестирование. Суть данного тестирования заключается в проверке ожидаемых для пользователя условий работы ПО.

3.7.2 Описание программного кода тестирования АИС

В среде разработки NetBeans есть отдельный пакет test для модульного тестирования. Сгенерируем модульные тесты для всех классов, работающий с

БД. Суть проведения тестирования заключается в том, что необходимо сравнивать возвращаемое методом значение с заведомо предполагаемым результатом. Для наглядности приведем код модульного тестирования для нескольких методов.

```
@Test
  public void testEditQuestions() {
    System.out.println("editQuestions");
    Questions quest = new Questions();
    quest.setIdQuestions(2L);
    quest.setQuestion("В каких единицах измеряется механическая работа?");
    quest.setAnswerTrue("Джоуль");
    quest.setAnswerTwo("Ватт");
    quest.setAnswerThree("Ньютон");
    quest.setAnswerFour("Паскаль");
    quest.setNumberQuestion(2);
    TestName tn = new TestName(5L);
    quest.setTestNameid(tn);

    boolean expResult = true;
    boolean result = WorkViewDbRequest.editQuestions(quest);
    assertEquals(expResult, result);
  }

  @Test
  public void testFindDiscipline() {
    System.out.println("findDiscipline");
    String name = "Физика";
    Discipline expResult = new Discipline(1L, "Физика");

    Discipline result = WorkViewDbRequest.findDiscipline(name);
    assertEquals(expResult, result);
  }
```

Показателем завершения теста является сравнение ожидаемого значения с фактическим. Это сравнение происходит в методе assertEquals(expResult, result). И если оба значения равны, то тест пройден успешно.

Для наглядности приведем скриншот результатов тестов, приведенных выше.

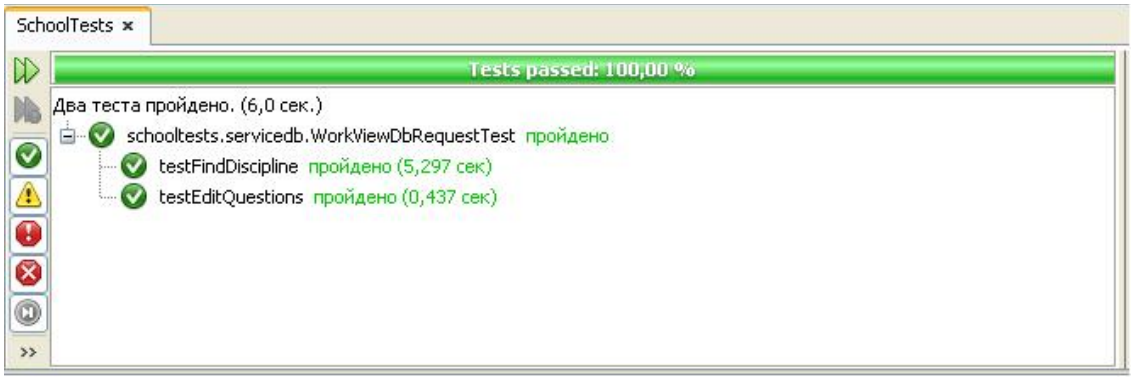

Рисунок 3.21 – Результат модульного тестирования методов

editQuestions() и findDiscipline() класса WorkViewDbRequest с заведомо верными ожидаемыми результатами

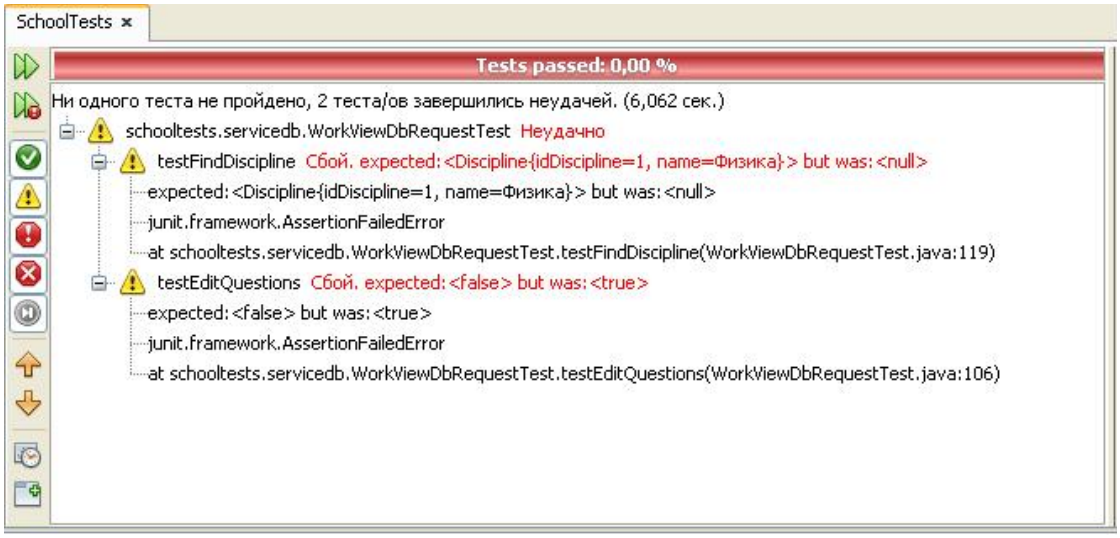

Рисунок 3.22 - Результат модульного тестирования методов

editQuestions() и findDiscipline() класса WorkViewDbRequest с заведомо ложными ожидаемыми результатами

Приведем пример скриншота тестирования тестирования методов editQuestions() и findDiscipline() класса WorkViewDbRequest с заведомо ложными ожидаемыми параметрами.

Сведем все результаты тестов в таблицу №3.3.

Таблица 3.3 – Результаты модульного тестирования.

Название класса	Название метода	Результат
JPQLNativeQuery	getCurrentValueId	+
		100%
JPQLQueryEntity	checkPassword	+
		100%
QueryCriteriaBuilder	findAllEntityCritBuilder	+
	findWithFilterEntityCritBuilder 3args_1	+
	findWithFilterEntityCritBuilder 3args_2	+
		100%
SingleEntity	addEntity	+
	findEntity	+
	removeEntity	+
		100%
WorkViewDbRequest	editTeacher	+
	findTeacher	+
	editQuestions	+
	findDiscipline	+
	findClassSchool	+
	findTestName 3args	+
	findTestName String_int	+
	findQuestions	+
	findClassLearner	+
	findAllMadeTry	+
	findAllDateMadeTry	+
	findLearner	+
	findTry	+
	findAnswerTry	+
	findSumAnswerTry	+
		100%
DateView	dateToString	+
	stringToDate	++
	localDateToDate	
		100%

Результаты модульного тестирования во всех случаях показали соответствие ожидаемого результата к фактическому. Поэтому, можно считать, что модульное тестирование данное программное обеспечение успешно прошло.

Также было проведено системное тестирование. В ходе данного тестирование была проведена работа со всеми окнами программы. А именно:

- неоднократно пройдены тесты;

- формировались результаты тестирования;
- добавлялись, редактировались и удалялись преподаватели;
- формировались, редактировались и удалялись наборы тестов;
- проверялась корректность заполнения текстовых полей.

В целом можно сказать, что ПО полностью готово к работе

Выводы по главе 3

В данной главе была выбрана архитектура АИС, выбрана технология разработки АИС, также обоснован выбор самой СУБД. Как следствие была построена физическая модель данных автоматизированной информационной системы тестирования с элементами геймификация для школьников.

Реализовано средствами программного обеспечения в СУБД postgreSQL база данных с именем SchoolTests, в которой создано 7 таблиц. Данные таблицы полностью покрывают все нужды автоматизированной системы.

На основании всех требований было разработано программное обеспечение для управление данной информационной системой. Полностью описан состав программного продукта, а также подробно расписан функционал ПО.

В ходе проведения тестирования всей информационной системы недочетов выявлено не было, поэтому можно считать работу успешно выполненной.

ЗАКЛЮЧЕНИЕ

В ходе проведения данной работы была спроектирована и разработана автоматизированная информационная система тестирования с элементами геймификации для школьников.

Сформулированы цель, задачи проектирования, требования к разрабатываемой ИС и определены потребности конечных пользователей. Проведен анализ и выбор проектных решений в рамках информационного и программного обеспечений.

В процессе работы достигнуты следующие поставленные задачи:

1. Произведен анализ предметной области – процесс организации проведения тестирования учащихся в общеобразовательных учебных учреждениях.

2. Выявлена проблематика, характерная для данной области.

3. Обоснована необходимость разработки ИС.

4. Были исследованы средства для разработки и проектирования ИС, выбраны необходимые с учетом потребностей.

5. Была разработана рабочая информационная система с необходимым функционалом.

В процессе достижения цели данной работы было произведено концептуальное, логическое и физическое проектирование базы данных и сама ее реализация, так же было разработано клиентское приложение для работы с БД. СУБД и программное обеспечение выбраны на бесплатной основе, а именно СУБД – PostgreSQL 9.6 и язык программирования Java 1.8, а также бесплатная среда разработки ПО NetBeans 8.2.

Результатом работы является информационная система, применение которой позволит автоматизировать процесс проведение тестирования учащихся, завлечение в сам процесс тестирования, а также автоматизировать подсчет результатов тестирования, а значит и свести внесение ошибки в результаты к нулю.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

Научная и методическая литература

1. Буч, Г. Объектно-ориентированный анализ и проектирование с примерами приложений на C++, 2-е изд. – М.: Бином, СПб: Невский диалект, 1999.

2. Гонсалвес, Э. Изучаем Java EE 7. – СПб.: Питер, 2014. – 640с.: ил.

3. Грэхем, И. Объектно-ориентированные методы. Принципы и практика. Пер. с англ. – М.: Издательский дом «Вильямс». 2004. – 880 с.

4. Исаев, Г.Н. Проектирование информационных систем. Учебное пособие. - М.: Омега-Л, 2015. - 432с.

5. Карвин, Б. Программирование баз данных SQL. Типичные ошибки и их устранение / Б. Карвин. – М.: Рид Групп, 2012. – 336с. – (Профессиональные компьютерные книги).

6. Карпова, И.П. Базы данных: Учебное пособие / И.П. Карпова. - СПб.: Питер, 2013. - 240 с.

7. Карпова, И. П. Базы данных : курс лекций и материалы для практ. занятий : учеб. пособие для студентов техн. фак. / И. П. Карпова. - Санкт-Петербург : Питер, 2013. - 240 с.

8. Кириллов, В.В. Введение в реляционные базы данных. - СПб.: БХВ-Петербург, 2012. - 464 с.

9. Коваленко, В.В. Проектирование информационных систем. - М.: Форум, 2012. - 320с.

10. Машнин, Т.С. JavaFX 2.0: разработка RIA-приложений. - СПб.: БХВ-Петербург, 2012. - 320 с.: ил. - (Профессиональное программирование).

11. Татур, Ю.Г. Высшее образование: методология и опыт проектирования : учеб. пособие / Ю. Г. Татур. – Гриф УМО. – М.: Логос, 2006. – 252 с.

12. Форта, Б. SQL за 10 минут, 4-е изд.: Пер.с англ. – М.: ООО «И.Д.Вильямс», 2014. – 288с.:ил. – Парал.тит.англ.

13. Шилдт, Г. Java 8. Полное руководство, 9-е изд.: Пер. с англ. – М.: ООО «И.Д. Вильямс», 2015. – 1376с.:ил. – Парал. тит. англ.

Электронные ресурсы

14. Диаграмма классов. Режим доступа - https://ru.wikipedia.org/wiki/Диаграмма_классов

15. Диаграмма последовательности. Режим доступа - https://ru.wikipedia.org/wiki/Диаграмма_последовательности

16. Диаграмма прецедентов. Режим доступа - https://ru.wikipedia.org/wiki/Диаграмма_прецедентов.

17. Документация к Postgres Pro Standard 9.6.8.2. Режим доступа - https://postgrespro.ru/docs/postgrespro/9.6/.

18. Модель процесса разработки автоматизированной информационной системы. Режим доступа - http://rf-biz.ru/50.php.

19. Системы классификации и кодирования информации. Режим доступа - http://villian2008.narod.ru/5.htm

20. ER-Win Data Modeler. Режим доступа - https://ru.wikipedia.org/wiki/ERwin_Data_Modeler.

21. Java Database Connectivity. Режим доступа - https://ru.wikipedia.org/wiki/Java_Database_Connectivity

22. MyTestXPro. Режим доступа - http://mytest.klyaksa.net/htm/index.htm.

23. Rational Rose. Режим доступа - http://www.kpms.ru/Automatization/Rational_Rose.htm.

24. tTester. Режим доступа https://sunrav.ru/ttester.html.

25. UML. Режим доступа - https://ru.wikipedia.org/wiki/UML.

Литература на иностранном языке

26. Daniela Baglieri, Concetta Metallo, Cecilia Rossignoli, Mario Pezzillo Iacono. Information Systems, Management, Organization and Control: Smart Practices and Effects, p.304, 2014

27. Dhillon, G. (2007). Principles of Information Systems Security: Text and Cases. Hoboken, NJ: John Wiley & Sons.

28. Ebbers, H. Mastering JavaFX 8 Controls, Oracle Publishing Group, 2014.

29. Efrem, G. Mallach. Information Systems: What Every Business Student Needs to Know, p.458, 2015.

30. Hevner, Alan, Chatterjee, Samir. Design Research in Information Systems - Theory and Practice, 2010.

31. Mandic, D, Lalic, N., Bandjur, V..: Computer Aided Research in Managing Educational Process, in the book 7th WSEAS International Conference on ENGINEERING EDUCATION (EDUCATION '10 Corfu, Greece, 2010.

ПРИЛОЖЕНИЕ А

Код программы всех классов для работы с базой данных

```
public class DefaultCreateManager{

   private static EntityManagerFactory emf = null;
   private static EntityManager em = null;
   private static Map properties;

   private DefaultCreateManager() {
   }

   public static EntityManager createManager() {
      if(properties==null){
         createConnectionConfig();
      }
         emf = Persistence.createEntityManagerFactory("SchoolTestsPersist", properties);
         em = emf.createEntityManager();

      return em;
   }
   public static void closeManager() {
      if(emf.isOpen())  emf.close();
      if(em.isOpen())  em.close();
   }
   /**
    * Метод инициализации данных для соединения с базой данных
    */
   private static void createConnectionConfig(){
      properties = new HashMap();
      String URL  = null;
      String USER  = null;
      String PASSWORD  = null;
      Path path = Paths.get("config.conf");
      try {
         List<String>  list = Files.readAllLines(path);
         URL  = "jdbc:postgresql://" + list.get(0) + "/" + list.get(1);
         USER  = list.get(2);
         PASSWORD  = list.get(3);
      } catch (IOException  ex) {
         Logger.getLogger(DefaultCreateManager.class.getName()).log(Level.SEVERE,  null, ex);
      }
      //properties.put(TRANSACTION_TYPE,  PersistenceUnitTransactionType.RESOURCE_LOCAL.name());
      properties.put(JDBC_DRIVER,  "org.postgresql.Driver");
      properties.put(JDBC_URL,  URL);
      properties.put(JDBC_USER,  USER);
      properties.put(JDBC_PASSWORD,  PASSWORD);
      properties.put(SCHEMA_GENERATION_DATABASE_ACTION,  "create");
      //properties.put(LOGGING_LEVEL,  "FINE");
      //properties.put(LOGGING_TIMESTAMP,  "false");
      //properties.put(LOGGING_THREAD,  "false");
      //properties.put(LOGGING_SESSION,  "false");
      //properties.put(TARGET_SERVER,  TargetServer.None);
   }
}

/**
 * Класс шифрования информации
 * @author User
 */
```

```
public class Encryption {

    /**
     * Метод шифрования информации
     * @param info информация подлежащая шифрованию
     * @param algoritm алгоритм шифрования данных
     * @return Хеш-представление информации
     */
    public static String md5HashCoding(String info, String algoritm){
        //Создание ссылки на класс шифрования
        MessageDigest md = null;
        try {
            //Инициализация объекта шифрования с конкретным алгоритмом
            md = MessageDigest.getInstance(algoritm);
        } catch (NoSuchAlgorithmException ex) {
            Logger.getLogger(Encryption.class.getName()).log(Level.SEVERE, null, ex);
            System.err.println("ERROR: md5HashCoding()");
        }
        //Получение массива байт шифрования
        byte[] bytes = md.digest(info.getBytes());
        StringBuilder builder = new StringBuilder();
        for(byte b : bytes) {
            //Приведение в 16тиричное значение массива байт
            builder.append(String.format("%02X", b));
        }
        return builder.toString();
    }

}

/**
 * Класс для работы с "родными" SQL-запросами
 * @author Admin
 */
public class JPQLNativeQuery {

    private static EntityManager em;

    /**
     * Метод получения сгенерированного базой id при помощи "родного" SQL-запроса
     * @param sqlString - родной SQL-запрос
     * @return значение последнего сгенерированного базой id
     */
    public static long getCurrentValueId(String sqlString) {
        em = DefaultCreateManager.createManager();
        Query query = em.createNativeQuery(sqlString);
        long id = Long.parseLong(String.valueOf(query.getResultList().get(0)));
        DefaultCreateManager.closeManager();
        return id;
    }
}

public class JPQLQueryEntity {

    private EntityManager em;
    private Query query;


    /**
     * Метод для входа в программу для администратора/преподавателя.
     * Если введенные данные верны, то вернется true
     * @param lastName фамилия пользователя
```

```
     * @param password пароль пользователя
     * @return 1 - если есть права администратора, -1 - если нет прав администратора,
     * 0 - если нет доступа к программе.
     */
    public byte checkPassword(String lastName, String password){

        em = DefaultCreateManager.createManager();

        String checkQuery = "SELECT e FROM Teacher e where e.lastName = :lname AND e.password = :pass";
        query = em.createQuery(checkQuery);
        query.setParameter("lname", lastName);
        query.setParameter("pass", password);
        //Запрос вернет объект типа Teacher
        //Если поля объекта равны запрашиваемым полям, то вернем true
        try {
        Teacher tech = (Teacher)query.getSingleResult();

        DefaultCreateManager.closeManager();

        if(tech.getLastName().equals(lastName) && tech.getPassword().equals(password)
              && tech.getAdministrator()) {
            return 1;
        } else if(tech.getLastName().equals(lastName) && tech.getPassword().equals(password)) {
            return -1;
        }
        } catch (Exception exc) {
            System.err.println(exc.getMessage());
            System.err.println("ERROR -> checkPassword()");
            //return 0;
        }
        return 0;
    }
}


/**
 * Класс для работы с объектно-ориентированными запросами
 * @author Admin
 * @param <T> запрашиваемая сущность
 */
public class QueryCriteriaBuilder<T> {
    private static EntityManager em;
    private static CriteriaBuilder builder;
    private static CriteriaQuery criteriaQuery;
    private static Root root;
    public QueryCriteriaBuilder() {
    }
    /**
     * Метод находжения всех объектов заданной сущности
     * @param clazz экземпляр класса (сущности), с которым нужно работать
     * @return объект типа List с типом сущности T
     */
    public static <T> List<T> findAllEntityCritBuilder(Class clazz){
        createBuilder(clazz);
        //Используется для установки типа списка запроса
        criteriaQuery.select(root);
        Query query = em.createQuery(criteriaQuery);
        List<T> list = query.getResultList();
        DefaultCreateManager.closeManager();
```

```
        return list;
    }
    /**
     * Метод нахождения объектов по заданному фильтру
     * @param clazz экземпляр класса (сущности), с которым нужно работать
     * @param fieldName имя поля для которого устанавливается фильтрация
     * @param findName текстовое значение, по которому происходит фильтрация
     * @return коллекция типа List типа T с найденными значениями
     */
    public static <T> List<T> findWithFilterEntityCritBuilder(Class clazz, String fieldName, String findName){
        createBuilder(clazz);
        //Используется для установки типа списка запроса
        criteriaQuery.select(root);
        Predicate field = builder.equal(root.get(fieldName), findName);
        criteriaQuery.where(field);
        Query query = em.createQuery(criteriaQuery);
        List<T> list = query.getResultList();
        DefaultCreateManager.closeManager();
        return list;
    }
    /**
     * Метод нахождения объектов по заданному фильтру
     * @param clazz экземпляр класса (сущности), с которым нужно работать
     * @param fieldName имя поля для которого устанавливается фильтрация
     * @param findNumber числовое значение, по которому происходит фильтрация
     * @return коллекция типа List типа T с найденными значениями
     */
    public static <T> List<T> findWithFilterEntityCritBuilder(Class clazz, String fieldName, int findNumber){
        createBuilder(clazz);
        //Используется для установки типа списка запроса
        criteriaQuery.select(root);
        Predicate field = builder.equal(root.get(fieldName), findNumber);
        criteriaQuery.where(field);
        Query query = em.createQuery(criteriaQuery);
        List<T> list = query.getResultList();
        DefaultCreateManager.closeManager();
        return list;
    }
    /**
     * Метод создания объектов Criteria API
     * @param clazz экземпляр класса (сущности), с которым нужно работать
     */
    private static void createBuilder(Class clazz) {
        em = DefaultCreateManager.createManager();
        //Главная фабрика запросов
        builder = em.getCriteriaBuilder();
        //Используется для создания объекта запроса
        criteriaQuery = builder.createQuery(clazz);
        //Используется для установки корня запроса
        root = criteriaQuery.from(clazz);
    }
}

/**
```

```
 *Работа с одиночными сущностями
 * @author Admin
 */
public class SingleEntity<E,  T> {

    private static EntityManager em;
    private static EntityTransaction tx;
    /**
     * Добавление данных в сущность
     * @param e добавляемая сущность
     * @return true если сущность добавлена;
     * false если сущность не добавлена
     */
    public static <E> boolean addEntity(E e){
        em = DefaultCreateManager.createManager();
        //Обеспечение постоянства сущности в базе данных
        tx = em.getTransaction();
        try {
        tx.begin();
        em.persist(e);
        em.flush();//синхронизация контента с БД
        tx.commit();
        } catch (Exception exc) {
            tx.rollback();
            System.err.print("ERROR  -> addEntity(E e)");
            return false;
        }
        DefaultCreateManager.closeManager();
        return true;
    }
    /**
     * Поиск сущности типа E по ее идентификатору (id). Если сущность не найдена
     * вернется null
     * @param e объект сущности для которой выполняется поиск
     * @param id идентификатор для поиска
     * @return объект того же типа E. Если сущность не найдена вернется null
     */
    public static <E>  E findEntity(E e, long id) {
        em = DefaultCreateManager.createManager();
        E ee = em.find((Class<E>)  e.getClass(), id);
        DefaultCreateManager.closeManager();
        return ee;
    }
    /**
     * Удаление сущности по значению id
     * @param e заданная сущность
     * @param id идентификатор для поиска
     * @return true если сущность найдена и удалена;
     * false если сущность не найдена
     */
    public static <E> boolean removeEntity(E e, long id){
        boolean check = false;
        em = DefaultCreateManager.createManager();
        //Обеспечение постоянства сущности в базе данных
```

```
        tx = em.getTransaction();
        try {
            tx.begin();
            E b = em.find((Class<E>) e.getClass(), id);
            em.remove(b);
            //em.flush();//синхронизация контента с БД
            tx.commit();
            check = true;
        } catch (Exception exc) {
            tx.rollback();
            System.err.println(exc.getMessage());
            //System.out.println(exc.getCause().getMessage());
            System.err.println("ERROR -> removeEntity(E e, int id)");
        } finally {
              DefaultCreateManager.closeManager();
              return check;
            }
    }
}
/**
 * Класс для работы с базой данных
 * @author User
 */
public class WorkViewDbRequest {
    /**
     * Метод иземения данных об преподавателе
     * @param tech измененный объект
     * @return true - если изменение прошло успешно, иначе - false
     */
    public static boolean editTeacher(Teacher tech){
        boolean check = false;
        EntityManager em = DefaultCreateManager.createManager();
        EntityTransaction tx = em.getTransaction();
        try {
            tx.begin();
            Teacher t = em.find(Teacher.class, tech.getIdTeacher());
            em.merge(tech);
            tx.commit();
            check = true;
        } catch (Exception exc) {
            tx.rollback();
        }
        DefaultCreateManager.closeManager();
    return check;
    }
    /**
     * Метод нахождения учителя(преподавателя) по заданному условию
     * @param lastName фамилия преподавателя
     * @param pass пароль преподавателя
     * @return объект класса Teacher, null - если не найден
     */
    public static Teacher findTeacher(String lastName, String pass){
        EntityManager em = DefaultCreateManager.createManager();
        //Фабрика по созданию экземпляров CriteriaQuery, а также всех видов выражений запросов
```

```
    CriteriaBuilder cb = em.getCriteriaBuilder();
    CriteriaQuery<Teacher> cq = cb.createQuery(Teacher.class);
    Root<Teacher> rootTeacher = cq.from(Teacher.class);
    //---Predicate - Утверждение (условие на фильтрацию строк по заданному критерию)
    //Условие сравнения строки из поля name и заданной строки
    Predicate lastNameTeach = cb.equal(rootTeacher.get(Teacher_.lastName), lastName);
    Predicate passTeach = cb.equal(rootTeacher.get(Teacher_.password), pass);
    cq.where(lastNameTeach, passTeach);
    //Типизированный запрос
    TypedQuery<Teacher> tq = em.createQuery(cq);
    //Получение id со значение из БД
    Teacher teach = null;
    try {
        teach = tq.getSingleResult();
    } catch(Exception exc) {

    }
    DefaultCreateManager.closeManager();
    return teach;
}
/**
 * Метод иземения вопроса в тестах
 * @param quest измененный объект
 * @return true - если изменение прошло успешно, иначе - false
 */
public static boolean editQuestions(Questions quest){
    boolean check = false;
    EntityManager em = DefaultCreateManager.createManager();
    EntityTransaction tx = em.getTransaction();
    try {
        tx.begin();
        Questions t = em.find(Questions.class, quest.getIdQuestions());
        em.merge(quest);
        tx.commit();
        check = true;
    } catch (Exception exc) {
        tx.rollback();
    }
    DefaultCreateManager.closeManager();
return check;
}
/**
 * Метод нахождения объекта класса Discipline по его названию (имени)
 * @param name название дисциплины
 * @return объект класса Discipline. null - если объект не найден
 */
public static Discipline findDiscipline(String name){
    EntityManager em = DefaultCreateManager.createManager();
    //Фабрика по созданию экземпляров CriteriaQuery, а также всех видов выражений запросов
    CriteriaBuilder cb = em.getCriteriaBuilder();
    CriteriaQuery<Discipline> cq = cb.createQuery(Discipline.class);
    Root<Discipline> rootDiscipline = cq.from(Discipline.class);
    //---Predicate - Утверждение (условие на фильтрацию строк по заданному критерию)
    //Условие сравнения строки из поля name и заданной строки
```

```
    Predicate nameDiscipline = cb.equal(rootDiscipline.get(Discipline_.name), name);
    cq.where(nameDiscipline);
    //Типизированный запрос
    TypedQuery<Discipline> tq = em.createQuery(cq);
    //Получение id со значение из БД
    Discipline disc = null;
    try {
        disc = tq.getSingleResult();
    } catch(Exception exc) {
    }
    DefaultCreateManager.closeManager();
    return disc;
}
/**
 * Метод нахождения объекта класса ClassSchool по его номеру
 * @param number номер класса
 * @return объект класса ClassSchool, null - если объект не найден
 */
public static ClassSchool findClassSchool(int number){
    EntityManager em = DefaultCreateManager.createManager();
    //Фабрика по созданию экземпляров CriteriaQuery, а также всех видов выражений запросов
    CriteriaBuilder cb = em.getCriteriaBuilder();
    CriteriaQuery<ClassSchool> cq = cb.createQuery(ClassSchool.class);
    Root<ClassSchool> rootClassSchool = cq.from(ClassSchool.class);
    //---Predicate - Утверждение (условие на фильтрацию строк по заданному критерию)
    //Условие сравнения строки из поля name и заданной строки
    Predicate nameClassSchool = cb.equal(rootClassSchool.get(ClassSchool_.number), number);
    cq.where(nameClassSchool);
    //Типизированный запрос
    TypedQuery<ClassSchool> tq = em.createQuery(cq);
    //Получение id со значение из БД
    ClassSchool cs = null;
    try {
        cs = tq.getSingleResult();
    } catch(Exception exc) {
    }
    DefaultCreateManager.closeManager();
    return cs;
}
/**
 * Метод нахождения объекта класса TestName по заданным критериям
 * @param nameTest название теста
 * @param nameDisc название дисциплины (предмета)
 * @param numClass номер класса
 * @return список с объектами класса TestName
 */
public static List<TestName> findTestName(String nameTest, String nameDisc, int numClass){
    EntityManager em = DefaultCreateManager.createManager();
    //Фабрика по созданию экземпляров CriteriaQuery, а также всех видов выражений запросов
    CriteriaBuilder cb = em.getCriteriaBuilder();
    CriteriaQuery<TestName> cq = cb.createQuery(TestName.class);
    Root<TestName> rootTestName = cq.from(TestName.class);
    //Объединение таблиц по общим полям
    Join<TestName, Discipline> joinDisc = rootTestName.join(TestName_.disciplineId);
```

```
    Join<TestName, ClassSchool> joinClass = rootTestName.join(TestName_.classsschoolId);
    //---Predicate - Утверждение (условие на фильтрацию строк по заданному критерию)
    //Условие сравнения строки из поля name и заданной строки
    Predicate nameTestName = cb.equal(rootTestName.get(TestName_.name), nameTest);
    Predicate nameDiscipline = cb.equal(joinDisc.get(Discipline_.name), nameDisc);
    Predicate numberClassSchool = cb.equal(joinClass.get(ClassSchool_.number), numClass);
    if(nameTest!=null){
        cq.where(nameTestName, nameDiscipline, numberClassSchool);
    } else {
        cq.where(nameDiscipline, numberClassSchool);
    }
    //Типизированный запрос
    TypedQuery<TestName> tq = em.createQuery(cq);
    //Получение id со значение из БД
    List<TestName> tn = null;
    try {
        tn = tq.getResultList();
    } catch(Exception exc) {
    }
    DefaultCreateManager.closeManager();
    return tn;
}
/**
 * Метод нахождения всех объектов класса TestName по заданным критериям
 * @param nameDisc название дисциплины (предмета)
 * @param numClass номер класса
 * @return список с объектами класса TestName
 */
public static List<TestName> findTestName(String nameDisc, int numClass){
    return findTestName(null, nameDisc, numClass);
}
/**
 * Метод нахождения всех объектов класса Questions по заданным критериям
 * @param nameDisc название дисциплины (предмета)
 * @param numClass номер класса
 * @param nameTest название теста
 * @return список объектов класса Questions
 */
public static List<Questions> findQuestions(String nameDisc, int numClass, String nameTest){
    EntityManager em = DefaultCreateManager.createManager();
    //Фабрика по созданию экземпляров CriteriaQuery, а также всех видов выражений запросов
    CriteriaBuilder cb = em.getCriteriaBuilder();
    CriteriaQuery<Questions> cq = cb.createQuery(Questions.class);
    Root<Questions> rootQuestion = cq.from(Questions.class);
    //Объединение таблиц по общим полям
    Join<Questions, TestName> joinTestName = rootQuestion.join(Questions_.testNameid);
    Join<TestName, ClassSchool> joinClassSchool = joinTestName.join(TestName_.classsschoolId);
    Join<TestName, Discipline> joinDiscipline = joinTestName.join(TestName_.disciplineId);
    //---Predicate - Утверждение (условие на фильтрацию строк по заданному критерию)
    //Условие сравнения строки из поля name и заданной строки
    Predicate nameTestName = cb.equal(joinTestName.get(TestName_.name), nameTest);
    Predicate nameDiscipline = cb.equal(joinDiscipline.get(Discipline_.name), nameDisc);
    Predicate numberClassSchool = cb.equal(joinClassSchool.get(ClassSchool_.number), numClass);
    cq.where(nameTestName, nameDiscipline, numberClassSchool);
```

```
        //Типизированный запрос
        TypedQuery<Questions> tq = em.createQuery(cq);
        //Получение id со значение из БД
        List<Questions> tn = tq.getResultList();
        DefaultCreateManager.closeManager();
        return tn;
    }
    /**
     * Метод нахождения всех классов учащихся которые сдавали задынний тест по данным критериям
     * @param nameDisc название дисциплины
     * @param numClass номер класса
     * @param nameTest название теста
     * @return список классов учащихся
     */
    public static List<String> findClassLearner(String nameDisc, int numClass, String nameTest){
        EntityManager em = DefaultCreateManager.createManager();
        //Фабрика по созданию экземпляров CriteriaQuery, а также всех видов выражений запросов
        CriteriaBuilder cb = em.getCriteriaBuilder();
        CriteriaQuery<String> cq = cb.createQuery(String.class);
        Root<Results> rootResults = cq.from(Results.class);
        //Объединение таблиц по общим полям
        Join<Results, Learner> joinLearner = rootResults.join(Results_.learnerId);
        Join<Results, Questions> joinQuestions = rootResults.join(Results_.questionsId);
        Join<Questions, TestName> joinTestName = joinQuestions.join(Questions_.testNameid);
        Join<TestName, Discipline> joinDiscipline = joinTestName.join(TestName_.disciplineId);
        Join<TestName, ClassSchool> joinClassSchool = joinTestName.join(TestName_.classschoolId);
        //---Predicate - Утверждение (условие на фильтрацию строк по заданному критерию)
        //Условие сравнения строки из поля name и заданной строки
        Predicate nameTestName = cb.equal(joinTestName.get(TestName_.name), nameTest);
        Predicate nameDiscipline = cb.equal(joinDiscipline.get(Discipline_.name), nameDisc);
        Predicate numberClassSchool = cb.equal(joinClassSchool.get(ClassSchool_.number), numClass);
        cq.select(joinLearner.get(Learner_.className)).where(nameTestName, nameDiscipline,
numberClassSchool).groupBy(joinLearner.get(Learner_.className));
        //Типизированный запрос
        TypedQuery<String> tq = em.createQuery(cq);
        //Получение id со значение из БД
        List<String> tn = tq.getResultList();
        DefaultCreateManager.closeManager();
        return tn;
    }

    /**
     * Метод нахождения всех сделанных попыток по заданному фильтру
     * @param nameDisc название дисциплины
     * @param numClass номер класса
     * @param testName название теста
     * @param classLear название класса учащихся
     * @return номер последней попытки
     */
    public static List<Integer> findAllMadeTry(String nameDisc, int numClass, String testName, String classLear){
        EntityManager em = DefaultCreateManager.createManager();
        //Фабрика по созданию экземпляров CriteriaQuery, а также всех видов выражений запросов
        CriteriaBuilder cb = em.getCriteriaBuilder();
        CriteriaQuery<Integer> cq = cb.createQuery(Integer.class);
```

```
        Root<Results> rootResults = cq.from(Results.class);
        Join<Results, Learner> joinLear = rootResults.join(Results_.learnerId);
        Join<Results, Questions> joinQuest = rootResults.join(Results_.questionsId);
        Join<Questions, TestName> joinTestName = joinQuest.join(Questions_.testNameid);
        Join<TestName, ClassSchool> joinClssSch = joinTestName.join(TestName_.classsschoolId);
        Join<TestName, Discipline> joinCDisc = joinTestName.join(TestName_.disciplineId);
        //---Predicate - Утверждение (условие на фильтрацию строк по заданному критерию)
        Predicate predClsLeanr = cb.equal(joinLear.get(Learner_.className), classLear);
        Predicate predNameTest = cb.equal(joinTestName.get(TestName_.name), testName);
        Predicate predClassSch = cb.equal(joinClssSch.get(ClassSchool_.number), numClass);
        Predicate predNameDisc = cb.equal(joinCDisc.get(Discipline_.name), nameDisc);
        //cq.select(cb.max(rootResults.get(Results_.tryNumber))).where(predClsLeanr, predNameTest, predClassSch,
predNameDisc).groupBy(rootResults.get(Results_.tryNumber));
        cq.select(rootResults.get(Results_.tryNumber)).where(predClsLeanr, predNameTest, predClassSch,
predNameDisc).groupBy(rootResults.get(Results_.tryNumber));
        //Типизированный запрос
        TypedQuery<Integer> tq = em.createQuery(cq);
        List<Integer> lastTry = tq.getResultList();
        DefaultCreateManager.closeManager();
        return lastTry;
    }
    /**
     * Метод нахождения всех дат, в которые сдавали данный тест по заданному фильтру
     * @param nameDisc название дисциплины
     * @param numClass номер класса
     * @param testName название теста
     * @param classLear название класса учащихся
     * @param numTry номер попытки
     * @return номер последней попытки
     */
    public static List<Date> findAllDateMadeTry(String nameDisc, int numClass, String testName, String classLear, int
numTry){
        EntityManager em = DefaultCreateManager.createManager();
        //Фабрика по созданию экземпляров CriteriaQuery, а также всех видов выражений запросов
        CriteriaBuilder cb = em.getCriteriaBuilder();
        CriteriaQuery<Date> cq = cb.createQuery(Date.class);
        Root<Results> rootResults = cq.from(Results.class);
        Join<Results, Learner> joinLear = rootResults.join(Results_.learnerId);
        Join<Results, Questions> joinQuest = rootResults.join(Results_.questionsId);
        Join<Questions, TestName> joinTestName = joinQuest.join(Questions_.testNameid);
        Join<TestName, ClassSchool> joinClssSch = joinTestName.join(TestName_.classsschoolId);
        Join<TestName, Discipline> joinCDisc = joinTestName.join(TestName_.disciplineId);
        //---Predicate - Утверждение (условие на фильтрацию строк по заданному критерию)
        Predicate predClsLeanr = cb.equal(joinLear.get(Learner_.className), classLear);
        Predicate predNameTest = cb.equal(joinTestName.get(TestName_.name), testName);
        Predicate predClassSch = cb.equal(joinClssSch.get(ClassSchool_.number), numClass);
        Predicate predNameDisc = cb.equal(joinCDisc.get(Discipline_.name), nameDisc);
        Predicate predNumTry = cb.equal(rootResults.get(Results_.tryNumber), numTry);
        cq.select(rootResults.get(Results_.dateTest)).where(predClsLeanr, predNameTest, predClassSch, predNameDisc,
predNumTry).groupBy(rootResults.get(Results_.dateTest));
        //Типизированный запрос
        TypedQuery<Date> tq = em.createQuery(cq);
        List<Date> lastTry = tq.getResultList();
        DefaultCreateManager.closeManager();
```

```
        return lastTry;
    }
/**
     * Метод нахождения объекта класса Learner
     * @param name имя ученика
     * @param lastName фамилия ученика
     * @param numClass номер класса
     * @param year год обучения
     * @return объект класса Learner. null - если объект не найден
     */
    public static Learner findLearner(String  name, String lastName, String numClass, int year){
        EntityManager em = DefaultCreateManager.createManager();
        //Фабрика по созданию экземпляров CriteriaQuery, а также всех видов выражений запросов
        CriteriaBuilder  cb = em.getCriteriaBuilder();
        CriteriaQuery<Learner>  cq = cb.createQuery(Learner.class);
        Root<Learner> rootLearner = cq.from(Learner.class);
        //---Predicate - Утверждение (условие на фильтрацию строк по заданному критерию)
        //Условие сравнения строки из поля name и заданной строки
        Predicate nameLearner = cb.equal(rootLearner.get(Learner_.firstName), name);
        Predicate lastNameLearner = cb.equal(rootLearner.get(Learner_.lastName), lastName);
        Predicate classLearner = cb.equal(rootLearner.get(Learner_.className), numClass);
        Predicate yearLearner = cb.equal(rootLearner.get(Learner_.years), year);
        cq.where(nameLearner, lastNameLearner, classLearner, yearLearner);
        //Типизированный запрос
        TypedQuery<Learner> tq = em.createQuery(cq);
        //Получение id со значение из БД
        Learner lear = null;
        try {
            lear = tq.getSingleResult();
        } catch(Exception exc) {
        }
        DefaultCreateManager.closeManager();
        return lear;
    }

    /**
     * Метод нахождения последней сделанной попытки учеником для выбранного теста
     * @param idLeanr id ученика
     * @param disc название дисциплины
     * @param numClass номер класса
     * @param testName название теста
     * @return номер последней попытки
     */
    public static int findTry(long idLeanr, String disc, int numClass, String testName){
        EntityManager em = DefaultCreateManager.createManager();
        //Фабрика по созданию экземпляров CriteriaQuery, а также всех видов выражений запросов
        CriteriaBuilder  cb = em.getCriteriaBuilder();
        CriteriaQuery<Integer>  cq = cb.createQuery(Integer.class);
        Root<Results> rootResults = cq.from(Results.class);
        Join<Results, Learner> joinLear = rootResults.join(Results_.learnerId);
        Join<Results, Questions> joinQuest = rootResults.join(Results_.questionsId);
        Join<Questions, TestName> joinTestName = joinQuest.join(Questions_.testNameid);
        Join<TestName, ClassSchool> joinClssSch = joinTestName.join(TestName_.classsschoolId);
        Join<TestName, Discipline> joinCDisc = joinTestName.join(TestName_.disciplineId);
```

```
        //---Predicate - Утверждение (условие на фильтрацию строк по заданному критерию)
        //Условие сравнения строки из поля name и заданной строки
        Predicate idLearner = cb.equal(joinLear.get(Learner_.idLearner), idLeanr);
        Predicate nameTest = cb.equal(joinTestName.get(TestName_.name), testName);
        Predicate classSch = cb.equal(joinClssSch.get(ClassSchool_.number), numClass);
        Predicate nameDisc = cb.equal(joinCDisc.get(Discipline_.name), disc);
        cq.select(cb.max(rootResults.get(Results_.tryNumber))).where(idLearner, nameTest, classSch, nameDisc);
        //Типизированный запрос
        TypedQuery<Integer> tq = em.createQuery(cq);
        //Получение id со значение из БД
        int lastTry = 0;
        try{
            lastTry = tq.getSingleResult();
        } catch(Exception exc){
            DefaultCreateManager.closeManager();
            return lastTry;
        }
        DefaultCreateManager.closeManager();
        return lastTry;
    }
    /**
     * Метод нахождения количества заработанных баллов учениками в заданном тесте
     * @param nameTest название теста
     * @param nameDisc название дисциплины
     * @param numClass номер класса
     * @param date дата прохождения теста
     * @param numTry номер попытки
     * @param all true - найти сумму всех правильных ответов, false - найти ответ
     * @return список учеников и суммарное количество баллов
     */
    private static List<ResultsLearner> findAnswerTryTest(String nameTest, String nameDisc, String learnClass, int
numClass, Date date, int numTry, long idQuest, boolean all){
        EntityManager em = DefaultCreateManager.createManager();
        //Фабрика по созданию экземпляров CriteriaQuery, а также всех видов выражений запросов
        CriteriaBuilder cb = em.getCriteriaBuilder();
        CriteriaQuery<ResultsLearner> cq = cb.createQuery(ResultsLearner.class);
        Root<Results> rootResults = cq.from(Results.class);
        Join<Results, Questions> joinQuestion = rootResults.join(Results_.questionsId);
        Join<Questions, TestName> joinTestName = joinQuestion.join(Questions_.testNameid);
        Join<TestName, Discipline> joinDisc = joinTestName.join(TestName_.disciplineId);
        Join<TestName, ClassSchool> joinClsSch = joinTestName.join(TestName_.classschoolId);
        Join<Results, Learner> joinLear = rootResults.join(Results_.learnerId);
        //---Predicate - Утверждение (условие на фильтрацию строк по заданному критерию)
        Predicate predDate = cb.equal(rootResults.get(Results_.dateTest), date);
        Predicate predTry = cb.equal(rootResults.get(Results_.tryNumber), numTry);
        Predicate predTestName = cb.equal(joinTestName.get(TestName_.name), nameTest);
        Predicate predClassSchool = cb.equal(joinClsSch.get(ClassSchool_.number), numClass);
        Predicate predDiscipline = cb.equal(joinDisc.get(Discipline_.name), nameDisc);
        Predicate predQuest = cb.equal(joinQuestion.get(Questions_.idQuestions), idQuest);
        Predicate predLeanrClass = cb.equal(joinLear.get(Learner_.className), learnClass);
        //Получение колонок ученика и суммарного количества баллов
        if(all){
            cq.multiselect(joinLear.get(Learner_.firstName), joinLear.get(Learner_.lastName),
joinLear.get(Learner_.className),
```

```
                cb.sum(rootResults.get(Results_.timeAnswer)), cb.sum(rootResults.get(Results_.answer)))
                //Фильтр
                .where(predDate, predTry, predTestName, predClassSchool, predDiscipline, predLeanrClass)
                //Группировка по ученикам
                .groupBy(rootResults.get(Results_.learnerId));
        } else {
            cq.multiselect(joinLear.get(Learner_.firstName),  joinLear.get(Learner_.lastName),
joinLear.get(Learner_.className),
                rootResults.get(Results_.timeAnswer), rootResults.get(Results_.answer))
                //Фильтр
                .where(predDate, predTry, predTestName, predClassSchool, predDiscipline, predQuest, predLeanrClass);
        }
        //Типизированный запрос
        TypedQuery<ResultsLearner> tq = em.createQuery(cq);
        //Получение id со значение из БД
        List<ResultsLearner> resLear = tq.getResultList();
        DefaultCreateManager.closeManager();
        return resLear;
    }
    /**
     * Метод нахождения результатов тестирования для конкретного вопроса
     * @param nameTest название теста
     * @param nameDisc название дисциплины
     * @param learnClass название класса
     * @param numClass номер класса
     * @param date дата проведения теста
     * @param numTry номер попытки
     * @param idQues id вопроса
     * @return список типа ResultsLearner
     */
    public static List<ResultsLearner> findAnswerTryTest(String nameTest, String nameDisc, String learnClass, int
numClass, Date date, int numTry, long idQues){
        return findAnswerTryTest(nameTest, nameDisc, learnClass, numClass, date, numTry, idQues, false);
    }
    /**
     * Метод нахождения результатов тестирования для всего теста
     * @param nameTest название теста
     * @param nameDisc название дисциплины
     * @param learnClass название класса
     * @param numClass номер класса
     * @param date дата проведения теста
     * @param numTry номер попытки
     * @return список типа ResultsLearner
     */
    public static List<ResultsLearner> findSumAnswerTryTest(String nameTest, String nameDisc, String learnClass, int
numClass, Date date, int numTry){
        return findAnswerTryTest(nameTest, nameDisc, learnClass, numClass, date, numTry, -1, true);
    }
}
```