**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**«РЯЗАНСКИЙ ГОСУДАРСТВЕННЫЙ РАДИОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**Факультет** ФВТ **«К защите»**

**Кафедра** ВПМ Зав. кафедрой \_\_\_\_\_\_\_\_ Пылькин А.Н.

**Направление** 231000 “\_\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_г.

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к выпускной квалификационной работе бакалавра на тему

**«*Разработка обучающей системы программирования SQL запросов*»**

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
| Студент: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Буянов Е.А.) |
| Руководитель проекта: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Каширин И.Ю.) |
| Консультант кафедры: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Каширин И.Ю.) |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

“\_\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_г.

Рязань 2015

РЕФЕРАТ

Пояснительная записка — 76 страницы, 10 табл., 30 рисунков, 11 источников.

ИНФОРМАЦИОННАЯ СИСТЕМА ПО ОБУЧЕНИЮ SQL ЗАПРОСАМ.

Цель проекта — разработка информационной системы для тестирования знаний и обучению в области SQL запросов.

Средства разработки — Microsoft Visual Studio 2010, СУБД MS SQL, OS Windows 7.

Назначение: автоматизация обучения в области SQL запросов и автоматизация в области обработки данных результатов тестирования.

Область применения: сфера образования.

**Министерство образования и науки Российской Федерации**

Федеральное государственное образовательное учреждение

высшего профессионального образования

«Рязанский государственный радиотехнический университет»

**(ФГБОУ ВПО «РГРТУ», РГРТУ)**

**УТВЕРЖДАЮ**

**Зав. кафедрой ВПМ**

д.т.н., профессор\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.Н. Пылькин

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2015 г.

**ЗАДАНИЕ**

**на выпускную квалификационную работу**

Студенту Буянову Евгению Александровичу, гр. 143.

1. **Тема проекта:** *«Разработка обучающей системы программирования SQL запросов»*.
2. **Дата представления законченного проекта:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
3. **Руководитель работы**: Каширин Игорь Юрьевич, доцент технических наук, Профессор каф. ВПМ
4. **Исходные данные к работе:**

*Перечень документов:* Федеральный закон от 02.10.2007 N 229-ФЗ (ред. от 06.04.2015) "Об исполнительном производстве"

*Системные требования:* ОС Windows XP/Vista/7/8.

1. **Основные разделы пояснительной записки**
2. *Введение.*
3. *Постановка задачи.*
4. *Технико-экономическое обоснование темы*
5. *Теоретическая часть.*
6. *Практическая часть.*
7. *Разработка программной системы:*
   1. *Разработка серверной части.*
   2. *Разработка клиентской части.*
8. *Программная документация:*
   1. *Руководство пользователя.*
   2. *Руководство администратора.*
9. *Тестирование системы.*
10. *Заключение.*
11. *Список литературы.*
12. *Приложения.*
13. **Содержание графической части** (с указанием числа листов формата **А1**)

*Постановка задачи* — 1 лист.

*Анализ предметной области* — 1 лист.

*Общая структура* — 1 листа.

*Схема базы данных* — 2 листа.

*Интерфейс пользователя* — 2 листов.

*Формы отчетов* — 1 листа.

1. **Консультанты по проекту**

Консультант кафедры \_\_\_\_\_\_\_\_\_ Каширин И.Ю.

Дата выдачи задания «\_\_\_\_» \_\_\_\_\_\_\_\_ 2015 г.

Руководитель \_\_\_\_\_\_\_\_\_Каширин И.Ю.

Задание принял к исполнению «\_\_\_\_» \_\_\_\_\_\_\_\_ 2015 г.

Студент \_\_\_\_\_\_\_\_\_Буянов Е.А.

Содержание

[Содержание 5](#_Toc421572748)

[Введение 6](#_Toc421572749)

[1. Постановка задачи и технико-экономическое обоснование 8](#_Toc421572750)

[1.1. Постановка задачи 8](#_Toc421572751)

[1.2. Технико-экономическое обоснование 9](#_Toc421572752)

[2. Проектирование системы 11](#_Toc421572753)

[2.1. Теоретическая часть 11](#_Toc421572754)

[2.1.1. Выбор языка для написания клиентского приложения 11](#_Toc421572755)

[2.1.2. Выбор СУБД 12](#_Toc421572756)

[2.1.3. Выводы 16](#_Toc421572761)

[3. Анализ предметной области 17](#_Toc421572762)

[3.1. Разработка алгоритмов 18](#_Toc421572763)

[4. Разработка базы данных 25](#_Toc421572774)

[4.1. Моделирование данных 25](#_Toc421572775)

[4.2.Разработка базы данных 27](#_Toc421572776)

[5. Разработка интерфейса пользователя 31](#_Toc421572777)

[6. Разработка классов и процедур 40](#_Toc421572778)

[7. Техническая документация 43](#_Toc421572779)

[7.1.Назначение программы 43](#_Toc421572780)

[7.2 Условия приминения 43](#_Toc421572782)

[7.3. Руководство программиста 44](#_Toc421572783)

[7.4. Руководство пользователя 46](#_Toc421572784)

[8. Тестирование системы 56](#_Toc421572785)

[8.1. Программа и методика испытаний 56](#_Toc421572786)

[8.2. Тестирование программного обеспечения 56](#_Toc421572787)

[8.3. Выводы по результатам тестирования 59](#_Toc421572788)

[Заключение 60](#_Toc421572789)

[Список использованных источников 61](#_Toc421572790)

[ПРИЛОЖЕНИЕ 1 62](#_Toc421572791)

[ЛИСТИНГ НАИБОЛЕЕ ЗНАЧИТЕЛЬНЫХ ЧАСТЕЙ ПРОГРАММЫ 62](#_Toc421572792)

Введение

Постоянное увеличение объема информации и ограниченность учебного времени обуславливают необходимость интенсификации обучения, разработки и внедрения технологий, базирующихся на использовании вычислительной техники с применением активных методов обучения во всем их разнообразии и комплексности. Реализация активных методов обучения – одна из основных задач дидактики, которая предполагает активизацию всего процесса, выявление системы, способов, приемов, способствующих повышению активности обучаемых через формирование положительной мотивационной структуры учебно-познавательной деятельности.

Развитие информационных технологий предоставило новую, уникальную возможность проведения занятий с внедрением автоматизированных обучающих систем по дисциплинам в вузах. Она, во-первых, позволяет самому обучаемому выбрать и время и место для обучения, во-вторых, дает возможность использовать в обучении новые информационные технологии, в-четвертых, в определенной степени сокращает расходы на обучение. С другой стороны, внедрение в образование новых автоматизированных обучающих систем усиливает возможности индивидуализации обучения.

Достоинствами автоматизированных обучающих систем (АОС), являются: во-первых, их мобильность, во-вторых, доступность связи с развитием компьютерных сетей, в-третьих, адекватность уровню развития современных научных знаний. С другой стороны, создание АОС способствует также решению и такой проблемы, как постоянное обновление информационного материала. В них также может содержаться большое количество упражнений и примеров, подробно иллюстрироваться в динамике различные виды информации. Кроме того, при помощи АОС осуществляется контроль знаний - компьютерное тестирование.

В настоящее время традиционные подходы в области преподавания информатики и программирования в вузе не способны отследить быстроменяющуюся действительность в области информационных технологий, связанную с бурным развитием вычислительной техники, операционных систем, парадигм программирования, организацией, анализом, представлением информации и обеспечением доступа к ней, в том числе и в сетях.

Выход из создавшегося положения видится в несколько иной расстановке акцентов, как на принципы обучения, так и на сам процесс и условия обучения, позволяющие не только и не столько учить в прямом смысле этого слова, сколько помогать учиться, организовать процесс обучения так, чтобы развивались не только практические навыки в области информатики и программирования, но и соответствующее мировоззрение и творческий потенциал, позволяющие будущему специалисту с минимальными затратами осуществлять доступ к требуемым информационным ресурсам (в том числе и мировым), самостоятельно адаптироваться к действительности, определяемой появлением новых парадигм, сред и инструментальных средств.

Целью квалификационного проекта является автоматизация процесса обучения, а также тестирования имеющихся знаний в области программирования SQL запросов, для чего требуется разработать программный пакет, представляющий собой базу данных и клиентское приложение.

К практической ценности проекта, помимо автоматизации процесса обучения, можно отнести и достаточно гибкую систему предотвращения занесения ошибочной информации в базу.

1. Постановка задачи и технико-экономическое обоснование

1.1. Постановка задачи

Целью квалификационного проекта является разработка программного пакета, представляющего собой базу данных и клиентское приложение, выполняющих автоматизацию процесса обучения, а также тестирования имеющихся знаний в области программирования SQL запросов.

Проект должен соответствовать следующим требованиям:

* выполнение поставленных задач;
* удобство оконного интерфейса;
* пользовательская настройка (настройка соединения, настройка тестов);
* подробная документация.

База данных должна хранить проиндексированную клиентским приложением информацию, которая включает в себя:

* список тестируемых пользователей;
* обучающую базу;
* базу тестов;
* результаты тестирования.

Клиентское приложение должно реализовать следующие механизмы:

* настройка соединения с базой данных;
* регистрация пользователей;
* авторизация пользователей;
* настройка базы обучения;
* настройка базы тестов;
* просмотр результатов тестирования.

1.2. Технико-экономическое обоснование

Ни одно учебное заведение не обходится без тестирования учащихся по полученным знаниям с помощью письменных тестов. Преподавателям приходится сталкиваться с некоторыми недостатками такого рода тестирования. Рассмотрим эти недостатки подробнее.

*Выбор способа разделения и представления информации индивидуально*. К этому недостатку можно отнести затраты времени на составление и рапределение тестов между студентами.

*Отсутствие ссылочной целостности*. К этому недостатку можно отнести случаи, когда тестирование было произведено заново, а результаты не были перезаписаны у преподавателя.

*Долгий поиск необходимой информации*. Очевидно, чтобы найти результаты выполнения определенным студентом, необходимо искать их среди записей в журналах или личных тетрадях, что делается относительно небыстро.

*Пополнение базы знаний новой информацией или тестами*. Этот недостаток выявляется при необходимости дополнить базы. В этом случае нужно ставить в известность преподавателей, преподносящих данную информацию или выполняющих тестирование студентов.

Таким образом, было бы эффективнее использовать систему, позволяющую выполнять необходимый набор операций, исключая перечисленные недостатки.

Поиск готовой системы, выполняющей подобный набор функций, не привел к результатам. В связи с этим было принято решение о создании программного пакета, решающего поставленные задачи.

Проект состоит из базы данных и клиентского приложения.

База данных — наилучший способ организации хранения и предоставления по запросу большого объёма данных. В ней хранится вся накапливаемая информация о программировании SQL запросов. Удобное клиентское приложение обеспечивает лёгкость в проведении необходимых операций, обращённых к базе данных.

Разработка и внедрение программного обеспечения для автоматизации процесса обучения, а также тестирования имеющихся знаний в области программирования SQL запросов актуальна, так как повышает качество обучения студентов, а также ускоряет процесс тестирования.

В дальнейшем планируется перспектива развития данного программного продукта: добавление новых областей обучения и тестирования для расширения областей применения.

Вне сомнений – разработанная система может быть внедрена в любое нуждающееся учебное заведение.

Для функционирования разработанного программного обеспечения необходим компьютер, работающий под операционной системой *Windows Vista/7/8,* сустановленным *Microsoft Framework 4.5(клиентская часть)*, и СУБД *Microsoft SQL Server*. Программный продукт *Microsoft Visual Studio 2010* выбран из-за широкого спектра решаемых задач. Язык *C#* привлекателен новизной, перспективностью и рациональным способом связи интерфейса с базой данных, а также удобным в использовании регулярных выражений. СУБД *MS SQL Server* является одной из крупнейших на рынке, постоянно развивается и удобна в написании сохраненных процедур, манипулирующих данными.

Программа разработана с помощью современных программных средств, актуальна и несколько экономически выгодна, что обеспечит ее востребованность на рынке.

2. Проектирование системы

2.1. Теоретическая часть

2.1.1. Выбор языка для написания клиентского приложения

Наиболее распространёнными языками программирования, для разработки оконных приложений, на сегодняшний день являются *C++*, *C#*, *Java* и *Delphi*.

Все эти языки объектно-ориентированы, то есть описывают не столько саму задачу, сколько её решение.

Проанализируем наиболее существенные для данной работы возможности указанных языков. Все они поддерживают обработку исключений с помощью блоков *try/catch* или эквивалентных конструкций (*try/except*), хотя в *C++* не определён раздел *finally*. Автоматический процесс сборки мусора (освобождения динамической памяти, занятой неиспользованными объектами) предусмотрен в *C#* и *Java* [5]. Также *Java* позволяет записывать целые числа сколь угодной длины с помощью литералов — полезная особенность при необходимости соблюдения высокой точности чисел, но достаточно далёкая от проблем поставленной задачи. Зато в *Delphi* доступно определение целых чисел интервальных типов. Удобным свойством установки значений параметров по умолчанию обладают *Delphi* и *C++*. А в *С++* также имеет место множественное наследование, которое может стать причиной дублирования полей и прочих проблем, какие устраняются при замене этой возможности в *C#* интерфейсами.

Хотя *Java* идентичен *C#*, он использование *Java* не желательно по следующим причинам

1. По сравнению c C# построение одного и того же интерфейса в Java более трудоемкая и рутинная задача.
2. По сравнению c IDE C# в любой IDE java удобство работы с отладчиком заметно уступает IDE C#.
3. Для работы с технологией Java необходима установка дополнительных пакетов, которые по умолчанию не входят ОС windows (JRE и JDK).

Исходя из этих причин, java не может быть использована в данном проекте.

Остаются *C++*, *Delphi* и *C#*. *C++* удобен при написании высокопроизводительных приложений, поскольку позволяет контролировать код на более низком уровне, чем *C#* и *Delphi*, но разработка с его использованием более длительная. Остаются *Delphi* и *С#*. Они подходят оба, но в *Delphi* нет поддержки самодокументирующегося кода и автоматической чистки памяти, что также увеличивает сложность и время написания приложения. Также механизм работы с регулярными выражениями в C# реализован удобнее. Поэтому выбор пал на C# в качестве языка разработки приложения.

2.1.2. Выбор СУБД

Среди современных СУБД (Систем Управления Базами Данных) наиболее ярко выделяются *Oracle*, *Microsoft SQL Server*, *MySQL* и *Firebird*. Рассмотрим их отличительные особенности.

1. *MySQL*

Это относительно небольшая и быстрая реляционная СУБД, основанная на традициях *Hughes Technologies Mini SQL* (*mSQL*). Основными чертами пакета *MySQL* являются:

* многопоточность — поддержка нескольких одновременных запросов;
* оптимизация связей с присоединением многих данных за один проход;
* записи фиксированной и переменной длины;
* гибкая система привилегий и паролей;
* большое количество ключей в таблице — до 16, причём каждый может иметь до 15 полей;
* возможность обращения к ключевым и специальным полям в операторе *CREATE*;
* поддержка чисел длинной от 1 до 4 байт (*ints*, *float*, *double*, *fixed*), строк переменной длины и меток времени;
* быстрая система памяти, основанная на потоках;
* утилита проверки и ремонта таблицы (*isamchk*);
* операции работы со строками, не чувствительные к регистру символов;
* применимость псевдонимов, как к таблицам, так и к столбцам;
* наличие значений по умолчанию для всех полей, что позволяет использовать *INSERT* на любом подмножестве;
* лёгкость управления таблицей, включая добавление и удаление ключей и полей.

1. *Firebird*

Это компактная СУБД, основанная на ядре *Borland InterBase*, представляющая собой полнофункциональный *SQL*-сервер. Рассмотрим характерные качества *Firebird*:

* высокая эффективность и мощная языковая поддержка для хранимых процедур и триггеров;
* многоверсионная архитектура, позволяющая серверу обрабатывать различные версии одной и той же записи параллельно таким образом, что каждая транзакция видит свою версию данных, не мешая соседним;
* высокая масштабируемость ⎯ возможность использования как в качестве настольного, однопользовательского приложения БД, так и в качестве мощного сервера для работы десятков и сотен пользователей;
* кроссплатформенность: поддерживаются все версии *Windows*, начиная с *Windows 98*, *Linux* и несколько платформах *UNIX*;
* хранимые процедуры и триггеры способны генерировать события, на которые может подписаться клиент, открывая специально настроенный *TCP/IP* порт;
* «универсальные» триггеры, позволяющие обрабатывать вставки, обновления и удаления записей таблицы в одном триггере;
* декларативное описание ссылочной целостности, обеспечивающее непротиворечивость и корректность многоуровневых взаимосвязей таблиц;
* реализация ограничений проверки (*check constraints*) на уровне доменов и полей, отображений (*views*), исключений, ролей и управления правами доступа;
* поддержка больших баз данных, которые могут быть расположены в нескольких файлах, размер которых ограничен файловой системой и метом на жёстком диске.

1. *Oracle*

Эта СУБД является исторически первой и одной из наиболее развитых на сегодняшний день реализацией архитектуры клиент/сервер. Среди основных свойств Oracle следует отметить следующие свойства:

* высокая надежность;
* открытость ⎯ совокупность переносимости и масштабируемости;
* возможность разбиения крупных баз данных на разделы (*large-database partition*);
* свободные таблицы, которые не обязательно заполнять сразу при создании;
* распараллеливание операций в запросе;
* поддержка известных платформ: *Windows*, *AIX*, *Compaq Tru64 UNIX*, *HP 9000 Series HP-UX*, *Linux Intel*, *Sun Solaris*;
* средство передачи журналов, позволяющее синхронизировать отдельные системы посредством автоматизированного обновления журнала транзакций, что повышает надежность системы и устойчивость к авариям;
* поддержка большого объема памяти и симметричной многопроцессорной обработки, позволяющая управлять приложениями с высокими нагрузками, связанными с большим объемом транзакций.

1. *Microsoft SQL Server*

Это продукт нацелен на решение широкого круга задач во всех областях, связанных с хранением и обработкой данных, требующих быстрого получения и анализа информации. *MS SQL Server* имеет следующие возможности:

* надёжность и масштабируемость, обеспечивающие практически неограниченный рост объёмов памяти;
* высокая скорость построения решений;
* простота администрирования;
* наличие средств удалённого доступа;
* многозадачность и параллельная обработка данных: разделение потока данных между серверами, синхронизация данных на всех серверах в кластере независимо от их местонахождения, переустановка и восстановление любой ветви при сбое в кластере, не затрагивая остальные;
* сервисы анализа и безопасности: закрытие данных при использовании системы безопасности для массивов и ячеек, ограничение доступа к специальным наборам ячеек, высокий уровень защиты данных по умолчанию.

*Microsoft SQL Server* и *Oracle*, существующие как коммерческие программные продукты, развиваются уже в течение многих лет, и их можно охарактеризовать как основу для промышленных решений в области обработки данных. Таким образом, для разработки проекта следует выбрать одну из этих СУБД ⎯ *MS SQL Server* или *Oracle*.

Следует отметить ряд преимуществ *SQL Server* перед *Oracle*, важных для разработки проекта:

* безопасность;
* отказоустойчивость;
* интегрируемость с *Microsoft Visual Studio*;
* масштабируемость;
* относительно небольшая стоимость;
* лёгкость установки, использования и поддержки.

Несмотря на то, что СУБД *MySQL* и *Firebird* являются бесплатными, в рамках данного квалификационного проекта в качестве СУБД был выбран *Microsoft SQL Server* благодаря удобным средствам разработки, хорошей интеграцией с *Microsoft Visual Studio* и наибольшему опыту работы разработчика с ним. Для подготовки проекта можно использовать версию *Developers Edition* данной СУБД ⎯ полнофункциональная версия для разработчиков, распространяемая бесплатно. Однако для реальной работы приложения в локальной сети будет целесообразно использовать одну из бесплатных СУБД (в силу высокой стоимости лицензий на *MSSQL*).

2.1.3. Выводы

В соответствии с кругом задач, определённых для разрабатываемого программного комплекса, была выбрана СУБД *Microsoft SQL Server*, язык *C#*, и среда разработки *Microsoft Visual Studio 2010*.

3. Анализ предметной области

Для тестирования учащихся необходимо рапределить, определить время выполнения, а также собрать результаты тестирования. Если тестирование не пройдено одним из учащихся, то необходимо указать на ошибки и, в последствие, провести новый тест.

Результаты теста должны содержать:

* название теста;
* правильные и неправильные ответы;
* дату и время тестирования;
* срок выполнения тестирования;
* данные по студенту для последующей идентификации.

Таким образом, преподаватель, проводящий тест, записывает результаты в отдельный журнал или тетрадь для последующего доступа к записанной информации.

3.1. Разработка алгоритмов

В клиентском приложении все пользователи разделены на две группы.

* Администраторы
* Ученики

В связи с этим, в информационной системе необходимо реализовать систему авторизации. Алгоритм системы авторизации представлен на рис. 3.1.



Рис. 3.1. Схема алгоритма авторизации в программе

Пользователи клиенсткого приложения могут:

* пополнять информацией или тестами базу данных;
* выполнять тестирование;
* просматривать результаты тестирования.

Общая схема алгоритма работы программы представлена на рис. 3.2:



Рис. 3.2. Общая схема алгоритма программы

В программе предусмотрены некоторые настройки, которые впоследствии могут быть сохранени во избежание затрат времени на их установку (рис. 3.3). Настройки параметров соединения автоматически сохраняются в конфигурационный файл приложения.



Рис. 3.3. Схема алгоритма настроек приложения

Рассмотрим алгоритмы работы с программой по основным назначениям. В клиентской части приложения должно быть реализованы процес создания тестов, процесс обучения созданному тесту, процесс прохождения теста. Для увеличения детализации вопроса решено все вопросы и пояснения к вопросам хранить в виде графических изображений. Это позволяет более гибко предствить вопрос для обучающегося (добавив в вопрос не только текст, но и графику). Блок схема процесса создания тестирования представлена на рисунке 3.4.

На рис. 3.4 показан алгоритм создания теста.



Рис. 3.5. Схема алгоритма тестирования

Как видно из схемы, после выполнения теста пользователю представляется результат, после чего результат сохраняется в базу данных и отображается начальная форма выбора режима работы с программой.

Алгоритм обучения приведен на рис. 3.6:



Рис. 3.6. Схема алгоритма обучения

Как и в случае тестирования, пользователю отображается начальная форма выбора режима работы с программой после завершения обучения.

Далее представлен алгортим составления отчета о пройденном тесте рис. 3.7



Рис. 3.7. Схема алгоритма составления отчета о тестировании

4. Разработка базы данных

4.1. Моделирование данных

Наиболее распространенным средством моделирования диаграммы данных «сущность-связь» является *Entity Relationship Diagram*— *ERD*. С помощью этих диаграмм определяются важные для предметной области объекты (сущности), их свойства (атрибуты) и отношения друг с другом (связи). *ERD* непосредственно используются для проектирования реляционных баз данных.

Построение моделей «сущность-связь» включает идентификацию важных для организации объектов (сущностей), их свойств (атрибутов) и их отношений (связей). Получаемая в результате информационная модель не зависит от режима хранения данных или метода доступа.

На рис. 4.1 изображена *ER*-диаграмма рассматриваемой предметной области на уровне сущностей, подготовленная средствами программного пакета *VISIO* корпорации *Microsoft*.



Рис. 4.1. ER-диаграмма на уровне сущностей

На рис. 4.2 изображена *ER*-диаграмма сущностей фактических данных.

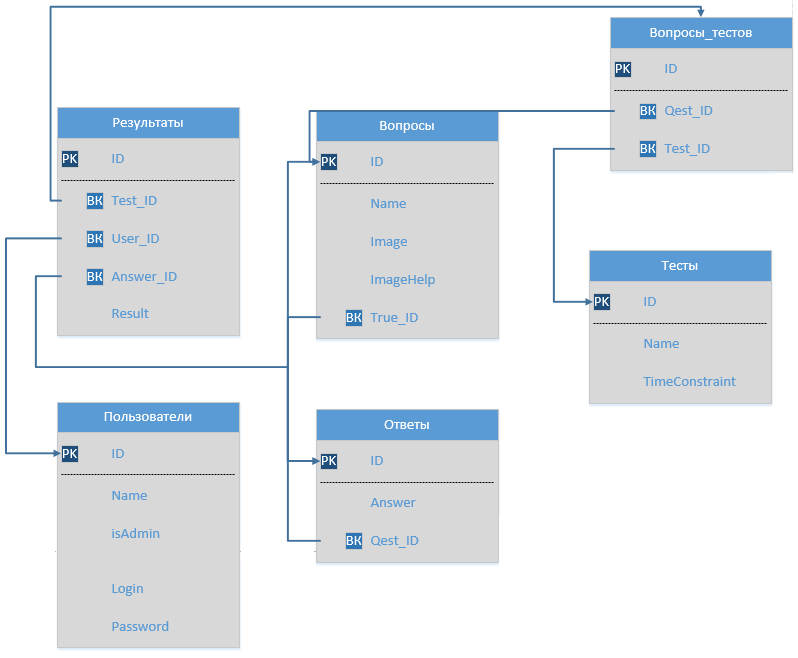


Рис. 4.2. ER-диаграмма сущностей фактических данных

Произведенный анализ позволил разработать модель данных базы данных.

4.2.Разработка базы данных

Для хранения большого количества информации по тестам, результатов тестирования – рационально использовать базу данных.

В базе необходимо хранить следующую информацию:

* данные пользователей (Ф.И.О., логин, пароль);
* вопросы;
* тесты;
* результаты.

Структуру таблицы пользователя можно увидеть в табл. 4.1:

Таблица 4.1

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Описание** | **Тип** | **Огра-ничение** |
| **users — тaблицa-cпpaвoчник пользователей программы** | | | |
| ID | Идентификатор | int | PK |
| User | ФИО пользователя | nvarchar(60) |  |
| Login | Логин | nvarchar(255) |  |
| Password | Пароль | nvarchar(255) |  |

Структуру таблицы вопросов можно увидеть в табл. 4.2:

Таблица 4.2

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Описание** | **Тип** | **Огра-ничение** |
| **questions — тaблицa-cпpaвoчник вопросов** | | | |
| ID | Идентификатор | int | PK |
| Question | Вопрос | text |  |
| ImageHelp | Пояснение изображение | image |  |
| Image | Вопрос изображение | image |  |

Структуру таблицы тестов можно увидеть в табл. 4.3:

Таблица 4.3

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Описание** | **Тип** | **Огра-ничение** |
| **tests — тaблицa-cпpaвoчник тестов** | | | |
| ID | Идентификатор | int | PK |
| Name | Название теста | text |  |
| Constraint | Время на выполнение теста, мин. | int |  |

Структуру таблицы вопросов тестов можно увидеть в табл. 4.4:

Таблица 4.4

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Описание** | **Тип** | **Огра-ничение** |
| **testsQuestions — тaблицa-cпpaвoчник вопросов к тестам** | | | |
| ID | Идентификатор | int | PK |
| TestID | ID теста | int | UK |
| QuestionID | ID вопроса | int | UK |

Структуру таблицы вариантов ответа на вопросы можно увидеть в табл. 4.5:

Таблица 4.5

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Описание** | **Тип** | **Огра-ничение** |
| **Answers — тaблицa-cпpaвoчник вариантов ответа на вопросы** | | | |
| ID | Идентификатор | int | PK |
| QuestionID | ID вопроса | int |  |
| Answer | Вариант ответа | text |  |
| IsTrueAnswer | Правильный ли ответ | bool |  |

Структуру таблицы результатов выполнения тестов можно увидеть в табл. 4.6:

Таблица 4.6

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Описание** | **Тип** | **Огра-ничение** |
| **testsResults — тaблицa-cпpaвoчник результатов выполнения тестов** | | | |
| UserID | ID пользователя | int | UK |
| QuestionID | ID вопроса | int | UK |
| AnswerID | ID ответа | int | UK |

5. Разработка интерфейса пользователя

Интерфейс пользователя поддерживает следующие формы:

* форма настройки соединения с сервером БД;
* форма авторизации;
* форма регистрации;
* основная форма приложения;
* форма результатов тестирования;
* форма проведения теста;
* форма режима обучения;
* форма администрирования;
* форма добавления теста;
* форма добавления вопроса.

А также диалоговые окна подтверждения действий при ошибках.

При первом запуске программа проверяет наличие конфигурационного файла в своей директории. В этом файле хранится информация о подключении. Формат файла *XML*. В нем записана строка подключения к серверу. Если такого файла нет (то он создастся при первой настройке подключения) или есть, но по данным подключения в нем не получается подключиться к базе данных, то пользователю открывается окно настроек (рис. 5.1):

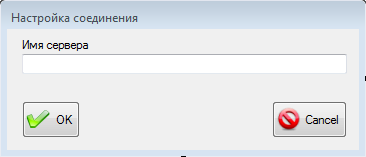


Рис. 5.1. Окно настройки соединения с БД

Здесь указываются стандартные данные:

* IP сервера, на котором находится база данных;
* порт, на который она настроена.

Нажав кнопку «ОК», программа попытается установить соединение. В случае успешного соединения данное окно закрывается, данные сохраняются в конфигурационный файл, и перед пользователем появляется форма авторизации в приложении (см. рис. 5.3). В случае ошибки появляется диалоговое окно, представленное на рис. 5.2:

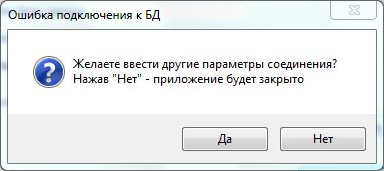


Рис. 5.2. Диалог выбора действий при ошибке подключения

Если нажать кнопку «Да», то останется форма настройки соединения (рис. 5.1), иначе – приложение закроется.

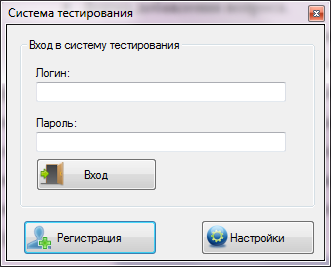


Рис. 5.3. Форма авторизации

В форме авторизации пользователь может войти под своей учетной записью в систему (если такая имеется) или зарегистрироваться для получения учетной записи (рис. 5.4).

Из данной формы можно войти в режим администрирования при вводе логина и пароля администратора после нажатия на кнопку «Настройки».

Форма регистрации содержит в себе поля для ввода Ф.И.О., логина и пароля. Для подтверждения правильности ввода пароля необходимо повторно ввести пароль в соответствующее поле (рис. 5.4).

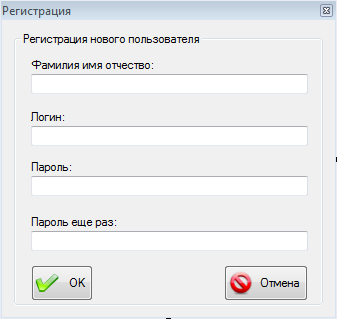


Рис. 5.4. Форма регистрации

В случае успешной авторизации перед пользователем возникает основное окно приложения (рис. 5.5).

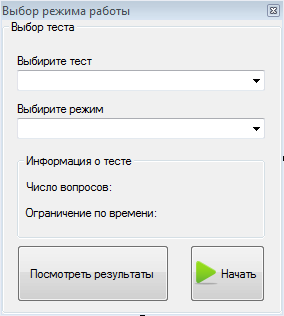


Рис. 5.5. Основная форма приложения

Главное окно приложения предоставляет пользователю право выбора дальнейших действий: прохождение теста, обучение или просмотр результатов прошлых тестирований. Здесь выбирается необходимый тест в списке тестов, режим (тестирование или обучение) и кнопка «Начать» или «Просмотреть результаты» для начала тестирования/обучения или просмотра результатов выбранного теста соответственно (рис. 5.5).

В случае выбора просмотра результатов тестирования, отобразится окно, представленное на рис. 5.6.

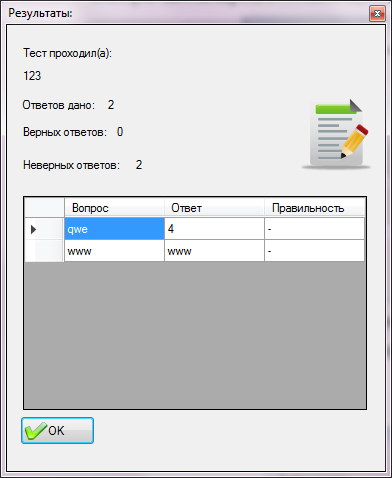


Рис. 5.6. Форма результатов тестирования

Если тестирование по выбранному тесту еще не было проведено, то окно останется без дополнительной информации, как на рис. 5.6. Иначе отобразится количество данных ответов, количество верных ответов, количество неверных ответов и список вопросов с поставленными на них ответами.

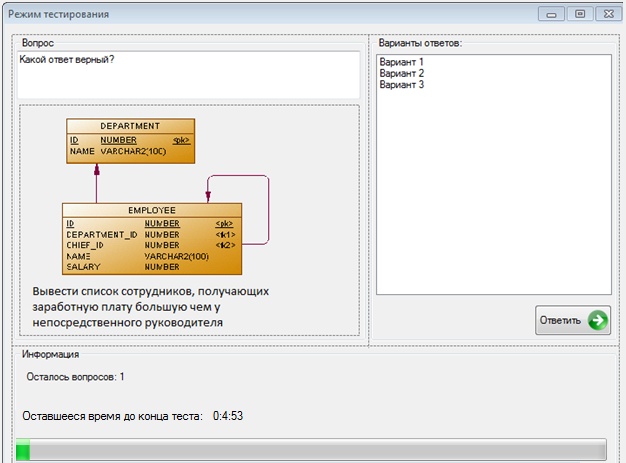


Рис. 5.7. Окно тестирования

На рис. 5.7 отображено окно тестирования. В этом окне отображаются:

* вопрос;
* изображение для вопроса, если таковое имеется;
* варианты ответа;
* оставшееся количество вопросов;
* оставшееся время выполнения теста с графической шкалой процесса.

Если был выбран режим обучения, то будет отображено окно, представленное на рис. 5.8.

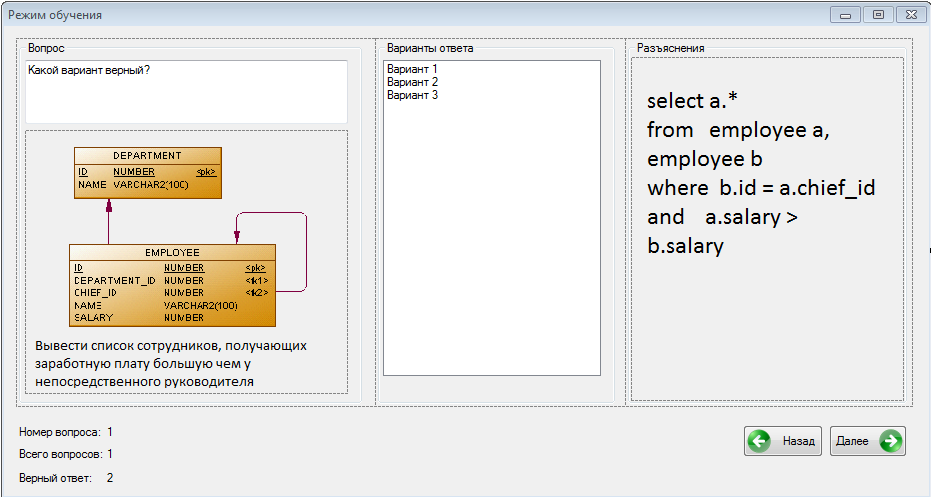


Рис. 5.8. Форма обучения

Здесь отображаются:

* вопрос;
* изображение для вопроса, если таковое имеется;
* варианты ответа;
* разъяснения к вопросу (возможно изображение);
* номер разъясняемого вопроса;
* количество вопросов для разъяснения;
* верный ответ на данный вопрос.

В случае выбора режима администрирования, перед пользователем отображается окно, представленное на рис. 5.9.

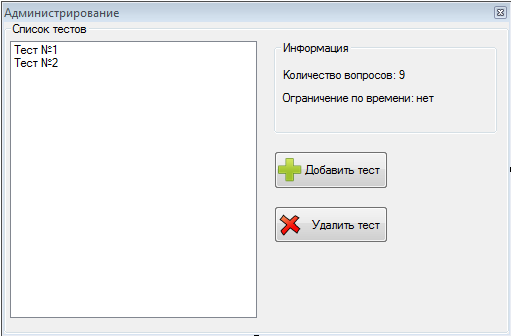


Рис. 5.9. Форма администрирования

В данном окне будут отображены все имеющиеся тесты в базе данных.

При необходимости удалить какой-либо из тестов, нужно выбрать тест в списке и нажать на кнопку «Удалить тест». После этого выбранный тест пропадет из списка тестов.

При необходимости добавить новый тест, необходимо нажать на кнопку «Добавить тест». В результате отобразиться окно, представленное на рис. 5.10.

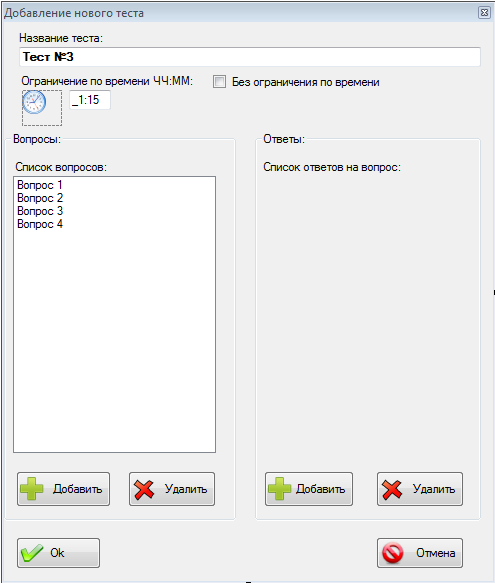


Рис. 5.10. Форма добавления нового теста

Здесь необходимо ввести название теста, время, отведенное на тест, добавить нужные вопросы и варианты ответа. По завершению формирования теста нужно нажать на кнопку «Ок» для сохранения. Новый тест появится в списке тестов (рис. 5.9).

Во время добавления теста необходимо создавать новые вопросы. Окно создания вопроса представлено на рис. 5.11:

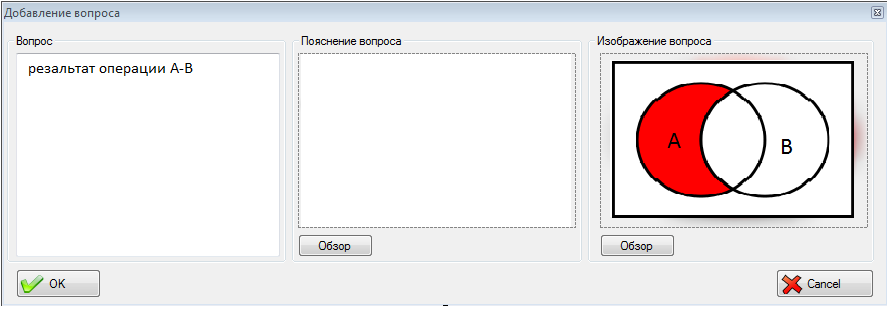


Рис. 5.11. Форма добавления вопроса

В данном окне необходимо ввести текст вопроса, пояснение (возможно графическое) и выбрать картинку вопроса, если необходимо. Для сохранения вопроса нужно нажать кнопку «ОК».

6. Разработка классов и процедур

В разрабатываемом приложении каждый класс выполняет одну специализированную задачу, то есть содержит необходимый набор свойств, полей и методов для достижения поставленной задачи. Рассмотрим назначение классов проекта.

* AddQuestion

Предназначен для отображения формы добавления вопроса.

* AddTestForm

Предназначен для отображения формы добавления теста.

* AdminForm

Предназначен для отображения формы администрирования тестов.

* ConnectionOptions

Предназначен для отображения формы настройки подключения.

* LearnForm

Предназначен для отображения формы режима обучения.

* LoginForm

Предназначен для отображения формы авторизации в программе.

* RegisterForm

Предназначен для отображения формы регистрации в программе.

* ResultForm

Предназначен для отображения формы результата тестирования.

* TesterForm

Предназначен для отображения формы режима тестирования.

* TestPicker

Предназначен для отображения формы выбора теста.

* AnswerRecord

Предназначен для работы с ответами на вопросы.

* QuestionRecord

Предназначен для работы с вопросами.

* ResultRecord

Предназначен для работы с результатом теста.

* ResultView

Предназначен для работы с результатами.

* TaskRecord

Предназначен для работы с тестом.

* TaskTestRecord

Предназначен для работы с тестированием.

* Test

Предназначен для работы с тестами.

* TestRecord

Предназначен для работы с составом теста.

* DataBaseWorker

Класс работы с базой данных. Позволяет подключаться к базе данных, содержит набор методов по работе с базой.

На рис. 6.1 изображены классы программы.

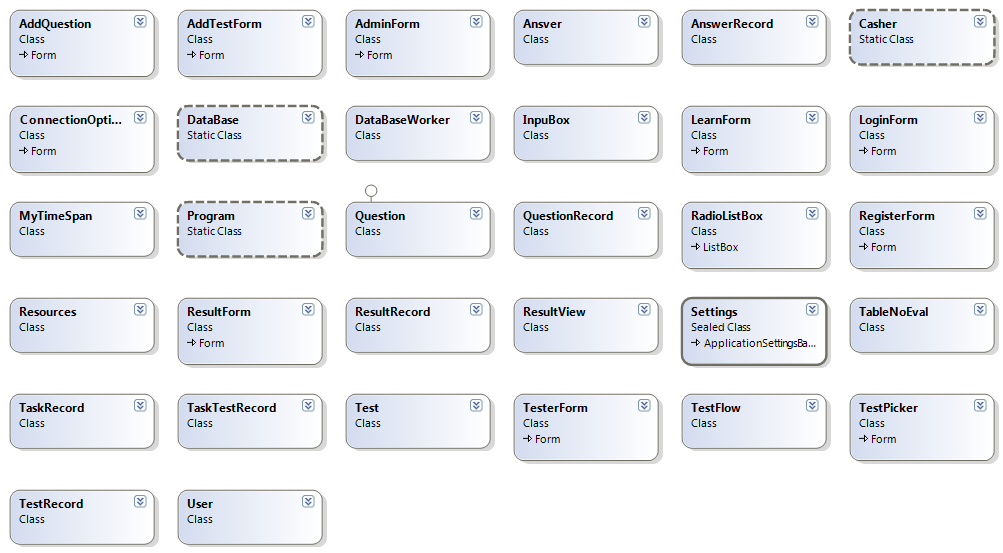


Рис. 6.1. Классы программы

7. Техническая документация

7.1.Назначение программы

Данная информационная система предназначена для автоматизированного проведения тестирования и обучения SQL запросам. А также автоматизированного хранения результатов тестирования.

7.2 Условия приминения

Требования к программному и аппаратному обеспечению компьютера для информационной системы таковы.

* Минимальные требования к аппаратному и программному обеспечению серверной стороны представлены в таблице 7.1;
* Минимальные требования к аппаратному и программному обеспечению клиентской стороны представленны в таблице 7.2;

Таблица 7.1 Минимальные требования к аппаратному обеспечению серверной стороны

|  |  |
| --- | --- |
| Характеристика | Минимальный параметр |
| Процессор | Intel core 2 duo 2000 ггц |
| Оперативная память | 4 ГБ |
| Жесткий диск | 500 ГБ |
| Операционная система | Windows server 2000 |

Таблица 7.2 Минимальные требования к аппаратному обеспечению клиентской стороны

|  |  |
| --- | --- |
| Характеристика | Минимальный параметр |
| Процессор | Intel core 2 duo 2000 ггц |
| Оперативная память | 2 ГБ |
| Жесткий диск | 100 ГБ |
| Операционная система | Windows 7 |
| .Net Framework |  |

7.3. Руководство программиста

Для работы программы необходимы следующие исходные данные:

* операционная система *Windows Vista/7/8*;
* система управления базами данных *Microsoft SQL Server*;
* платформа *Microsoft Faramework 4.5 (клиент)*.

Файлами базы данных являются «*InformationDataBase.mdf*» и «*InformationDataBase\_log.LDF*». Их следует записать в существующую общую папку «*Program Files\Microsoft SQL Server\* *MSSQL10\_50.MSSQLSERVER\MSSQL\Data*.

Подключить базу данных можно двумя способами:

* через графический интерфейс *Microsoft SQL Management Studio;*
* с помощью командной строки сервера.

В первом случае приходится следовать инструкциям в диалоговых окнах. В случае с командной строкой времени уйдет гораздо меньше. Рассмотрим данный способ как основной для подключения базы данных к серверу.

1. Откройте на сервере командную строку.
2. В командной строке подключитесь к экземпляру *SQL Server* с помощью следующей команды ***sqlcmd***:

sqlcmd -S Server\Instance

Здесь ***Server*** — это имя компьютера, а ***Instance*** — имя экземпляра.

1. После подключения введите следующие команды:

USE [master]

GO

CREATE DATABASE *Jobs* ON

( FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL10\_50.MSSQLSERVER\MSSQL\Data\ *InformationDataBase*.mdf' ),

( FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL10\_50.MSSQLSERVER\MSSQL\Data\ *InformationDataBase\_log*.ldf' )

FOR ATTACH ; GO

Где Jobs представляет собой имя базы данных, которую необходимо подключить, FileName — путь и имя файла базы данных и файла журнала, а FOR ATTACH указывает на то, что база данных будет создана подключением к существующему набору файлов операционной системы.

1. Для проверки добавления базы данных введите следующие две команды:

select name from sys.databases

go

1. В программе sqlcmd отобразятся названия всех баз данных, добавленных к этому экземпляру SQL Server Express. В списке должно появиться название базы данных, указанное в шаге 3 (Jobs) [7].

Стоит отметить, что не будет сложным создать базу данных из описания структуры базы данных (п. 2.4), так как в ней содержится всего одна таблица.

Клиентское приложение запускается из файла «*ShoolTester.exe*».

Программа представляет собой интеллектуальную систему, обеспечивающую контроль входных данных, хранение и представление результатов, а также информирование пользователя в понятных терминах на русском языке.

7.4. Руководство пользователя

При первом запуске программы или в случае неудачной попытки подключения к базе данных, если настройка уже была сделана ранее, появляется диалоговое окно настройки подключения к базе данных (см. рис. 7.1).

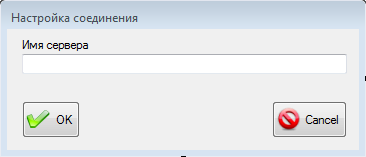


Рис. 7.1. Окно настройки подключения к базе данных

Следует заполнить:

* IP сервера, на котором находится база данных. Имеет маску ХХХ.ХХХ.ХХХ.ХХХ с максимальным значением в 255.
* порт, на который она настроена;
* имя самой базы.

Нажав на кнопку «ОК» возможен один из двух вариантов:

* откроется диалоговое окно ошибки;
* откроется форма авторизации в программе.

В первом случае отобразится окно, представленное на рис. 7.2:

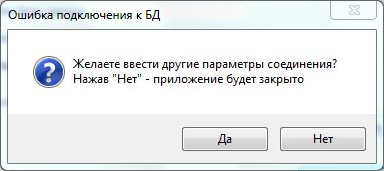


Рис. 7.2. Диалоговое окно ошибки подключения к БД

Если нажать кнопку «Да», то останется форма настройки соединения (рис. 7.1), где можно будет отредактировать параметры, иначе – приложение закроется.

Во втором случае откроется окно авторизации в программе (рис. 7.3).

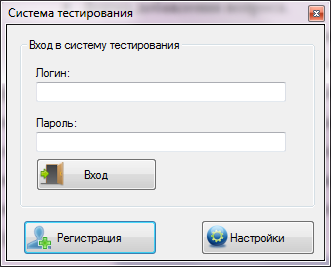


Рис. 7.3. Форма авторизации в программе

Окно авторизации в программы позволяет:

* авторизоваться под известной учетной записью (кнопка «Вход»);
* зарегистрироваться в системе (кнопка «Регистрация»);
* настроить имеющуюся информационную базу (кнопка «Настройки»).

Для авторизации достаточно ввести известный логин и пароль в соответствующие поля и нажать кнопку «Вход». В случае неверных данных появится окно об ошибке. В случае успешного прохождения авторизации откроется основное окно программы (рис. 7.4).

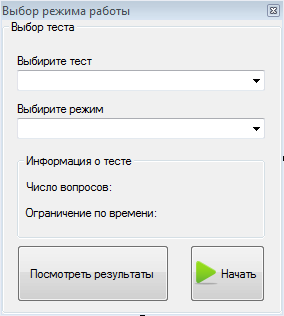


Рис. 7.4. Основное окно программы

Здесь возможен выбор дальнейшего режима работы:

* тестирование (необходимо выбрать нужный тест из списка, выбрать режим «Тестирование» и нажать на кнопку «Начать»);
* обучение (необходимо выбрать нужный тест из списка, выбрать режим «Обучение» и нажать на кнопку «Начать»);
* просмотр результатов (необходимо выбрать нужный тест из списка и нажать на кнопку «Просмотреть результаты»).

В последнем случае, отобразится окно, представленное на рис. 7.5.

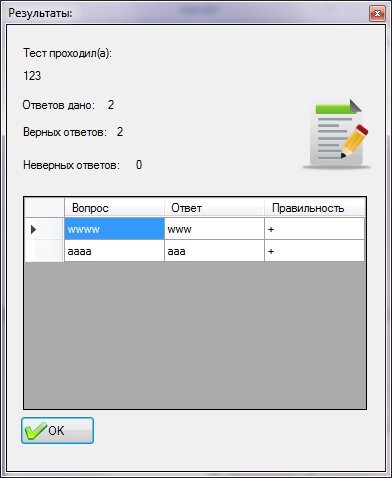


Рис. 7.5. Форма результатов тестирования

Если тестирование по выбранному тесту еще не было проведено, то окно останется без дополнительной информации, как на рис. 7.5. Иначе отобразится количество данных ответов, количество верных ответов, количество неверных ответов и список вопросов с поставленными на них ответами. Кнопка «ОК» закрывает данное окно.

Если был выбран режим обучения, то будет отображено окно, представленное на рис. 7.6.

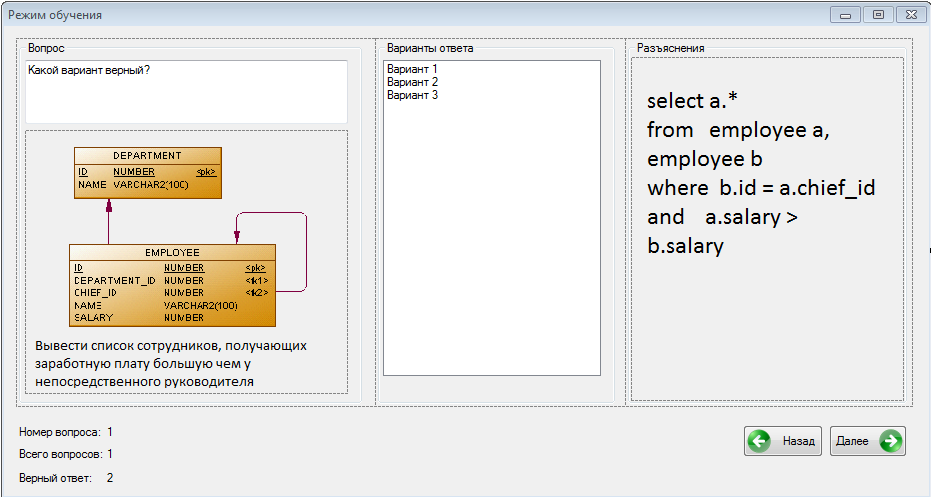


Рис. 7.6. Форма обучения

Здесь отображаются:

* вопрос;
* изображение для вопроса, если таковое имеется;
* варианты ответа;
* разъяснения к вопросу (возможно изображение);
* номер разъясняемого вопроса;
* количество вопросов для разъяснения;
* верный ответ на данный вопрос.

Кнопки «Назад» и «Далее» позволяют возвращаться к предыдущему или переходить к следующему вопросу. Если вопрос был последний, то форма обучения закроется, оставив перед пользователем окно на рис. 7.4.

В случае выбора режима тестирования, перед пользователем отобразиться следующее окно (рис. 7.7):

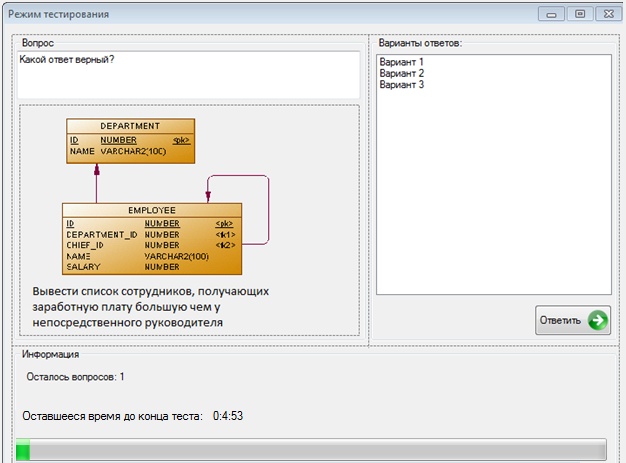


Рис. 7.7. Окно тестирования

В этом окне (рис. 7.7) отображаются:

* вопрос;
* изображение для вопроса, если таковое имеется;
* варианты ответа;
* оставшееся количество вопросов;

оставшееся время выполнения теста с графической шкалой процесса.

Для ответа на вопрос достаточно мышкой выделить нужный вариант ответа в списке «Варианты ответов» и нажать на кнопку «Ответить». После этого в окне отобразиться следующий вопрос, если такой имеется, или будет выведен результат, если был дан ответ на последний вопрос.

Если из основной формы (рис. 7.4) была нажата кнопка «Регистрация», то появится окно, представленное на рис. 7.8.

Форма регистрации содержит в себе поля для ввода Ф.И.О., логина и пароля. Для подтверждения правильности ввода пароля необходимо повторно ввести пароль в соответствующее поле (рис. 7.8).

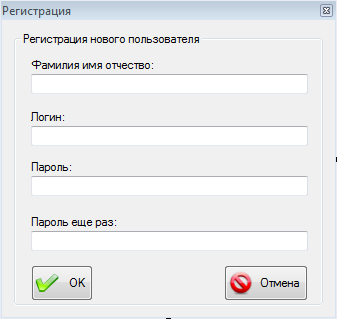


Рис. 7.8. Форма регистрации

Оставлять пустыми ни логин, ни пароль нельзя, т.к. появится предупреждающее сообщение.

В строке Ф.И.О. обязательно должно быть 2 пробела между фамилией, именем и отчеством.

Пароль не должен быть менее 4 символов. Причем, один и тот же символ нельзя задавать в качестве пароля, иначе появится сообщение об ошибке.

В случае успешного заполнения данного окна (рис. 7.8) после нажатия кнопки «ОК» отобразится основное окно программы (рис. 7.4).

Если из основной формы (рис. 7.4) была нажата кнопка «Настройки», то после успешного ввода пароля администратора появится окно, представленное на рис. 7.9.

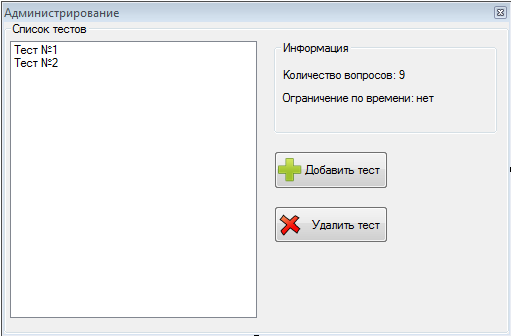


Рис. 7.9. Форма администрирования

В данном окне будут отображены все имеющиеся тесты в базе данных.

Слева отображен список имеющихся тестов. При выделении мышкой любого из них справа отображается краткая информация по тесту:

* количество вопросов в тесте;
* ограничения по времени теста.

При необходимости удалить какой-либо из тестов, нужно выбрать его в списке и нажать на кнопку «Удалить тест». После этого выбранный тест пропадет из списка тестов.

При необходимости добавить новый тест, необходимо нажать на кнопку «Добавить тест». В результате отобразиться окно, представленное на рис. 7.10.

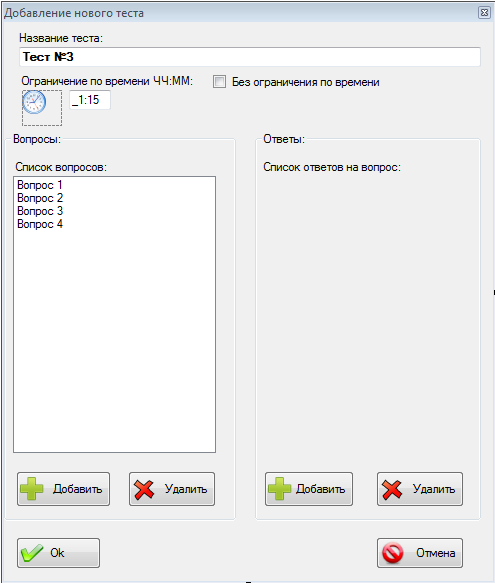


Рис. 7.10. Форма добавления нового теста

Здесь необходимо ввести название теста, время, отведенное на тест, добавить нужные вопросы и варианты ответа. По завершению формирования теста нужно нажать на кнопку «Ок» для сохранения. Новый тест появится в списке тестов (рис. 7.9).

Название теста не должно быть пустым.

Если ограничения по времени теста нет, то необходимо поставить галку на «Без ограничения по времени», иначе перед сохранением будет показано сообщение об ошибке.

Список вопросов не должен быть пустым, иначе перед сохранением появится окно об ошибке.

Список ответов на вопрос не может быть пустым. В каждом вопросе должно быть минимум два варианта ответа, один из которых должен быть правильным.

Во время добавления теста необходимо создавать новые вопросы. Окно создания вопроса представлено на рис. 7.11:

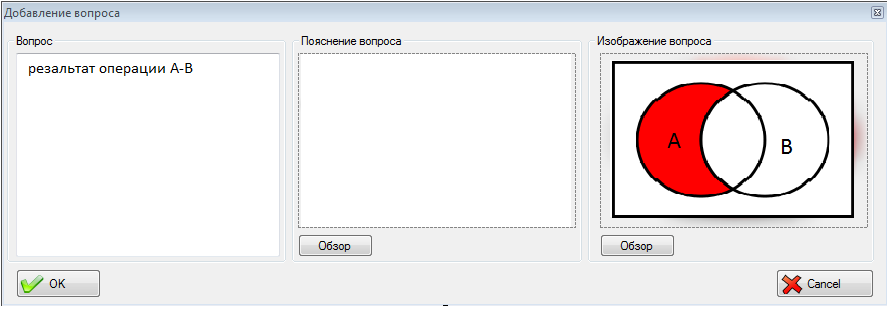


Рис. 7.11. Форма добавления вопроса

В данном окне необходимо ввести текст вопроса (в левой части окна), пояснение (возможно графическое в центральной части окна, нажав на кнопку «Обзор» и выбрав нужное изображение) и выбрать картинку вопроса, если необходимо (в правой части окна, нажав на кнопку «Обзор» и выбрав нужное изображение). Для сохранения вопроса нужно нажать кнопку «ОК». Для отмены сохранения вопроса необходимо нажать на кнопку «Cancel».

По завершению формирования теста (рис. 7.10) нужно нажать на кнопку «Ок» для сохранения или кнопку «Cancel» для отмены сохранения. Новый тест появится в списке тестов (рис. 7.9).

Подводя итог, работать с программой достаточно просто, в ней нет избыточности информации, а все выполняемые действия находятся в логической цепочке последовательностей.

8. Тестирование системы

8.1. Программа и методика испытаний

Разработанная система состоит из клиенского приложения и базы данных. Чтобы не загружать сеть и сервер операциями, качественно способными выполняться на стороне клиента, контроль вводимых данных целиком возложен на клиенскую часть. Программа уже содержит ограничения, соответствующие ограничениям в таблице базы данных.

Обозначим этапы тестирования:

* попытка добавления пустых данных в обязательных полях;
* попытка вставки вместо картинки файла другого формата;
* пропадание соединения с БД.

8.2. Тестирование программного обеспечения

Проведём несколько тестов для различных полей данных.

Так, если попытаться оставить название теста без имени, то возникнет ошибка (рис. 8.1).

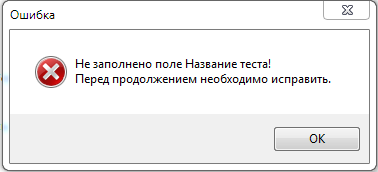


Рис. 8.1. Окно ошибки в указанных данных

Если выбрать файл, который не соответсвует файлу изображения, то возникнет ошибка, представленная на рис. 8.2.

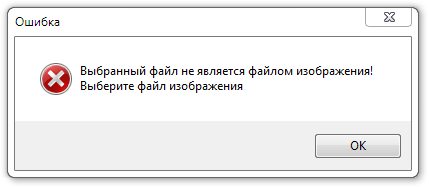


Рис. 8.2. Окно ошибки загрузки файла

Если во время работы пропадает соединение с базой данных, то программа пытается переподключиться к базе в фоновом режиме. Если эта попытка неудачна, то выводится следующее диалоговое окно (рис. 8.3):

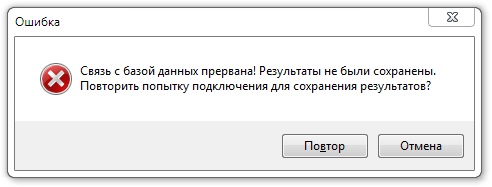


Рис. 8.3. Диалоговое окно пропадания подключения к БД

Как видно, пользователь будет проинформирован о проблеме работы программы.

Для защиты от невнимательности операции по удалению тестов, впросов, а также перед изменением перечисленных предусмотрены диалоговые окна подтверждения выполняемых действий.

Далее проведем тестирование с базой данных. Проведём несколько тестов для различных атрибутов: произведём попытки добавления записей через клиентское приложение. Характеристики параметров приведены в табл. 8.1.

Таблица 8.1.

Характеристики тестируемых значений

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Форма | Введенные данные | Поле | Полученное значение | Действие |
| Форма регистрации | **Поле пустое** | **Фамилия** | Цифры – буквы от 1 до 20 символов | **Нажатие кнопки OK** |
| Форма регистрации | **Поле пустое** | **Имя** | Цифры – буквы от 1 до 20 символов | **Нажатие кнопки OK** |
| Форма регистрации | **Поле пустое** | **Отчество** | Цифры – буквы от 1 до 20 символов | **Нажатие кнопки OK** |
| Форма ввода вопроса | **Поле пустое** | **Вопрос** | Цифры – буквы от 1 до 20 символов | **Нажатие кнопки OK** |

Результаты тестирования значений этих атрибутов сведены в табл. 8.2.

Таблица 8.2.

Результаты тестирования значений атрибутов таблиц

|  |  |  |  |
| --- | --- | --- | --- |
| Форма | Введенные данные | Поле | Полученное значение |
| Ввод перс. данных | **123йцу** | **Фамилия** | Ошибка формата фамилии |
| Ввод перс. данных | **Йцч12фы4** | **Имя** | Ошибка формата имени |
| Ввод перс. данных | **Йцу3 23ц4** | **Отчество** | Ошибка формата отчества |
| Ввод перс. данных | **Иыфы фцвц** | **Имя** | Ошибок не обнаружено |
| Ввод перс. данных | **ПКвыа ывак** | **Отчество** | Ошибок не обнаружено |
| Ввод вопроса | **«пустая строка»** | **Вопрос** | Ошибка формата вопроса |

8.3. Выводы по результатам тестирования

В программе предусмотрены механизмы предотвращения появления наивероятнейших ошибок. Таким образом, у пользователя нет возможности ввести некорректные данные - отобразится сообщение об ошибке.

Заключение

В процессе работы над дипломным проектом была разработана база данных и клиентское приложение. Система позволяет автоматизировать процесс обучения, а также тестирования имеющихся знаний в области программирования SQL запросов.

Подготовленный программный пакет обладает следующим рядом свойств:

* предусмотрено редактирование базы тестов;
* просмотр результатов тестирования;
* выполнение обучения;
* создана система контроля входных данных, отвечающая за коррекцию и информирование о состоянии процесса индексации;

Разработанный комплекс может быть введён в эксплуатацию на любой кафедре ВУЗа.

Перечисленные выше достоинства разработки делают данный проект актуальным и перспективным для использования в учебных заведениях.

В дальнейшем планируется доработка проекта – расширение областей тестирования.

Список использованных источников

1. Дипломное проектирование по специальностям 220400 и 351400: Методические указания / Рязан. гос. радиотехн. акад.; Сост. И.А. Цветков. Рязань, 2005. 48 с.
2. Троелсен Э. C# и платформа .NET. Библиотека программиста. — СПб.: Питер, 2007. — 796 с.: ил.
3. Джеффри Фридл. Регулярные выражения (3-е издание). — СПб.: Символ-Плюс, 2008. — 608 с.: ил.
4. <http://ru.wikibooks.org/wiki/Объектно-ориентированное_программирование>
5. <http://msdn.microsoft.com/ru-ru/library/az24scfc.aspx>
6. <http://technet.microsoft.com/ru-ru/library/ms165673(v=sql.105).aspx>
7. ГОСТ 19.505–79 Руководство пользователя.
8. ГОСТ 19.503–79 Руководство системного программиста.
9. ГОСТ 19.001-77 Единая система программной документации.
10. ГОСТ 19.103-77 Обозначение программ и программных документов.
11. ГОСТ 19.701-90 Схемы алгоритмов программ данных и систем.

ПРИЛОЖЕНИЕ 1

ЛИСТИНГ НАИБОЛЕЕ ЗНАЧИМЫХ ЧАСТЕЙ ПРОГРАММЫ

Файл AddQuestion.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.IO;

namespace Tester

{

/// <summary>

/// Форма добавления вопроса

/// </summary>

public partial class AddQuestion : Form

{

public byte[] Image { get; set; }

public byte[] ImageQuestion { get; set; }

public String QuestionName { get; set; }

public AddQuestion()

{

InitializeComponent();

}

/// <summary>

/// Открытие картинки которая

/// служит пояснениями

/// </summary>

///

private void button3\_Click(object sender, EventArgs e)

{

OpenFileDialog openFileDialog = new OpenFileDialog();

openFileDialog.Filter = "изображения | \*.jpg;\*.jpeg;\*.png;\*.bmp";

var result = openFileDialog.ShowDialog();

if (result == System.Windows.Forms.DialogResult.OK)

{

pictureBox1.Image = System.Drawing.Image.FromFile(openFileDialog.FileName);

MemoryStream ms = new MemoryStream();

pictureBox1.Image.Save(ms, System.Drawing.Imaging.ImageFormat.Jpeg);

Image = new byte[ms.Length];

ms.Position = 0;

ms.Read(Image, 0, Image.Length);

}

}

/// <summary>

/// Добавление вопроса

/// </summary>

private void button1\_Click(object sender, EventArgs e)

{

DialogResult = System.Windows.Forms.DialogResult.OK;

if (this.ImageQuestion == null)

ImageQuestion = new byte[0];

if (this.Image == null)

{

var im = System.Drawing.Image.FromFile(Environment.CurrentDirectory + @"\Images\img.jpg");

MemoryStream ms = new MemoryStream();

im.Save(ms, System.Drawing.Imaging.ImageFormat.Jpeg);

Image = new byte[ms.Length];

ms.Position = 0;

ms.Read(Image, 0, Image.Length);

}

QuestionName = textBox1.Text;

}

/// <summary>

/// Закрытие окна

/// </summary>

private void button2\_Click(object sender, EventArgs e)

{

Close();

}

private void button4\_Click(object sender, EventArgs e)

{

OpenFileDialog openFileDialog = new OpenFileDialog();

openFileDialog.Filter = "изображения | \*.jpg;\*.jpeg;\*.png;\*.bmp";

var result = openFileDialog.ShowDialog();

if (result == System.Windows.Forms.DialogResult.OK)

{

pictureBox2.Image = System.Drawing.Image.FromFile(openFileDialog.FileName);

MemoryStream ms = new MemoryStream();

pictureBox2.Image.Save(ms, System.Drawing.Imaging.ImageFormat.Jpeg);

ImageQuestion = new byte[ms.Length];

ms.Position = 0;

ms.Read(ImageQuestion, 0, ImageQuestion.Length);

if (pictureBox2.Image == null)

ImageQuestion = new byte[0];

}

}

}

}

Файл AdminForm.cs

using System;

using System.Collections.Generic;

using System.Windows.Forms;

using ShoolTester.Properties;

using Tester.Models;

using Tester.Properties;

namespace Tester

{

/// <summary>

/// Администрирование тестов

/// </summary>

public partial class AdminForm : Form

{

Dictionary<int, Test> \_tests;

public AdminForm(Dictionary<int, Test> tests)

{

InitializeComponent();

listBox\_Testes.DisplayMember = "Name";

\_tests = tests;

RefreshList();

}

/// <summary>

/// Обновление списка тестов

/// </summary>

private void RefreshList()

{

\_tests.Clear();

\_tests = DataBase.GetAllTests();

listBox\_Testes.Items.Clear();

if (\_tests.Count != 0)

foreach (var t in \_tests)

listBox\_Testes.Items.Add(t.Value);

}

/// <summary>

/// Выделение теста

/// </summary>

private void ListBoxTestesSelectedIndexChanged(object sender, EventArgs e)

{

// Кнопка удалить

if (listBox\_Testes.Items.Count != 0 && listBox\_Testes.SelectedIndex != -1)

button\_DeleteTest.Enabled = true;

else

button\_DeleteTest.Enabled = false;

// Вывод краткой информации о тесте

if (\_tests.Count != 0 && listBox\_Testes.Items.Count != 0 && listBox\_Testes.SelectedIndex != -1)

{

var t = (Test)listBox\_Testes.SelectedItem;

label\_TimeConstr.Text = t.TimeConstraint.TimeOfDay.ToString();

if (label\_TimeConstr.Text == Resources.AdminForm\_ListBoxTestesSelectedIndexChanged\_\_00\_00\_00)

label\_TimeConstr.Text = Resources.AdminForm\_ListBoxTestesSelectedIndexChanged\_Без\_ограничения\_;

label\_QuestCount.Text = t.QuestCount.ToString();

}

}

/// <summary>

/// Удаление теста

/// </summary>

private void button\_DeleteTest\_Click(object sender, EventArgs e)

{

if (listBox\_Testes.SelectedItem != null)

{

var t = (Test)listBox\_Testes.SelectedItem;

DataBase.DeleteTest(t.Id);

listBox\_Testes.Items.RemoveAt(listBox\_Testes.SelectedIndex);

}

}

/// <summary>

/// Добавление теста

/// </summary>

private void button\_AddTest\_Click(object sender, EventArgs e)

{

var addTestForm = new AddTestForm();

Hide();

var result = addTestForm.ShowDialog();

if (result == DialogResult.OK ||result == System.Windows.Forms.DialogResult.Cancel )

{

button\_DeleteTest.Enabled = false;

Show();

RefreshList();

}

}

}

}

Файл AdminForm.cs

using System;

using System.Collections.Generic;

using System.Windows.Forms;

using ShoolTester.Properties;

using System.IO;

using Tester.Models;

namespace Tester

{

public partial class LearnForm : Form

{

readonly List<ResultRecord> \_results = new List<ResultRecord>();

readonly TestFlow \_tf;

Question \_q;

MemoryStream ms;

int currentQuest = 1;

public LearnForm()

{

InitializeComponent();

}

public LearnForm(TestFlow parTf)

{

InitializeComponent();

\_tf = parTf;

\_q = \_tf.GetNextQuestion();

ms = new MemoryStream(\_q.Image);

ms.Seek(0, SeekOrigin.Begin);

pictureBox1.Image = System.Drawing.Image.FromStream(ms);

if (\_q.QuestionImage.Length != 0)

{

ms = new MemoryStream(\_q.QuestionImage);

ms.Seek(0, SeekOrigin.Begin);

pictureBox2.Image = System.Drawing.Image.FromStream(ms);

}

else

pictureBox2.Image = null;

textBox1.Text = \_q.Name;

for (int i = 0; i < \_q.Count; i++)

Ansvers\_listBox.Items.Add(\_q[i]);

QuestCount\_label.Text = currentQuest.ToString();

label3.Text = \_tf.QuestionCount().ToString();

foreach (var tr in \_q.Ansvers)

if (tr.ID == \_q.TrueAnsverID)

label5.Text = tr.Text;

}

// NEXT

private void button2\_Click(object sender, EventArgs e)

{

if (\_tf.QuestionCount() > \_tf.Current + 1)

{

ms = new MemoryStream(\_q.Image);

ms.Seek(0, SeekOrigin.Begin);

pictureBox1.Image = System.Drawing.Image.FromStream(ms);

currentQuest = \_tf.Current + 2;

QuestCount\_label.Text = currentQuest.ToString();

\_q = \_tf.GetNextQuestion();

ms = new MemoryStream(\_q.Image);

ms.Seek(0, SeekOrigin.Begin);

pictureBox1.Image = System.Drawing.Image.FromStream(ms);

if (\_q.QuestionImage.Length != 0)

{

ms = new MemoryStream(\_q.QuestionImage);

ms.Seek(0, SeekOrigin.Begin);

pictureBox2.Image = System.Drawing.Image.FromStream(ms);

}

else

pictureBox2.Image = null;

foreach (var tr in \_q.Ansvers)

if (tr.ID == \_q.TrueAnsverID)

label5.Text = tr.Text;

textBox1.Text = \_q.Name;

Ansvers\_listBox.Items.Clear();

for (int i = 0; i < \_q.Count; i++)

Ansvers\_listBox.Items.Add(\_q[i]);

}

}

// Back

private void button1\_Click(object sender, EventArgs e)

{

if (\_tf.Current > 0)

{

currentQuest = currentQuest - 1;

QuestCount\_label.Text = currentQuest.ToString();

\_q = \_tf.GetBackQuestion();

ms = new MemoryStream(\_q.Image);

ms.Seek(0, SeekOrigin.Begin);

pictureBox1.Image = System.Drawing.Image.FromStream(ms);

if (\_q.QuestionImage.Length != 0)

{

ms = new MemoryStream(\_q.QuestionImage);

ms.Seek(0, SeekOrigin.Begin);

pictureBox2.Image = System.Drawing.Image.FromStream(ms);

}

else

pictureBox2.Image = null;

foreach (var tr in \_q.Ansvers)

if (tr.ID == \_q.TrueAnsverID)

label5.Text = tr.Text;

textBox1.Text = \_q.Name;

Ansvers\_listBox.Items.Clear();

for (int i = 0; i < \_q.Count; i++)

Ansvers\_listBox.Items.Add(\_q[i]);

}

}

private void button3\_Click(object sender, EventArgs e)

{

Close();

}

}

}

Файл TesterForm.cs

using System;

using System.Collections.Generic;

using System.Windows.Forms;

using ShoolTester.Properties;

using System.IO;

using Tester.Models;

using Tester.Properties;

namespace Tester

{

/// <summary>

/// Форма для прохождения тестирования

/// </summary>

public partial class TesterForm : Form

{

readonly List<ResultRecord> \_results = new List<ResultRecord>();

readonly TestFlow \_tf;

Question \_q;

readonly int \_max;

/// <summary>

/// Инициализация

/// </summary>

public TesterForm(TestFlow parTf)

{

InitializeComponent();

\_tf = parTf;

\_tf.TestTick += TfTestTick;

\_tf.TestFinished += TfTestFinished;

\_q = \_tf.GetNextQuestion();

if (\_q.QuestionImage.Length != 0)

{

var ms = new MemoryStream(\_q.QuestionImage);

ms.Seek(0, SeekOrigin.Begin);

pictureBox1.Image = System.Drawing.Image.FromStream(ms);

}

else

pictureBox1.Image = null;

textBox1.Text = \_q.Name;

for (int i = 0; i < \_q.Count; i++)

Ansvers\_listBox.Items.Add(\_q[i]);

if (\_tf.TimeConstraint().Hour == 0 && \_tf.TimeConstraint().Minute == 0)

label\_Time\_Last.Text = "Без ограничения по времени";

\_max = \_tf.Span().\_minutes \* 60 + \_tf.Span().\_hours \* 60 \* 60 + \_tf.Span().\_seconds;

progressBar1.Maximum = \_max;

QuestCount\_label.Text = \_tf.QuestionCount().ToString();

}

/// <summary>

/// Вызывается если время вышло

/// вставка результатов

/// </summary>

/// <param name="e"></param>

void TfTestFinished(EventArgs e)

{

DataBase.InsertResults(\_results);

MessageBox.Show(Resources.TesterForm\_TfTestFinished\_Тест\_завершон);

List<ResultView> datasourse = DataBase.GetResults(\_tf.ID, DataBase.UserId);

var resultFormDoalog = new ResultForm(datasourse);

Hide();

if (resultFormDoalog.ShowDialog() == DialogResult.Cancel)

Close();

}

/// <summary>

/// Таймер

/// </summary>

void TfTestTick(EventArgs e)

{

label\_Time\_Last.Text = \_tf.Span().ToString();

if (progressBar1.Value < \_max)

progressBar1.Value = ++progressBar1.Value;

}

/// <summary>

/// Кнопка ответа

/// </summary>

private void AnsverButtonClick(object sender, EventArgs e)

{

int current = Convert.ToInt32(QuestCount\_label.Text);

current--;

QuestCount\_label.Text = current.ToString();

var res = new ResultRecord();

string s = "";

int ida = 0;

// Если ответ выбран

if (Ansvers\_listBox.SelectedItem != null)

{

s = (string)Ansvers\_listBox.SelectedItem;

foreach (var r in \_q.Ansvers)

{

if (s == r.Text)

ida = r.ID;

}

// заполнение результата

res.TestID = \_tf.ID;

res.Ans = ida;

res.Trues = \_q.TrueAnsverID;

res.Quest = \_q.ID;

res.UserID = DataBase.UserId;

\_results.Add(res);

// Следующий вопрос

\_q = \_tf.GetNextQuestion();

if (\_q.QuestionImage != null && \_q.QuestionImage.Length > 0)

{

var ms = new MemoryStream(\_q.QuestionImage);

ms.Seek(0, SeekOrigin.Begin);

pictureBox1.Image = System.Drawing.Image.FromStream(ms);

}

else

pictureBox1.Image = null;

textBox1.Text = \_q.Name;

Ansvers\_listBox.Items.Clear();

for (int i = 0; i < \_q.Count; i++)

Ansvers\_listBox.Items.Add(\_q[i]);

}

}

}

}

Файл TestPicker.cs

using System;

using System.Collections.Generic;

using System.Windows.Forms;

using Tester.Models;

namespace Tester

{

public partial class TestPicker : Form

{

public int TestID;

private TestFlow \_tf;

readonly Dictionary<int, Test> \_tests;

public TestPicker()

{

InitializeComponent();

}

public TestPicker(Dictionary<int, Test> tests)

{

InitializeComponent();

foreach (var t in tests)

comboBox\_Tests.Items.Add(t.Value.Name);

\_tests = tests;

}

private void GetTestID()

{

if (comboBox\_Tests.SelectedIndex != -1)

{

var s = (string)comboBox\_Tests.Items[comboBox\_Tests.SelectedIndex];

foreach (var t in \_tests)

{

if (t.Value.Name == s)

TestID = t.Key;

}

}

else

throw new System.ArgumentException("Не выбран елемент");

}

private void ButtonStartTestClick(object sender, EventArgs e)

{

GetTestID();

DataBase.ClearResults(DataBase.UserId, TestID);

\_tf = DataBase.FillTest(TestID);

switch (comboBox1.SelectedIndex)

{

case 0:

{

var learnForm = new LearnForm(\_tf);

Hide();

\_tf.Start();

if (learnForm.ShowDialog() == DialogResult.Cancel)

Show();

break;

}

case 1:

{

var testreForm = new TesterForm(\_tf);

Hide();

\_tf.Start();

if (testreForm.ShowDialog() == DialogResult.Cancel)

Show();

break;

}

case -1: { MessageBox.Show("Выберите режим"); break; }

}

}

private void ButtonSeeResultsClick(object sender, EventArgs e)

{

GetTestID();

List<ResultView> datasourse = DataBase.GetResults(TestID, DataBase.UserId);

var resultFormDoalog = new ResultForm(datasourse);

resultFormDoalog.Show();

}

private void comboBox\_Tests\_SelectedIndexChanged(object sender, EventArgs e)

{

try

{

GetTestID();

QuestionCount\_Label.Text = \_tests[TestID].QuestCount.ToString();

TimeConstraint\_label.Text = \_tests[TestID].TimeConstraint.TimeOfDay.ToString();

if (TimeConstraint\_label.Text == "00:00:00")

TimeConstraint\_label.Text = "Без ограничения";

}

catch (ArgumentException ex)

{

MessageBox.Show(ex.Message);

}

}

}

}