МИНИСТЕРСТВО ОБРАЗОВАНИЯ И МОЛОДЁЖНОЙ ПОЛИТИКИ СВЕРДЛОВСКОЙ ОБЛАСТИ
государственное автономное профессиональное образовательное учреждение
Свердловской области
«Верхнесалдинский авиаметаллургический колледж имени А.А. Евстигнеева»
(ГАПОУ СО «ВСАМК им. А.А. Евстигнеева»)

ДИПЛОМНЫЙ ПРОЕКТ

на тему: Разработка автоматизированной информационной системы «Формирование междисциплинарных тестовых заданий»

Специальность 09.02.07
Информационные технологии и программирование
ДП.09.02.07.ПЗ.09.00.00

Выпускник Лунев Леонид Михайлович
группа ИСП-301

Верхняя Салда 2021

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И МОЛОДЁЖНОЙ ПОЛИТИКИ СВЕРДЛОВСКОЙ ОБЛАСТИ

государственное автономное профессиональное образовательное учреждение Свердловской области
«Верхнесалдинский авиаметаллургический колледж имени А.А. Евстигнеева»
(ГАПОУ СО «ВСАМК им. А.А. Евстигнеева»)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к дипломному проекту

на тему: Разработка автоматизированной информационной системы «Формирование междисциплинарных тестовых заданий»

Специальность 09.02.07
Информационные технологии и программирование

Выпускник гр. ИСП-301	_____ (подпись)	Лунев Л.М. (Фамилия И.О.)
Руководитель работы	_____ (подпись)	Федюкович С.В. (Фамилия И.О.)
Рецензент	_____ (подпись)	_____ (Фамилия И.О.)
Консультант	_____ (подпись)	_____ (Фамилия И.О.)
Нормоконтроль	_____ (подпись)	_____ (Фамилия И.О.)

Допустить к защите
Директор колледжа _____ (подпись) Ракитина Н.А. (Фамилия И.О.)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью данного дипломного проекта является разработка основы создания программного обеспечения, предназначенного для создания междисциплинарных тестовых заданий , то есть тестов для проверки усвоения учебного материала учащимися, пригодных для организации учебного процесса в колледже и применения в других учебных заведениях и организациях, где необходимо обучение персонала и контроля уровня теоретической подготовки.

Для достижение указанной цели будет решена задача создания работающей программы, в которой можно будет размещать наборы вопросов и ответов по различным учебным дисциплинам с пометками, какие из ответов являются правильными. Исходный код данной программы будет размещён в открытом доступе под системой управления версиями Git на Интернет-ресурсе так, чтобы сделать возможной дальнейшую разработку нашей программы, доработку её под разнообразные требования учебного процесса, которые могут появится или быть сформулированными в будущем.

Выпускнику колледжа за годы учёбы приходится овладевать очень большим объёмом знаний, причём с каждым годом этот объём всё более увеличивается. Знания, относящиеся к избранной специальности, становятся всё разнообразнее и шире, появляется всё больше мест и способов, откуда и как их можно подчерпнуть. Усвоение знаний ведётся с помощью компьютерной техники, видео на Youtube, разнообразных лекций, семинаров онлайн, которые прочно вошли в нашу жизнь, поэтому контроль за усвоением знаний также следует вести с помощью компьютерных методов, проверяя, насколько качественно учащиеся усвоили преподаваемый предмет, насколько глубоко они понимают изучаемый материал.

Чем более разнообразными и доступными будут средства компьютерного контроля и проверки знаний, тем легче и быстрее будет идти процесс обучения. Привычка использования компьютерных методов контроля

результатов учёбы поможет учащимся больше внимания обратить на огромные образовательные возможности, которые предоставляет Интернет.

Поскольку наш колледж обеспечивает подготовку специалистов по специальностям авиаметаллургии, уникальном направлении, для которого трудно подобрать аналоги в мировом учебном процессе, нам тем более нужны настраиваемые учебные тестовые материалы, которые годятся для всех предметов, преподаваемых в колледже. Решению этой задачи призвана способствовать. данная дипломная работа, посвящённая созданию универсальной программной оболочки для подготовки тестов.

Создавать с нуля тест по каждому предмету может явиться весьма непростой задачей, которую может облегчить готовый тестовый программный каркас, который от предмета к предмету будет заполняться разными вариантами вопросов и ответов на них, оставляя неизменной общую организацию тестовой программы, разработку которой мы ставим свой целью.

В тексте пояснительной записки мы укажем, что готовые программные средства хотя и могут быть применены в образовательной работе в колледже и на предприятии, но не всегда в достаточной степени годятся для глубокого и планомерного изучения учебных дисциплин, повышения уровня профессионального образования и проверки знаний на производстве.

Наилучшим способом получить наиболее эффективное программное обеспечение и научиться его разрабатывать является его конкретная разработка, максимально приспособленная к необходимым нам задачам обучения, тестирования, проверки знаний.

В тексте пояснительной записки будет показано, что желаемыми свойствами расширяемости, адаптируемости, гибкости в применении может обладать только программа с открытым программным кодом, предпочтительно выложенная на GitHub, крупнейший репозиторий, хранилище программного обеспечения, которое позволяет сотрудничать программистам всего мира, всем

тем, кто заинтересован в развитии данной программы. Мы подробно изложим, как будет использован механизм открытой разработки.

Чем шире будет коллектив программистов, желающих отдать свой труд и время совершенствованию нашей программы, тем более качественной она будет, тем быстрее и с большей отдачей мы сможем её применять в учебном процессе.

Поэтому для написания программы следует выбрать распространённый язык программирования, для которого существует хороший набор инструментальных средств. Почти вне конкуренции здесь оказывается язык C#([1]), по версии TIOBE([2]) пятый по популярности язык программирования, прекрасно приспособленный для написания программ под операционной системой Windows, с огромным количеством библиотек, обеспечивающих программиста всеми необходимыми функциями.

Мы начнём разработку с десктопного варианта программы, но при дальнейшем расширении нашей системы язык C# может быть использован и для программ , работающих на мобильных устройствах, и для клиент-серверных Интернет-приложений.
Мы будем использовать итеративный метод разработки , в противоположность водопадному (каскадному) методу, при котором создание программного обеспечения идёт поэтапно, от проектирования до программирования и тестирования.
Выбрав итеративный метод, мы будем сразу стремиться получить работающую модель программы с минимумом необходимых функций, но с возможностью добавления новых так, чтобы в итоге получить полезную на практике программу.

Выложив на GitHub.com программу с минимумом функций, но с возможностью дальнейшей доработки, мы создадим рабочую среду, из которой наш колледж будет получать программное обеспечение для учебных нужд и которая сама по себе является прекрасным учебным виртуальным классом для подготовки будущих работников информационной индустрии - программистов.

В последующий главах мы подробно опишем все этапы разработки нашей программы: в первой главе проанализируем предметную область, рассмотрим существующие аналоги нашей программы, выполняющие схожие с требуемыми для нас функции выработаем общие принципы, которыми будем руководствоваться при разработке, которые будут заложены в её основу.

Вторая глава будет посвящена анализу того, какие программные средства следует использовать для реализации задуманного проекта , докажем правильность выбора C# в качестве языка программирования, выбору графической оболочки для взаимодействия с пользователем и базы данных для хранения самих тестов и результатов тестирования.

Наконец, в третьей главе мы детально опишем весь процесс создания программы, в результате которого будет получена рабочая копия проекта, расположенная в свободном доступе на GiHub, основываясь на которой можно дальше вести разработку программных систем, пригодных для организации учебного процесса вокруг формирования междисциплинарных тестовых заданий, создания программ, пригодных для использования как в нашем колледже , так и в других учебных заведениях и на производстве.

ГЛАВА 1

Теоретическая часть. Анализ требований.

1.1 Анализ предметной области

Для выбора языка программирования, типа создаваемой программы и инструментальных средств для работы нам в первую очередь следует внимательно проанализировать предметную область, то есть более внимательно рассмотреть ту среду, для функционирования в которой в первую очередь предназначена наша программа.

Следует убедится, что выбор темы нашей работы соответствует реальным потребностям потребителей программного обеспечения, в первую очередь колледжа и предприятия ПАО «Корпорация ВСМПО-АВИСМА», поскольку выпускники нашего колледжа призваны в первую очередь обеспечивать кадрами градообразующее предприятие нашего города, выпускающее изделия из титановых сплавов для нужд авиационной промышленности, что и отражено в названии нашего учебного заведения «Верхнесалдинский авиаметаллургический колледж». Наше предприятие всегда славилось наукоёмкостью, большим удельным весом НИОКР (Научно-исследовательских, опытно-конструкторских работ) в своей деятельности, поэтому тестирование персонала, проверка знаний, квалификации требуется там постоянно, так что можно надеяться на применение разрабатываемой программы тестирования ив производственных условиях.

Выбор языка программирования и инструментальных средств тоже диктуется тем, что предприятие ПАО «Корпорация ВСМПО-Ависма» является , как гласит его название, корпорацией, то есть объединением большого числа более мелких предприятий, находящихся на разных местах в производственной цепочке, которые тем не менее должны действовать согласованно, отсюда происходит большая важность выбора единых стандартов для всех подразделений.

Для корпораций такого типа стандартом де-фактом для создания программного обеспечения является выбор языка C#. Действительно, на ПАО «Корпорация ВСМПО-Ависма» C# широко применяется, и владеющие им лица будут с готовностью востребованы на производстве. Для создания

программного обеспечения на языке C# фактически стандартным выбором для инструментальной среды программирования является выбор интегрированной среды разработки Microsoft Visual Studio, которая также широко применяется на ПАО «Корпорация ВСМПО-Ависма». Учебный вариант этой среды Community Edition широко доступен и может быть бесплатно скачен с сайта Microsoft([3]).

Данный инструментарий предназначен для работы под управлением операционной системы Windows 10, самой современной версии из семейства Windows, желательно обновлённой до самой современной её модификации. Эта операционная система является фактически стандартной для всех достаточно новых компьютеров, используемых в учебном процессе и на производстве.

1.2 Сравнительный анализ систем – аналогов

В наш информационный век все идеи должны быть кем-то реализованы, поэтому по теме нашей работы сделаем поиск в Интернете, набрав в строке поисковика «программы для создания учебных тестов», можем провести поиск, используя английскую фразу «test generator software», количество вариантов программ для создания тестов будет ещё большим.

Из множества ссылок для нашего исследования выберем следующие: «Универсальный тест»([5]), «Конструктор тестов»([6]), «Конструктор интерактивных квиз-тестов»([7])

Название теста	Стоимость полной лицензии(руб.)
«Конструктор тестов»	14990
«Универсальный тест»	150
«Конструктор интерактивных квиз-тестов»	4990/месяц

Таблица 1

Знакомясь ближе со всеми предлагаемыми вариантами программного обеспечения для создания тестов, проверки знаний учащихся, убеждаемся, что эти варианты, в качестве достоинств, позволяют:

1) Организовывать разнообразные многопредметные опросы по всем отраслям знаний
2) Применять всевозможные аудио и видео средства, видеоклипы, звуковые файлы, рисунки, чертежи.
3) Использовать, как правило, при наличии в учреждении достаточно современных компьютеров, уже имеющуюся технику, без закупки дополнительного оборудования.

Наряду с перечисленными достоинствами, приведённые программные средства обладают следующими недостатками:

1) Нуждаются в значительных затратах времени на освоение, чтобы педагоги научились наиболее эффективно применять их в учебной практике.
2) Требуют оплаты для полноценного применения.
3) Не могут быть расширены или полностью адаптированы к учебной практике колледжа или производственной деятельности предприятия, когда в процессе применения программы возникнут пожелания по изменению её работы или появятся новые требования в ходе учебного процесса.

1.3 Основные требования к разрабатываемой программе

Ознакомившись с приведёнными программами – аналогами и другими, приходим к выводу, что самостоятельное создание с нуля проверочной, тестирующей знания программы не только доставит нам программный продукт с нужными свойствами, но и позволит овладеть всеми навыками и приёмами, которыми должен обладать современный программист-разработчик.

Основной приём, который нужно использовать при создании такой программы – это иметь запас готовых ответов на вопросы, из которых учащийся будет выбирать нужные. Набор ответов может составлять как сам учитель, так и учащийся. Если программа будет использоваться не в режиме проверки знаний, а в режиме обучения. Выбор нужного ответа может происходить как из представленного списка готовых ответов, так и вводом ключевых слов. Если

проверка идёт через ввод ключевых слов, то ответ засчитывается правильным, если введённая фраза содержит нужные ключевые слова,. Для данного вопроса может быть один правильный ответ, может быть ни одного, может быть несколько правильных ответов. Информация о том, какое количество правильных ответов и каких именно, должна находится в базе данных вместе с самими ответами в виде специальной отметки напротив каждого ответа.

Вопросы могут задаваться в текстовом виде, могут быть представлены в виде графических изображений, видеоклипов, математических формул, дат, таблиц, графиков, чертежей, электронных схем, географических карт, звуковых файлов.

В качестве заданий могут выступать такие действия как: выбрать нужное, выделить неправильное, убрать лишнее, добавить недостающее. Ответами могут служить не только галочки, отметки возле заранее заданных текстовых ответов, но и напечатанные на клавиатуре слова, фразы, графические знаки, начерченные мышкой, области, выделенные на экране, слова, произнесённые вслух. При оценке итогов теста могут учитываться количества попыток, сделанных для нахождения правильного ответа. Все полученные в результате тестирования данные должны сохранятся в базе данных наряду с вопросным материалом тестов и быть доступны для последующего анализа, печати в виде бумажных документов, передачи в другие программы и базы данных.

Следует учесть необходимость разделения ролей при работе программы с соответствующим наборами прав доступа к разным её функциям. Права учащегося должны, как правило подразумевать возможность доступа к вопросам по некоторому учебному предмету, тогда как права преподавателя должны, как правило, подразумевать возможность составлять списки вопросов по преподаваемым предметам и подготавливать варианты ответов к ним. Права администратора предполагают возможность контроля и изменения всех функций программы.

Чтобы разграничить все три функции, предполагается при входе в систему указывать логин и пароль, по которым система будет опознавать пользователя и

отводить ему соответствующую роль согласно заложенным в систему администратором данных.

ГЛАВА 2
Проектирование программного обеспечения.

2.1 Выбор типа создаваемой программной системы

Выбрав операционную систему, среду разработки и язык программирования, мы должны определить тип создаваемой программной системы, должна ли она принадлежать к классу клиент-серверных разработок, мобильных приложений или десктопных программ. Как уже указывалось, выбор языка C# и Visual Studio нисколько не ограничивает нас в дальнейшем выборе типа приложения, поскольку на C# и Visual Studio можно создавать приложения любого типа!
Мы делаем выбор в пользу десктопной системы, потому что десктопная система не требует разработки серверной части кода, и позволит быстро создать действующий прототип для нашей программы тестирования. Мы начнём с десктопного варианта, имея ввиду, что полученный код сможет быть впоследствии доработан и для клиент-серверных применений и использован для работы мобильных приложений.

2.2 Среды разработки, сравнение и выбор

Для создания компьютерной программы необходимо средство создания кода, то есть текстовый редактор. Он может быть таким простым и распространённым, как notepad.exe, который входит в состав пакета обязательных программ на любой Windows, или менее известным, таким как Emacs или Vim.
Одно условие – это должен быть простой текстовый редактор, Microsoft Word, например, подходит не очень для этой цели, поскольку добавляет элементы форматирования, удобные для визуального представления текста, но ненужные для перевода текста в вид, пригодный для выполнения на компьютере(машинный код). И в дополнение к средствам создания и редактирования текстовых файлов нам понадобятся средства компиляции, преобразования этих файлов в машинный код. Мы можем загрузить все эти средства отдельно с сайта Microsoft, и компилятор, и средства загрузки библиотек, но гораздо проще загрузить на компьютер сразу интегрированную среду , в которой будет и текстовый редактор кода, и средства преобразования этого кода в машинный, а также средства

автоматической проверки правильности программы, подсказки при редактировании кода, средства автозаполнения и многое другое.
Поэтому хорошим выбором для нас будет специальная интегрированная среда для разработки программ(IDE, Integrated Development Environment) Microsoft Visual Studio, например, её бесплатный вариант Community([3]). Последнее время популярность набирает интегрированная среда с похожим названием Microsft Visual Studio Code, но её чаще применяют для Web-разработок.
Microsoft Visual Studio позволяет разрабатывать как Web-приложения, так и традиционные десктопные приложения для настольных компьютеров. Microsoft Visual Studio имеет уже более чем двадцатилетнюю историю своего использования миллионами программистов, активно разрабатывается и обновляется по настоящее время. Почти ежемесячно выходят обновления для последней версии Visual Studio 2019, и только что, в июне 2021 года, вышла в свет предварительная версия Visual Studio 2022([9]). Visual Studio 2022 является первой 64-битной версией продукта, поэтому мы ожидаем от неё увеличения скорости загрузки и обработки кода, и , конечно, новых возможностей, облегчающих труд программиста и ускоряющих его работу.

2.3 Сравнительная характеристика языков программирования

Намеченный нами выбор языка C# в качестве основного языка для осуществления нашего проекта может быть обоснован его сравнением с другими языками. C# - это объектно-ориентированный язык с богатой библиотекой типов, содержащихся в .NET Framework, в том числе её последней версии .NET Core, позволяющих выполнить любую необходимую задачу в области программирования. Среди других объектно-ориентированных языков сопоставимое по популярности и возможностям место занимает схожий с C# язык Java, тоже имеющий чрезвычайно богатые библиотеки, но принадлежащий другому набору программных продуктов и соответствующих им средств разработки, не фирмы Microsoft. Но среди языков, которые созданы и используются Microsoft, есть и другие языки, кроме C#.
Существует, кроме объектно-ориентированных , ещё одно обширное семейство

языков так называемого функционального программирования, среди продуктов Microsoft таким языком является F#(F Sharp). С этим языком также можно работать в Visual Studio, и он также может использовать все средства библиотеки .NET. Языки, поддерживающие парадигму функционального программирования, приобретают всё большую популярность , поскольку хорошо поддерживают многопоточное параллельное программирование, с наиболее эффективным использованием на многопроцессорных системах и многоядерных процессорах. Кроме C# и F# языком, использующим .NET и поддерживаемым средой разработки Visual Studio, является Visual Basic. Разновидности этого языка используются внутри знаменитых прикладных программ Microsoft, имеющих широкое распространение, таких как Word, Excel, Access. Visual Basic используется как внутренний язык в этих приложениях, позволяющий расширять функциональность этих приложений, передавать данные между ними. Кроме того, Visual Basic используется как язык для написания командных файлов, которые могут запускаться из командных оболочек, таких как CMD.exe или PowerShell. Visual Basic как C# и F# может пользоваться всем богатством возможностей, которые предоставляют библиотеки .NET, каждый из этих языков использует классы из этих библиотек согласно правилам, которые определяются синтаксисом этого языка.

Предпочтение, которое мы отдадим C# перед этими двумя языками определяются его уникальными возможностями: на нём написан исходный код всех библиотек NET, из всех трёх он является наиболее популярным, базовым и основным. Изучив его, научившись использовать C# в наших проектах мы сможем вполне овладеть всей структурой программного обеспечения Microsoft, изучая исходный код библиотек. Впервые в полном объёме он выложен на GitHub, причём есть возможность его компиляции, описанная в официальной документации так, что результатом будут dll, которые можно использовать вместо тех , что идут в официальных дистрибутивах. То есть , владея языком C#, мы в самой полной мере владеем всеми возможностями .NET, вплоть до тех, что проистекают из возможностей, связанных со знанием кода исходных библиотек и

возможностью его изменения.

Кроме описанного свойства фундаментального значения для .NET, C# обладает набором возможностей, которые пришли к нему из других языков, он постоянно развивается и обогащается новыми инструментами. Например, среди многих появившихся в ходе его развития возможностей, в своём составе он имеет встроенные в него средства работы с данными, LINQ, что придаёт ему достоинства функционального языка программирования.

Все свойства языка C# прекрасно служат программисту, когда он используется в качестве языка backend, то есть для написания кода, обслуживающего тыловую часть процессов, то есть ту часть программ, которые выполняют процедуры, невидимые для пользователя, рабочие вычисления, необходимые, но не отражающиеся на экране непосредственно. Но язык C# пригоден и для написания frontend, то есть программных частей, прямо взаимодействующих с пользователем, он работает в составе программных графических оболочек Windows Forms и Windows Presentation Foundation, служащих для создания десктопных приложений. Более того, язык C# теперь используется в Интернет – приложениях не только на серверной стороне, но и на пользовательской, то есть в Интернет-браузерах, непосредственно участвуя в формировании образа экрана, управляя взаимодействием с пользователем.

В случае с десктопными приложениями при работе с библиотеками WPF(Windows Presentation Foundation) партнёром для C# выступает язык разметки XAML, который служит для описания внешнего вида экрана. Ниже приведён фрагмент кода, описывающий главное окно программы, полностью код приведён в Приложении:

```
<Window
x:Class="Формирование_междисциплинарных_тестовых_заданий.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
```

```
        xmlns:local="clr-
    namespace:Формирование_междисциплинарных_тестовых_заданий"
        mc:Ignorable="d"
        Title="Формирование междисциплинарных тестовых заданий"
    Height="801" Width="800" Loaded="Window_Loaded"
    WindowState="Maximized" >
    ….
    </Window>
```

Теги разметки в угловых скобках служат для оформления элементов, располагающихся на экране, окна, кнопок на нём, меню, использования разнообразных оформительских приёмов. Файлы, написанные на C#, служат в качестве так называемого code-behind, кода программной части. Пример взаимодействия части окна, написанной на XAML, с его программной частью, можно увидеть в приведённом фрагменте. Внутри первой части тега <Window…> находится атрибут Loaded="Window_Loaded", с помощью которого при загрузке окна вызывается метод, написанный на языке C#:

```
        private void Window_Loaded(object sender, RoutedEventArgs e)
        {
                …
     }
```

Для работы в Интернет – браузерах Microsoft разработала новую систему библиотек Blazor, тоже на основе языка C#, в которых используется язык разметки HTML, такой же, как у всех Web-страниц. Но большинство Web-страниц использует для своей работы язык Javascript, тогда как библиотеки Blazor используют для тех же целей C#. Разметка Web-страниц на языке HTML и код на C# располагаются в файлах с расширением .razor Можно заметить архитектурное сходство в построении систем библиотек WPF и Blazor и отразить его в следующей таблице(Таблица 2):

Название Framework	Сфера применения	Язык разметки	Язык кода программной части
Windows Presentation Foundation, (WPF)	Настольные(десктопные) приложения	XAML	C#
Blazor	Интернет-приложения	HTML	Javascript

Таблица 2

2.4 Выбор графической оболочки

Конечно же, наша программа должна использовать типовые элементы графической оболочки Windows, чтобы быть максимально дружественной пользователю, чтобы UX (User Experience, англ. Ощущения пользователя) были для него/её ожидаемыми, чтобы начало работы с программой было для него/её безболезненным, по возможности не нуждалось в подробных инструкциях, и все возможности программы при её использовании раскрывались естественно и постепенно. Если бы мы изначально создавали Интернет – приложение, то тогда наш выбор был бы из второй строчки таблицы 2, то есть мы могли бы использовать фреймворк Blazor, но нашим решением было создавать десктопное приложение.

Исходя из вышеуказанных требований, мы выбираем для создания интерфейса пользователя UI (User Interface) готовую графическую оболочку Windows Presentation Framework (WPF), библиотеку классов, специально предназначенных для этой цели, о которой мы уже рассказали как о построенной на двух языках C# и XAML.

Эта библиотека сделана на основе объектно – ориентированного подхода к программированию, и обеспечивает понятный , прозрачный процесс конструирования интерфейса пользователя, его изменения и доработки. WPF является самой последней разработкой Microsoft специально для десктопных приложений, уже долгие годы используется, накоплен большой опыт её использования, и она полностью перенесена для работы на самых последних выпусках .NET: .Net Core 3.1 - Net 6.0

2.5 Выбор базы данных

Для хранения вариантов тестовых заданий нам понадобится база данных. Воспользуемся рекомендацией, которую можно извлечь из учебно-тренировочной сессии Microsoft https://docs.microsoft.com/en-us/ef/core/get-started/wpf
и используем для хранения наших проектов движок базы данных SQlite. Наш проект не требует больших объёмов данных, высоких скоростей работы при одновременной обработке большого количества запросов пользователей, поэтому наш выбор вполне уместен, учитывая, что мы можем легко воспользоваться приведённым учебным примером.

Для работы с базой данных , подключения к ней , чтения из неё, занесения туда новых записей, то есть новых тестовых заданий, придерживаясь объектно – ориентированного подхода, изберём ERM Entity Framework. ERM(Entity Relational Management) – это библиотека классов, служащая для перевода информации, содержащейся в виде записей в реляционной базе данных в специально упакованном для хранения виде, в вид классов в памяти компьютера, удобном для использования, отображения на экране.

Чтобы контролировать появление записей в базе данных, создание в ней таблиц в процессе написания программы, её отладки, очень удобно иметь дополнительное средство просмотра баз данных SQLite. Выберем браузер баз данных DB Browser for SQLite, который можно загрузить, перейдя по адресу https://sqlitebrowser.org/. Он снабжён удобным типовым графическим оконным интерфейсом со стандартной системой меню. (См. рисунок 1) В этом браузере мы можем раскрыть

и посмотреть файл базы данных, увидеть, какая у него сформировалась структура,

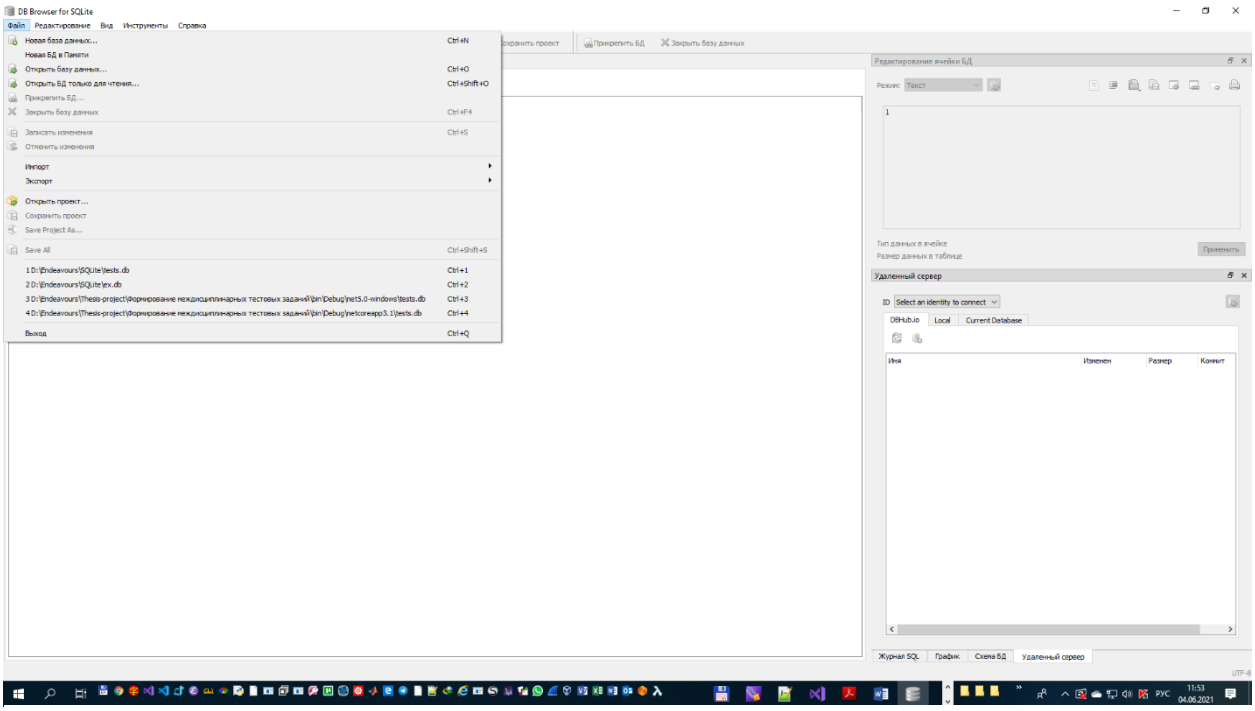

Рисунок 1

какая в него будет заносится информация.

Очень удобным средством просмотра DB-файлов и контроля за их содержимым является средство командной строки sqlite3.exe, которое можно взять по адресу https://sqlite.org/download.html Оно располагается там внутри архива sqlite-tools-win32-x86-3350500.zip Архив следует загрузить, извлечь sqlite.exe и расположить в любом удобном для вызова месте, например, %SystemRoot%\System32 Команды для работы с командной строкой описаны по адресу https://sqlite.org/cli.html

Но описанные средства, как графическое, так и текстовая командная строка служат лишь для контроля и проверки базы данных. Вся работа с ней, создание базы данных и изменение информации в ней будет вестись через создаваемую нами программу.

2.5 Выбор системы управления версиями

Выбор системы управления версиями практически предопределён выбором среды программирования Microsoft Visual Studio 2019, потому что одной из её замечательных особенностей является тесная интеграция с системой управления версиями Git. От выпуска к выпуску поддержка Git в Visual Studio развивается и улучшается, становится всё проще и удобней. Майкрософт принадлежит

крупнейший в мире репозитарий программного обеспечения , расположенный на сайте GitHub.com, созданный и функционирующий на основе Git. Чтобы начать пользоваться этой системой, рекомендуется изучить руководство ([9]), расположенное на сайте https://git-scm.com/ и загрузить саму программу, перейдя по ссылке https://git-scm.com/download/win. После этого мы сможем использовать Git из командной строки, что является полезным дополнением к работе с Git из среды Visual Studio. Например, чрезвычайно полезной является команда git reset –hard, введённая из командной строки, которая позволяет отменить все изменения, которые были сделаны в проекте и которые оказались нежелательны, например, программа полностью перестала работать и установить причину этого крайне затруднительно.

Чтобы работать с Git из Visual Studio и сохранять все стадии нашего проекта в депозитарии на GitHub, необходимо обзавестись там учётной записью. Проходим регистрацию, зарегистрировав пользователя с учётным именем(логином) LeonidLunev. Теперь на этом аккаунте мы сможем создать из Visual Studio репозитарий с нашим проектом и периодически обновлять его по мере разработки. Код здесь будет надёжно защищён от возможных аварий и поломок жёсткого диска, кроме того, разработка может вестись с разных компьютеров откудв угодно, в учебном заведении, на работе и дома. В любой момент на рабочий компьютер может быть скопирована самая актуальная рабочая версия проекта и разработка продолжена любым желающим присоединиться к разработке.

(См. рисунок 2)

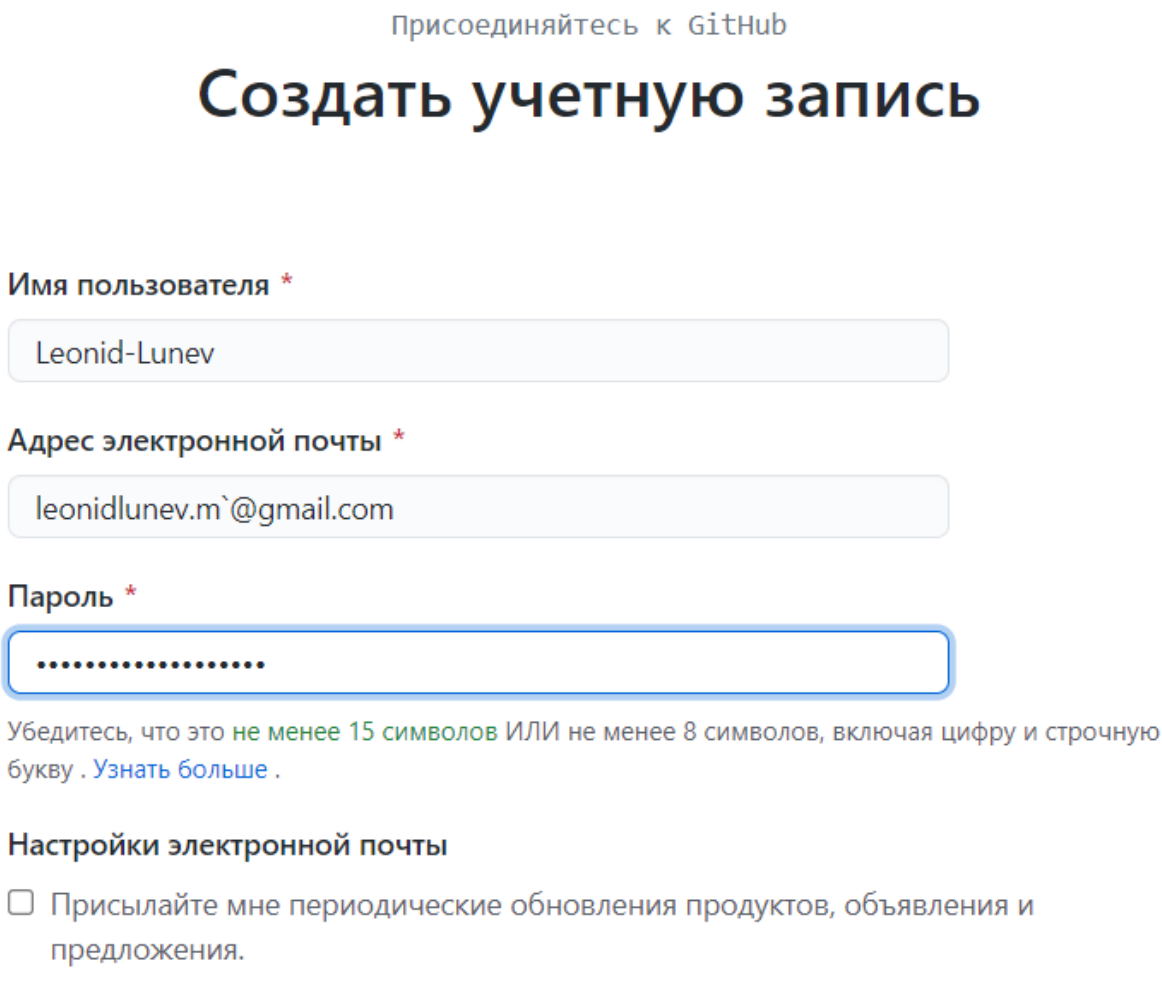
Присоединяйтесь к GitHub
Создать учетную запись
Имя пользователя *
Leonid-Lunev
Адрес электронной почты *
leonidlunev.m`@gmail.com
Пароль *
Убедитесь, что это не менее 15 символов ИЛИ не менее 8 символов, включая цифру и строчную букву . Узнать больше .
Настройки электронной почты
Присылайте мне периодические обновления продуктов, объявления и предложения.
подтвердите ваш аккаунт

Рисунок 2

ГЛАВА 3
Создание программного обеспечения.

3.1 Технические требования к компьютеру

Подготавливаем компьютер с техническими характеристиками, пригодными для установки инструментальной среды разработки Visual Studio 2019. Рекомендуется 4-х ядерный процессор, тактовая частота должна быть не меньше 1.8 гигагерц, не менее 2 гигабайт оперативной памяти, причём рекомендуется 8 гигабайт для более быстрой работы и дисковой памяти 20 Гб, хотя дисковой памяти чем больше , тем лучше, при полной загрузке указывается потребность до 210 гигабайт: по характеристикам подойдёт любой современный системный блок.

3.2 Подготовка инструментального программного обеспечения

3.2.1 Подготовка проекта в Visual Studio

Производим загрузку Visual Studio 2019. Загрузку можно произвести с адреса https://visualstudio.microsoft.com/ru/downloads/, выбрав кнопку «Бесплатная загрузка» и установив полученный файл как обычное приложение Windows.

После того, как Visual Studio будет установлена, запускаем её и начинаем работу. Выбираем вариант работы «Создание нового проекта»

Начинаем создание проекта через выбор шаблона проекта. Шаблон проекта выбираем, исходя из выбранного языка C#, целевой операционной системы, под которой будет работать наша программа, то есть Windows, и типа создаваемого приложения, в нашем случае при создании проекта мы выбираем Рабочий стол, что и означает задуманный нами выбор десктопного приложения. Из списка шаблонов проектов, удовлетворяющих выбранным нами критериям, выбираем шаблон Проект для создания приложения WPF .NET Core. Проекту даём название, совпадающее с темой диплома: «Формирование междисциплинарных тестовых заданий». Проект является частью более широкого понятия, используемого в Visual Studio, частью Решения(Solution),

которое тоже должно носить имя , и которое мы назовём «Дипломный проект Лунева».

На диске образуется дерево папок, в котором будут располагаться как файлы с кодом, которые будут составлять создаваемую нами программу, так и служебные, вспомогательные файлы, которые описывают наш проект и помогают системе работать над ним. В этот проект через Проводник Windows удобно положить текст Дипломного проекта , презентации PowerPoint, другие , необходимые при подготовке диплома файлы, чтобы работать над ними параллельно с созданиям программы. Внешние программы, Word, PowerPoint

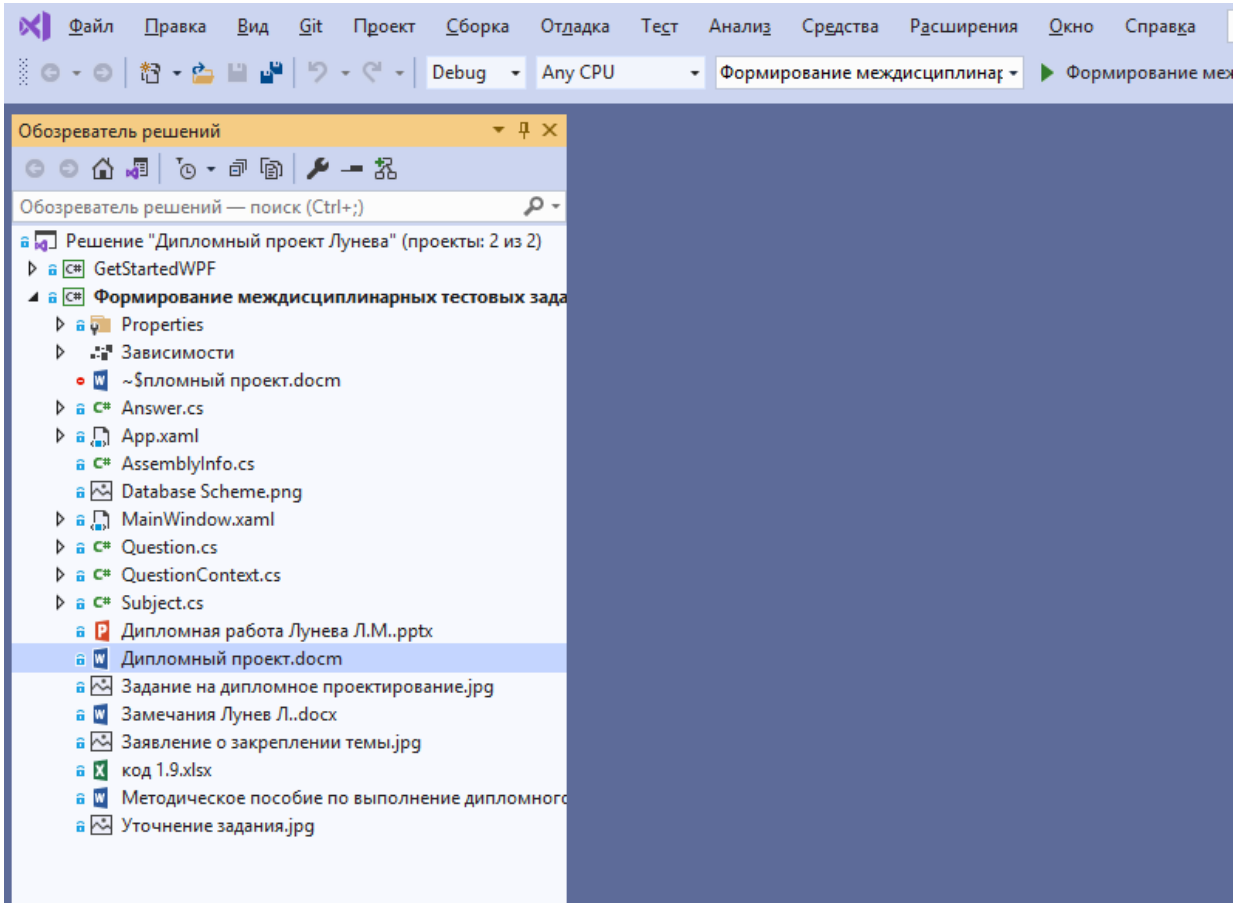

Рисунок 3

графический редактор вызываются для работы над этими файлами так же, как из Проводника Windows.(См. рисунок 3)

3.2.2 Использование системы управления версиями Git

После того, как Решение и Проект созданы, нам предстоит настроить использование Git. Для этого мы должны зайти на Github под нашей учётной

записью LeonidLunev так, что путь к нашему проекту будет выглядеть в виде следующей строки:

https://github.com/LeonidLunev/Thesis-project

Теперь по этому адресу мы будем сохранять все шаги разработки нашего решения так, чтобы надёжно сохранять и документировать все шаги создания программного продукта, при необходимости корректируя ошибки и возвращаясь к предыдущим стадиям, если что-то пойдёт не так.

Работу с системой управления версиями Git будем вести непосредственно из программной среды Visual Studio 2019, для этого там созданы богатые возможности

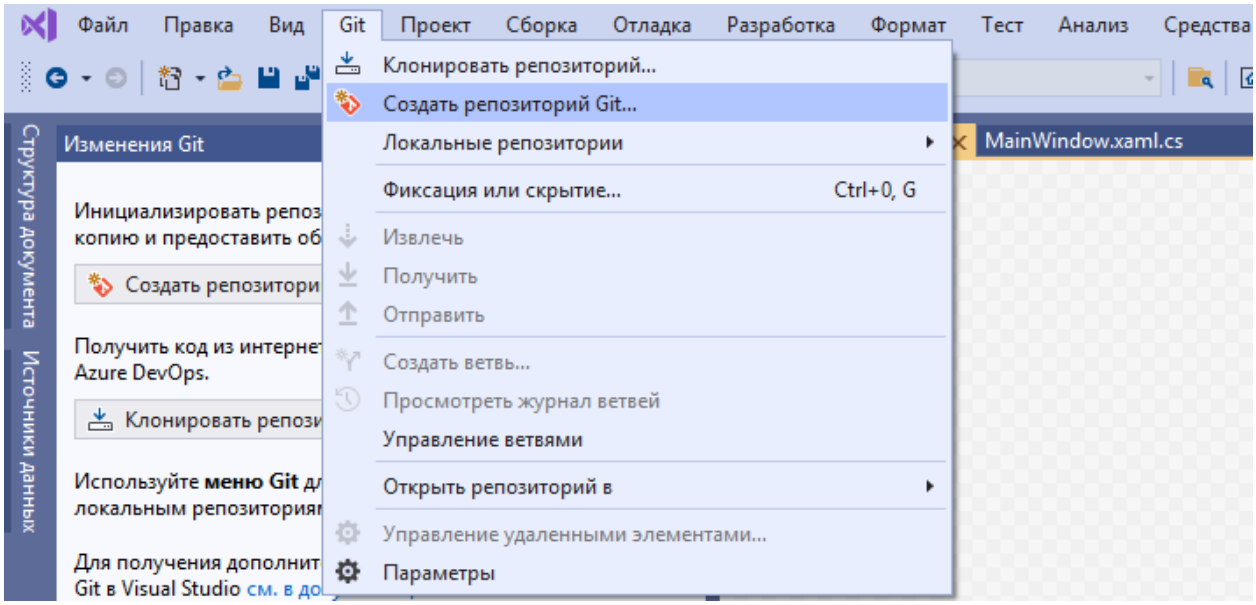

Рисунок 4

В Главном меню Visual Studio 2019 есть пункт Git, в котором есть все необходимые опции для работы с локальным репозиторием, который будет располагается на диске нашего компьютера, и с удалёнными ветвями, которые расположатся в удалённом репозитарии созданного нами аккаунта на GitHub.(См. рисунок 4)

Выбираем пункт меню «Создать репозиторий Git…», возникает окно создания репозитория (см. рисунок 5)

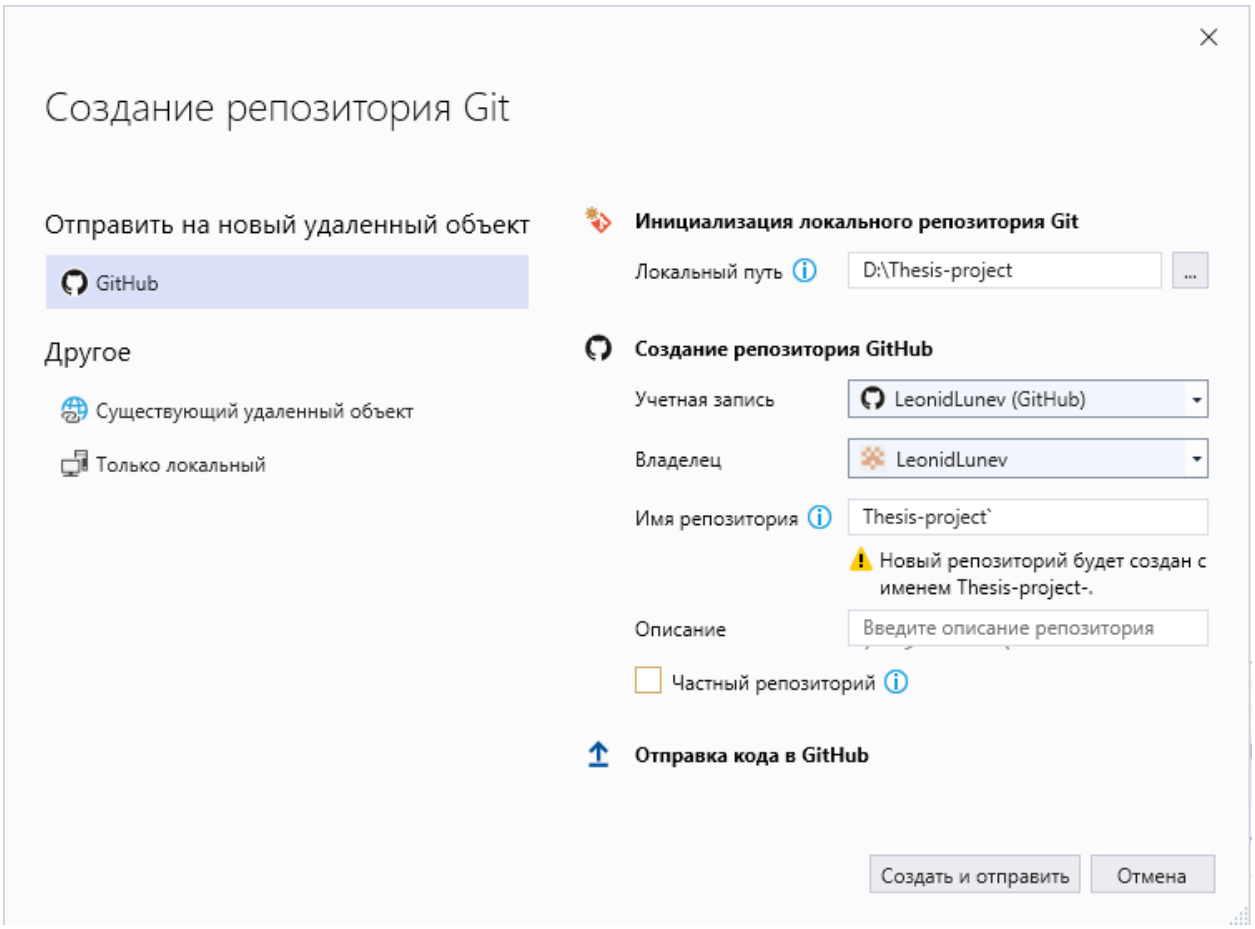

Рисунок 5

Придумываем название для нашего репозитория на GitHub, называем его Thesis-project, (“дипломный-проект”, англ.)

Нажав кнопку «Создать и отправить», создаём на GitHub под именем Thesis-project копию нашего локального проекта.

Для дальнейшего использования Git и GitHub очень удобной является работа с Git из специальной вкладки, располагающейся в окне Обозревателя решений. Она вызывается через пункт меню «Фиксация или скрытие …» (см. рисунок 6).

Из этой вкладки (см. рисунок 7) очень удобно производить сохранение проделанной работы после достаточно существенных изменений : делаем краткое описание сделанных изменений, нажимаем кнопку «Зафиксировать всё» и щёлкаем маленькую кнопочку со стрелкой вверх рядом с тремя горизонтальными точками, после чего происходит отправка рабочей ветви в удалённый репозиторий.

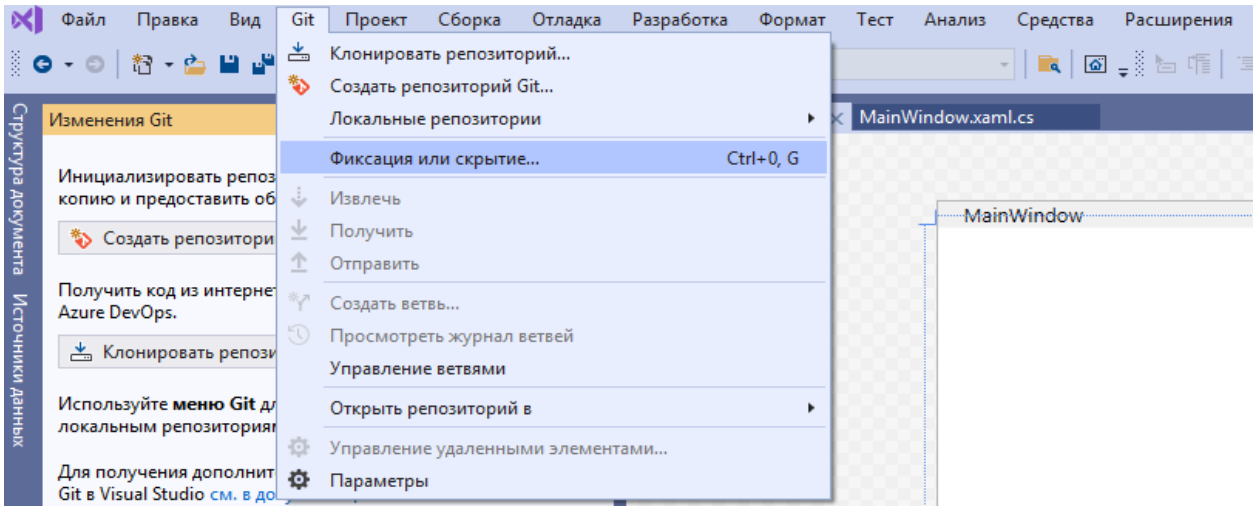

Рисунок 6

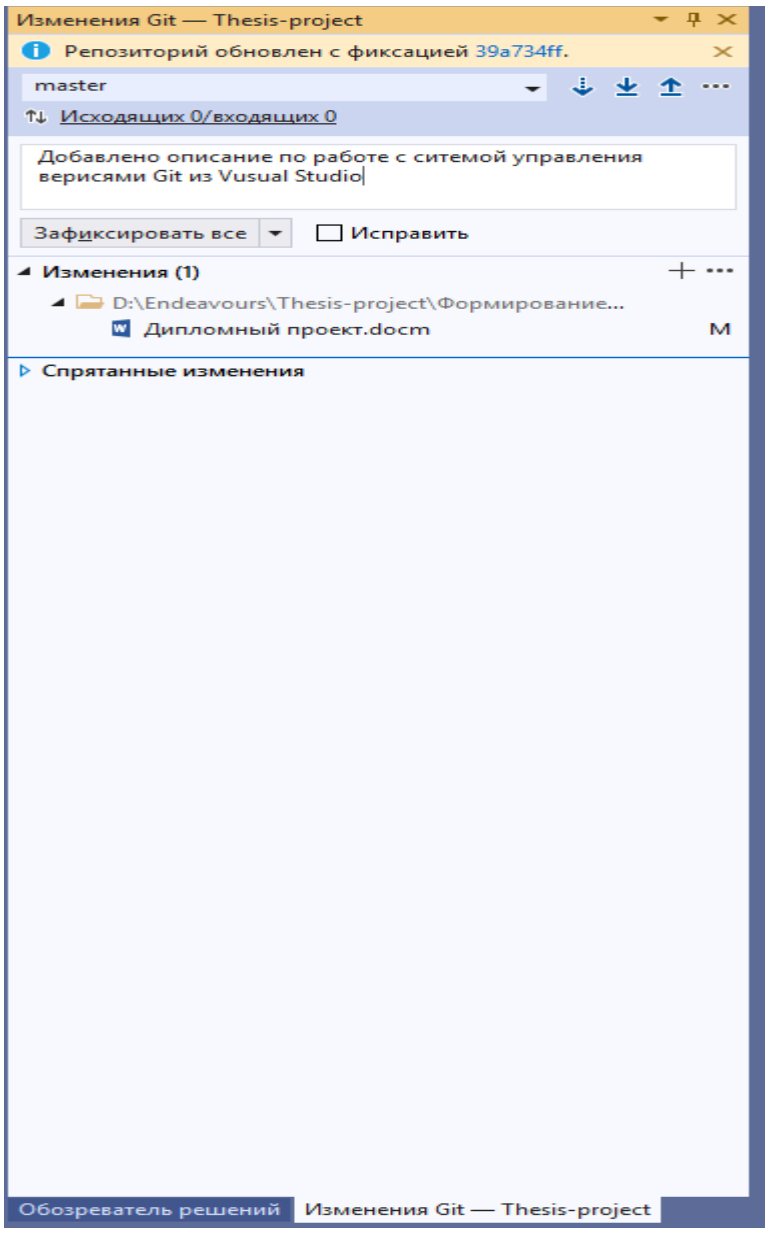

Рисунок 7

3.3 Создание пользовательского интерфейса

Частично пользовательский интерфейс уже создан для нас средой разработки Visual Studio, достаточно через меню Отладка или просто нажатием комбинации клавиш Ctrl+F5 запустить выполнение уже готового кода, как мы увидим основу пользовательского интерфейса – главное окно программы. Чтобы убедится, что мы можем работать с кодом, и программа отзывается на наши изменения, сделаем редакцию оконного кода, изменив название окна в .xaml файл проекта MainWindow.xaml, Title= “Формирование междисциплинарных тестовых заданий”. Запустив вновь на выполнение наш код, убедимся, что кастомизация окна произошла, его заголовок сменился на

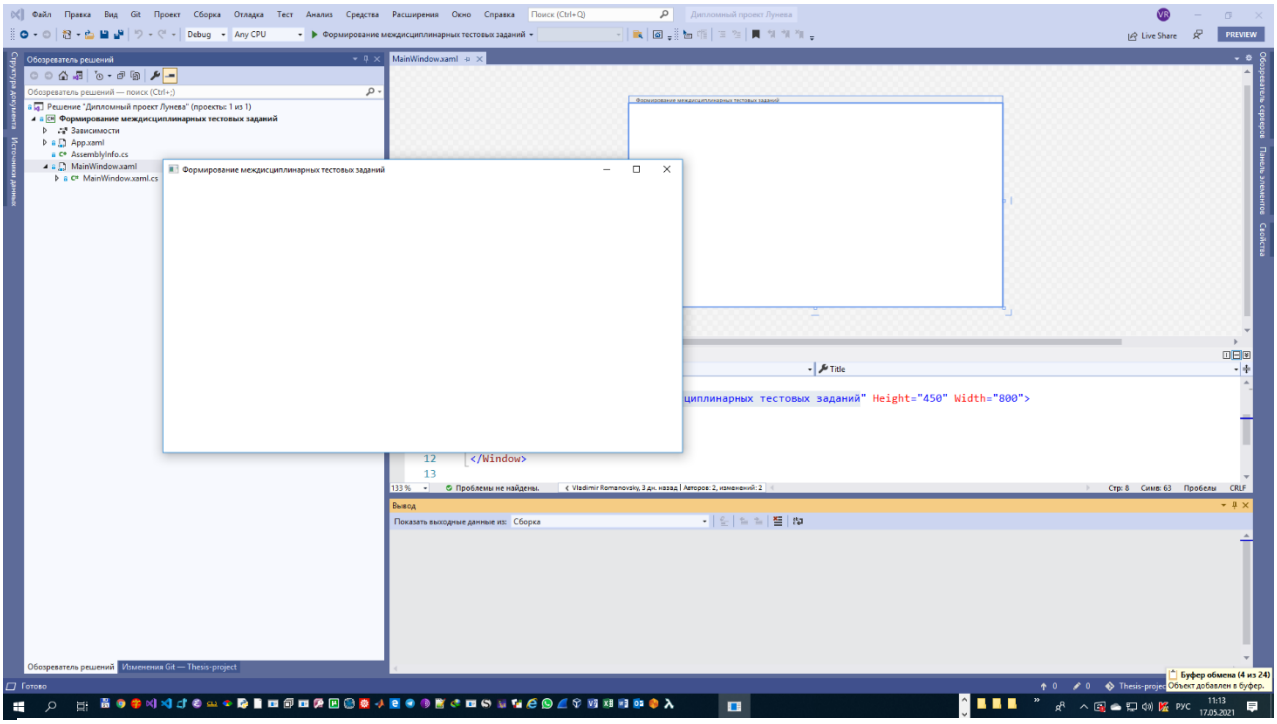

Рисунок 8

желаемый.(см. рисунок 8)

Первый шаг в создании интерфейса сделан, создана его основа – главное окно, в которое будут помещаться элементы управления. Богатая коллекция

элементов управления располагается на специальной палетке, окне, которое удобно вызывать нажатием комбинации клавиш Ctrl+Shift+X. (См. рисунок 9). Оттуда нужные элементы управления будем перетаскивать мышью, редактируя, при необходимости, код разметки XAML.

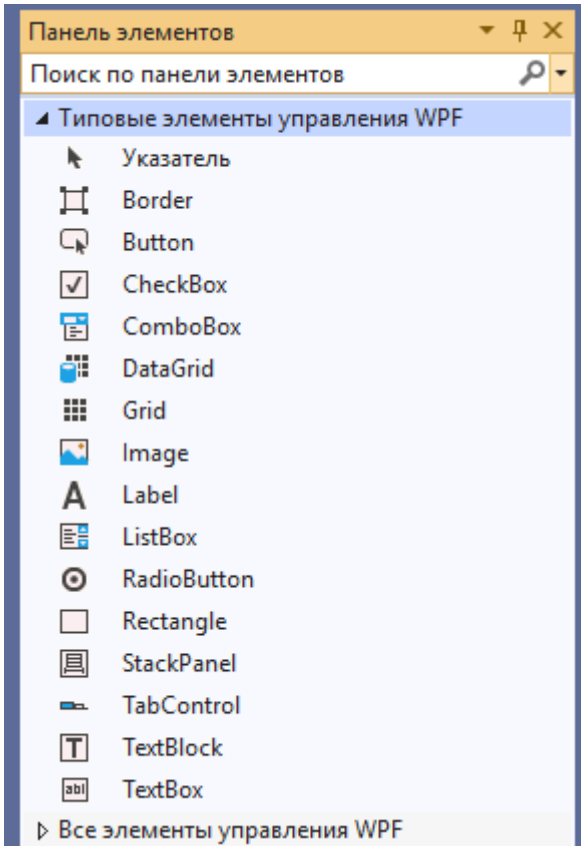

Рисунок 9

А теперь перейдём к тому материалу, который будет наполнять наш интерфейс, откуда будут поступать и куда отправляться данные: к сущностям, программным классам.

3.4 Создание сущностей для работы с базой данных

3.4.1 Определение состава сущностей

Анализируя предметную область, убеждаемся: в основе тестов лежат три сущности:

а) Предмет, преподаваемая дисциплина, по которой идёт тестирование

б) Вопросы, которые преподаватель составил для проверки знаний по своей дисциплине

б) Ответы на эти вопросы, среди которых тестируемый должен определить правильные и неправильные.

Проверка знаний ученика будет заключаться в определении того, сколько верных ответов он привёл на заданные ему вопросы.

Для каждой из преподаваемых дисциплин должен быть свой набор тестовых вопросов, причём желательно несколько вариантов. Между сущностями есть очевидная связь: с каждым предметом должно быть связано несколько вопросов, а с каждым вопросом несколько возможных ответов. Рисунок 10 показывает, как будут выглядеть структуры записей в таблицах базы данных, которые будут созданы на основе классов сущностей в памяти программы

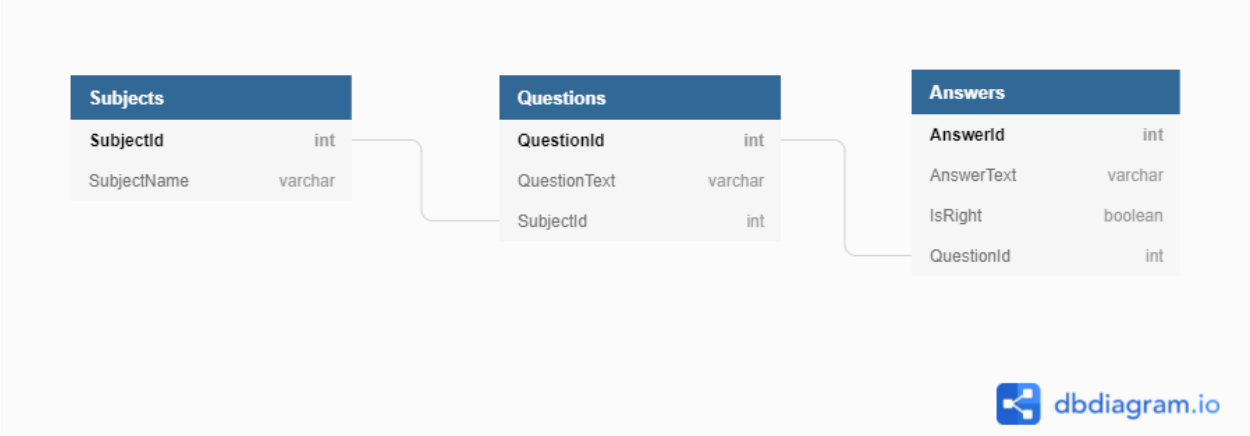

Рисунок 10

Полезное средство для рисования структур баз данных можно найти на ресурсе([4]).

Чтобы база данных была создана автоматически вместе со всеми необходимыми таблицами на основе классов сущностей, необходимо загрузить

нужные библиотеки для работы с этими сущностями из онлайнового репозитория библиотек Microsoft NuGet. Для работы с Nuget в VisualStudio есть специальные встроенные средства. Например, можно воспользоваться Средством управления пакетами NuGet из контекстного меню Проводника проектов(Solution Explorer) или пункта меню Инструменты(Tools). С помощью этого средства находим и устанавливаем для нашего решения пакеты Microsoft.EntityFrameworkCore.Sqlite и Microsoft.EntityFrameworkCore.Proxies. Этого будет достаточно, чтобы по написанному коду у нас возникли таблицы базы данных , в которых у нас будут хранится созданные нами тесты.

3.4.2 Создание сущностей для хранения в базе данных

Итак, нам предстоит хранить в базе данных три типа сущностей и работать с ними: вопрос, ответ и изучаемый предмет, к которому относятся вопрос и ответ. Каждой из этих сущностей соответствует свой класс программы, записанный в отдельный файл с таким же названием и расширением .cs, что означает, что содержимое этих файлов написано на языке C# (C Sharp). Согласно правилам языка C#, код каждого из классов начинается служебным словом class, за которым в фигурных скобках следуют свойства и методы класса. Названия свойств и методов должны удовлетворять определённым правилам не только языка C#, но и тем соглашениям, которые устанавливаются Entity Framework, которая в соответствии с ними будет формировать таблицы базы данных.

Вот наши три класса:

```
public class Subject
    {
        public int SubjectId { get; set; }
        public string SubjectName { get; set; }

        public virtual ICollection<Question>
            Questions
        { get; private set; } =
            new ObservableCollection<Question>();
    }

public class Question
    {
        public int QuestionId { get; set; }
        public string QuestionText { get; set; }


        public virtual Subject Subject { get; set; }

        public virtual ICollection<Answer>
        Answers
        { get; private set; } =
        new ObservableCollection<Answer>();
    }
public class Answer
    {

        public int AnswerId { get; set; }
        public string AnswerText { get; set; }
```

```
        public bool IsRight { get; set; }

        public virtual Question Question { get; set; }
    }
```

Первые два класса Subject(Предмет), Question(Вопрос) содержат почти одинаковый код, а именно, в каждом есть место (свойство, property) для хранения текста, названия преподаваемого предмета и текста вопроса по нему соответственно, и уникальные идентификационные номера этих текстов, удобные для ссылок. Кроме этого, в этих классах есть важные, так называемые навигационные свойства, по которым устанавливаются соответствия между заведёнными наборами данных: ведь для каждого предмета у нас должен быть не один вопрос, а целый набор вопросов, образующих тест по данному предмету, и для каждого вопроса должен быть набор ответов, из которых учащийся должен выбрать правильный.

Для класса Answer(Ответ) нужно особое свойство, в котором должна содержаться информация, правильный ли это ответ или нет, задаваемая преподавателем. Назовём это свойство IsRight(Ответ правильный) со значениями Yes/No (Да/Нет),
конечно, логического (булевского) типа bool.

Для того, чтобы созданные нами классы сущностей стали сохранятся в базе данных, необходим ещё один управляющий класс, который является

потомком специального класса DbContext, текст которого приводится ниже:

```
class QuestionContext : DbContext
    {
        public DbSet<Answer> Answers { get; set; }
        public DbSet<Question> Questions { get; set; }
        public DbSet<Subject> Subjects { get; set; }

        protected override void OnConfiguring(
            DbContextOptionsBuilder optionsBuilder)
    {
            optionsBuilder.UseSqlite(
                "Data Source=tests.db");
            optionsBuilder.UseLazyLoadingProxies();
            base.OnConfiguring(optionsBuilder);
     }
    }
```

В этом управляющем классе задаётся имя базы данных, в которой будет хранится вся информация, относящаяся к тестам, в нашем случае имя базы данных tests.db . Очень важными являются свойства с типами DbSet <Subject>,DbSet<Question>,DbSet<Answers>, их имена определяют названия таблиц в базе данных Subjects, Questions, Answers, в которых будут храниться наборы названий изучаемых дисциплин(«Предметы»), вопросы по эти дисциплинам(«Вопросы»), ответы на эти вопросы(«Ответы»).

С помощью указанных классов мы получили единицы хранения для нужной нам информации как на диске, так и в памяти для оперативной работы, но теперь перед нами стоит задача организовать саму эту работу, то есть дополнить наше окно ввода, главное окно программы средствами, элементами управления, с помощью которых информация будет заводится лицами, участвующими в учебном процессе : студентом, преподавателем, администратором системы.

Итак, для каждого действующей роли в сценариях использования программы следует предусмотреть своё окно ввода данных и свой набор функций. Роли у нас три, следовательно, необходимы три окна. Их удобно совместить на одном экране в виде вкладок элемента управления TabControl. Функции, которые нам необходимы для работы на каждой из вкладок, будут вызываться из элемента управления Ribbon (Лента), самого современного и универсального, разработанного Microsoft.

Чтобы совместить окно ввода данных и управляющие функции на ленте, ленту Ribbon помещаем в первую строку элемента Grid, а во вторую строку элемент управления TabControl, и два базовых, основных элемента интерфейса у нас готовы.

3.4 Обеспечение безопасной работы с информацией

Для работы с нашей системой могут быть выделены три роли:

1. Роль администратора, который назначает пользователей системы.
2. Роль преподавателя, который формирует список вопросов по своему предмету и готовит наборы ответов для них.
3. Роль студента, который отвечает на эти вопросы.

При начале работы программы возможен вход только с ролью администратора

Для того, чтобы программа обслуживала независимо эти две роли, чтобы при входе в систему у каждого пользователя были свои права, нужна ещё третья роль администратора, то есть человека, который имеет право присвоения логинов и паролей.

При первоначальном входе в систему зайти в неё может только человек с правами администратора, с логином admin и паролём admin. После входа в неё администратор может поменять свой пароль и логин, а также завести список преподавателей с логинами и паролям. Затем каждый преподаватель (или администратор от его имени) может завести список студентов, для которых предназначаются тесты. Затем каждый студент, получивший логин и пароль,

может зайти в систему под ним, и отвечать на вопросы тестов. При наличии такой многоступенчатой системы входа можно настроить любой желаемый режим тестирования, например, защитить тесты от нежелательных изменений со стороны учащихся.

Для ввода паролей через графическую оболочку предусмотрены специальные элементы управления PasswordBox(см. рисунок 11):

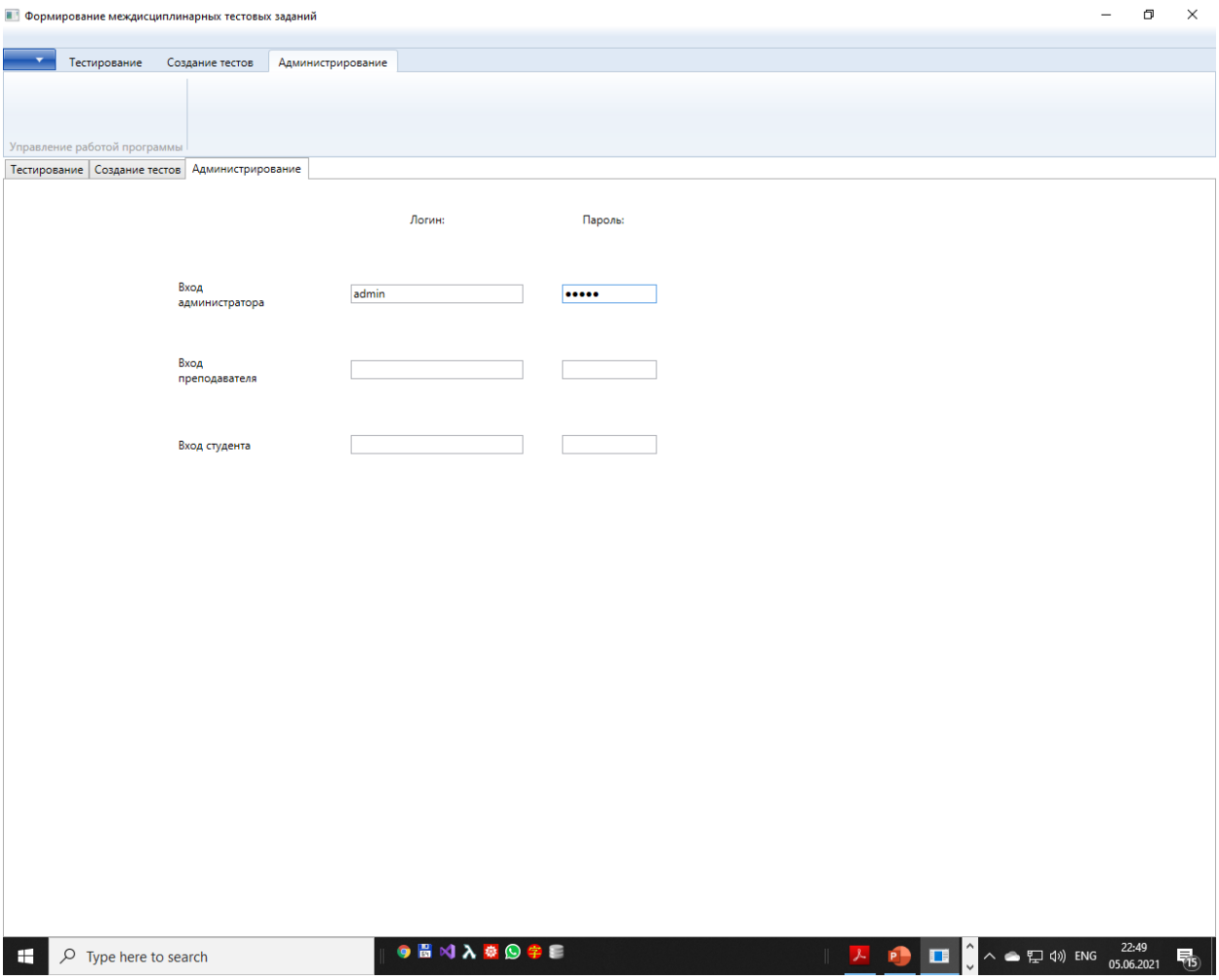

Рисунок 11

ГЛАВА 4
Охрана труда

Вопросы сбережения здоровья и жизни людей в процессе трудовой деятельности занимают исключительно важное место в наше время. Соблюдение правил техники безопасности согласно производственным инструкциям, правил поведения в условиях эпидемиологической опасности

является строго обязательным для всех работников всех производств, между тем это знание не обеспечивается автоматически, а требует изучения работающими многочисленных правил, распоряжений, инструкций, касающихся безопасности труда.

Для многих профессий регулярная демонстрация знаний положений и правил инструкций по технике безопасности является обязательным условиям допуска к исполнению своих служебных обязанностей. Поэтому применение разрабатываемой нами автоматизированной информационной системы «Формирование междисциплинарных тестовых заданий» к области контроля знаний правил техники безопасности для различных типов производств может принести значительный эффект в виде снижения травматизма и заболеваемости. Достаточно традиционные вопросы и ответы из существующих инструкций загрузить в нашу базу данных, как мы получим готовый тренажёр по охране труда.

Немаловажное отношение к охране труда имеет принцип построения программного обеспечения, которое мы использовали в процессе выполнения нашего дипломного проекта, а именно, итеративный, циклический принцип с использованием системы управления версиями Git. Использование этой методологии позволяет избежать стрессов, больших психических нагрузок, неизбежных спутников работы программиста при использовании традиционных методологий, когда довлеет требование соблюдения жёстких сроков на каждом этапе разработки, и когда нет возможности вернуться назад в процессе разработки, вернуть состояние проекта на несколько шагов назад, чтобы найти допущенную ошибку или изменить направление разработки. Система управления версиями Git и хранение проекта на GitHub позволяет полностью устранить угрозу потери наработанного кода, настоящий бич профессии программиста.

Поэтому мы считаем, что разработкой нашего дипломного проекта внесли вклад в общее дело охраны труда, бережного отношения к человеческим ресурсам – главной ценности нашей жизни.

Оптимизация работы информационной системы

Разработанная схема опроса учеников, проверки знаний, которая лежит в основе созданного в Дипломном проекте программного обеспечения может быть с успехом применена в учебном процессе, оптимизируя его, уменьшая затраты труда и времени преподавателя: при компьютерном опросе нет необходимости печатать и раздавать опросные листы, затем вводить полученные результаты в компьютер или вручную обрабатывать. Исключается субъективный фактор, непонимания, обид со стороны учащихся, поскольку может быть составлен и явно объявлен алгоритм вывода оценки по результатам опроса. При дальнейшем развитии предложенного программного обеспечения возможно создание мобильного варианта, для использования на смартфонах и планшетах. Из нежелательной помехи для учителя смартфон в руках студента может превратиться в эффективное средство обучения. Как отмечает создатель тестовой программы «Универсальный тест» *Квашнин М. Г. «*Например, чтобы учащиеся не играли в свои любимые игрушки на смартфонах во время уроков, можно проводить компьютерное тестирование на проверку их знаний по различным предметам через Internet. Действительно, когда надо отвечать на вопросы на оценку, тут уже не до игр.» http://timk.ru/informatsionnaya-tsivilizatsiya-v-razrabotke/25-programming-love.html.

Кроме прямой задачи организации учебного процесса вокруг проверки знаний, наша информационная система может оптимизировать и непосредственно сам учебный процесс, связывая задачи проверки и получения знаний, то есть вовлекая учеников в процесс составления тестов. Пытаясь сформулировать интересный вопрос, учащийся углубляется в изучаемый материал, запоминает терминологию, а в поисках ответа на него вникает в причинно-следственные связи, составляющие суть изучаемого предмета. Ученикам для составления тестов можно давать логины/пароли преподавателей, что может служить мерой поощрения, знаком отличия, разновидностью хорошей оценки.

Немаловажным фактором оптимизации учебного процесса с помощью нашей программной системы является то, что её код выложен в открытом доступе на

GitHub, и может , и должен находится в непрерывной разработке, служа сам по себе отличным учебным пособием для будущих поколений студентов. Таким образом, с одной стороны, разработанное в дипломном проекте программное обеспечение призвано облегчать, упрощать, оптимизировать процесс обучения по всем дисциплинам в колледже или ином учебном или производственном учреждении, а с другой стороны, является само по себе учебным материалом при освоении основных учебных предметов по специальности 09.02.07 «Информационные технологии и программирование». В нашей информационной системе мы осуществили положительную обратную связь: дальнейшая разработка созданного нами программного обеспечения ведёт к улучшению, оптимизации учебного процесса , а повышение качества обучения будет привлекать интерес к нашему программному обеспечению и предлагаемым нами способам его разработки и дальнейшему их совершенствованию.

ЗАКЛЮЧЕНИЕ

В рамках данного проекта была создана основа программной системы тестирования знаний и выработаны способы её дальнейшей доработки для использования в учебном процессе с целью его оптимизации. Создаваемое таким путём программное обеспечение может быть использовано в учебных заведениях и производственных учреждениях любого типа с целью тестирования студентов и сотрудников, причём с неограниченными возможностями адаптации к особенностям и потребностям этих учреждений и организаций.

Такая универсальность приложений и широкие возможности использования разработанного программного обеспечения достигнуты за счёт использования при его создании универсального языка программирования C#, библиотеки классов .NET Core, средства объектного представления баз данных Entity Framework Core, широко распространённое средство создания и управления

реляционными базами данных SQLite. Удобство использования, быстрое освоение возможностей программы достигнуто использованием мощной, богатой изобразительными средствами и элементами управления графической библиотеки WPF.

Созданный нами проект открыт для дальнейшей разработки, адаптации к различным способам тестирования, проверки знаний и приобретения знаний , поскольку находится в открытом доступе по адресу https://github.com/LeonidLunev/Thesis-project.git, откуда может быть клонирован и использован для развития по различным веткам. Мы опробовали GitHub для работы с проектом и можем рекомендовать эту среду совместной разработки и управления версиями, которая прекрасно интегрирована в Microsoft Visual Studio 2019, в основной инструмент, на котором велась разработка нашего проекта.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Джозеф Албахари, Бен Албахари C# 5.0. Справочник. Полное описание языка пер. с англ. – 5-е изд. – Москва ; Санкт-Петербург: Вильямс, 2013. – 1008 с.

2. Индекс TIOBE на июнь 2021 г. – Режим доступа: https://www.tiobe.com/tiobe-index/ . – Загл. с экрана.

3. Домашняя страница документации и учебных ресурсов Майкрософт для разработчиков и технических специалистов.. – Режим доступа: https://docs.microsoft.com/ru-ru/ 03.06.2021. – Загл. с экрана.

4. Разработка с использованием Entity Framework Core– Режим доступа: https://docs.microsoft.com/ru-ru/ef/ 03.06.2021. – Загл. с экрана.

5. Нарисуйте диаграммы сущность-взаимосвязь. Бесплатный простой инструмент для рисования диаграмм ER. Режим доступа: https://dbdiagram.io/home 03.06.2021. – Загл. с экрана.

6. Универсальный тест Тесты для школ Тесты для вузов и колледжей Тесты для проверки проф. навыков
Режим доступа: http://timk.ru/ 04.06.2021. – Загл. с экрана

7. Конструктор тестов программный комплекс для проведения компьютерного тестирования персонала и учащихся. Режим доступа: https://www.keepsoft.ru/simulator/about.php 04.06.2021. – Загл. с экрана

8. Конструктор интерактивных квиз-тестов. Создавайте квизы без программистов и дизайнеров. Простой и гибкий редактор, быстрое размещение, настройка CTA и встроенная статистика. https://madtest.ru/ 04.06.2021. – Загл. с экрана

9. Блог разработчиков Майкрософт Visual Studio 2022 . – Режим доступа: https://devblogs.microsoft.com/visualstudio/visual-studio-2022/ – Загл. с экрана.

10. Вся книга Pro Git, написанная Скоттом Чаконом и Беном Штраубом и опубликованная Apress, доступна здесь . – Режим доступа: https://git-scm.com/book/ru/v2 – Загл. с экрана.

ПРИЛОЖЕНИЯ

XAML код главного окна:

```
<Window x:Class="GetStartedWPF.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:GetStartedWPF"
        mc:Ignorable="d"
        Title="MainWindow" Height="450" Width="800"
Loaded="Window_Loaded">
    <Window.Resources>
        <CollectionViewSource x:Key="categoryViewSource"/>
        <CollectionViewSource x:Key="categoryProductsViewSource"
                              Source="{Binding Products, Source={StaticResource
categoryViewSource}}"/>
    </Window.Resources>
    <Grid>
        <DataGrid x:Name="categoryDataGrid" AutoGenerateColumns="False"
                  EnableRowVirtualization="True"
                  ItemsSource="{Binding Source={StaticResource
categoryViewSource}}"
                  Margin="13,13,43,229"
RowDetailsVisibilityMode="VisibleWhenSelected">
            <DataGrid.Columns>
                <DataGridTextColumn Binding="{Binding CategoryId}"
                                    Header="Category Id" Width="SizeToHeader"
                                    IsReadOnly="True"/>
                <DataGridTextColumn Binding="{Binding Name}" Header="Name"
                                    Width="*"/>
            </DataGrid.Columns>
        </DataGrid>
```

```
        <DataGrid x:Name="productsDataGrid" AutoGenerateColumns="False"
                  EnableRowVirtualization="True"
                  ItemsSource="{Binding Source={StaticResource
categoryProductsViewSource}}"
                  Margin="13,205,43,108"
RowDetailsVisibilityMode="VisibleWhenSelected"
                  RenderTransformOrigin="0.488,0.251">
            <DataGrid.Columns>
                <DataGridTextColumn Binding="{Binding CategoryId}"
                                    Header="Category Id" Width="SizeToHeader"
                                    IsReadOnly="True"/>
                <DataGridTextColumn Binding="{Binding ProductId}"
Header="Product Id"
                                    Width="SizeToHeader" IsReadOnly="True"/>
                <DataGridTextColumn Binding="{Binding Name}" Header="Name"
Width="*"/>
            </DataGrid.Columns>
        </DataGrid>
        <Button Content="Save" HorizontalAlignment="Center"
Margin="0,240,0,0"
                Click="Button_Click" Height="20" Width="123"/>
    </Grid>
</Window>
```

Программный код главного окна:

```
using Microsoft.EntityFrameworkCore;
using System.ComponentModel;
using System.Windows;
using System.Windows.Data;

namespace GetStartedWPF
{
    /// <summary>
```

```
/// Interaction logic for MainWindow.xaml
/// </summary>
public partial class MainWindow : Window
{
    private readonly ProductContext _context =
        new ProductContext();

    private CollectionViewSource categoryViewSource;

    public MainWindow()
    {
        InitializeComponent();
        categoryViewSource =
            (CollectionViewSource)FindResource(nameof(categoryViewSource));
    }

    private void Window_Loaded(object sender, RoutedEventArgs e)
    {
        // this is for demo purposes only, to make it easier
        // to get up and running
        _context.Database.EnsureCreated();

        // load the entities into EF Core
        _context.Categories.Load();

        // bind to the source
        categoryViewSource.Source =
            _context.Categories.Local.ToObservableCollection();
    }
```

```
        private void Button_Click(object sender, RoutedEventArgs e)
        {
            // all changes are automatically tracked, including
            // deletes!
            _context.SaveChanges();

            // this forces the grid to refresh to latest values
            categoryDataGrid.Items.Refresh();
            productsDataGrid.Items.Refresh();
        }

        protected override void OnClosing(CancelEventArgs e)
        {
            // clean up database connections
            _context.Dispose();
            base.OnClosing(e);
        }
    }
}
```

Программный код с классом сущности Subject.cs:

```
using System.Collections.Generic;
using System.Collections.ObjectModel;

namespace Формирование_междисциплинарных_тестовых_заданий
{
    public class Subject
    {
        public int SubjectId { get; set; }
        public string SubjectName { get; set; }
```

```
        public virtual ICollection<Question>
            Questions
        { get; private set; } =
            new ObservableCollection<Question>();
    }
}
using System.Collections.Generic;
using System.Collections.ObjectModel;
```

Программный код файла с классом сущности Question.cs:

```
namespace Формирование_междисциплинарных_тестовых_заданий
{
    public class Question
    {
        public int QuestionId { get; set; }
        public string QuestionText { get; set; }



        public virtual Subject Subject { get; set; }

        public virtual ICollection<Answer>
        Answers
        { get; private set; } =
        new ObservableCollection<Answer>();
    }
}
```

Программный код файла с классом сущности Answer.cs:

```
using System;
```

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

namespace Формирование_междисциплинарных_тестовых_заданий
{
    public class Answer
    {

        public int AnswerId { get; set; }
        public string AnswerText { get; set; }

        public bool IsRight { get; set; }

        public virtual Question Question { get; set; }

    }
}
```

Программный код файла с классом контекста QuestionContext.cs:

```
using Microsoft.EntityFrameworkCore;
namespace Формирование_междисциплинарных_тестовых_заданий
{
    class QuestionContext : DbContext
    {
        public DbSet<Answer> Answers { get; set; }
        public DbSet<Question> Questions { get; set; }
        public DbSet<Subject> Subjects { get; set; }
```

```
        protected override void OnConfiguring(
            DbContextOptionsBuilder optionsBuilder)
    {
            optionsBuilder.UseSqlite(
                "Data Source=tests.db");
            optionsBuilder.UseLazyLoadingProxies();
            base.OnConfiguring(optionsBuilder);
        }
    }
}
```