|  |
| --- |
| **МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ** |
| **ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕХНОЛОГИЙ И УПРАВЛЕНИЯ ИМЕНИ К.Г. РАЗУМОВСКОГО (ПЕРВЫЙ КАЗАЧИЙ УНИВЕРСИТЕТ)»**  **(ФГБОУ ВО «МГУТУ ИМ. К.Г. РАЗУМОВСКОГО (ПКУ)»)** |
|  |
| **УНИВЕРСИТЕТСКИЙ КОЛЛЕДЖ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ** |

|  |
| --- |
| **КУРСОВОЙ ПРОЕКТ** |
|  |
| по междисциплинарному курсу: МДК.02.02. Технология разработки и защиты баз данных |
|  |
| на тему: Разработка генератора билетов к экзамену |
|  |
| студента группы 090203-9o-20/1  специальности 09.02.03 Программирование в компьютерных системах |
| Смирнова Константина Вадимовича |

|  |  |  |
| --- | --- | --- |
| Студент |  | К.В Смирнов |
| Руководитель курсового проекта |  | Е.А. Ларионова |
| Председатель ПЦК специальности 09.02.03 Программирование в компьютерных системах |  | А.И. Глускер |

|  |  |  |
| --- | --- | --- |
| Дата защиты «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2023 г. | | |
| Оценка:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | |
| Заведующий отделением № 1 |  | И.А. Миланова | |

Москва

2023

ЗАДАНИЕ КУРСОВОЙ РАБОТЫ

Разработка генератора билетов к экзамену

**ОГЛАВЛЕНИЕ**

[ВВЕДЕНИЕ 3](#_Toc134709767)

[1 ОБЩАЯ ЧАСТЬ 4](#_Toc134709768)

[1.1 Назначения и цели создания системы 4](#_Toc134709769)

[1.2 Обзор и анализ предметной области 4](#_Toc134709770)

[1.3 Жизненный цикл базы данных 6](#_Toc134709771)

[1.4 Выбор и характеристика СУБД 7](#_Toc134709772)

[1.5 Выбор и характеристика среды разработки приложения 9](#_Toc134709773)

[2 СПЕЦИАЛЬНАЯ ЧАСТЬ 12](#_Toc134709774)

[2.1 Постановка задачи 12](#_Toc134709775)

[2.2 Архитектура информационной системы 12](#_Toc134709776)

[2.3 Логическая модель базы данных 14](#_Toc134709777)

[2.4 Нормализация таблиц 14](#_Toc134709778)

[2.5 Описание таблиц 15](#_Toc134709779)

[2.6 Другие объекты базы данных (представления, хранимые процедуры, триггеры, имена входа, пользователи) 17](#_Toc134709780)

[2.7 Разработка приложения 19](#_Toc134709781)

[2.7.1 Диаграмма вариантов использования 19](#_Toc134709782)

[2.7.2 Решение главной задачи проекта 19](#_Toc134709783)

[2.7.3 Тестирование приложения 24](#_Toc134709784)

[2.7.4 Защита информационной системы 27](#_Toc134709785)

[2.8 Инструкция пользователю 28](#_Toc134709786)

[2.8.1 Общие сведения об информационной системе 30](#_Toc134709787)

[2.8.2 Требования к техническим средствам 30](#_Toc134709788)

[2.8.3 Требования к программным средствам 30](#_Toc134709789)

[2.8.4 Настройка информационной системы 31](#_Toc134709790)

[2.8.5 Формы ввода 32](#_Toc134709791)

[2.8.6 Отчёты 35](#_Toc134709792)

[ЗАКЛЮЧЕНИЕ 36](#_Toc134709793)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 37](#_Toc134709794)

[ПРИЛОЖЕНИЕ A.SQL скрипты на создание и заполнение базы данных, на создание учётных записей пользователей, на назначение привилегий 38](#_Toc134709795)

[ПРИЛОЖЕНИЕ Б. Код программы 50](#_Toc134709796)

Введение

Когда заходит речь о Word документах, то обычному пользователю или сотруднику, который работает с документацией, приходится заполнять документы самостоятельно, тратя человеко-часы на рутину, уставая и направляя своё внимание на излишние действия, что сказывается на усталости человека. Такой подход может привести к наличию сотрудников, которых при автоматизации заполнения документов можно заменить, сэкономив, направив бюджет и силы компании на более полезные проекты или повысить зарплату других сотрудников.

Использование приложения, которое позволяет взаимодействовать через интуитивно понятный и простой интерфейс поможет избежать лишней работы, в следствие повышения скорости и удобства заполнения документации, используя шаблоны с тегами, вместо которых будут подставляться записи из базы данных.

Основная задач проекта исходя основываясь на предметной области - создание базы данных для автоматизированной генерации экзаменационных билетов. В базе данных имеется информации и билетах, вопросах в билетах, учителях, дисциплинах и специальности, от выбора которых зависят перечисленные таблицы. Так же имеется промежуточная таблица, в которой содержатся первичные ключи от курсов, семестров, протоколов и председателей пцк. Предусмотрена таблица пользователей для реализации ограничений в зависимости от роли.

Актуальность курсовой работы обусловлена тем, что при получении навыков в создание Word документов по средству языка c#, можно автоматизировать создание необходимой документации, что положительно скажется на продуктивности, скорости и качестве производимых документов. В данном курсовом проекте осуществляется комплексное решение задач, направленных на достижение этих целей.

Цель исследования: систематизировать теоретические знания создания базы данных, взаимодействия с Entity Framework, Word.

Задачи курсовой работы:

1. исследовать, изучить, проанализировать и использовать источники по работе Word и базы данных
2. Применение полученной информации на практике для получения практических навыков

Результатом выполнения курсовой работы представлены в виде c# приложения и теоретический подготовки студента. При разработке приложения были рассмотрены существующие приложения в данной предметной области.

1. Общая часть
   1. Назначение и цели создания системы

Назначение: Система предназначена для генерации экзаменационных билетов

Целью создания системы является получение опыта в работе с базой данных, реализация учебного плана, C#, WPF, Entity Framework и библиотеками Word.

1.2 Обзор и анализ предметной области

Основная задача проектируемого приложения состоит в создание экзаменационного билета на основе выбранных пользователем данных. Система должна предусматривать авторизацию пользователей и администраторов, регистрацию, формирование билетов, редактирование таблиц вопросов, пользователей и дисциплин.

Формирование билета в приложение состоит из выбора: специальности, дисциплины, семестра, количества билетов, преподавателя, председателя цикловой комиссии, курса, протокола и сложности вопроса. Соответственно возможно наличие нескольких билетов в базе данных, которые отличаются только вопросами. В соответствии с предметной областью система строится с учётом следующих особенностей:

1. Предоставить пользователю выбор составляющих для формирования билетов.
2. Формировать билеты в формате Word документ

Пользователь, у которого роль является администратор, может:

1. Просматривать таблицы пользователей, дисциплин, вопросов и билетов
2. Добавлять записи в таблицы пользователей, дисциплин, вопросов и билетов
3. Редактировать записи в таблице пользователей, дисциплин, вопросов и билетов

При открытии страницы, на которой осуществляется выбор данных для формирования билета, должны выводиться следующие характеристики:

* Специальность
* Дисциплина
* Семестр
* Количество билетов, от числа которых зависит сколько будет сформировано билетов
* Преподаватель
* Председатель цикловой комиссии
* Курс
* Протокол
* Сложность вопроса

Базовые сущности предметной области:

Пользователи. Атрибуты пользователей – номер пользователя, логин, пароль, роль. Для хранения данных о логинах и паролей администраторов используется таблица Пользователи. Разница между функционалом приложения для администратора и клиента определяется полем роль. Администраторы имеют доступ к просмотру таблиц, добавлять и изменять данные в таблицах.

Комплект билетов. Атрибуты комплекта билетов – курс, семестр, протокол, идентификатор председателя пцк.

Вопросы. Атрибуты вопросов – идентификатор вопросов, идентификатор дисциплины, вопрос, тип вопроса, сложность. Вопросы напрямую связаны с Билетами.

Билеты. Атрибуты билетов – идентификатор билетов, номер комплект, идентификатор вопроса 1, идентификатор вопроса 2, идентификатор вопроса 3, идентификатор учителя. Билеты являются главной таблицей, на основе которых формируется Word документ.

Логическая модель базы данных представлена на Рисунке 2.2.

Возможности приложения определялись исходя из рассмотрения уже существующих вариантов.

Рассмотренные приложения:

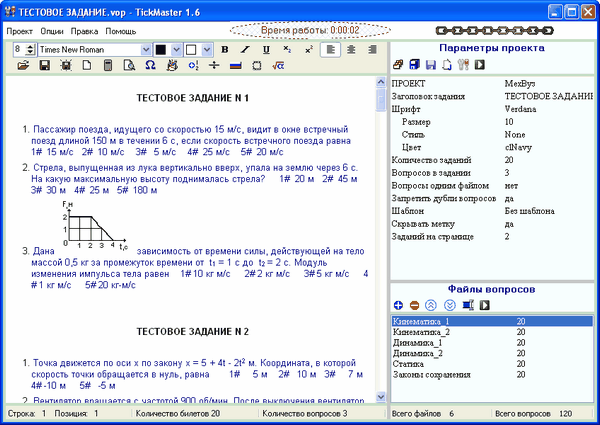


Рисунок 1.2. Приложение TickMaster

Основное назначение программы - генератор билетов - создание билетов из предварительно подготовленной базы вопросов - RTF файлов, содержащих вопросы.



Рисунок 1.2. Логотип приложения EasyTestMaker

EasyTestMaker - программа, которая автоматически создает экзаменационные билеты с различными типами вопросов, включая множественный выбор, короткие ответы и эссе. Она также позволяет генерировать ответы на вопросы, тестовые листы и ключи.

* 1. Жизненный цикл базы данных

Жизненный цикл базы данных — это процесс создания, сопровождения и удаления базы данных. Жизненный цикл включает в себя несколько этапов:

1. Анализ и проектирование базы данных: на этом этапе определяются требования к базе данных и создается концептуальная модель данных. Затем проектируется логическая модель данных и создаются схемы базы данных.
2. Разработка базы данных: на этом этапе создаются таблицы, индексы и другие объекты базы данных.
3. Тестирование и отладка: на этом этапе выполняется тестирование базы данных для выявления ошибок и багов.
4. Реализация и развертывание: база данных развертывается на реальной платформе и настраивается для работы.
5. Мониторинг и оптимизация: на этом этапе производится мониторинг базы данных для определения производительности и выявления проблем. В случае необходимости производится оптимизация базы данных.
6. Сопровождение и обслуживание: на этом этапе выполняются задачи по управлению базой данных, включая резервное копирование, восстановление данных в случае сбоев и обновление версии базы данных.
7. Удаление: на этом этапе база данных полностью удаляется из системы и выполняется очистка всех связанных с ней объектов.

Выбор и характеристика СУБД

Во время разработки информационной системы использовалась концепция клиент-серверной архитектуры, которая включает в себя наличие серверов и клиентских устройств, которые используют определённые сервисы.

Данная архитектура была использована из-за того, что серверы и клиенты имеют возможность одновременно работать на разных системах в сети, что позволяет распределять нагрузку и обеспечивать более высокую доступность и производительность приложений.

Разделение приложения на клиентскую и серверную части позволяет изменять или обновлять каждую часть независимо от другой.

Серверная часть приложения может обеспечивать более высокий уровень безопасности с помощью механизмов аутентификации, авторизации и шифрования данных.

Клиент-серверная архитектура позволяет централизованно управлять хранилищами данных, масштабировать приложения по мере необходимости, добавляя дополнительные серверы или клиенты, оптимизировать производительность приложения, разделяя задачи между клиентом и сервером. обеспечивает возможность добавления новых функций и модулей без необходимости изменения всего приложения.

Для реализации клиент-серверной архитектуры была выбрана СУБД MS SQL Server, ибо предоставляется бесплатный доступ. Также ПО MS SQL Server Management Studio для работы с MS SQL Server упрощает разработку, так как предоставляет удобный и интуитивно понятный интерфейс, а также имеет встроенный компилятор для кода на SQL языке.

Сравнение SQL Server с другими СУБД представлено на Рисунке 1.4.



Рисунок 1.4. Сравнение СУБД

1.5 Выбор и характеристика среды разработки приложения

ADO.NET и Entity Framework являются двумя разными подходами к решению задач, связанных с доступом к данным в приложениях, использующих технологии .NET.

ADO.NET - это набор библиотек для работы с данными в .NET, который включает в себя классы, используемые для подключения к базам данных, выполнения запросов и получения результатов. ADO.NET предоставляет более прямой доступ к данным, но с его использованием не всегда удобно и быстро писать код.

Entity Framework - это ORM фреймворк для .NET, который использует подход Code First (код первым). Он предоставляет набор инструментов, с помощью которых можно создавать сущности данных, отображающие таблицы в базе данных и хранить их в коде приложения. Entity Framework позволяет работать с данными в объектно-ориентированном стиле, что упрощает и ускоряет разработку приложений.

Однако существует множество факторов, влияющих на выбор между ADO.NET и Entity Framework, такие как опыт разработчиков, сложность приложения, требования к быстродействию и т.д. В каждом конкретном случае необходимо выбрать тот инструмент, который позволяет достичь наилучших результатов при оптимальных затратах времени и усилий.

Для разработки приложения был выбран Entity Framework в связи с удобством выполнения запросов в языке программирования c#, позволяя представить данные как реальные объекты, что позволяет абстрагироваться от базы данных и её таблиц, работая с данными как с объектами.

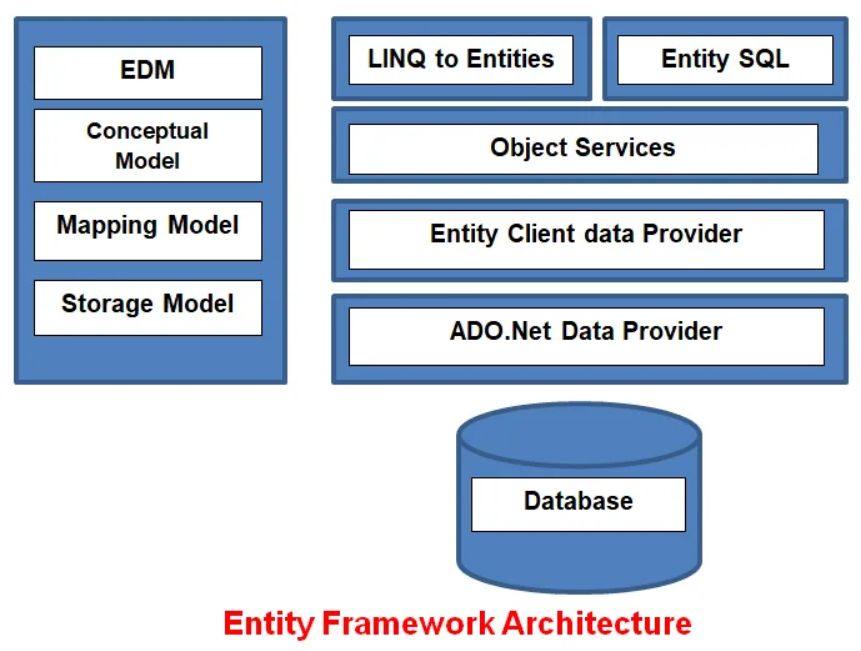


Рисунок 1.5. Схема работы Entity Framework

Entity Framework - инструмент ORM (Object-Relational Mapping), позволяющий взаимодействовать с базами данных, используя объектно-ориентированный подход.

Минусами Entity Framework являются:

1. Большой объем данных, который нужно хранить в памяти при использовании внутренних объектов Entity Framework

Плюсами Entity Framework являются:

1. Простота в использовании и повышение уровня абстракции
2. Можно работать с данными на уровне объектов вместо работы непосредственно с запросами SQL

ADO.NET позволяет взаимодействовать с базами данных с помощью языка программирования C# и языка SQL

Минусами ADO.NET являются:

1. Неудобство работы с объектами
2. Много повторяющихся блоков кода
3. Необходимость написания SQL-запросов

Плюсами ADO.NET являются:

1. Простота в использовании, для работы достаточно хорошо знать SQL язык
2. Основной нужный функционал включен в ядро .NET Framework
3. Стабильность и быстрота работы
4. Простота в использовании, для работы достаточно хорошо знать SQL язык

Исходя из вышеперечисленного, выбор был сделан в пользу Entity Framework.

В качестве среды разработки была выбрана Microsoft Visual Studio.

Visual Studio содержит все необходимые инструменты и функции, такие как редактор кода, отладчик, компилятор. Имеет мощный отладчик, который позволяет находить и исправлять ошибки в коде. Он также поддерживает множество отладочных функций, таких как точки остановок, просмотр переменных и значения, трассировка стека вызовов и др.

Интерфейс Visual Studio разработан, чтобы сделать работу программиста простой и удобной. Инструменты и функции доступны благодаря простой структуре меню и панелей инструментов.

2  СПЕЦИАЛЬНАЯ ЧАСТЬ

2.1 Постановка задачи

2.2 Архитектура информационной системы

2.3 Логическая модель базы  данных

2.4 Нормализация таблиц

2.5 Описание таблиц

2.6 Другие объекты базы данных (представления, хранимые процедуры, триггеры, имена входа, пользователи)

2.7 Разработка приложения

2.7.1 Диаграмма вариантов использования

2.7.2 Решение главной задачи проекта

2.7.3 Тестирование приложения

2.7.4 Защита информационной системы

2.8 Инструкция пользователю

2.8.1 Общие сведения об информационной системе

2.8.2 Требования к техническим средствам

2.8.3 Требования к программным средствам

2.8.4 Настройка информационной системы

2.8.5 Формы ввода

2.8.6 Отчёты

1. **Специальная часть**

**2.1 Постановка задачи**

Постановка задачи

Спроектировать средствами MS SQL SERVER базу данных и создать приложение, позволяющее ввести вопросы   по дисциплинам для проведения комплексного экзамена и сформировать по ним билеты.  Предусмотреть, чтобы

1. На  одной странице  помещается  1 билет.
2. Число билетов выбирается пользователем.
3. В одном билете были вопросы по одинаковой дисциплине.
4. Ни один вопрос не должен дублироваться.
5. Число вопросов равняется 3
6. Билеты формируются с учётом сложности вопросов.

Предусмотреть возможность добавления, удаления, редактирования записей. Билеты должны быть представлены в виде отчёта или текстового файла. Обеспечить защиту базы данных и приложения.

В дополнение к вышеперечисленным этапам курсовая работа включает следующие задачи:

1. Разработка графического интерфейса для приложения, чтобы сделать его более удобным для пользователя.
2. Использование администраторов и пользователей для создания градации по функциональным возможностям при работе программы.
3. Просмотр, добавление и редактирование записей в таблицах.
4. Обеспечить защиту базы данных и приложения.

Курсовая работа будет включает в себя сочетание навыков программирования и знаний по управлению базами данных для создания практического и полезного программного приложения.

* 1. **Архитектура информационной системы**

Архитектура информационной системы для генератора экзаменационных билетов включает разработку и внедрение системы программного обеспечения, которую можно использовать для эффективного производства экзаменационных билетов.

База данных вопросов: хранит набор всех доступных вопросов вместе с дисциплиной, типом вопроса и уровнем сложности.

Компоненты, входящие в состав архитектуры информационной системы книжного магазина, будут включать клиентское приложение C# WPF, серверную часть и базу данных MS SQL Server. Клиентское приложение обеспечит пользователю взаимодействие с системой, предоставив пользователям удобный вид приложения для ввода параметров билета, таких как специальность, дисциплина, количество вопросов и другие, а также отобразит сгенерированный экзаменационный документ в формате Word для дальнейшего использования пользователем. В приложение предусмотрена авторизация и регистрация пользователей.

Генератор экзаменационных билетов реализует алгоритм, основанный на пользовательском вводе, который будет случайным образом выбирать билеты, которые доступны после выбора пользователем.

Систем разработана с использованием комбинации языков программирования, таких как c#, и системой управления базами данных MSSQL. Пользовательский интерфейс разработан с использованием таких библиотек wpf.

Архитектура информационной системы для генератора экзаменационных билетов требует сочетания принципов проектирования программного обеспечения, знаний по управлению базами данных и понимания процесса создания билета для разработки эффективной и полезной программной системы.

**2.3 Логическая модель базы данных**

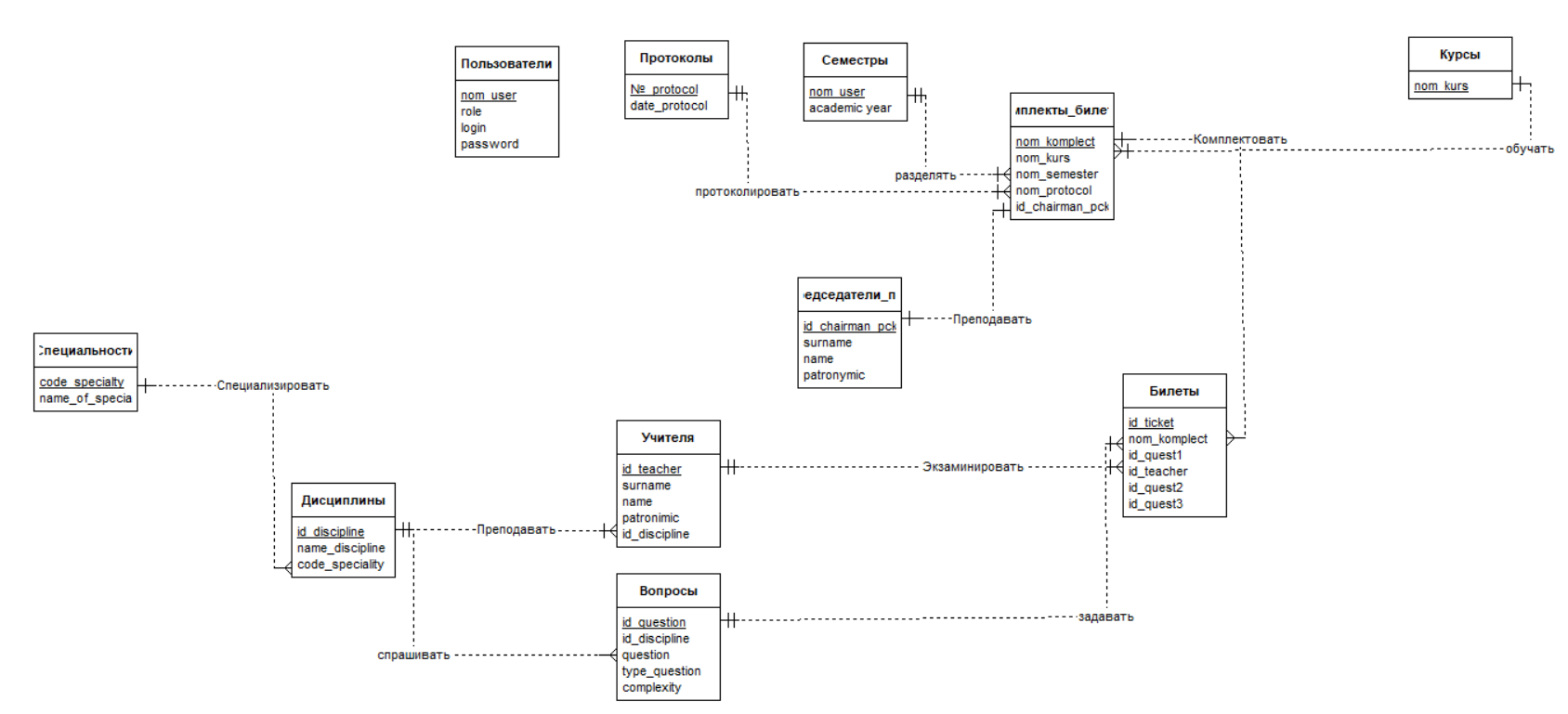


Рисунок 2.2. Логическая модель данных

* 1. **Нормализация таблиц**

Нормализация таблиц — это процесс проектирования базы данных с целью устранения повторений данных и установления связей между таблицами.

Как правило, нормализация таблиц проходит через несколько стадий, от первой нормальной формы до третьей нормальной формы. На каждой стадии устраняются определенные типы повторений данных и устанавливается определенный тип связи между таблицами.

Таблицы базы данных были приведены к третьей нормальной форме, в которой минимизировано повторение данных, устраняется аномалия данных, обеспечивается ссылочная целостность и относительно простое управление данными. Сущности в таблицах, соответствующих третьей нормальной форме, зависят только от первичного ключа и элементы таблиц являются неделимыми.

Все таблицы в базе данных приведены к третьей нормальной форме так как:

1. Каждое поле содержит только одно значение и что все поля уникальны, вся информация разделена на более мелкие и более точные единицы, что обеспечивает более эффективное и точное управление базой данных. (Первая нормальная форма)
2. Каждое поле таблиц зависит от первичного ключа. Это позволяет обеспечить более эффективное управление базой данных, так как можно избежать дублирования информации и других аномалий данных. (Вторая нормальная форма)
3. Отсутствует транзитивная зависимость. Транзитивная зависимость - неключевые столбцы зависят от значений других неключевых столбцов. Это позволяет избежать аномалий данных. (Третья нормальная форма)
   1. **Описание таблиц**

Таблица 1 – Users (Пользователи)

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Описание | Тип | Примечание |
| nom\_user | Номер пользователя | Int | FK |
| login\_ | Логин | nvarchar(50) |  |
| password\_ | Пароль | nvarchar(50) |  |
| role\_ | Роль | nvarchar(5) |  |

Таблица 2 – Kurs (Курсы)

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Описание | Тип | Примечание |
| nom\_kurs | Номер курса | Int | FK |

Таблица 3 – Semesters (Семестры)

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Описание | Тип | Примечание |
| nom\_semester | Номер семестра | Int | FK |
| academic\_year | Академический год | Date |  |

Таблица 4 – Protocols (Протоколы)

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Описание | Тип | Примечание |
| nom\_protocol | Номер протокола | Int | FK |
| date\_protocol | Дата протокола | Date |  |

Таблица 5 – Chairman\_pck (Председатель\_пцк)

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Описание | Тип | Примечание |
| id\_chairman\_pck | Идентификатор председателя ПЦК | Int | FK |
| Surname | Фамилия | nvarchar(100) |  |
| name\_ | Имя | nvarchar(100) |  |
| patronymic | Отчество | nvarchar(100) |  |

Таблица 6 – Komplect\_tickets (Комплекты\_билетов)

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Описание | Тип | Примечание |
| nom\_komplect | Идентификатор комплекта билетов | Int | FK |
| nom\_kurs | Номер курса | Int |  |
| nom\_semester | Номер семестра | Int |  |
| nom\_protocol | Номер протокола | Int |  |
| id\_chairman\_pck | Идентификатор Предсседателя\_пцк | Int |  |

1. Таблица 7 – Speciality (Специальности)

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Описание | Тип | Примечание |
| code\_speciality | Код специальности | nvarchar(100) | FK |
| name\_of\_speciality | Название специальности | nvarchar(100) |  |

1. Таблица8 – Disciplines (Дисциплины)

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Описание | Тип | Примечание |
| id\_discipline | Код специальности | int | FK |
| name\_discipline | Название специальности | nvarchar(500) |  |
| code\_speciality | Код специальности | nvarchar(100) |  |

1. Таблица 9 – Disciplines(Дисциплины)

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Описание | Тип | Примечание |
| id\_question | Идентификатор вопроса | int | FK |
| id\_discipline | Идентификатор дисциплины | Int |  |
| Question | Вопрос | nvarchar(500) |  |
| type\_question | Тип вопроса | nvarchar(14) |  |
| Complexity | Сложность вопроса | nvarchar(8) |  |

1. Таблица 10 – Teacher (Преподаватель)

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Описание | Тип | Примечание |
| id\_teacher | Идентификатор учителя | int | FK |
| Surname |  | nvarchar(100) |  |
| name\_ | Вопрос | nvarchar(100) |  |
| Patronymic | Тип вопроса | nvarchar(100) |  |
| id\_discipline | Идентификатор дисциплины | int |  |

1. Таблица 11 – Tickets (Билеты)

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Описание | Тип | Примечание |
| id\_ticket | Идентификатор билета | int | FK |
| id\_quest1 |  | int |  |
| id\_quest1 | Вопрос | int |  |
| id\_quest1 | Тип вопроса | int |  |
| nom\_komplect | Идентификатор дисциплины | int |  |
| id\_teacher | Идентификатор учителя | int |  |

* 1. Другие объекты базы данных (представления, хранимые процедуры, триггеры, имена входа, пользователи)

Другие объекты базы данных не используются.

* 1. Разработка приложения

Разработано приложение На основе предметной области. Создание происходило в несколько этапов.

На первом этапе был определён функционал приложения и была изучена информация о среде разработки Microsoft Visual Studio, языке C#, системы WPF и Entity Framework.

В функционал приложения вошли:

* Добавление записей в таблицы;
* Удаление записей в таблицах;
* Редактирование записей в таблицах;
* Просмотр информации из таблиц.

На следующем этапе был разработан относительно удобный и интуитивно понятный интерфейс, для простоты взаимодействия пользователя с приложением.

Далее была начата работа над реализацией приложения.

Для работы с базой данных были реализованы запросы на языке LINQ, которые вызываются в приложении при выборе пользователем данных.

На следующем этапе было проведено тестирование приложения и отладка.

2.7.1 Диаграмма вариантов использования

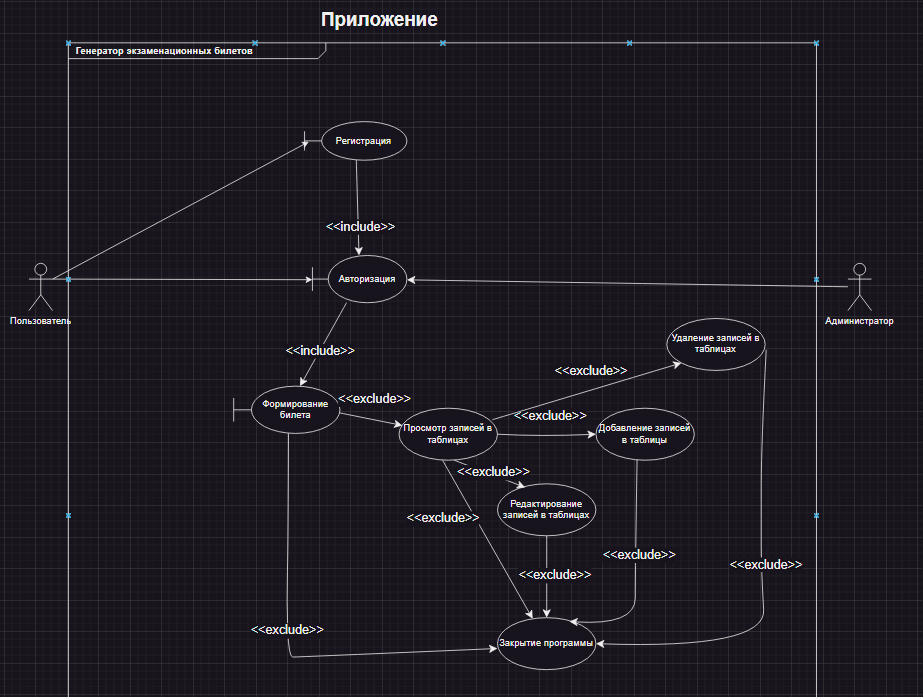


Рисунок 2.3. Диаграмма вариантов использования.

* + 1. Решение главной задачи проекта

Главная задача генератора экзаменационных билетов состоит в создании приложения, которое будет позволять эффективно управлять всеми процессами: авторизация пользователя, просмотр записи в таблице, редактирование записи в таблице, добавление записи в таблице, удаление записи из таблицы.

1. Реализовать вход пользователя в аккаунт

Код на c#:

/// <summary>

/// авторизация пользователя

/// </summary>

private void But\_authorization(object sender, RoutedEventArgs e)

{

string loginUser = textBox\_login.Text;

string roleUser = Role.Text;

string passUser = textBox\_password.Password.ToString();

if ((loginUser != "") && (roleUser != "") && (passUser != ""))

{

if (roleUser == "Admin" || roleUser == "User") //работает Admin

{

//сверяем введённые данные и данные в бд

if (db.Users.Any(o => (o.login\_ == loginUser) && (o.password\_ == passUser) && (o.role\_ == roleUser)))

{

MessageBox.Show("Успешная авторизация");

NavigationService.Navigate(new Choice\_admin(roleUser));

}

else

{

MessageBox.Show("Неправильный логин или пароль");

}

}

else

{

MessageBox.Show("Неверно указана роль пользователя");

}

}

else MessageBox.Show("Для продолжения заполните все поля");

}

Выполнение в приложении указано на Рисунке 2.8.

1. Реализовать формирование билета

Код на c#:

private void But\_Click\_Form\_Ticket(object sender, RoutedEventArgs e)

{

if (FindSpecId(FindSpecialityId()).ToString() == "")

{

MessageBox.Show("Выберите Дисциплину");

}

Initialize\_questions();

string disca\_content = Disca.Text;

var helper = new WordHelper("Ex\_Ticket\_Prac.docx");

string count\_tickets = count\_of\_tickets.Text;

string teacher\_content = Teacher.Text;

string Chairman\_pck\_content = Chairman.Text;

string kurs\_content = Kurs.Text;

string semester\_content = Semester.Text;

string speciality\_content = Spec.Text;

string protocol\_content = Protocol.Text;

string protocol\_date\_content = GetProtDateString();

string sem\_year\_content = GetSemYearString();

int nom\_ticket = 1;

int t;

if (int.TryParse(count\_tickets, out t))

{

if (t < 1)

{

MessageBox.Show("Число билетов должно быть >= 1");

return;

}

}

else

{

MessageBox.Show("Неверный формат");

return;

}

try

{

for (int i = 0; i < Convert.ToInt32(count\_tickets); i++)

{

var ticket = new List<string>(FindQuestions(NextTicketId(FindKomplectId(GetKursId(), GetSemesterId(), GetProtocolsId()))));

string quest1 = ticket[0];

string quest2 = ticket[1];

string quest3 = ticket[2];

var Items = new Dictionary<string, string>

{

{"<DISC>", disca\_content},

{"<PCK>", Chairman\_pck\_content},

{"<PREP>", teacher\_content},

{"<KURS>", kurs\_content},

{"<SEM>", semester\_content},

{"<SPEC>", speciality\_content},

{"<NOMPROT>", protocol\_content},

{"<DATEPROT>", protocol\_date\_content},

{"<YEARSEM> ", sem\_year\_content},

{"<NOMTICK>", nom\_ticket.ToString()},

{"<TEO1>", quest1},

{"<TEO2>", quest2},

{"<PRAC1>", quest3},

};

helper.Process(Items);

nom\_ticket++;

MessageBox.Show($"Билет №{nom\_ticket} сформирован");

}

}

catch (Exception ex)

{

MessageBox.Show("Билеты закончились");

}

}

/// <summary>

/// формирование word документа

/// </summary>

internal bool Process(Dictionary<string, string> items)

{

using (var doc = WordprocessingDocument.Create("FilePath", WordprocessingDocumentType.Document))

{

MainDocumentPart mainPart = doc.AddMainDocumentPart();

mainPart.Document = new Document();

Body body = mainPart.Document.AppendChild(new Body());

Paragraph para = body.AppendChild(new Paragraph());

Run run = para.AppendChild(new Run());

run.AppendChild(new Text("this new text for test"));

}

Word.Application app = null;

try

{

app = new Word.Application();

Object file = fileinfo\_.FullName;

Object missing = Type.Missing;

app.Documents.Open(file);

foreach (var item in items)

{

Word.Find find = app.Selection.Find;

find.Text = item.Key;

find.Replacement.Text = item.Value;

Object wrap = Word.WdFindWrap.wdFindContinue;

Object replace = Word.WdReplace.wdReplaceAll;

find.Execute(FindText: Type.Missing,

MatchCase: false,

MatchWholeWord: false,

MatchWildcards: false,

MatchSoundsLike: missing,

MatchAllWordForms: false,

Forward: true,

Wrap: wrap,

Format: false,

ReplaceWith: missing, Replace: replace

);

}

Object newFileName = Path.Combine("C:\\VS Projects\\Commit\\Kursach\\Kursach\\WpfApp1\\Tickets\\",

DateTime.Now.ToString("HHmmss") + fileinfo\_.Name);

app.ActiveDocument.SaveAs2(newFileName);

app.ActiveDocument.Close();

return true;

}

catch (Exception ex)

{

Console.WriteLine(ex.ToString());

return false;

}

finally

{

if (app != null)

{

app.Quit();

}

}

}

1. Реализовать просмотр информации таблицы

Код на c#:

private void ViewingTable\_admin\_IsVisibleChanged(object sender, DependencyPropertyChangedEventArgs e)

{

if (Visibility == Visibility.Visible)

{

RandomTicketGenerator.GetContext().ChangeTracker.Entries().ToList().ForEach(p => p.Reload());

DGridQuestion.ItemsSource = RandomTicketGenerator.GetContext().Questions.ToList();

}

}

1. Реализовать добавление записи в таблицу

Код на c#:

private void But\_Click\_Save\_Question(object sender, RoutedEventArgs e)

{

var currentQuest = GetQuestions();

if (string.IsNullOrWhiteSpace(currentQuest.question))

{

MessageBox.Show("Корректно напишите вопрос");

return;

}

RandomTicketGenerator.GetContext().Questions.Add(currentQuest);

try

{

RandomTicketGenerator.GetContext().SaveChanges();

MessageBox.Show("Информация сохранена");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message.ToString());

}

}

1. Изменения записи в таблице

Код на c#:

private void UpdateQuestions()

{

\_selectedQuestion.id\_discipline = GetDisciplineId();

\_selectedQuestion.question = question\_textbox.Text;

\_selectedQuestion.type\_question = Type\_question.Text;

\_selectedQuestion.complexity = Complexity\_question.Text;

}

private void But\_Click\_Save\_Question(object sender, RoutedEventArgse)

{

if (string.IsNullOrWhiteSpace(\_selectedQuestion.question))

{

MessageBox.Show("Корректно напишите вопрос");

return;

}

UpdateQuestions();

try

{

RandomTicketGenerator.GetContext().SaveChanges();

MessageBox.Show("Информация сохранена");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message.ToString());

}

}

* + 1. Тестирование приложения

Таблица 3 – Метод проверки требований к приложению

1. Тестирование клиентской части

|  |  |  |  |
| --- | --- | --- | --- |
| № | Действие | Результат | Ожидаемый результат |
| 1 | Запуск приложения | Приложение подключается к серверу и базе данных | Приложение подключается к серверу и базе данных |
| 2 | Пользователь нажимает на кнопку «Авторизация» | Пользователю показывается уведомление о том, что не все поля заполнены | Пользователю показывается уведомление о том, что не все поля заполнены |
| 3 | Пользователь вводит данные и нажимает кнопку «Авторизация» | Система проверяет, есть ли в базе данных пользователь с таким логином и паролем. Если нет, то выводится уведомление, что логин или пароль не верны. Если да -открывается страница формирования билета | Система проверяет, есть ли в базе данных пользователь с таким логином и паролем. Если нет - уведомление, что логин или пароль не верны. Если да -открывается страница формирования билета |
| 4 | Пользователь нажимает на кнопку «Регистрация» | Пользователю показывается уведомление о том, что не все поля заполнены | Пользователю показывается уведомление о том, что не все поля заполнены |
| 5 | Пользователь вводит данные и нажимает кнопку «Регистрация» | Регистрация пользователя и переход на страницу авторизации | Регистрация пользователя и переход на страницу авторизации |
| 4 | Пользователь выбирает доступные специальности | Система отправляет на сервер запрос списка специальностей и предоставляет его пользователю | Система отправляет на сервер запрос списка специальностей и предоставляет его пользователю |
| 5 | Пользователь нажимает на кнопку подтверждение специальности | Система отображает доступные дисциплины | Система отображает доступные дисциплины |
| 6 | Пользователь выбирает доступные дисциплины | Система отправляет на сервер запрос списка специальностей и предоставляет его пользователю | Система отправляет на сервер запрос списка специальностей и предоставляет его пользователю |
|  | Пользователь выбирает доступные семестры | Система отправляет на сервер запрос списка семестров и предоставляет его пользователю | Система отправляет на сервер запрос списка курсов и предоставляет его пользователю |
| 7 | Пользователь выбирает количество билетов | Система проверяет, правильно ли указаны введённые данные. Если нет, то выводится уведомление о неверном формате. Если да – в переменную сохраняется число формируемых билетов | Система проверяет, правильно ли указаны введённые данные. Если нет, то выводится уведомление о неверном формате. Если да – в переменную сохраняется число формируемых билетов |
| 8 | Пользователь выбирает доступных преподавателей | Система отправляет на сервер запрос списка преподавателей и предоставляет его пользователю | Система отправляет на сервер запрос списка преподавателей и предоставляет его пользователю |
| 9 | Пользователь выбирает доступных председателей пцк | Система открывает пользователю окно корзины и отображает товары в ней | Система открывает пользователю окно корзины и отображает товары в ней |
| 10 | Пользователь выбирает доступные семестры | Система предлагает пользователю сохранить чек и после этого приложение завершает работу | Система предлагает пользователю сохранить чек и после этого приложение завершает работу |
|  | Пользователь выбирает доступные протоколы | Система предлагает пользователю сохранить чек и после этого приложение завершает работу | Система предлагает пользователю сохранить чек и после этого приложение завершает работу |
|  | Пользователь выбирает сложность | Система предлагает пользователю сохранить чек и после этого приложение завершает работу | Система предлагает пользователю сохранить чек и после этого приложение завершает работу |

1. Тестирование части администратора

|  |  |  |  |
| --- | --- | --- | --- |
| № | Действие | Результат | Ожидаемый результат |
| 1 | Запуск приложения | Приложение подключается к серверу и базе данных | Приложение подключается к серверу и базе данных |
| 2 | Пользователь нажимает на кнопку «Войти» | Пользователю высвечивается уведомление о том, что не все поля заполнены | Пользователю высвечивается уведомление о том, что не все поля заполнены |
| 3 | Пользователь вводит данные и нажимает кнопку «Войти» | Система проверяет, есть ли в базе данных администратор с таким логином и паролем. Если нет, то пользователю выводится уведомление о том, что логин или пароль не верны. Если да, то открывается страница администратора | Система проверяет, есть ли в базе данных администратор с таким логином и паролем. Если нет, то пользователю выводится уведомление о том, что логин или пароль не верны. Если да, то открывается страница администратора |
| 4 | 4 | Пользователь нажимает на кнопку «Регистрация» | Пользователю показывается уведомление о том, что не все поля заполнены |
| 5 | 5 | Пользователь вводит данные и нажимает кнопку «Регистрация» | Регистрация пользователя и переход на страницу авторизации |
| 6 | 4 | Пользователь выбирает доступные специальности | Система отправляет на сервер запрос списка специальностей и предоставляет его пользователю |
| 7 | 5 | Пользователь нажимает на кнопку подтверждение специальности | Система отображает доступные дисциплины |
|  | Адмнистратор нажимает на кнопку “Просмотр база дынных” | Переход на страницу вопросов и просмотр таблицы вопросов | Переход на страницу вопросов и просмотр таблицы вопросов |
|  | Администратор нажимает на кнопку “Редактировать” | Редактирование таблицы вопросов | Редактирование таблицы вопросов |
|  | Администратор нажимает на кнопку “Добавить” | Система проверяет, заполнено ли поле ввода.  Если да, система добавляет | Система проверяет, заполнено ли поле ввода.  Если да, система добавляет |
|  | Администратор нажимает на кнопку “Удалить” | Удаление вопроса | Удаление вопроса |
|  | Пользователь нажимает на кнопку “Таблица дисциплин” | Переход на страницу “Просмотр дисциплин” | Переход на страницу “Просмотр дисциплин” |
|  | Администратор нажимает на кнопку “Редактировать” | Редактирование таблицы дисциплин | Редактирование таблицы дисциплин |
|  | Администратор нажимает на кнопку “Добавить” | Система проверяет, заполнено ли поле ввода.  Если да, система добавляет | Система проверяет, заполнено ли поле ввода.  Если да, система добавляет |
|  | Администратор нажимает на кнопку “Удалить” | Удаление дисциплины | Удаление дисциплины |
|  | Пользователь нажимает на кнопку “Таблица Пользователей” | Переход на страницу “Просмотр пользователей” | Переход на страницу “Просмотр пользователей ” |
|  | Администратор нажимает на кнопку “Редактировать” | Редактирование таблицы пользователей | Редактирование таблицы пользователей |
|  | Администратор нажимает на кнопку “Добавить” | Система проверяет, заполнено ли поле ввода.  Если да, система добавляет | Система проверяет, заполнено ли поле ввода.  Если да, система добавляет |
|  | Администратор нажимает на кнопку “Удалить” | Удаление пользователей | Удаление пользователей |
|  | Пользователь нажимает на кнопку “Таблица Билетов” | Переход на страницу “Просмотр билетов” | Переход на страницу “Просмотр билетов ” |
|  | Администратор нажимает на кнопку “Редактировать” | Редактирование таблицы билетов | Редактирование таблицы билетов |
|  | Администратор нажимает на кнопку “Добавить” | Система проверяет, заполнено ли поле ввода.  Если да, система добавляет | Система проверяет, заполнено ли поле ввода.  Если да, система добавляет |
|  | Администратор нажимает на кнопку “Удалить” | Удаление билетов | Удаление билетов |

* + 1. Защита информационной системы

Для обеспечения безопасности в базах данных использовались средства защиты в MS SQL Server и Entity Framework.

Использованные средства:

Защита от инъекций: Entity Framework позволяет использовать LINQ запросы, чтобы предотвратить ошибки при работе с данными.

* 1. Инструкция пользователю

Инструкция для пользователя:

1. Для начала использования приложения необходимо авторизоваться. Для этого введите логин и пароль в соответствующие поля на странице авторизации и нажмите кнопку "Войти".
2. Если у вас нет аккаунта, нажмите кнопку "Регистрация" и заполните поля Роль, Логин и Пароль. После этого вы сможете авторизоваться в приложении.
3. После авторизации переход на главную страницу приложения. На этой странице представлены составляющие билета. Выберите данные для формирования билета. Первоначально нужно выбрать специальность, чтобы получить доступ к выбору дисциплины и преподавателя. Также можно нажать кнопку "Авторизация", чтобы перейти к выбору пользователя.
4. После корректного выбора составляющих билета, при нажатии кнопки “Формирование билета” происходит формирование экзаменационного билета.
5. При нажатии на кнопку “Просмотр базы данных”, если при авторизации была указана роль Admin, то происходит переход на страницу просмотра вопросов. Можно просмотреть имеющиеся вопросы, добавить, редактировать или добавить записи.
6. При нажатии на кнопку “Таблица дисциплин происходит переход на страницу просмотра дисциплин. Можно просмотреть имеющиеся дисциплины, добавить, редактировать или добавить записи.
7. При нажатии на кнопку “Таблица пользователей” происходит переход на страницу просмотра пользователей. Можно просмотреть имеющихся пользователей, добавить, редактировать или добавить записи.
8. При нажатии на кнопку “Таблица билетов” происходит переход на страницу просмотра билетов. Можно просмотреть имеющиеся билеты, добавить, редактировать или добавить записи.
   * 1. Общие сведения об информационной системе

Генератор экзаменационных билетов– это комплект программных средств, которые помогают автоматизировать составление билетов.

Основными элементами информационной системы являются:

1. Компьютер с установленным на него соответствующим программным обеспечением, предназначенным для учета заказов и хранения информации о пользователях.
2. Приложение, написанное на C# WPF с использованием технологии Entity Framework, которое реализует подключение к базе данных генератора экзаменационного билета и позволяет получать из неё данные и изменять их.
3. База данных MS SQL Server, которая хранит в себе данные

Информационной системы позволяет повысить эффективность работы, что позволяет экономить время и ресурсы. Также, система упрощает работу и обработку данных о билетах.

* + 1. Требования к техническим средствам

Для корректной работы приложения требуется соблюдать данные технические характеристики:

1. Минимальный объём оперативной памяти: 4 ГБ
2. Минимальные требования к монитору: Super VGA с разрешением 800x600 пикселей или более высоким.
3. Свободное место на диске: 30 МБ (Без установки SQL Server)
4. Доступ в Интернет
5. Тип процессора: AMD Opteron, AMD Athlon 64, Intel Xeon с поддержкой Intel EM64T, Intel Pentium IV с поддержкой EM64T.
6. Быстродействие процессора: частота 2,0 ГГц и выше
   * 1. Требования к программным средствам

Для корректной работы приложения, программное обеспечение должно соответствовать данному списку:

1. Операционная система: Windows 10 или Windows 11
2. MS SQL Server Management Studio
3. База данных в MS SQL Server
4. .NET Framework
5. Последние драйверы для системы
   * 1. Настройка информационной системы

Для интеграции информационной системы

Генератора экзаменационных билетов для работы понадобится:

1. Проверить, что система соответствует минимальным системным требованиям.
2. Установить всё необходимое программное обеспечение.
3. Настроить MS SQL Server.
4. Выполнить скрипты на создание базы данных.
5. Заменить строку подключения в приложении.
6. Раздать администраторам логины и пароли для доступа к базе данных.
7. Распространить копии приложения преподавателям или другим пользователям.
   * 1. Формы ввода

В информационной системе генератора экзаменационных билетов использовались следующие формы ввода:

1. Форма авторизации.

<картинка авторизации>

Рисунок 2.4. Форма авторизации

1. Форма регистрации, на которую пользователь переходи в случае отсутствия данных для авторизации. После регистрации переход на страницу авторизации.

<картинка регистрации>

Рисунок 2.5. Форма регистрации

1. Форма формирования билета, в которой пользователь может выбрать данные, которые будут в билете.

<картинка формирования билета>

Рисунок 2.6. Форма формирования билета

1. Форма просмотра вопросов, в которой можно просмотреть, добавить, редактировать, удалить данные из таблицы вопросов.

<картинка вопросов>

Рисунок 2.6. Форма просмотра вопросов

1. Форма просмотра дисциплин, в которой можно просмотреть, добавить, редактировать, удалить данные из таблицы дисциплин.

<картинка дисциплин>

Рисунок 2.6. Форма просмотра дисциплин

1. Форма просмотра пользователей, в которой можно просмотреть, добавить, редактировать, удалить данные из таблицы дисциплин.

<картинка пользователей>

Рисунок 2.6. Форма просмотра пользователей

1. Форма просмотра билетов, в которой можно просмотреть, добавить, редактировать, удалить данные из таблицы пользователей.

<картинка билетов>

Рисунок 2.6. Форма просмотра билетов

* + 1. Отчёты

При нажатие на кнопку “Формирование билета” создаётся word документ с указанными пользователем данными

Сформированный билет:

Рисунок 2.13. Сформированный билет

# ЗАКЛЮЧЕНИЕ

В заключении курсовой работы по созданию генератора экзаменационных билетов на языке C# с использованием технологии Entity Framework и базы данных MSSQL, можно сказать, что данная работа была успешно выполнена. В процессе работы были изучены основные принципы работы с технологией Entity Framework и инструментами, используемыми для работы с ней, включая LINQ-запросы. Была разработана программа, которая позволяет генерировать экзаменационные билеты для преподавателей на основе заданных критериев, таких как специализация, курс, дисциплина, семестр, преподаватель, председатель пцк, курс, протокол, сложность вопросов и количество вопросов в билете. Были использованы современные методы программирования, такие как C# и база данных MSSQL.

Были преодолены сложности с проблемами формирования word документа, корректного отображения элементов таблиц, значений по умолчанию, с добавлением, удалением и редактированием записей, авторизацией, регистрацией и с запросами.

Результаты работы демонстрируют важность и актуальность использования технологии Entity Framework для упрощения работы с базами данных и выполнения сложных запросов. В итоге, генератор экзаменационных билетов на языке C# с использованием технологии Entity Framework и базы данных MSSQL является полезным инструментом для любой организации, проводящей массовую проверку знаний студентов или сдающих сертификационные экзамены.

В будущем планируется усовершенствование дизайна и предоставление возможности выбора количества вопросов.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Программные решения для бизнеса. Информационные и коммуникационные технологии: <https://nationalteam.worldskills.ru/skills/programmnye-resheniya-dlya-biznesa/>
2. Основы LINQ: <https://metanit.com/sharp/tutorial/15.1.php>
3. Руководство по WPF: <https://metanit.com/sharp/wpf/>
4. SQL запросы быстро: <https://habr.com/ru/articles/480838/>
5. Microsoft.Office.Interop.Word Пространство имен: <https://learn.microsoft.com/ru-ru/dotnet/api/microsoft.office.interop.word?view=word-pia>
6. Работа с Word файлом через Interop.Word. Замена слов в шаблоне: <https://www.youtube.com/watch?v=vQ7uW6g0z-U&t=938s>
7. Уроки C# – LINQ – Where, Select, GroupBy, AsParallel, x.Key – C#: <https://www.youtube.com/watch?v=gF4X3yr0nsA&t=886s>
8. Элемент ComboBox: <https://metanit.com/sharp/windowsforms/4.7.php>

**ПРИЛОЖЕНИЕ A.SQL скрипты на создание и заполнение базы данных.**