

**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 090203-9о-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

# СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	3
1 ОБЩАЯ ЧАСТЬ .....	5
1.1 Назначение и цели создания системы .....	5
1.2 Обзор и анализ предметной области.....	5
1.3 Жизненный цикл базы данных .....	8
1.4 Выбор и характеристика СУБД.....	8
1.5 Выбор и характеристика среды разработки приложения .....	9
2 СПЕЦИАЛЬНАЯ ЧАСТЬ.....	13
2.1 Постановка задачи .....	13
2.2 Архитектура информационной системы .....	13
2.3 Логическая модель базы данных.....	15
2.4 Нормализация таблиц.....	15
2.5 Описание таблиц.....	16
2.6 Другие объекты базы данных .....	19
2.7 Разработка приложения.....	19
2.7.1 Диаграмма вариантов использования .....	20
2.7.2 Решение главной задачи проекта .....	20
2.7.3 Тестирование приложения.....	26
2.7.4 Защита информационной системы.....	26
2.8 Инструкция пользователю .....	27
2.8.1 Общие сведения об информационной системе.....	28
2.8.2 Требования к техническим средствам .....	29
2.8.3 Требования к программным средствам .....	29
2.8.4 Настройка информационной системы .....	29
2.8.5 Форма выводы .....	30
2.8.6 Отчёты.....	34
ЗАКЛЮЧЕНИЕ .....	36
СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ.....	38
ПРИЛОЖЕНИЕ А. SQL скрипты на создание и заполнение базы данных....	40
ПРИЛОЖЕНИЕ Б. КОД ПРОГРАММЫ .....	51
Окно «Авторизация».....	51
Окно «Формирование билета».....	54
Формирование Word документа.....	69
Окно «Просмотр таблицы билетов» .....	74

## **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 с#, 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 с# приложения и теоретический подготовки студента. При разработке приложения были рассмотрены существующие приложения в данной предметной области.

# **1 ОБЩАЯ ЧАСТЬ**

## **1.1 Назначение и цели создания системы**

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 создании 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 приложении 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:

- 1) Специальность;
- 2) Дисциплина;
- 3) Семестр;
- 4) Количество билетов, от числа которых зависит сколько будет сформировано билетов;
- 5) Преподаватель;
- 6) Председатель цикловой комиссии;
- 7) Курс;
- 8) Протокол;
- 9) 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 билета, номер вопроса в билете, номер вопроса, номер комплект. 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:

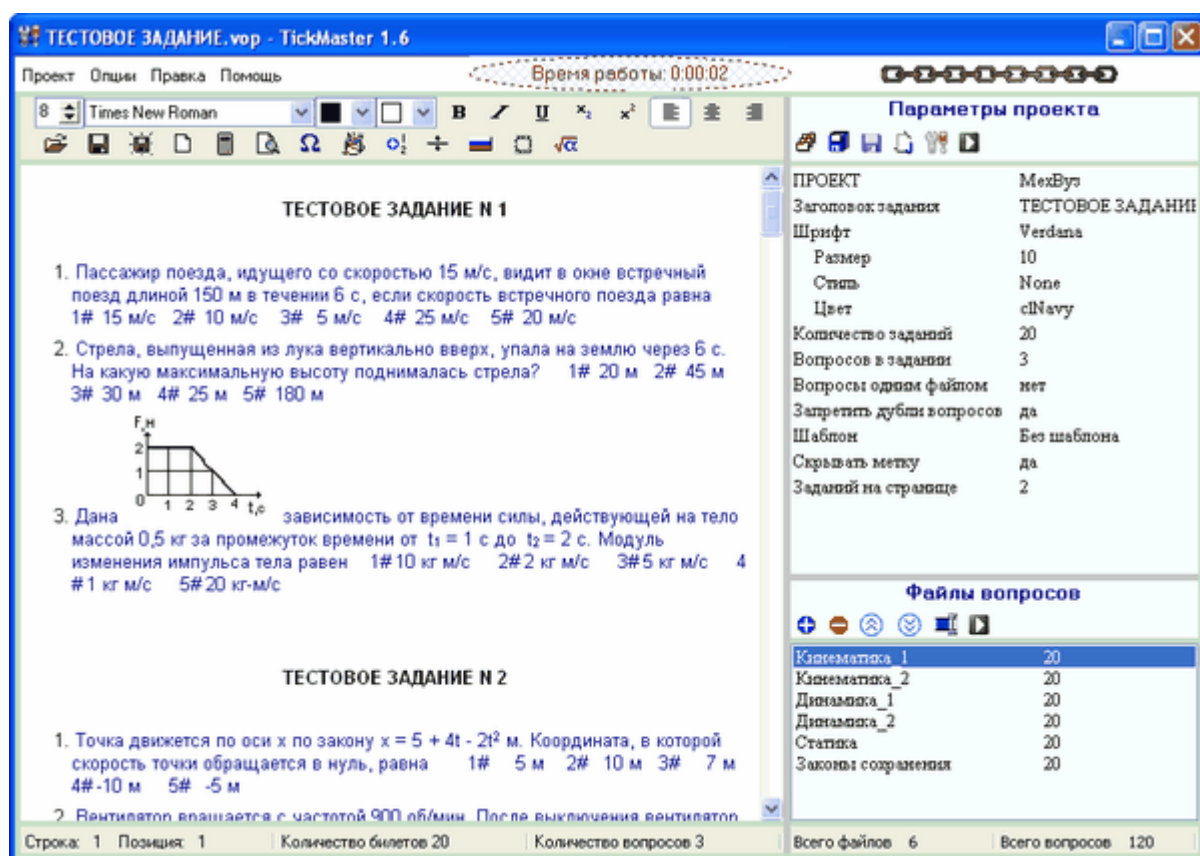


Рисунок 1.1. Приложение TickMaster

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.

### **1.3 Жизненный цикл базы данных**

Жизненный цикл базы данных — это процесс создания, сопровождения и удаления базы данных.

Жизненный цикл базы данных включает следующие этапы:

- 1) Анализ предметной области;
- 2) Проектирование;
- 3) Реализация;
- 4) Тестирование;
- 5) Эксплуатация и сопровождение.

### **1.4 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 1.3.

	Oracle	MS SQL Server	MySQL	FireBird
Бесплатность	нет	нет	да	да
Unix-совместимость	да	нет	да	да
Максимальный размер базы	неогр.	16 ТБ	неогр.	131 ТБ
Максимальный размер таблицы	неогр.	532 ГБ	256 ТБ	2,5 ТБ
Защита от подбора пароля	да	нет	нет	да
Макс. число пользователей	Не ограничено	Не ограничено	Не ограничено	1000
Наличие встраиваемых версий СУБД	нет	нет	да	да
Графический интерфейс	да	да	нет	Имеются утилиты

Рисунок 1.3. 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 с#, позволяя представить данные как реальные объекты, что позволяет абстрагироваться от базы данных и её таблиц, работая с данными как с объектами.

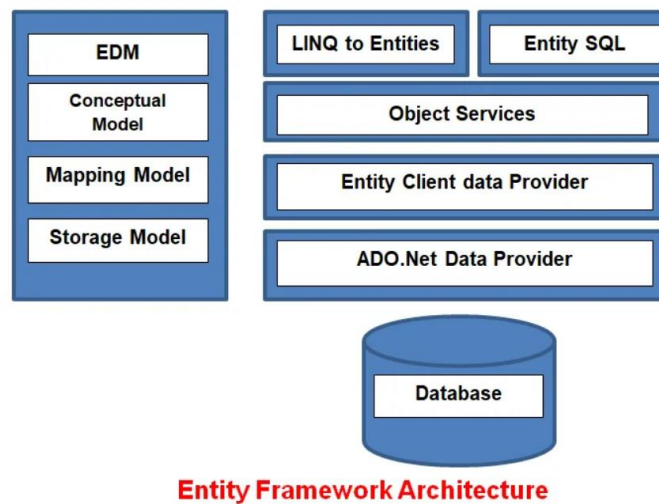


Рисунок 1.4. Схема работы Entity Framework

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**

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:

- 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;
- 7) возможность добавления, удаления, редактирования записей;
- 8) Билеты должны быть представлены в виде word документа;
- 9) 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 включает в себя сочетание навыков программирования и знаний по управлению базами данных для создания практического и полезного программного приложения.

## 2.2 Архитектура информационной системы

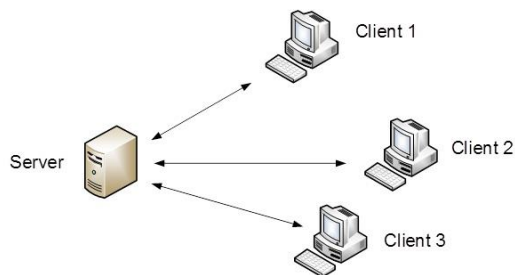


Рисунок 3 – Клиент-серверная модель

Клиент-серверная модель - это принцип построения распределенной системы, в которой компьютерные программы разбиваются на две части: клиентскую и серверную. Клиент обращается к серверу за выполнением определенного запроса, а сервер обрабатывает этот запрос и возвращает результат обратно клиенту.

Обычно в клиент-серверной модели клиент и сервер взаимодействуют через сеть, поэтому данная модель широко используется в сетевых приложениях, таких как веб-сайты, социальные сети, игры и приложения мессенджеров. Клиент - это программа, запущенная на компьютере или мобильном устройстве пользователя, которая обеспечивает пользовательский интерфейс для взаимодействия с приложением, отправляет запросы на сервер и обрабатывает ответы от сервера. Сервер, в свою очередь, это программа, выполняющая роль централизованной базы данных или сервиса, который обрабатывает запросы и возвращает данные клиенту.

Преимущества клиент-серверной модели включают:

- 1) Централизованность и контроль над системой;
- 2) Удобство распределения нагрузки;
- 3) Простота и надежность взаимодействия между клиентом и сервисом.

Недостатки включают:

- 1) Необходимость наличия соединения с сетью для работы;
- 2) Навыки настройки и обслуживания сервера.

## 2.3 Логическая модель базы данных

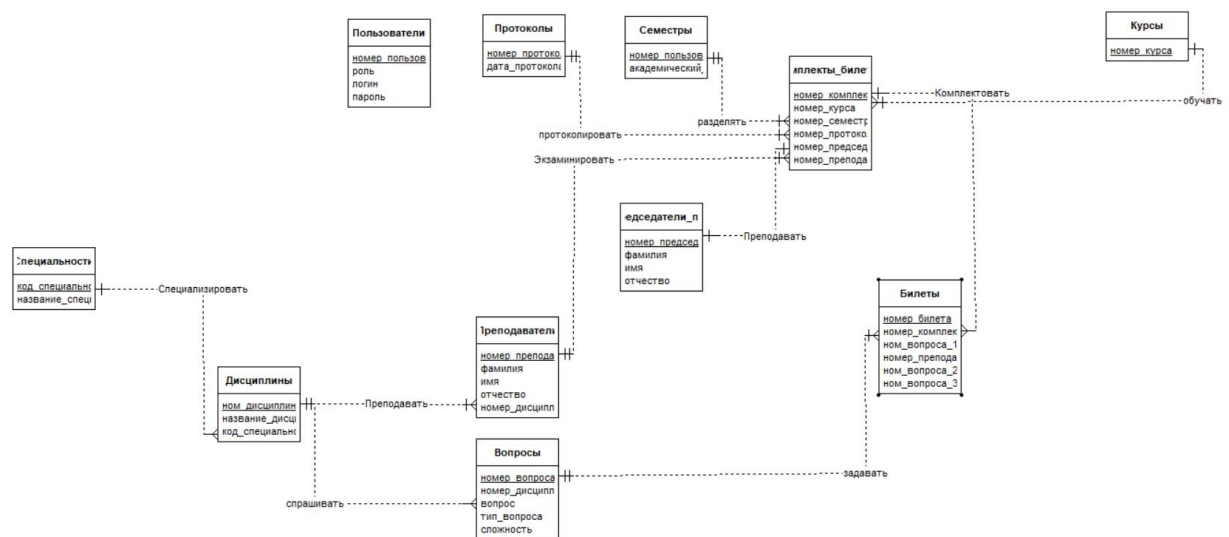


Рисунок 2.1. Логическая модель данных

## 2.4 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)

## 2.5 Описание таблиц

Таблица 1 – Users (Пользователи)

Название столбца	Описание	Тип	Примечание
nom_user	Номер пользователя	Int	РК
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	РК



Таблица 3 – Semesters (Семестры)

Название столбца	Описание	Тип	Примечание
nom_semester	Номер семестра	Int	PK
academic_year	Академический год	Date	

Таблица 4 – Protocols (Протоколы)

Название столбца	Описание	Тип	Примечание
nom_protocol	Номер протокола	Int	PK
date_protocol	Дата протокола	Date	

Таблица 5 – Chairman\_psk (Председатель\_пцк)

Название столбца	Описание	Тип	Примечание
id_chairman_psk	Номер председателя ПЦК	Int	PK
surname	Фамилия	nvarchar(100)	
name_	Имя	nvarchar(100)	
patronymic	Отчество	nvarchar(100)	

Таблица 6 – Komplekt\_tickets (Комплекты\_билетов)

Название столбца	Описание	Тип	Примечание
nom_komplekt	Номер комплекта билетов	Int	PK
nom_kurs	Номер курса	Int	FK
nom_semester	Номер семестра	Int	FK
nom_protocol	Номер протокола	Int	FK
id_chairman_psk	Номер Председателя пцк	Int	FK
id_teacher	Номер учителя	Int	FK

Таблица 7 – Speciality (Специальности)

Название столбца	Описание	Тип	Примечание
code_speciality	Код специальности	nvarchar(100)	PK
name_of_speciality	Название специальности	nvarchar(100)	

Таблица 8 – Disciplines (Дисциплины)

Название столбца	Описание	Тип	Примечание
id_discipline	Код специальности	Int	PK
name_discipline	Название специальности	nvarchar(500)	
code_speciality	Код специальности	nvarchar(100)	FK

Таблица 9 – Questions (Вопросы)

Название столбца	Описание	Тип	Примечание
id_question	Номер вопроса	Int	PK
id_discipline	Номер дисциплины	Int	FK
question	Вопрос	nvarchar(500)	
type_question	Тип вопроса	nvarchar(14)	
complexity	Сложность вопроса	nvarchar(8)	

Таблица 10 – Teachers (Преподаватели)

Название столбца	Описание	Тип	Примечание
id_teacher	Номер учителя	Int	PK
Surname	Фамилия	nvarchar(100)	
name_	Вопрос	nvarchar(100)	
Patronymic	Тип вопроса	nvarchar(100)	
id_discipline	Номер дисциплины	Int	FK

Таблица 11 – Tickets (Билеты)

Название столбца	Описание	Тип	Примечание
id_ticket	Номер билета	Int	PK
id_quest1	Вопрос 1	Int	
id_quest2	Вопрос 2	Int	
id_quest3	Вопрос 3	Int	
nom_komplect	Номер дисциплины	Int	FK
id_teacher	Номер учителя	Int	FK

## 2.6 Другие объекты базы данных

Другие объекты базы данных не используются.

## 2.7 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:

- 1) Добавление записей в таблицы;
- 2) Удаление записей в таблицах;
- 3) Редактирование записей в таблицах;
- 4) Просмотр информации из таблиц.

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

Далее, после преодоления проблем, связанных с проектированием, 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

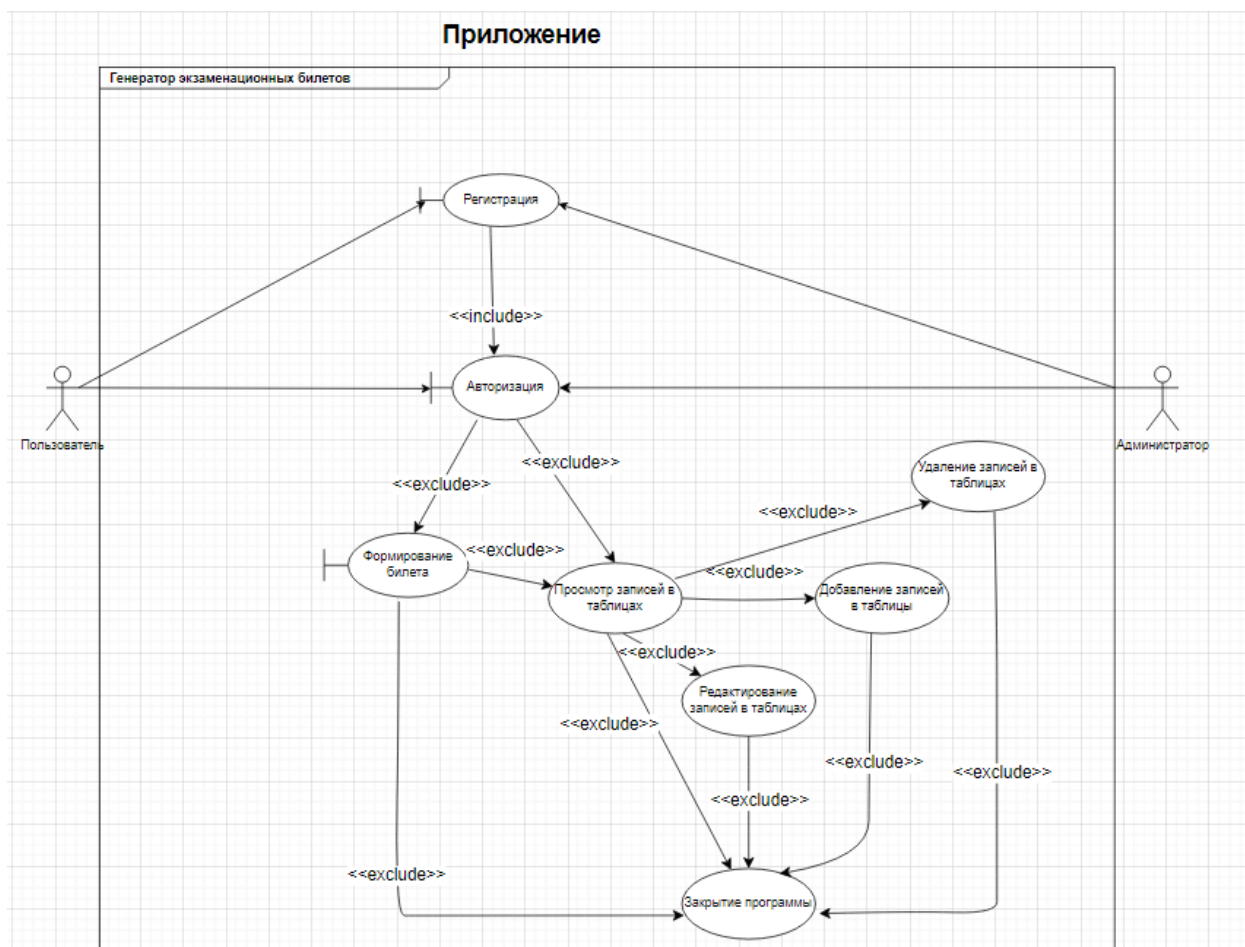


Рисунок 2.3. Диаграмма вариантов использования.

### 2.7.2 Решение главной задачи проекта

Главная задача генератора экзаменационных билетов состоит в создании экзаменационного билета в формате word документа по средству 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;

```

private void But_authorization(object sender, RoutedEventArgs e)
{
    string loginUser = textBox_login.Text;
    string roleUser = Role.Text;
    string passUser = textBox_password.Password.ToString().Encrypt();
}
    
```

```

if ((loginUser != "") && (roleUser != "") && (passUser != ""))
{
    if (roleUser == "Admin" || roleUser == "User")
    {
        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");
}

```

## 2) Реализовать формирование билета;

```

private void But_Click_Form_Ticket(object sender, RoutedEventArgs e)
{
    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(),
GetChairmanId(), FindTeacherId()))));
            string quest1 = ticket[0];
            string quest2 = ticket[1];
            string quest3 = ticket[2];
            MessageBox.Show("Формирование билета");
            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, disca_content, nom_ticket);
        MessageBox.Show($"Билет {disca_content} №{nom_ticket}
сформирован");
        nom_ticket++;
    }
}
catch (Exception ex)
{
    MessageBox.Show("Билеты закончились");
}
}

internal bool Process(Dictionary<string, string> items, string disca, int
count)
{
    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\\",
    count.ToString() + "_" + disca + " Билет.docx");
app.ActiveDocument.SaveAs2(newFileName);
app.ActiveDocument.Close();
return true;
}
catch (Exception ex)
{
    Console.WriteLine(ex.ToString());
    return false;
}
finally
{
    if (app != null)
    {
        app.Quit();
    }
}
}

```

### 3) Реализовать просмотр информации таблицы;

```
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();
    }
}
```

### 4) Реализовать добавление записи в таблицу;

```
private void But_Click_Save_Question(object sender, RoutedEventArgs e)
{
    var currentQuest = GetQuestions();

    if (string.IsNullOrEmpty(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());
    }
}
```

### 5) Изменения записи в таблице.

```
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, RoutedEventArgs)
    {
        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());
        }
    }
}

```

### 2.7.3 Тестирование приложения

№	Действие	Ожидаемый результат	Результат
1	Запуск приложения	Приложение подключается к серверу и базе данных	Приложение подключается к серверу и базе данных
2	Регистрация нового пользователя	Создается новая учетная запись в базе данных	Успешно
3	Авторизация пользователя	Зарегистрированный пользователь переходит на окно пользователя	Успешно
4	Авторизация администратора	Зарегистрированный администратор переходит на окно администратора	Успешно
5	Формирование экзаменационного билета	Выбор данных для формирования экзаменационного билета	Сформирован экзаменационный билет в формате word документа
6	Редактирование данных администратором	При выполнении одного из действий (Создания, удаления, изменения) происходит создание записи, удаление записи, и редактирование записи соответственно	Успешно
7	Выход	Выход из приложения	Успешно

Таблица 13 – Метод проверки требований к приложению.

### 2.7.4 Защита информационной системы

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

Entity Framework позволяет использовать LINQ запросы, чтобы предотвратить ошибки при работе с данными. При вводе некорректных данных в приложение выходит соответствующее предупреждающее сообщение об ошибке.

## 2.8 Инструкция пользователю

Для начала использования приложения необходимо авторизоваться. Для этого введите логин и пароль в соответствующие поля на странице авторизации и нажмите кнопку «Войти».

Если у вас нет аккаунта, нажмите кнопку «Регистрация» и заполните поля Роль, Логин и Пароль. После этого вы сможете авторизоваться в приложении.

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.

После корректного выбора составляющих билета, при нажатии кнопки «Формирование билета» происходит формирование экзаменационного билета.

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.

При нажатии на кнопку «Таблица дисциплин» происходит переход на страницу просмотра дисциплин. Можно просмотреть имеющиеся дисциплины, добавить, редактировать или добавить записи.

При нажатии на кнопку «Таблица пользователей» происходит переход на страницу просмотра пользователей. Можно просмотреть имеющихся пользователей, добавить, редактировать или добавить записи.

При нажатии на кнопку «Таблица билетов» происходит переход на страницу просмотра билетов. Можно просмотреть имеющиеся билеты, добавить, редактировать или добавить записи.

### 2.8.1 Общие сведения об информационной системе

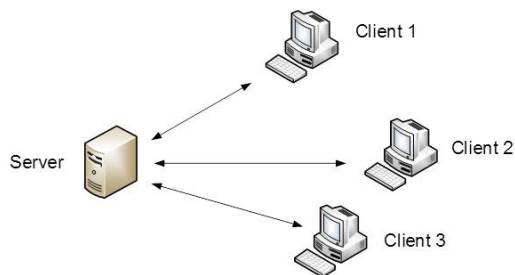


Рисунок 3 – Клиент-серверная модель

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.

### **2.8.2 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 EM64;
- 6) Быстродействие процессора: частота 2,0 ГГц и выше.

### **2.8.3 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.

### **2.8.4 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.

### 2.8.5 Форма вывода

Все формы ввода, имеющиеся в приложении проверяют корректность введённых данных. В случае введения некорректных данных, приложение предупреждает и не запрещает возможность вывода.

Генератор экзаменационных билетов

## Авторизация пользователя

**Роль:** Admin

**Логин:**

**Пароль:**

**Авторизация** **Регистрация**

Рисунок 2.4. Форма авторизации.

- 1) Форма регистрации, на которую пользователь переходит в случае отсутствия данных для авторизации. После регистрации переход на страницу авторизации;

**Генератор экзаменационных билетов**

## Регистрация пользователя

**Роль:** Admin

**Логин:**

**Пароль:**

**Регистрация**
**Авторизация**

Рисунок 2.5. Форма регистрации.

- 2) Форма формирования билета, в которой пользователь может выбрать данные, которые будут использованы в формируемом экзаменационном билете;

**Генератор экзаменационных билетов**

## Формирование билета

<b>Специальность:</b>	<b>Дисциплина:</b>	<b>Семестр:</b>	<b>Количество билетов:</b>	<b>Преподаватель:</b>	<b>Председатель цикловой комиссии:</b>
<span style="border: 1px solid #ccc; padding: 2px 10px;">Сетевое и системн</span>	<span style="border: 1px solid #ccc; padding: 2px 10px;"></span>	<span style="border: 1px solid #ccc; padding: 2px 10px;">1</span>	<span style="border: 1px solid #ccc; display: inline-block; width: 50px; height: 20px;"></span>	<span style="border: 1px solid #ccc; padding: 2px 10px;"></span>	<span style="border: 1px solid #ccc; padding: 2px 10px;">Солонин М.С.</span>

<b>Курс:</b>	<b>Протокол:</b>	<b>Сложность вопросов:</b>	
<span style="border: 1px solid #ccc; padding: 2px 10px;">1</span>	<span style="border: 1px solid #ccc; padding: 2px 10px;">1</span>	<span style="border: 1px solid #ccc; padding: 2px 10px;">Сложный</span>	

**Авторизация**

**Формирование билета**

**Просмотр базы данных**

Рисунок 2.6. Форма формирования билета.

- 3) Форма просмотра вопросов, в которой можно просмотреть, добавить, редактировать, удалить данные из таблицы вопросов;

Индекс	Индент	Название дисциплины	Вопрос	Тип вопроса	Сложность вопроса	
1	1	ОП.01. Операционные системы	Назначение и функции операционной системы.	Теоретический	Сложный	Редактировать
2	1	ОП.01. Операционные системы	WIMP-интерфейс. SILK-интерфейс. Виды реализации.	Теоретический	Сложный	Редактировать
3	1	ОП.01. Операционные системы	Порядок загрузки операционной системы. Управление параметрами загрузки операционной системы.	Практический	Сложный	Редактировать
4	1	ОП.01. Операционные системы	Файловая система Linux: структура каталогов файловой системы. Основные каталоги Linux	Теоретический	Сложный	Редактировать
5	1	ОП.01. Операционные системы	Состояние потоков на разных этапах их разработки. Алгоритм планирования процессов основанный на относительных приоритетах	Теоретический	Сложный	Редактировать
6	1	ОП.01. Операционные системы	Настройка параметров рабочей среды пользователя	Практический	Сложный	Редактировать
7	1	ОП.01. Операционные системы	Типы адресов (символьные, виртуальные, физические). Классификация методов распределения оперативной памяти	Теоретический	Сложный	Редактировать
8	1	ОП.01. Операционные системы	Понятие оперативной памяти. Распределение памяти перемещаемыми разделами	Теоретический	Сложный	Редактировать
9	1	ОП.01. Операционные системы	Выполнение конфигурирования аппаратных устройств. Распределение ресурсов	Практический	Сложный	Редактировать
10	1	ОП.01. Операционные системы	Свопинг, как частный случай виртуальной памяти	Теоретический	Сложный	Редактировать
11	1	ОП.01. Операционные системы	Понятие виртуальной памяти. Сегментно - страничное распределение.	Теоретический	Сложный	Редактировать
12	1	ОП.01. Операционные системы	Управление оперативной памятью. Распределение физической памяти	Практический	Сложный	Редактировать
13	1	ОП.01. Операционные системы	Файловая система FAT. Логические области раздела FAT	Теоретический	Сложный	Редактировать
14	1	ОП.01. Операционные системы	Понятие кэш-памяти. Принцип действия кэш-памяти	Теоретический	Сложный	Редактировать
15	1	ОП.01. Операционные системы	Динамическая настройка виртуальной памяти в Linux Ubuntu	Практический	Сложный	Редактировать
16	1	ОП.01. Операционные системы	Составные модули ядра ОС.	Теоретический	Сложный	Редактировать
17	1	ОП.01. Операционные системы	Администрирование учётных записей и групп в ОС Linux.	Теоретический	Сложный	Редактировать
18	1	ОП.01. Операционные системы	Изучение структуры файловой системы. Управление дисками и файловыми системами	Практический	Сложный	Редактировать
19	1	ОП.01. Операционные системы	Управление разделением ресурсов в локальной сети	Теоретический	Сложный	Редактировать

Добавить      Формирование билета      Таблица дисциплин      Удалить

Рисунок 2.6. Форма просмотра вопросов.

- 4) Форма просмотра дисциплин, в которой можно просмотреть, добавить, редактировать, удалить данные из таблицы дисциплин;

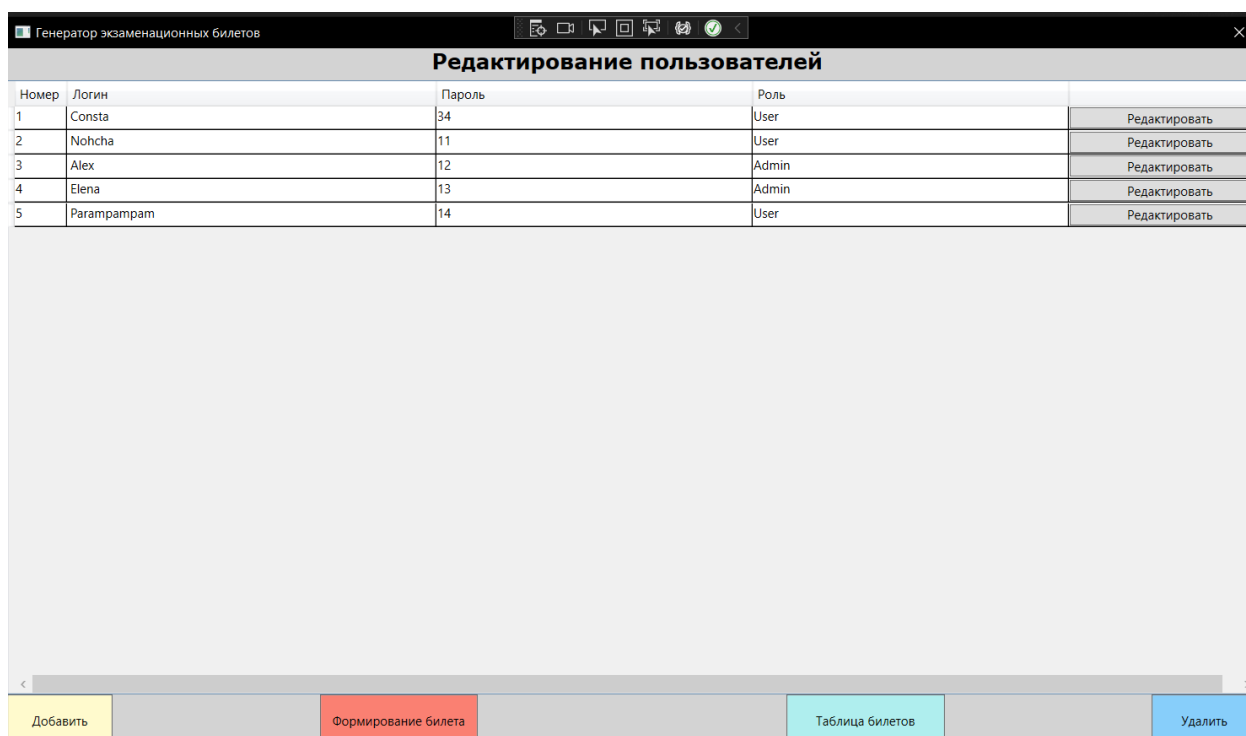
Индекс	Индентификатор дисциплины	
1	ОП.01. Операционные системы	Редактировать
2	ОП.10. Математическое моделирование	Редактировать
3	МДК.02.01. Информационные системы и сети	Редактировать
4	МДК.02.02. Технология разработки и защиты баз данных	Редактировать
5	ОГСЭ.03. Иностранный язык	Редактировать

Добавить      Формирование билета      Таблица Пользователей      Удалить

Рисунок 2.6. Форма просмотра дисциплин.



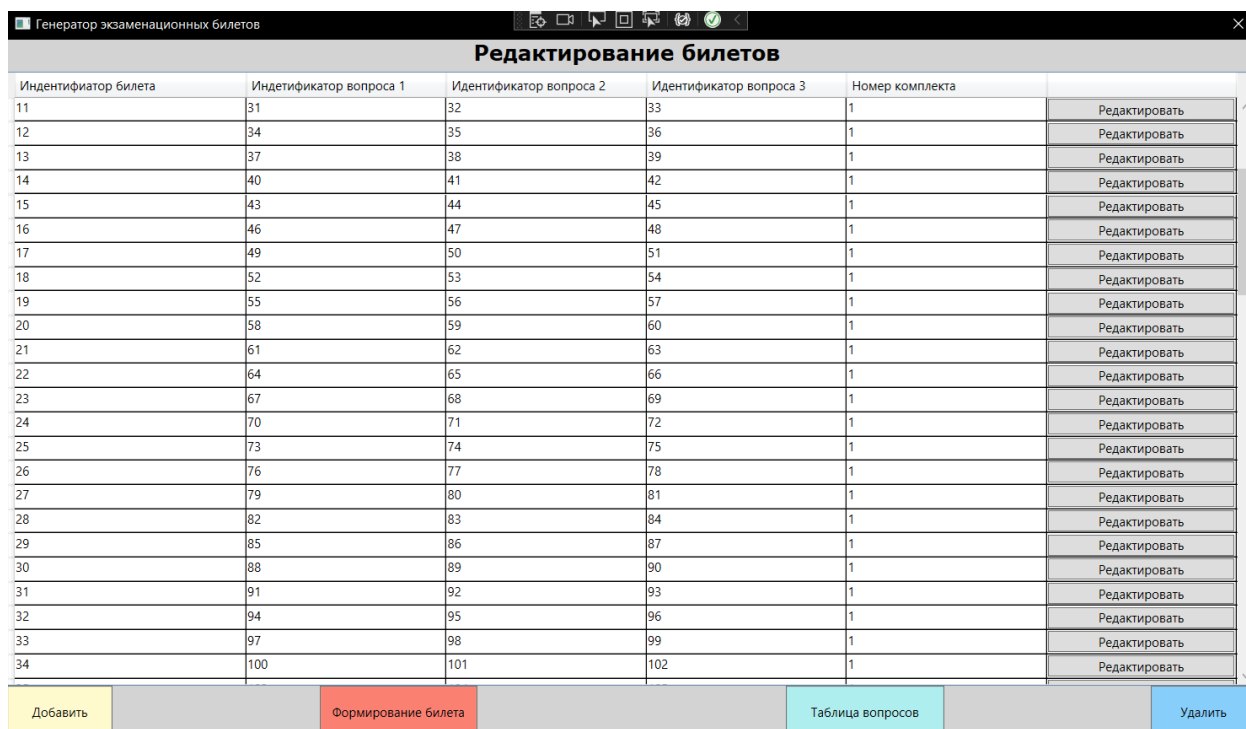
- 5) Форма просмотра пользователей, в которой можно просмотреть, добавить, редактировать, удалить данные из таблицы дисциплин;



Номер	Логин	Пароль	Роль	
1	Consta	34	User	Редактировать
2	Nohcha	11	User	Редактировать
3	Alex	12	Admin	Редактировать
4	Elena	13	Admin	Редактировать
5	Parampampam	14	User	Редактировать

Рисунок 2.6. Форма просмотра пользователей.

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



Идентификатор билета	Идентификатор вопроса 1	Идентификатор вопроса 2	Идентификатор вопроса 3	Номер комплекта	
11	31	32	33	1	Редактировать
12	34	35	36	1	Редактировать
13	37	38	39	1	Редактировать
14	40	41	42	1	Редактировать
15	43	44	45	1	Редактировать
16	46	47	48	1	Редактировать
17	49	50	51	1	Редактировать
18	52	53	54	1	Редактировать
19	55	56	57	1	Редактировать
20	58	59	60	1	Редактировать
21	61	62	63	1	Редактировать
22	64	65	66	1	Редактировать
23	67	68	69	1	Редактировать
24	70	71	72	1	Редактировать
25	73	74	75	1	Редактировать
26	76	77	78	1	Редактировать
27	79	80	81	1	Редактировать
28	82	83	84	1	Редактировать
29	85	86	87	1	Редактировать
30	88	89	90	1	Редактировать
31	91	92	93	1	Редактировать
32	94	95	96	1	Редактировать
33	97	98	99	1	Редактировать
34	100	101	102	1	Редактировать

Рисунок 2.6. Форма просмотра билетов.

## 2.8.6 Отчёты

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

- 1) Шаблон билета;

ФГБОУ ВО «МГУТУ им. К.Г. Разумовского (ПКУ)» Университетский колледж информационных технологий		
РАССМОТРЕНО	Экзаменационный билет № <NOMTICK>	УТВЕРЖДАЮ
предметной (цикловой) комиссией <SPEC> Протокол № <NOMPROT> от <DATEPROT>	Дисциплина <DISC>	Заместитель директора по учебно-методической работе
	Специальность <SPEC>	
Председатель ПИК _____ <PSK>	Курс <KURS>	Семестр <SEM> _____ Е.В. Вернар <YEARSEM> г.
1. Теоретический вопрос. <TEO1>		3 1
2. Теоретический вопрос. <TEO2>		3 2
3. Практическое задание. <PRAC1>		У 1, У 2, ОК 1 ПК 01
Преподаватель	<PREP>	

Рисунок 2.13. Шаблон билета.

2) Сформированный билет.

ФГБОУ ВО «МГТУ им. К.Г. Разумовского (ПКУ)» Университетский колледж информационных технологий			
РАССМОТРЕНО	Экзаменационный билет № 1		УТВЕРЖДАЮ
предметной (дисловой) комиссией <b>Сетевое и системное администрирование</b> Протокол № 1 от 02.02.2023	Дисциплина		Заместитель директора по учебно-методической работе
	<b>ОП.01. Операционные системы</b>		
	Специальность		
	<b>Сетевое и системное администрирование</b>		
Председатель ПИК <b>Соловьев М.С.</b>	Курс 1	Семестр 1	« _____ » _____ 2019г. Е.В. Вернер
1. Теоретический вопрос. <b>Операционная система Free-DOS</b>			3 1
2. Теоретический вопрос. <b>Использование меню в файлах конфигурации</b>			3 2
3. Практическое задание. <b>Назначение программы DiskDirectSuite. Размер раздела жесткого диска равен 2048 Кбайт. Сколько секторов будет содержать раздел? Создайте раздел такого размера с файловой системой fat.</b>			У 1, У 2, ОК 1 ПК 01
Преподаватель	<b>Соловьев К.В.</b>		

Рисунок 2.14. 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.

Получен опыт в проектирование и сделан вывод, что понимание на раннем этапе того, из чего должна состоять предметная область – способствует более быстрой реализации поставленной задачи.

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/> (Дата обращения: 30.03.2023)
2. Основы LINQ: <https://metanit.com/sharp/tutorial/15.1.php> (Дата обращения: 02.04.2023)
3. Руководство по WPF: <https://metanit.com/sharp/wpf/> (Дата обращения: 02.04.2023)
4. SQL запросы быстро: <https://habr.com/ru/articles/480838/> (Дата обращения: 02.04.2023)
5. Microsoft.Office.Interop.Word Пространство имен:  
<https://learn.microsoft.com/ru-ru/dotnet/api/microsoft.office.interop.word?view=word-pia> (Дата обращения: 20.04.2023)
6. Работа с Word файлом через Interop.Word. Замена слов в шаблоне:  
<https://www.youtube.com/watch?v=vQ7uW6g0z-U&t=938s> (Дата обращения: 30.04.2023)
7. Уроки C# – LINQ – Where, Select, GroupBy, AsParallel, x.Key – C#:  
<https://www.youtube.com/watch?v=gF4X3yr0nsA&t=886s> (Дата обращения: 30.04.2023)
8. Элемент ComboBox: <https://metanit.com/sharp/windowsforms/4.7.php> (Дата обращения: 30.04.2023)
9. C# WPF: <https://www.youtube.com/watch?v=LkYB9gLzKxc&t=291s> (Дата обращения: 30.04.2023)
10. Урок 3: Подключение базы к приложению, аутентификация пользователей | WPF + MSSQL:  
<https://www.youtube.com/watch?v=7SoudUnDk5k&t=677s> (Дата обращения: 30.04.2023)

11. Getting selected value of a combobox:

<https://stackoverflow.com/questions/6901070/getting-selected-value-of-a-combobox> (Дата обращения: 30.04.2023)

12. ЧТО ТАКОЕ ENTITY FRAMEWORK?

<https://www.youtube.com/watch?v=QPuFSSVxM4I&t=1087s> (Дата обращения: 30.04.2023)

## ПРИЛОЖЕНИЕ А. SQL скрипты на создание и заполнение базы данных

```
USE master;

IF EXISTS (SELECT * FROM SYS.DATABASES WHERE NAME = 'Base')
    DROP DATABASE Base

GO

CREATE DATABASE Base
ON
( NAME = Base_data,
  FILENAME = 'C:\VS Projects\Commit\Kursach\Create_base_script\All\Basedata.mdf',
  SIZE = 8,
  MAXSIZE = 100,
  FILEGROWTH = 10 )
LOG ON
( NAME = Base_log,
  FILENAME = 'C:\VS Projects\Commit\Kursach\Create_base_script\All\Baselog.ldf',
  SIZE = 5MB,
  MAXSIZE = 25MB,
  FILEGROWTH = 5MB );

GO

USE Base;

IF EXISTS (SELECT * FROM SYSOBJECTS WHERE NAME = 'Users')
    DROP TABLE Users

IF EXISTS (SELECT * FROM SYSOBJECTS WHERE NAME = 'Kurs')
    DROP TABLE Kurs

IF EXISTS (SELECT * FROM SYSOBJECTS WHERE NAME = 'Semesters')
    DROP TABLE Semesters

IF EXISTS (SELECT * FROM SYSOBJECTS WHERE NAME = 'Protocols')
    DROP TABLE Protocols

IF EXISTS (SELECT * FROM SYSOBJECTS WHERE NAME = 'Chairman_pck')
    DROP TABLE Chairman_pck

IF EXISTS (SELECT * FROM SYSOBJECTS WHERE NAME = 'Speciality')
    DROP TABLE Speciality

IF EXISTS (SELECT * FROM SYSOBJECTS WHERE NAME = 'Disciplines')
    DROP TABLE Disciplines

IF EXISTS (SELECT * FROM SYSOBJECTS WHERE NAME = 'Teacher')
    DROP TABLE Teacher

IF EXISTS (SELECT * FROM SYSOBJECTS WHERE NAME = 'Komplekt_tickets')
```



```

        DROP TABLE Komplekt_tickets
    IF EXISTS (SELECT * FROM SYSOBJECTS WHERE NAME = 'Questions')
        DROP TABLE Questions
    IF EXISTS (SELECT * FROM SYSOBJECTS WHERE NAME = 'Tickets')
        DROP TABLE Tickets

CREATE TABLE Users
(
    nom_user INT IDENTITY PRIMARY KEY,
    login_ nvarchar(50),
    password_ nvarchar(50),
    role_ nvarchar(5)
);

CREATE TABLE Kurs
(
    nom_kurs INT PRIMARY KEY
);

INSERT INTO Kurs (nom_kurs)
VALUES (1),
(2),
(3),
(4),
(5);

CREATE TABLE Semesters
(
    nom_semester INT IDENTITY PRIMARY KEY,
    academic_year DATE
);

INSERT INTO Semesters (academic_year)
VALUES ('01.01.2019'),
('05.05.2020'),
('01.01.2021'),
('05.05.2022'),
('01.01.2023');

CREATE TABLE Protocols
(
    nom_protocol INT IDENTITY PRIMARY KEY,
    date_protocol date,
);

--DELETE FROM Protocols;

```

```

INSERT INTO Protocols (date_protocol)
VALUES ('02.02.2023'),
('03.03.2023'),
('04.04.2023'),
('05.05.2023'),
('06.06.2023');
CREATE TABLE Chairman_pck
(
    id_chairman_pck INT IDENTITY PRIMARY KEY,
    surname nvarchar(100),
    name_ nvarchar(100),
    patronymic nvarchar(100)
);
INSERT INTO Chairman_pck (surname, name_, patronymic)
VALUES
('Солонин', 'Марк', 'Семёнович'),
('Суворов', 'Виктор', 'Богданович'),
('Ларионова', 'Елена', 'Анатольевна'),
('Ленин', 'Владимир', 'Ильич'),
('Керенский', 'Александр', 'Фёдорович');
CREATE TABLE Speciality
(
    code_speciality nvarchar(100) PRIMARY KEY,
    name_of_speciality nvarchar(100),
);
INSERT INTO Speciality (code_speciality, name_of_speciality)
VALUES ('09.02.06', 'Сетевое и системной администрирование'),
('09.02.07', 'Информационные системы и программирование'),
('10.02.05', 'Обеспечение информационной безопасности автоматизированных систем'),
('21.02.19', 'Землеустройство'),
('42.02.01', 'Реклама');
CREATE TABLE Disciplines
(
    id_discipline int IDENTITY PRIMARY KEY,
    name_discipline nvarchar(500),
    code_speciality nvarchar(100),
    FOREIGN KEY (code_speciality) REFERENCES Speciality (code_speciality) ON DELETE
CASCADE ON UPDATE CASCADE,

```

```

);

INSERT INTO Disciplines (name_discipline, code_speciality)
VALUES ('ОП.01. Операционные системы', '09.02.06'),
('ОП.10. Математическое моделирование', '09.02.07'),
('МДК.02.01. Инфокоммуникационные системы и сети', '10.02.05'),
('МДК.02.02. Технология разработки и защиты баз данных', '21.02.19'),
('ОГСЭ.03. Иностранный язык', '42.02.01');

CREATE TABLE Teacher
(
    id_teacher INT IDENTITY PRIMARY KEY,
    surname nvarchar(100),
    name_ nvarchar(100),
    patronymic nvarchar(100),
    id_discipline int,
    FOREIGN KEY (id_discipline) REFERENCES Disciplines (id_discipline)
    ON DELETE CASCADE ON UPDATE CASCADE

);

INSERT INTO Teacher (surname, name_, patronymic, id_discipline)
VALUES
('Смирнов', 'Константин', 'Вадимович', 1),
('Марцинкевич', 'Максим', 'Сергеевич', 2),
('Ларионова', 'Елена', 'Анатольевна', 3),
('Мурашов', 'Анатолий', 'Алексеевич', 4),
('Глускер', 'Александр', 'Игоревич', 5);

CREATE TABLE Komplect_tickets
(
    nom_komplect INT IDENTITY PRIMARY KEY,
    nom_kurs INT,
    nom_semester INT,
    nom_protocol INT,
    id_chairman_pck INT,
    id_teacher INT,
    FOREIGN KEY (nom_kurs) REFERENCES Kurs (nom_kurs) ON DELETE CASCADE ON UPDATE
CASCADE,
    FOREIGN KEY (nom_semester) REFERENCES Semesters (nom_semester) ON DELETE CASCADE
ON UPDATE CASCADE,
    FOREIGN KEY (nom_protocol) REFERENCES Protocols (nom_protocol) ON DELETE CASCADE
ON UPDATE CASCADE,

```

```

        FOREIGN KEY (id_chairman_pck) REFERENCES Chairman_pck (id_chairman_pck) ON
DELETE CASCADE ON UPDATE CASCADE,

        FOREIGN KEY (id_teacher) REFERENCES Teacher (id_teacher) ON DELETE CASCADE ON
UPDATE CASCADE,

    );

    INSERT INTO Komplect_tickets (nom_kurs, nom_semester, nom_protocol, id_chairman_pck,
id_teacher)

VALUES (1, 1, 1, 1, 1),

(2, 2, 2, 2, 2),

(3, 3, 3, 3, 3),

(4, 4, 4, 4, 4),

(5, 5, 5, 5, 5);

CREATE TABLE Questions

(

    id_question INT IDENTITY,

    id_discipline int,

    question nvarchar(500),

    type_question nvarchar(14),

    complexity nvarchar(8),

    PRIMARY KEY(id_question),

    FOREIGN KEY (id_discipline) REFERENCES Disciplines (id_discipline)

    ON DELETE CASCADE ON UPDATE CASCADE

);

INSERT INTO Questions (id_discipline, question, type_question, complexity)

VALUES

--ОП.01. Оцперационные системы практические и теоретические

--сложные вопросы

(1, 'Назначение и функции операционной системы.', 'Теоретический', 'Сложный'),

(1, 'WIMP-интерфейс. SILK-интерфейс. Виды реализации.', 'Теоретический', 'Сложный'),

(1, 'Порядок загрузки операционной системы.

Управление параметрами загрузки операционной системы.', 'Практический', 'Сложный'),

(1, 'Файловая система Linux: структура каталогов файловой системы. Основные каталоги

Linux',

    'Теоретический', 'Сложный'),

(1, 'Состояние потоков на разных этапах их разработки.Алгоритм планирования процессов

основанный на относительных приоритетах', 'Теоретический', 'Сложный'),

(1, 'Настройка параметров рабочей среды пользователя', 'Практический', 'Сложный'),

(1, 'Типы адресов (символьные, виртуальные, физические). Классификация методов

распределения оперативной памяти', 'Теоретический', 'Сложный'),

(1, 'Понятие оперативной памяти. Распределение памяти перемещаемыми разделами',

```

```

    'Теоретический', 'Сложный'),
    (1, 'Выполнение конфигурирования аппаратных
устройств. Распределение ресурсов', 'Практический', 'Сложный'),
CREATE TABLE Tickets
(
    id_ticket INT IDENTITY, --IDENTITY,
    id_quest1 INT,
        id_quest2 INT,
        id_quest3 INT,
    nom_komplect INT,
        PRIMARY KEY(id_ticket),
        FOREIGN KEY (id_quest1) REFERENCES Questions (id_question),
        FOREIGN KEY (id_quest2) REFERENCES Questions (id_question),
        FOREIGN KEY (id_quest3) REFERENCES Questions (id_question),
        FOREIGN KEY (nom_komplect) REFERENCES Komplect_tickets (nom_komplect) ON DELETE
CASCADE ON UPDATE CASCADE
);
INSERT INTO Tickets (id_quest1, id_quest2, id_quest3, nom_komplect)
VALUES
(1, 2, 3, 1 ),
(4, 5, 6, 1),
(7, 8, 9, 1);

```

## ПРИЛОЖЕНИЕ Б. Код программы

### Окно «Авторизация»

XAML:

```
<Page x:Class="WpfApp1.Authorization"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:local="clr-namespace:WpfApp1"
      mc:Ignorable="d"
      d:DesignHeight="700" d:DesignWidth="1200"
      Title="Authorization">
    <Grid Background="#FFD3D3D3">
        <Grid.ColumnDefinitions>
            <ColumnDefinition>
            </ColumnDefinition>
            <ColumnDefinition>
            </ColumnDefinition>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition>
            </RowDefinition>
            <RowDefinition>
            </RowDefinition>
            <RowDefinition>
            </RowDefinition>
            <RowDefinition>
            </RowDefinition>
            <RowDefinition>
            </RowDefinition>
            <RowDefinition>
            </RowDefinition>
        </Grid.RowDefinitions>
        <TextBlock Grid.Row="0" Grid.Column="0" Grid.ColumnSpan="2"
            Height="70" Width="680"
            FontSize="40" FontFamily="Verdana" FontWeight="Bold"
x:Name="Header">
            Авторизация пользователя
        </TextBlock>
        <TextBlock Grid.Row="1" Grid.Column="0"
            Height="70" Width="350"
            FontSize="30" FontFamily="Verdana" FontWeight="Bold"
x:Name="role">
            Роль:
        </TextBlock>
        <TextBlock Grid.Row="2" Grid.Column="0"
            Height="70" Width="350"
            FontSize="30" FontFamily="Verdana" FontWeight="Bold"
x:Name="login">
            Логин:
        </TextBlock>
        <TextBlock Grid.Row="3" Grid.Column="0"
            Height="70" Width="350"
            FontSize="30" FontFamily="Verdana" FontWeight="Bold"
x:Name="password">
            Пароль:
        </TextBlock>
        <ComboBox Name="Role"
            Grid.Row="1"
            Grid.Column="1">
```

```

        Height="70" Width="400"
        HorizontalAlignment="Center"
        VerticalAlignment="Top" Margin="0,41,0,0" />
<TextBox Grid.Row="2" Grid.Column="1"
        Height="70" Width="400"
        MaxLength="50"
        TextWrapping="Wrap"
        x:Name="textBox_login" Margin="100,41,100,29">
</TextBox>
<PasswordBox Grid.Row="3" Grid.Column="1"
        Height="70" Width="400"
        x:Name="textBox_password" Margin="100,40,100,30"
        MaxLength="50"
        >
</PasswordBox>
<Button Grid.Row="4" Grid.Column="0" Width="400"
        FontSize="30" FontFamily="Verdana" FontWeight="Bold"
        x:Name="But2"
        Content="Авторизация"
        Click="But_authorization" Margin="0,10,0,16">
<Button.Background>
    <SolidColorBrush Color="#FFFA8072" />
</Button.Background>
</Button>
<Button Grid.Row="4" Grid.Column="1" Width="400"
        FontSize="30" FontFamily="Verdana" FontWeight="Bold"
        x:Name="But1"
        Content="Регистрация"
        Click="But_registration" Margin="0,10,0,16">
<Button.Background>
    <SolidColorBrush Color="#FF87CEFA" />
</Button.Background>
</Button>
</Grid>
</Page>

```

C#:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Navigation;

namespace WpfApp1
{
    /// <summary>
    /// Логика взаимодействия для Authorization.xaml
    /// </summary>
    public partial class Authorization : Page
    {
        RandomTicketGenerator db;
        public Authorization()
        {
            InitializeComponent();
            bindcombo_Users();
            db = new RandomTicketGenerator();
        }
        //ComboBox kurs
        public List<Users> Users_list { get; set; }
        private void bindcombo_Users()

```

```

{
    Role.Items.Add("Admin");
    Role.Items.Add("User");
    Role.SelectedIndex = 0;
}
/// <summary>
/// сокрытие пароля
/// </summary>
private void log_in_load(object sender, EventArgs e)
{
    textBox_password.PasswordChar = '•';
    textBox_password.MaxLength = 100;
}
/// <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().Encrypt();
    if ((loginUser != "") && (roleUser != "") && (passUser != ""))
    {
        if (roleUser == "Admin" || roleUser == "User") //работает
Admin
        {
            //проверка на соответствие данных
            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");
}
/// <summary>
/// Переход к регистрации
/// </summary>
private void But_registration(object sender, RoutedEventArgs e)
{
    NavigationService.Navigate(new Sign_up());
}
}
}

```



## Окно «Формирование билета»

### XAML:

```
<Page x:Class="WpfApp1.Choice_admin"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:local="clr-namespace:WpfApp1"
      mc:Ignorable="d"
      d:DesignHeight="700" d:DesignWidth="800"
      Title="Choice_admin">
<Grid Background="#FFD3D3D3">
    <Grid.ColumnDefinitions>
        <ColumnDefinition>
        </ColumnDefinition>
        <ColumnDefinition>
        </ColumnDefinition>
        <ColumnDefinition>
        </ColumnDefinition>
        <ColumnDefinition>
        </ColumnDefinition>
        <ColumnDefinition>
        </ColumnDefinition>
        <ColumnDefinition>
        </ColumnDefinition>
        <ColumnDefinition>
        </ColumnDefinition>
        <ColumnDefinition>
        </ColumnDefinition>
    </Grid.ColumnDefinitions>
    <Grid.RowDefinitions>
        <RowDefinition>
        </RowDefinition>
        <RowDefinition>
        </RowDefinition>
        <RowDefinition>
        </RowDefinition>
        <RowDefinition>
        </RowDefinition>
        <RowDefinition>
        </RowDefinition>
        <RowDefinition>
        </RowDefinition>
        <RowDefinition>
        </RowDefinition>
        <RowDefinition>
        </RowDefinition>
    </Grid.RowDefinitions>
    <TextBlock Grid.Row="0"
                Grid.Column="1"
                Grid.ColumnSpan="4"
                Height="70"
                Width="260"
                FontSize="20"
                FontFamily="Verdana"
                FontWeight="Bold"
                x:Name="Header"
                >
        Формирование билета
    </TextBlock>
    <TextBlock Grid.Row="1"
                Height="70"
                Width="130"
                FontSize="14"
                FontFamily="Verdana"
                FontWeight="Bold"
                HorizontalAlignment="Center"
                VerticalAlignment="Center"
                x:Name="spec_header"
                >
        Специальность:
```

```

</TextBlock>
<TextBlock Grid.Row="1"
            Grid.Column="3"
            Height="70"
            Width="120"
            TextWrapping="Wrap"
            FontSize="14"
            FontFamily="Verdana"
            FontWeight="Bold"
            HorizontalAlignment="Center"
            VerticalAlignment="Center"
            x:Name="count_of_tickets_header" >
    Количество билетов:
</TextBlock>
<TextBlock Grid.Row="1"
            Grid.Column="4"
            Height="70"
            Width="130"
            TextWrapping="Wrap"
            FontSize="14"
            FontFamily="Verdana"
            FontWeight="Bold"
            HorizontalAlignment="Center"
            VerticalAlignment="Center"
            x:Name="teacher_header" >
    Преподаватель:
</TextBlock>
<TextBlock Grid.Row="1"
            Grid.Column="5"
            Height="70"
            Width="120"
            TextWrapping="Wrap"
            FontSize="14"
            FontFamily="Verdana"
            FontWeight="Bold"
            HorizontalAlignment="Center"
            VerticalAlignment="Center"
            x:Name="ck_header" >
    Председатель цикловой комиссии:
</TextBlock>
<TextBlock Grid.Row="3"
            Grid.Column="0"
            Height="70"
            Width="120"
            FontSize="14"
            FontFamily="Verdana"
            FontWeight="Bold"
            x:Name="kurs_header"
            HorizontalAlignment="Center"
            VerticalAlignment="Center">
    Курс:
</TextBlock>
<TextBlock Grid.Row="1"
            Grid.Column="1"
            Height="70"
            Width="120"
            FontSize="14"
            FontFamily="Verdana"
            FontWeight="Bold"
            x:Name="Disca_header"
            HorizontalAlignment="Center"
            VerticalAlignment="Center">
    Дисциплина:
</TextBlock>
<TextBlock Grid.Row="1"
            Grid.Column="2"
            Height="70"
            Width="135"

```

```

        FontSize="14"
        FontFamily="Verdana"
        FontWeight="Bold"
        x:Name="Semester_header"
        HorizontalAlignment="Center"
        VerticalAlignment="Center">
Семестр:
</TextBlock>
<TextBlock Grid.Row="3"
        Grid.Column="1"
        Height="70"
        Width="120"
        FontSize="14"
        FontFamily="Verdana"
        FontWeight="Bold"
        x:Name="Prot_header"
        HorizontalAlignment="Center"
        VerticalAlignment="Center">
Протокол:
</TextBlock>
<TextBlock Grid.Row="3"
        Grid.Column="2"
        TextWrapping="Wrap"
        Height="70"
        Width="120"
        FontSize="14"
        FontFamily="Verdana"
        FontWeight="Bold"
        x:Name="complexity_header"
        HorizontalAlignment="Center"
        VerticalAlignment="Center">
Сложность вопросов:
</TextBlock>
<ComboBox Name="Disca"
        Grid.Row="2"
        Grid.Column="1"
        Width="133"
        Height="30"
        HorizontalAlignment="Center"
        VerticalAlignment="Top"
/>
<TextBox Grid.Row="2"
        Grid.Column="3"
        Height="40"
        Width="133"
        TextWrapping="Wrap"
        MaxLength="20"
        x:Name="count_of_tickets"
        Background="White"
        HorizontalAlignment="Center"
        VerticalAlignment="Top"
/>
<ComboBox Name="Teacher"
        Grid.Row="2"
        Grid.Column="4"
        HorizontalAlignment="Center"
        VerticalAlignment="Top"
        Width="133"
        Height="30">
</ComboBox>
<ComboBox Name="Chairman"
        Grid.Row="2"
        Grid.Column="5"
        HorizontalAlignment="Center"
        VerticalAlignment="Top"
        Width="133"
        Height="30">
</ComboBox>

```

```

<ComboBox Name="Kurs"
    Grid.Row="4"
    Grid.Column="0"
    Height="30"
    HorizontalAlignment="Center"
    VerticalAlignment="Top"
    Width="133" />
<ComboBox Name="Semester"
    Grid.Row="2"
    Grid.Column="2"
    Height="30"
    HorizontalAlignment="Center"
    VerticalAlignment="Top"
    Width="133" />
<ComboBox Name="Spec"
    Grid.Row="2"
    Grid.Column="0"
    Height="30"
    HorizontalAlignment="Center"
    VerticalAlignment="Top"
    Width="133" />
<ComboBox Name="Protocol"
    Grid.Row="4"
    Grid.Column="1"
    Height="30"
    HorizontalAlignment="Center"
    VerticalAlignment="Top"
    Width="133" />
<ComboBox x:Name="complexity_of_question"
    Grid.Row="4"
    Grid.Column="2"
    Height="30"
    HorizontalAlignment="Center"
    VerticalAlignment="Top"
    Width="133"
    SelectedIndex="0"
    Background="White">
    <TextBlock>Сложный</TextBlock>
    <TextBlock>Средний</TextBlock>
    <TextBlock>Простой</TextBlock>
</ComboBox>
<Button Grid.Row="5" Grid.Column="2" Grid.ColumnSpan="2"
    Height="70" Width="250"
    FontSize="12" FontFamily="Verdana" FontWeight="Bold"
    x:Name="But_Form"
    Content="Формирование билета"
    Click="But_Click_Form_Ticket">
    <Button.Background>
        <SolidColorBrush Color="#FFFA8072" />
    </Button.Background>
</Button>
<Button Grid.Row="5" Grid.Column="0" Grid.ColumnSpan="2"
    Height="70" Width="250"
    FontSize="12" FontFamily="Verdana" FontWeight="Bold"
    x:Name="But_Authorization"
    Content="Авторизация"
    Click="But_Auto">
    <Button.Background>
        <SolidColorBrush Color="#FFFFFFACD" />
    </Button.Background>
</Button>
<Button Grid.Row="5" Grid.Column="4" Grid.ColumnSpan="2"
    Height="70" Width="250"
    FontSize="12" FontFamily="Verdana" FontWeight="Bold"
    x:Name="But_viewing"
    Content="Просмотр базы данных"

    Click="But_Click_Viewing_Table_Data">

```

```

        <Button.Background>
            <SolidColorBrush Color="#FF87CEFA" />
        </Button.Background>
    </Button>
    <Button Grid.Row="4" Grid.Column="4" Grid.ColumnSpan="2"
        Height="70" Width="250"
        FontSize="12" FontFamily="Verdana" FontWeight="Bold"
        x:Name="But_Confirmation"
        Content="Подтверждение специальности"
        Click="But_Confirmation_Click">
        <Button.Background>
            <SolidColorBrush Color="#FFAFEEEE" />
        </Button.Background>
    </Button>
</Grid>
</Page>

```

C#:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Navigation;
using System.Data;

namespace WpfApp1
{
    public class ExaminerItem
    {
        public int Id { get; set; }
        public string Fio { get; set; }

        public ExaminerItem(int id, string fio)
        {
            this.Id = id;
            this.Fio = fio;
        }
    }

    public class Teach
    {
        public int Id { get; set; }
        public string Fio { get; set; }

        public Teach(int id, string fio)

```

```

        {
            this.Id = id;
            this.Fio = fio;
        }
    }
}

public partial class Choice_admin : Page
{
    private string roleUser;

    private List<Tickets> tick;

    public Choice_admin(string role)
    {
        InitializeComponent();
        InitializationSpeciality();
        InitializationKurs();
        InitializationSemester();
        InitializationProtocols();
        InitializationChairman();
        var rng = new Random();
        tick = RandomTicketGenerator.GetContext().Tickets.ToList();
        tick = tick.OrderBy(x => rng.Next()).ToList();
        roleUser = role;
    }

    private string FindSpecialityId()
    {
        return ((Speciality)Spec.SelectedItem).code_speciality;
    }

    /// <summary>
    /// получение id специальности
    /// </summary>
    private string FindSpecId(string code)
    {
        string id =
        (from i in RandomTicketGenerator.GetContext().Speciality.ToList()
        where i.code_speciality == code//FindSpecialityId()
        select i.code_speciality).First();
        return id;
    }
}

```

```

    /// <summary>
    /// заполнение ComboBox Специальности
    /// </summary>
    private void InitializationSpeciality()
    {

        IEnumerable<Speciality> Speciality_list = from i in
RandomTicketGenerator.GetContext().Speciality.ToList()

        select i;

        DataContext = Speciality_list;
        Spec.ItemsSource = Speciality_list;
        Spec.SelectedValuePath = "";
        Spec.DisplayMemberPath = "name_of_speciality";
        Spec.SelectedIndex = 0;
    }
    /// <summary>
    /// получение id Курса
    /// </summary>
    private int GetKursId()
    {
        int id =
(from i in RandomTicketGenerator.GetContext().Kurs.ToList()
where i.nom_kurs == Convert.ToInt32(Kurs.Text)
select i.nom_kurs).First();
        return id;
    }
    private int GetKurs()
    {
        return ((Kurs)Kurs.SelectedItem).nom_kurs;
    }

    private void InitializationKurs()
    {

        IEnumerable<Kurs> Kurs_list = from i in
RandomTicketGenerator.GetContext().Kurs.ToList()

        select i;

        DataContext = Kurs_list;
        Kurs.ItemsSource = Kurs_list;
        Kurs.SelectedValuePath = "";
        Kurs.DisplayMemberPath = "nom_kurs";

```

```

        Kurs.SelectedIndex = 0;
    }
    /// <summary>
    /// получение id Семестров
    /// </summary>
    private int GetSemesterId()
    {
        int id =
(from i in RandomTicketGenerator.GetContext().Semesters.ToList()
 where i.nom_semester == Convert.ToInt32(Semester.Text)
 select i.nom_semester).First();
        return id;
    }
    private DateTime GetSemDate()
    {
        return ((Semesters)Semester.SelectedItem).academic_year.Value;
    }

    private string GetSemYearString()
    {
        return GetSemDate().ToString("yyyy");
    }
    /// <summary>
    /// заполнение ComboBox Специальности
    /// </summary>
    private void InitializationSemester()
    {
        IEnumerable<Semesters> Semesters_list = from i in
RandomTicketGenerator.GetContext().Semesters.ToList()
                                                select i;

        DataContext = Semesters_list;
        Semester.ItemsSource = Semesters_list;
        Semester.SelectedValuePath = "";
        Semester.DisplayMemberPath = "nom_semester";
        Semester.SelectedIndex = 0;
    }
    /// <summary>
    /// получение id протокола
    /// </summary>

```



```

        private int GetProtocolsId()
        {
            int id =
            (from i in RandomTicketGenerator.GetContext().Protocols.ToList()
             where i.nom_protocol == Convert.ToInt32(Protocol.Text)
             select i.nom_protocol).First();
            return id;
        }
        /// <summary>
        /// заполнение ComboBox Протоколов
        /// </summary>
        private void InitializationProtocols()
        {
            IEnumerable<Protocols> Protocols_list = from i in
RandomTicketGenerator.GetContext().Protocols.ToList()
                                                    select i;

            DataContext = Protocols_list;
            Protocol.ItemsSource = Protocols_list;
            Protocol.SelectedValuePath = "";
            Protocol.DisplayMemberPath = "nom_protocol";
            Protocol.SelectedIndex = 0;
        }
        private DateTime GetProtDate()
        {
            return ((Protocols)Protocol.SelectedItem).date_protocol.Value;
        }
        private string GetProtDateString()
        {
            return GetProtDate().ToString("dd.MM.yyyy");
        }
        private void InitializationChairman()
        {
            IEnumerable<ExaminerItem> Chairman_list = from i in
RandomTicketGenerator.GetContext().Chairman_pck.ToList()
                                                    let fio = i.surname + " " +
i.name_[0] + "." + i.patronymic[0] + "."
                                                    select
                                                    new
ExaminerItem(i.id_chairman_pck, fio);

            DataContext = Chairman_list;
            Chairman.ItemsSource = Chairman_list;

```

```

        Chairman.DisplayMemberPath = "Fio";
        Chairman.SelectedIndex = 0;
    }
    List<Questions> Questions_list { get; set; }
    /// <summary>
    /// Инициализация Вопросов
    /// </summary>
    private void Initialize_questions()
    {
        IEnumerable<Questions> Quest_list = (from i in
RandomTicketGenerator.GetContext().Questions.ToList()
where i.id_discipline ==
FindDisciplineId(FindSpecId(FindSpecialityId()))
select i);

        var rng = new Random();
        Questions_list = Quest_list.OrderBy(x => rng.Next()).ToList();
    }

    /// <summary>
    /// получение id Председателя
    /// </summary>
    private int GetChairmanId()
    {
        return ((ExaminerItem)Chairman.SelectedItem).Id;
    }

    /// <summary>
    /// получение id учителя
    /// </summary>
    private int FindTeacher()
    {
        return ((Teach)Teacher.SelectedItem).Id;
    }

    /// <summary>
    /// получение id комплекта билетов
    /// </summary>
    private int FindKomplectId(int kursId, int semesterId, int protocolId, int
chairmanId, int teacherId)
    {
        int id =

```

```

(int)(from i in RandomTicketGenerator.GetContext().Komplekt_tickets.ToList()
    where i.nom_kurs == kursId//GetKursId() //+
    && i.nom_semester == semesterId//GetSemesterId() //+
    && i.nom_protocol == protocolId//GetProtocolsId() //+
        && i.id_chairman_pck == GetChairmanId() //+
    && i.id_teacher == FindTeacher() //6, а нужен 2
    select i.nom_komplekt).First();
    return id;
}
/// <summary>
/// получение id билета
/// </summary>
private int NextTicketId(int kompletId)
{
    int res = (from i in tick.ToList()
        where i.nom_komplekt == kompletId
        select i.id_ticket).First();
    tick.RemoveAll((ticket) => ticket.id_ticket == res);
    return res;
}
private IEnumerable<string> FindQuestions(int tickId) //NextTicketid
{
    var tickets = (from i in
RandomTicketGenerator.GetContext().Tickets.ToList()
    where i.id_ticket == tickId
    select i).First();

    var questions = (from quest in
RandomTicketGenerator.GetContext().Questions.ToList()
    where
        (quest.id_question == tickets.id_quest1
        || quest.id_question == tickets.id_quest2
        || quest.id_question == tickets.id_quest3)
        && quest.id_discipline
FindDisciplineId(FindSpecId(FindSpecialityId())) ==
        select quest.question);
    return questions;
}
private int FindDisciplineId(string specfind) //+
{

```

```

        int id =
        (from i in RandomTicketGenerator.GetContext().Disciplines.ToList()
         where i.code_speciality == specfind //GetSpecialityId()
         select i.id_discipline).First();
        return id;
    }

    private int FindTeacherId()
    {
        int id =
        (from i in RandomTicketGenerator.GetContext().Teacher.ToList()
         where i.id_discipline == FindDisciplineId(FindSpecId(FindSpecialityId()))
//discId//GetDisciplineId()
         && i.id_teacher == FindTeacher()
         select i.id_teacher).First();
        return id;
    }

    private void But_Confirmation_Click(object sender, RoutedEventArgs e)
    {

        IEnumerable<Disciplines>      Disc_list      =      from      i      in
RandomTicketGenerator.GetContext().Disciplines.ToList()

                                where      i.code_speciality      ==
FindSpecId(FindSpecialityId())

                                &&      i.id_discipline      ==
FindDisciplineId(FindSpecId(FindSpecialityId())) //2//FindDisciplineId(FindSpecialityId())

                                select i;

        DataContext = Disc_list;
        Disca.ItemsSource = Disc_list;
        Disca.SelectedValuePath = "";
        Disca.DisplayMemberPath = "name_discipline";
        Disca.SelectedIndex = 0;

        IEnumerable<Teach>      Teacher_list      =      from      i      in
RandomTicketGenerator.GetContext().Teacher.ToList()

                                where      i.id_discipline      ==
FindDisciplineId(FindSpecId(FindSpecialityId()))//2//FindDiscipline()

                                let fio = i.surname + " " + i.name_[0] +
"." + i.patronymic[0] + "."

                                select new Teach(i.id_teacher, fio);

        DataContext = Teacher_list;
        Teacher.ItemsSource = Teacher_list;

```

```

        Teacher.DisplayMemberPath = "Fio";
        Teacher.SelectedIndex = 0;
    }

private void But_Click_Form_Ticket(object sender, RoutedEventArgs e)
{
    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(),
        GetProtocolsId(),
        GetChairmanId(), FindTeacherId()))));
        string quest1 = ticket[0];
        string quest2 = ticket[1];
        string quest3 = ticket[2];
        MessageBox.Show("Формирование билета");
        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()},
            {"<TE01>", quest1},
            {"<TE02>", quest2},
            {"<PRAC1>", quest3},
        };
        helper.Process(Items, disca_content, nom_ticket);
        MessageBox.Show($"Билет {disca_content} №{nom_ticket}
сформирован");
        nom_ticket++;
    }
}
catch (Exception ex)
{
    MessageBox.Show("Билеты закончились");
}
}
private void But_Click_Viewing_Table_Data(object sender, RoutedEventArgs e)
{
    if (roleUser == "Admin")
    {
        NavigationService.Navigate(new ViewingTableData_admin(roleUser));
    }
}

```

```
    }  
    else  
    {  
        MessageBox.Show("Только администратор может редактировать таблицы");  
    }  
}  
private void But_Auto(object sender, RoutedEventArgs e)  
{  
    NavigationService.Navigate(new Authorization());  
}  
}  
}
```

## Формирование Word документа

C#:

```
using System;
using System.Collections.Generic;
using Word = Microsoft.Office.Interop.Word;
using System.IO;
using DocumentFormat.OpenXml.Packaging;
using DocumentFormat.OpenXml.Wordprocessing;
using DocumentFormat.OpenXml;

namespace WpfApp1
{
    internal class WordHelper
    {
        private FileInfo fileinfo_;
        public WordHelper(string filename)
        {
            if (File.Exists(filename))
            {
                fileinfo_ = new FileInfo(filename);
            }
            else
            {
                throw new ArgumentException("File not found");
            }
        }
        /// <summary>
        /// формирование word документа
        /// </summary>
        internal bool Process(Dictionary<string, string> items, int count)
        {
            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;
```





```

</Grid.ColumnDefinitions>
<TextBlock Grid.Row="0"
            Grid.Column="1"
            Grid.ColumnSpan="2"
            Height="30"
            Width="300"
            FontSize="20"
            FontFamily="Verdana"
            FontWeight="Bold"
            x:Name="Header"
            >
                Редактирование билетов
</TextBlock>
<DataGrid x:Name="DGridTicket"
            AutoGenerateColumns="False"
            IsReadOnly="True"
            Grid.Column="0"
            Grid.ColumnSpan="4"
            Grid.Row="1">
    <DataGrid.Columns >
        <DataGridTextColumn Header="Идентификатор билета"
                            Binding="{Binding id_ticket}"
                            Width="220">
            <DataGridTextColumn.ElementStyle>
                <Style>
                    <Setter Property="TextBlock.TextWrapping"
Value="Wrap" />
                    <Setter Property="TextBlock.TextAlignment"
Value="Left"/>
                </Style>
            </DataGridTextColumn.ElementStyle>
        </DataGridTextColumn>
        <DataGridTextColumn Header="Идентификатор вопроса 1"
                            Binding="{Binding id_quest1}"
                            Width="190">
            <DataGridTextColumn.ElementStyle>
                <Style>
                    <Setter Property="TextBlock.TextWrapping"
Value="Wrap" />
                    <Setter Property="TextBlock.TextAlignment"
Value="Left"/>
                </Style>
            </DataGridTextColumn.ElementStyle>
        </DataGridTextColumn>
        <DataGridTextColumn Header="Идентификатор вопроса 2"
                            Binding="{Binding id_quest2}"
                            Width="190">
            <DataGridTextColumn.ElementStyle>
                <Style>
                    <Setter Property="TextBlock.TextWrapping"
Value="Wrap" />
                    <Setter Property="TextBlock.TextAlignment"
Value="Left"/>
                </Style>
            </DataGridTextColumn.ElementStyle>
        </DataGridTextColumn>
        <DataGridTextColumn Header="Идентификатор вопроса 3"
                            Binding="{Binding id_quest3}"
                            Width="190">
            <DataGridTextColumn.ElementStyle>
                <Style>

```



```

        Grid.Column="2"
        HorizontalAlignment="Right"
        Width="150"
        Name="QuestionTable"
        Click="NextTable" >
        <Button.Background>
            <SolidColorBrush Color="#FFAFEEEE" />
        </Button.Background>
    </Button>
</Grid>
</Page>

```

C#:

```

using System;
using System.Linq;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Navigation;

namespace WpfApp1
{
    /// <summary>
    /// Логика взаимодействия для ViewingTicketTable.xaml
    /// </summary>
    public partial class ViewingTicketTable : Page
    {
        private string roleUser;
        public ViewingTicketTable(string roleUser)
        {
            InitializeComponent();
            this.roleUser = roleUser;
        }

        //страница редактирования
        private void BtnEdit_Click(object sender, RoutedEventArgs e)
        {
            NavigationService.Navigate(new EditingTicket((sender as Button).DataContext as Tickets, roleUser));
        }

        //страница добавления
        private void BtnAdd_Click(object sender, RoutedEventArgs e)
        {
            NavigationService.Navigate(new AddTicket(roleUser));
        }

        private void BtnDelete_Click(object sender, RoutedEventArgs e)
        {
            var ticketForRemoving = DGridTicket.SelectedItems.Cast<Tickets>().ToList();

            if (MessageBox.Show($"Вы точно хотите удалить следующие {ticketForRemoving.Count()} элементов?", "Внимание",
                MessageBoxButton.YesNo, MessageBoxImage.Question) ==
                MessageBoxResult.Yes)
            {
                try
                {
                    RandomTicketGenerator.GetContext().Tickets.RemoveRange(ticketForRemoving);
                    RandomTicketGenerator.GetContext().SaveChanges();
                    MessageBox.Show("Данные удалены");

                    DGridTicket.ItemsSource =
                    RandomTicketGenerator.GetContext().Tickets.ToList();
                }
            }
        }
    }
}

```

```

        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message.ToString());
        }
    }
}
private void ViewingTable_admin_IsVisibleChanged(object sender,
DependencyPropertyChangedEventArgs e)
{
    if (Visibility == Visibility.Visible)
    {
        RandomTicketGenerator.GetContext().ChangeTracker.Entries().ToList().ForEach(p =>
p.Reload());
        DGridTicket.ItemsSource =
RandomTicketGenerator.GetContext().Tickets.ToList();
    }
}
private void Form_Ticket_Click(object sender, RoutedEventArgs e)
{
    NavigationService.Navigate(new Choice_admin(roleUser));
}
private void NextTable(object sender, RoutedEventArgs e)
{
    NavigationService.Navigate(new ViewingTableData_admin(roleUser));
}
}
}

```