МИНОБРНАУКИ РОССИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**“ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”**

Факультет компьютерных наук

Кафедра информационных технологий в управлении предприятием

Курсовая работа по курсу

«Технологии программирования»

*«*Система для составления расписания факультета»

Преподаватель \_\_\_\_\_\_\_\_\_\_\_ В.С. Тарасов, ассистент

Обучающийся\_\_\_\_\_\_\_\_\_\_\_\_О.В. Пастухова,3 курс, д/о

Обучающийся\_\_\_\_\_\_\_\_\_\_\_\_М.А. Дьяконова,3 курс, д/о

Руководитель \_\_\_\_\_\_\_\_\_\_\_ Х.А. Полещук, ассистент

Воронеж 2019

Оглавление

[Введение 3](#_Toc10726479)

[Глава 1. Постановка задачи 5](#_Toc10726480)

[Глава 2. Анализ предметной области 6](#_Toc10726481)

[2.1 Глоссарий 6](#_Toc10726482)

[2.2 Аналоги решения задачи 7](#_Toc10726483)

[2.3 Методология 8](#_Toc10726484)

[2.4 Моделирование бизнес процессов составления расписания 9](#_Toc10726485)

[2.5 Анализ требований 10](#_Toc10726486)

[2.6 Анализ процесса получения списка аудиторий 12](#_Toc10726487)

[2.7 Анализ процесса изменения преподавателя 13](#_Toc10726488)

[2.8 Анализ процесса изменения расписания 15](#_Toc10726489)

[2.9 Анализ работы приложения 16](#_Toc10726490)

[Глава 3. Архитектура приложения 20](#_Toc10726491)

[3.1 Анализ структуры данных 20](#_Toc10726492)

[3.2 Визуализация элементов системы 23](#_Toc10726493)

[3.3 Проектирование пользовательского интерфейса 24](#_Toc10726494)

[Глава 4. Реализация приложения 26](#_Toc10726495)

[4.1 Средства реализации 26](#_Toc10726496)

[4.2 Реализация кода 27](#_Toc10726497)

[4.3 Реализация интерфейса 35](#_Toc10726498)

[Глава 5. Тестирование 40](#_Toc10726499)

[Заключение 42](#_Toc10726500)

[Список источников 43](#_Toc10726501)

## **Введение**

Для того чтобы облегчить работу человека, в настоящее время во многие сферы общества внедряются различные программные продукты. Не исключением стала и выбранная нами область деятельности.

Одной из основных задач учебного процесса является грамотное составление расписания занятий. Оно должно обеспечивать равномерную загрузку студенческих групп и преподавательского состава, так как это напрямую влияет на качество обучения. При составлении расписания нужно учитывать требования по организации учебного процесса, пожелания преподавателей, наличие свободных аудиторий и при всем этом надо стремиться исключить большие перерывы между занятиями, как у преподавателей, так и у студентов.

Также немаловажно, что учебное расписание должно быть достаточно информативным. Там должны быть отражены не только аудитория и время проведения занятий, но и номер группы и курса, название предмета и имя преподавателя.

В итоге, задача составления расписания становится очень сложной, так как при ее реализации существует много нюансов. При его заполнении не всегда получается минимизировать затраты во времени. Также не всегда корректно подбираются аудитории и часто можно столкнуться с большими перерывами между предметами в расписании, что составляет некоторые неудобства как студентам, так и преподавателям.

Все это приводит к низкой результативности обучения, поэтому автоматизация данного процесса является одной из основных проблем в учебных заведениях.

Противоречие связано с тем, что проблема достижения положительных результатов, а точнее минимизация незадействованного для учебы времени, решена, однако методов для её решения недостаточно, ведь большинство подобных приложений имеют либо платную основу, либо непонятный интерфейс.

Отсюда вытекает актуальность работы. Именно поэтому в качестве направления деятельности была выбрана разработка приложения для пользователя с любыми навыками владения компьютером.

Цель работы – реализовать приложение для составления расписания занятий, которое позволит учесть все требования и пожелания, предъявляемые к нему, оптимизировать его и сократить время, потраченное на его составление.

Предмет исследования – учебный план и аудиторный фонд университета.

Объектом исследования является расписание учебного процесса для университета.

Рабочая гипотеза заключается в предположении, что организация занятий в течение учебного года может быть оптимизирована путем применения программного продукта, реализованного в рамках научной работы. Процесс оптимизации будет выполняться на основе автоматизации выбора либо аудиторий, либо времени. А также вывод результата в удобном для пользователя формате.

Практическая значимость состоит в возможности применения данного приложения для решения реальных задач, возникающих перед организациями. Оно призвано помочь заинтересованным в составлении расписания лицам в упрощении данного процесса.

## **Глава 1. Постановка задачи**

Целью данной работы является реализация клиент-серверного приложения для составления расписания занятий в учебных заведениях, которое позволит учесть все требования и пожелания, предъявляемые к нему, сократить время, потраченное на его составление, сделать процесс создания расписания более удобным.

Для достижения поставленной цели необходимо решить следующие задачи **:**

* Определить способы составления корректного расписания.
* Спроектировать БД, содержащую все необходимые данные.
* Сформулировать основные методы, доступные для администратора системы.
* Разработать программный продукт для автоматизации составления расписания.
* Произвести тестирование разработанного продукта.

Требования к данной системе прописаны в документе «Техническое Задание», которое прилагается к курсовой работе.

## **Глава 2. Анализ предметной области**

### **2.1 Глоссарий**

Для конкретного понимания темы работы составлен небольшой справочник узкоспециализированных слов. Ниже представлены понятия и определения к ним.

1. Расписание - вид календаря (т.е. упорядоченность по времени), для которого указана информация о предстоящих (планируемых) событиях. Обычно представляется в виде таблицы.
2. Аудитория - помещение для проведения лекций. Не оборудовано компьютерной и специализированной техникой.
3. Лаборатория - помещение для проведения практических работ. Оборудовано специализированной техникой (компьютерами).
4. Преподаватель - человек, педагог, который занимается преподаванием какого-либо предмета. Работник вуза.
5. Читаемый курс (предмет) – учебная дисциплина, преподаваемая педагогом.
6. Лабораторные занятия (практические) - занятия в отдельных группах. Проведение учащимися по заданию преподавателя опытов с использованием спец оборудования. Проводятся в лабораториях.
7. Потоковая лекция - лекция, читаемая для нескольких групп одновременно. Проводятся в аудиториях.
8. Группа студентов - учётная единица. Характеризуется названием (номером) и относительно постоянным составом.
9. Учебная пара - аудиторное занятие, которое происходит в виде двух совмещённых академических часов.

### **2.2 Аналоги решения задачи**

Можно рассмотреть такое программное средство как 1С: Автоматизированное составление расписания: Университет. Данное программное обеспечение предназначается для решения задач по успешной автоматизации составления расписания в ВУЗах. Система позволяет составлять расписания учебных занятий в смешанном, ручном и автоматическом режиме. Продукт предназначен для решения задач автоматизированного составления учебных расписаний и оперативного управления помещениями в ВУЗах.   Вид программы 1С:Автоматизированное составление расписания: Университет представлен на рисунке 1.

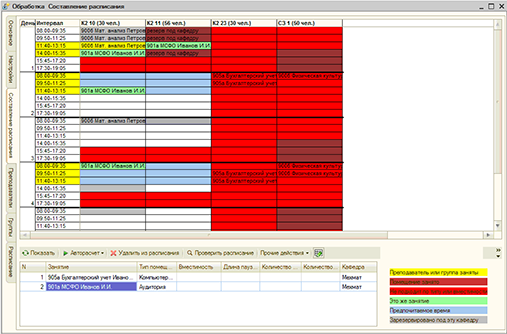


Рисунок 1 - Интерфейс программы 1С:Автоматизированное составление расписания

Следующее программное средство, которое используется для составления расписания - экспресс-расписание ВУЗ. Экспресс-расписание ВУЗ - программа для автоматизации составления расписания учебных занятий в ВУЗах. Программа автоматически составляет основное расписание, позволяет вести ежедневные изменения расписания, учет выполненных часов, формирует разнообразные отчеты. Вид данной программы представлен на рисунке 2.

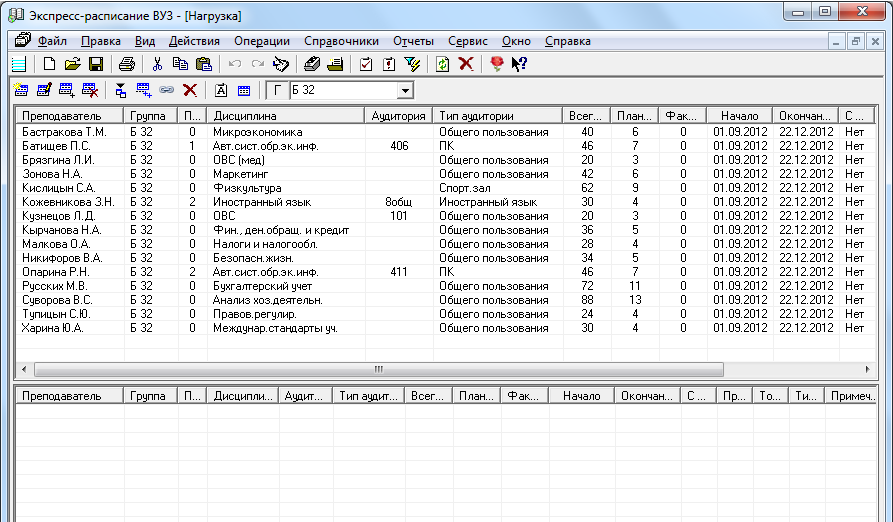


Рисунок 2 - Интерфейс программы Экспресс-расписание ВУЗ

Минус этих программных средств заключается в их стоимости. 1С: Автоматизированное составление расписания. Университет.1 стоит около семидесяти тысяч, а Экспресс-расписание ВУЗ – двадцать тысяч. А также их недостаток в сложном понимании для неопытного пользователя ПК.

### **2.3 Методология**

Если использовать модель водопада, то это последовательный переход от одной стадии проекта к другой. Только после успешного завершения этапа определения требований от разработчика, можно перейти к проектированию ПО, а затем уже к реализации. Например, после реализации уже не осуществляется переход назад к определению требований, что сокращает сроки выполнения проекта.

Но эта модель не подходит для нашего проекта, так как разработчики и заказчик не могут ознакомиться с вариантами системы во время разработки, и видят результат только в самом концу. Тем самым, заказчики не могут повлиять на процесс создания системы.

Поэтому в разработке нашего программного обеспечения мы используем методику Agile. Методика Agile позволяет создавать качественное ПО. Agile легко реагирует на изменения со стороны заказчика, и разработчики могут быстро подстроиться под требования заказчика.

### **2.4 Моделирование бизнес процессов составления расписания**

Для отображения функций системы, а также имеющихся связей между функциями и внешней средой была построена диаграмма IDEF0. Центральным бизнес-процессом является составление расписания, который на схеме отображается в виде функционального блока на рисунке 3.

На вход (стрелки слева) подаются списки дисциплин, групп, преподавателей и аудиторий, которые преобразуются в процессе.

Вход сверху - это поток управления в виде учебного плана. Он не преобразуется в процессе, но нужен для его выполнения. На основе учебного плана будут производиться все бизнес процессы, продемонстрированные на рисунке 4.

Вход снизу - это механизм. Он показывает людей, технические средства, информационные системы, при помощи которых бизнес процесс реализуется. В данном случае - механизмом является администратор.

На выходе системы получается составленное учебное расписание.

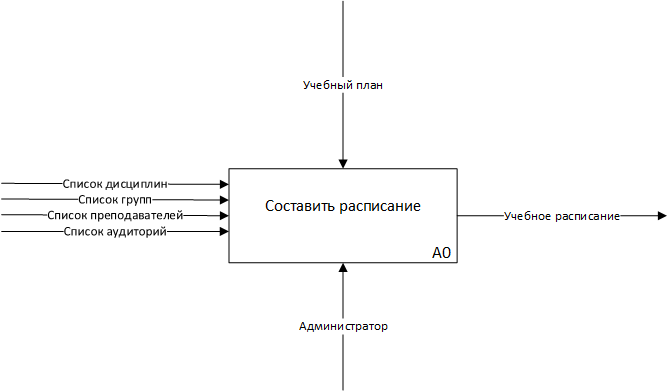


Рисунок 3 - Функциональный блок A0

На рисунке 4 представлена декомпозиция блока А0. Сложный процесс составления расписания разбит на составляющие его функции для понимания того, какие бизнес-процессы должна решать система и как они должны быть связаны между собой.

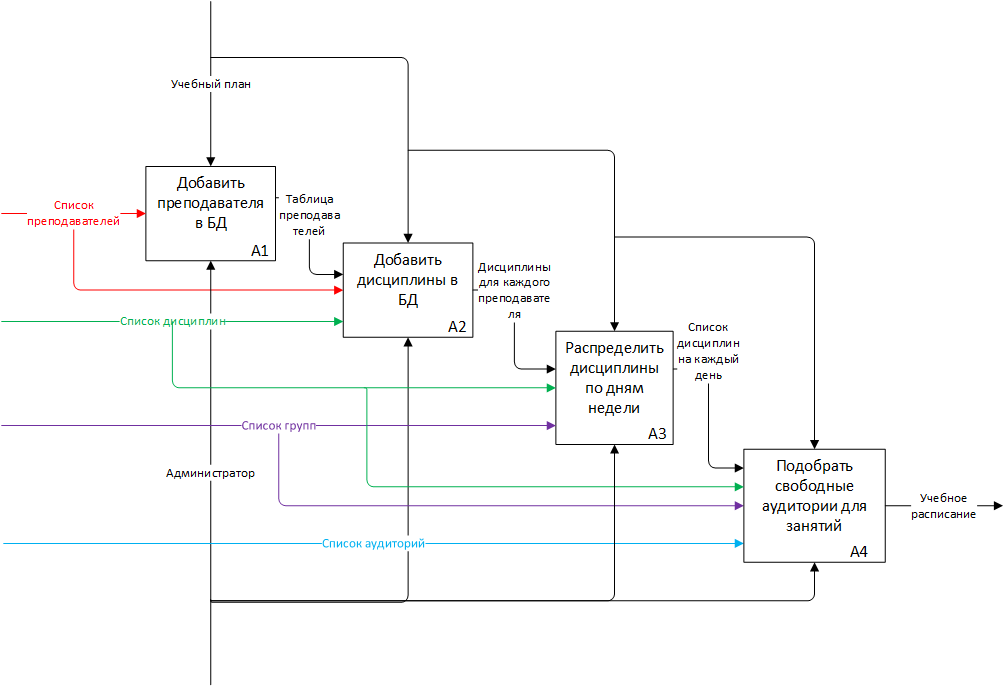


Рисунок 4 - Декомпозиция А0

### **2.5 Анализ требований**

В данной работе общая идея проекта состоит из реализации некоторого количества заданий. Для уточнения требований рассмотрим диаграмму вариантов использования на рисунке 5.

На диаграмме видно, что есть такой актер, как администратор. Он выполняет все функции в системе. Эти функции называются вариантами использования. Для актера возможны следующие варианты использования:

1. Получить список преподавателей. Включает в себя получение списка предметов для каждого преподавателя, который может редактироваться путем удаления или добавления нового занятия к преподавателю. Расширяется вариантами: добавление преподавателя, удаление преподавателя и изменение преподавателя в списке преподавателей.
2. Получить список всех предметов. Имеет расширения: добавление предмета, изменение предмета и удаление предмета в данном списке.
3. Составить расписание. Включает получение списка аудиторий и расширяется вариантом экспорт расписания в документ Microsoft Exel из приложения в удобную папку.
4. Получить список групп. Включает в себя получение списков предметов для каждой группы.
5. Внести изменения в расписание. Расширяется вариантами: изменить аудиторию, добавить занятие и удалить занятие из расписания.

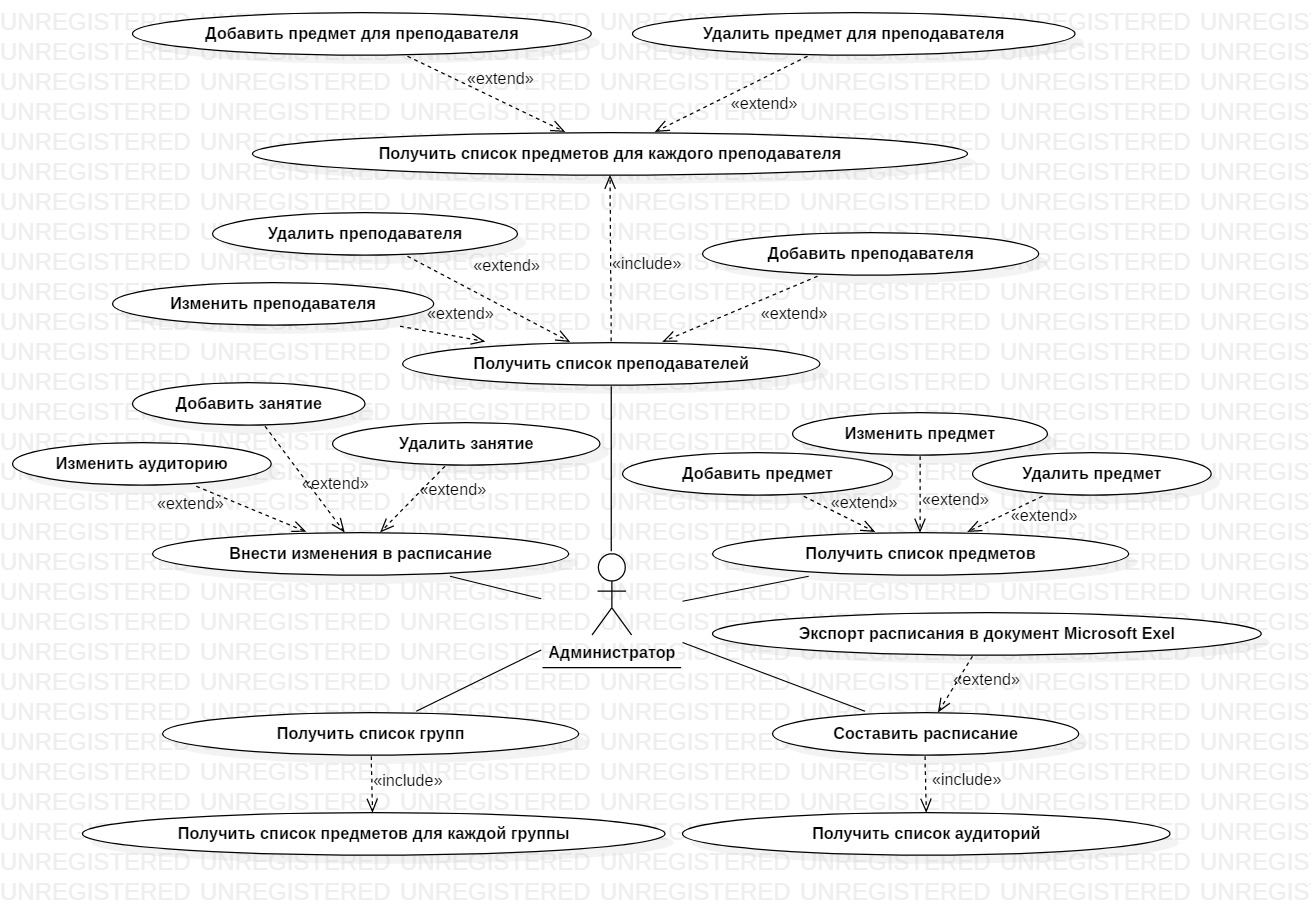
****

Рисунок 5 - Диаграмма вариантов использования

### **2.6 Анализ процесса получения списка аудиторий**

Процесс получения списка аудиторий представлен диаграммой последовательности на рисунке 6.

Данная диаграмма последовательности соответствует варианту использования получить список аудиторий. Диаграмма содержит 3 объекта “Приложение”, “Сервер” и “База данных” и одно действующее лицо - “Пользователь”. Для того чтобы получить список аудиторий, пользователю необходимо перейти на вкладку аудитории. Делается HTTP-запрос методом GET на сервер. Сервер обращается к БД и отвечает на запрос. В результате ответа приложение отображает пользователю список аудиторий.

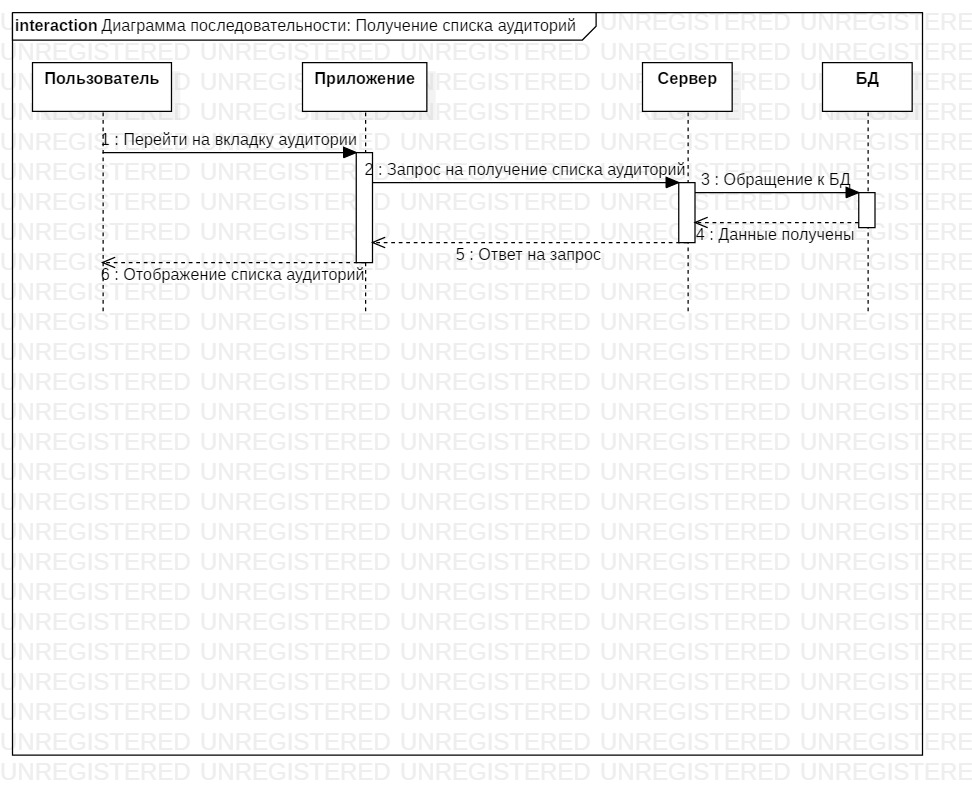


Рисунок 6 - Диаграмма последовательности “Получение списка аудиторий”

Диаграмма коммуникации для описанного процесса представлена на рисунке 7. Она явно указывает отношения между объектами в рамках процесса получения списка аудиторий, а время как отдельное измерение не используется (применяются порядковые номера вызовов). Сообщения в диаграмме коммуникации используются такие же, как и в диаграмме последовательности для получения списка аудиторий.

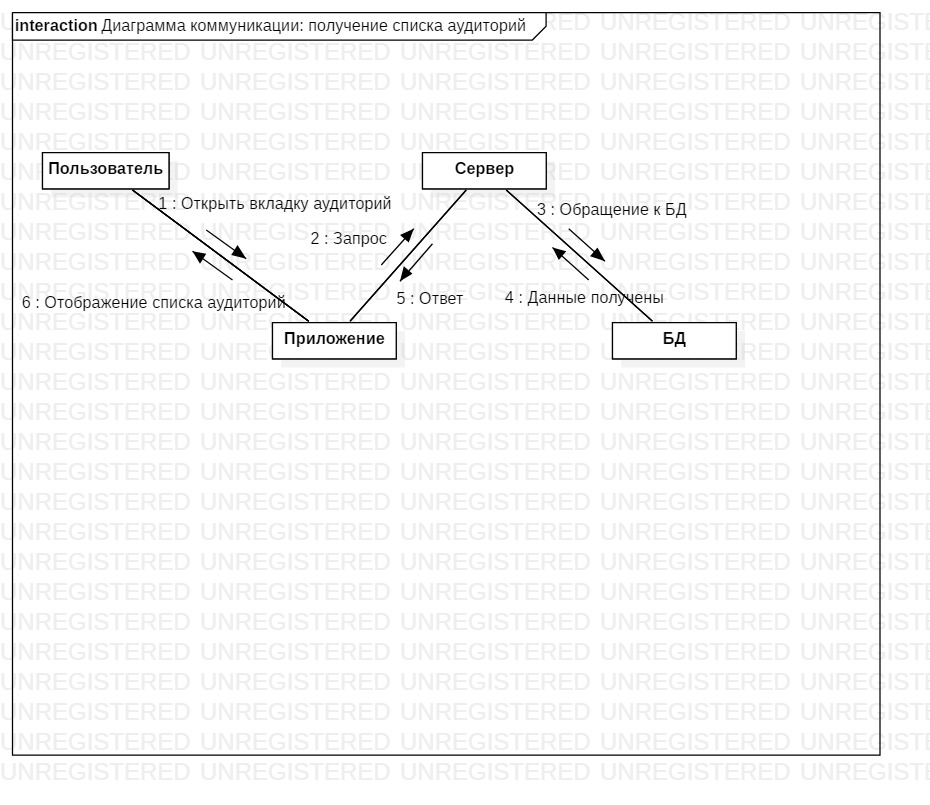


Рисунок 7- Диаграмма коммуникации “Получение списка аудиторий”

Аналогично, как показано на данных диаграммах, происходят процессы для получения списков предметов, преподавателей и групп.

### **2.7 Анализ процесса изменения преподавателя**

На рисунках 8 и 9 представлен процесс редактирования преподавателя пользователем диаграммами последовательности и коммуникации соответственно.

Для начала пользователю необходимо перейти на вкладку Преподаватели. Приложение сделает HTTP-запрос на сервер на получение списка преподавателя, сервер обратится к БД и ответит на запрос. В результате, приложение отобразит пользователю список преподавателей. Для того чтобы отредактировать какого-либо преподавателя, пользователю нужно выбрать его из списка, для того, чтобы он появился в форме для ввода. Далее, пользователь вносит изменения, причем приложение допустит к вводу только корректные символы(а именно символы русского или английского алфавита), и нажимает на кнопку Изменить. Приложение делает HTTP-запрос методом PUT на сервер, введенные изменения сохраняются в БД. Сервер формирует ответ на запрос и приложение отображает введенные изменения в списке преподавателей.

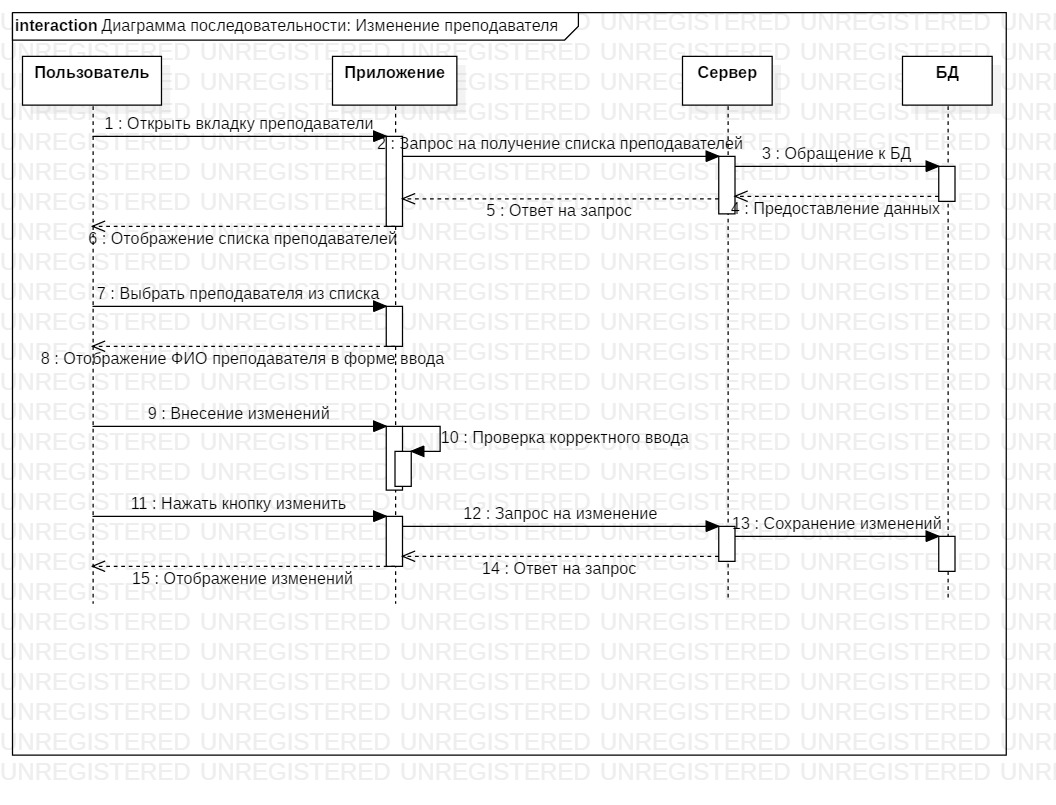


Рисунок 8 - Диаграмма последовательности “Изменение преподавателя”

Ниже на рисунке 9 представлена диаграмма коммуникации для процесса изменения преподавателя. Она имеет те же сообщения, что и на диаграмме последовательности для изменения преподавателя, поэтому они здесь продублированы не будут.

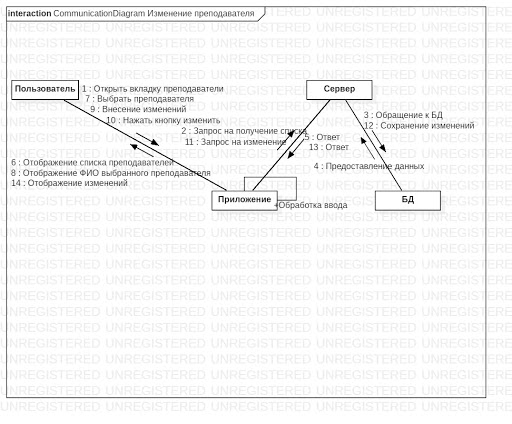


Рисунок 9 - Диаграмма коммуникации “Изменение преподавателя”

Процессы изменения других имеющихся списков в базе данных происходят аналогично, как показано на данных диаграммах.

### **2.8 Анализ процесса изменения расписания**

На рисунке 10 представлен процесс изменения расписания диаграммой последовательности.

Действующим лицом является пользователь. Для изменения расписания пользователю нужно перейти на вкладку Расписание. Приложение делает HTTP-запрос методом GET на сервер для получения расписания, сервер обращается к БД на получение списка, формирует ответ на запрос, и приложение отображает существующее в БД расписание. Затем пользователь выбирает ячейку с группой, для которой нужно изменить расписание. Далее, действия пользователя могут разделиться в соответствии с двумя вариантами использования: добавить занятие в выбранную ячейку или удалить его оттуда.

Если пользователю необходимо добавить занятие в пустую ячейку, он выбирает занятие из выпадающего списка занятий, приложение формирует HTTP-запрос методом POST на добавление занятия и сервер сохраняет введенное занятие в БД. Сервер отвечает на запрос и приложение отображает введенное занятие в таблице расписания для выбранной группы.

Если пользователю нужно удалить занятие, он нажимает кнопку удалить, приложение формирует HTTP-запрос методом DELETE к серверу, сервер обращается к БД для удаления и отвечает на запрос. В результате приложение очищает выбранную ячейку.

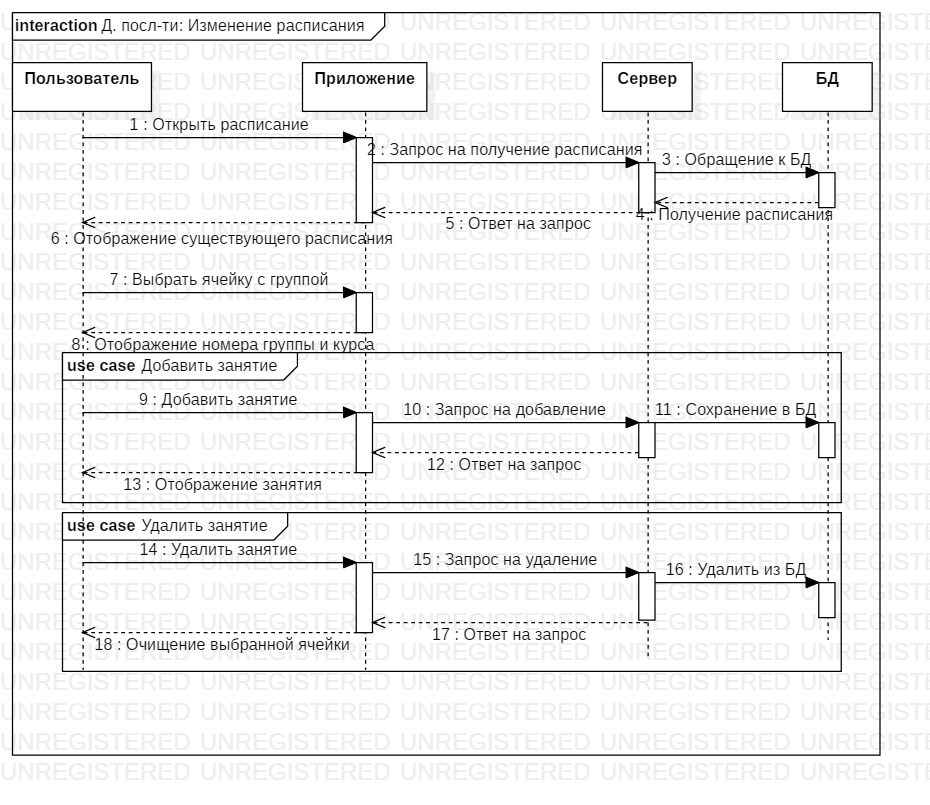
****

Рисунок 10 - Диаграмма последовательности “Изменение расписания”

### **2.9 Анализ работы приложения**

На рисунке 11 представлен процесс составления расписания. Изначально ожидается действие от пользователя, затем происходит операция сформировать расписание путем отправления HTTP-запроса на сервер для получения данных. Если пользователь формирует новое расписание, то приложение получает список групп, предметов, преподавателей, аудиторий, предметов для определенных преподавателей и предметов для групп. Но если расписание уже существует, то пользователь может его отредактировать, путем добавления или удаления занятий. После составления расписания, пользователь может экспортировать составленное расписание в Microsoft Excel. Затем можно вернуться в режим ожидания действия или выйти из системы.

### 

Рисунок 11 - Диаграмма деятельности

На рисунке 12 показан процесс смены состояний в результате составления расписания. Изначально приложение находится в состоянии ожидания действий от пользователя. Затем происходит отправление HTTP-запроса на сервер для получения данных. Если расписание существует, то можно перейти в состояние редактирования, либо в формирование расписания, где система находится в стадии получения списка предметов, преподавателей и аудиторий, предметов для определенных преподавателей и предметов для групп. Заключительная стадия в системе - составление расписания, путем добавления или удаления нужного занятия. Далее пользователь может экспортировать составленное расписание в Microsoft Excel. Затем можно вернуться в режим ожидания действия или выйти из системы.



Рисунок 12- Диаграмма состояния

На рисунке 13 показана диаграмма объектов для нашего приложения. Составление расписания требует взаимодействия нескольких различных объектов. Сценарий начинается с объекта класса Расписания. В свою очередь, объект Расписания может вызвать объекты других классов(Предмет, Группа, Аудитория, Преподаватель).

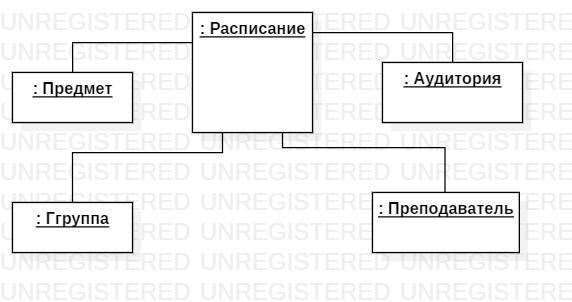


Рисунок 13- Диаграмма объектов

## **Глава 3. Архитектура приложения**

### **3.1 Анализ структуры данных**

Для детального изучения работы можно рассмотреть ER-диаграмму, представленную на рисунке 14.

Модель «Сущность - Отношение» используется для проектирования баз данных с целью удаления всякой избыточности данных.

На диаграмме видно, что между сущностями выстраиваются различные отношения взаимодействия. Опишем связи между сущностями.

1. Преподаватель - Дисциплина. Степень связи m:m (многие ко многим). Класс принадлежности сущности «Преподаватель» к сущности «Дисциплина» обязательный, и наоборот. Каждый преподаватель может преподавать 1 или несколько дисциплин, каждая дисциплина может преподаваться одним или несколькими преподавателями.

2. Преподаватель – Пара. Степень связи 1:1. Класс принадлежности сущности «Преподаватель» к сущности «Пара» обязательный, и наоборот. Каждый преподаватель может преподавать только 1 пару в определенное время, а у каждой пары может быть только 1 преподаватель.

3. Дисциплина – Пара. Степень связи 1:1. Класс принадлежности сущности «Дисциплина» к сущности «Пара» обязательный, и наоборот. Каждая дисциплина преподается вна одной паре, а на каждой паре преподается только одна дисциплина.

4. Группа – Пара. Степень связи m:1 (многие к 1). Класс принадлежности сущности «Группа» к сущности «Пара» обязательный, и наоборот. На каждой паре могут присутствовать несколько групп, но у каждой группы может быть только 1 пара в определенное время.

5. Аудитория – Пара. Степень связи 1:m (1 ко многим). Класс принадлежности сущности «Аудитория» к сущности «Пара» обязательный, и наоборот. В каждой аудитории может проходить несколько пар, но каждая пара проходит в одной аудитории.

6. Дни\_недели – Пара. Степень связи 1:1. Класс принадлежности сущности «Дни\_недели» к сущности «Пара» обязательный, и наоборот. В каждый день недели проходит только одна конкретная пара, и одна конкретная пара проходит в определенный день недели.

7. Дисциплина – Тип\_занятия. Степень связи m:1 (многие к 1). Класс принадлежности сущности «Дисциплина» к сущности «Тип\_занятия» обязательный, и наоборот. У каждой дисциплины может быть только 1 тип занятия (лекция/практика), а у каждого типа занятия может быть несколько дисциплин.

8. Аудитория – Тип\_занятия. Степень связи m:1 (многие к 1). Класс принадлежности сущности «Аудитория» к сущности «Тип\_занятия» обязательный, и наоборот. В каждой аудитории проводятся занятия определенного типа, но для каждого типа занятий может быть несколько аудиторий.

9. Группа - Дисциплина. Степень связи m:m (многие ко многим). Класс принадлежности сущности «Группа» к сущности «Дисциплина» обязательный, и наоборот. У каждой группы есть много дисциплин, как и у каждой дисциплины есть много групп.



Рисунок 14 - ER-диаграмма системы

### **3.2 Визуализация элементов системы**

Для визуализации элементов и компонентов системы рассмотрим диаграмму развертывания, представленную на рисунке 15.

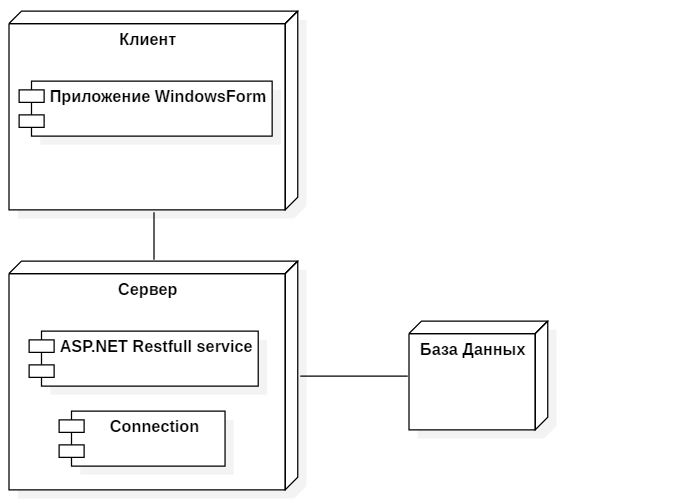
****

Рисунок 15 - Диаграмма развертывания

На клиенте находится приложение WindowsForm, c которого будут посылаться HTTP-запросы на сервер 4-мя методами:

* GET - получение данных с сервера.
* POST - добавление новых данных на сервере.
* PUT - модификация существующих данных на сервере
* DELETE - удаление данных на сервере.

На сервере запросы обрабатываются с помощью ASP.NET Restfullservice. А класс Connection хранит подключение к базе данных.

### **3.3 Проектирование пользовательского интерфейса**

Схема взаимодействия страниц приложения представлена на рисунке 18.

При запуске приложения на стартовой странице будет отображено существующее на данный момент расписание в виде таблицы(оно может быть и пустое). Есть возможность добавления занятия в расписание и его редактирование. На этой же странице можно экспортировать расписание в формат Microsoft Exсel, нажав кнопку экспорт. На главной странице будет возможность переключения на другие страницы.

Нажав на вкладку Группы откроется соответствующая страница. На ней можно будет получить список групп и предметы для каждой группы. Здесь же производится редактирование списка группы (удаление/ изменение/ добавление группы в список). Отредактированный список сохраняется в базе данных.

При нажатии вкладки Преподаватели откроется страница, на которой можно получить список преподавателей и предметов для каждого из них. Также будет возможность редактирования каждого из списков и сохранение изменений в базе данных.

Перейдя по вкладке Дисциплины откроется соответствующая страница со списком дисциплин, который при необходимости можно будет изменить.

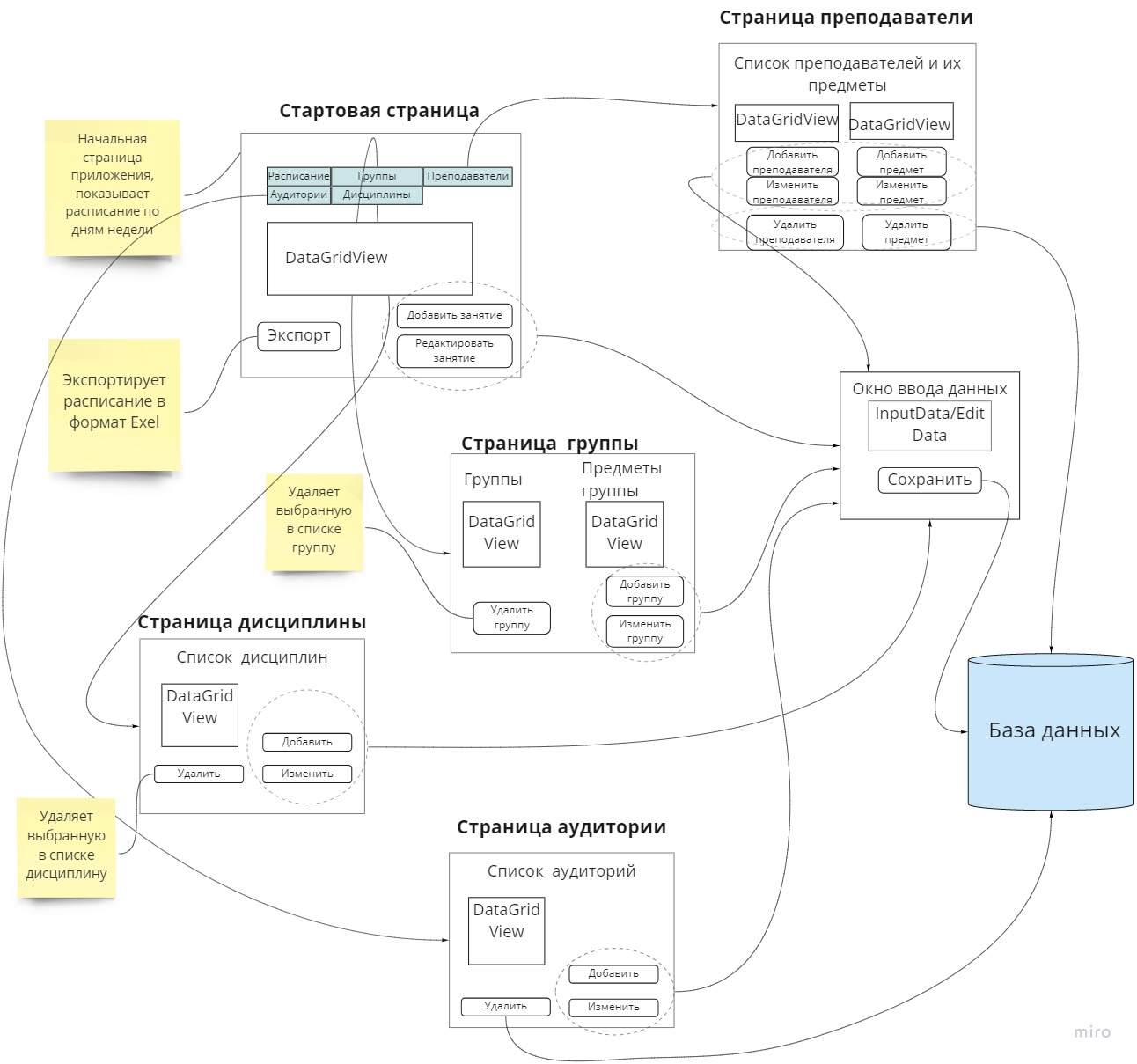


Рисунок 18- Архитектура приложения

# **Глава 4. Реализация приложения**

### **4.1 Средства реализации**

В качестве языка программирования был выбран С#. Причинами выбора данного языка для разработки нашей системы явилось наличие большого количества готовых конструкций, которые упрощают написание кода и делают его более читаемым. С# является языком с С-подобным синтаксисом, он будет понятен для большинства разработчиков, что будет плюсом в случае необходимости расширения системы.

В качестве архитектуры приложения был выбран стиль REST. Она очень проста в использовании. По виду пришедшего запроса сразу можно определить, что он делает, не разбираясь в форматах (в отличие от SOAP, XML-RPC). Данные передаются без применения дополнительных слоев, поэтому REST считается менее ресурсоемким. Особенность REST в том, что сервер не запоминает состояние пользователя между запросами. И в каждом запросе передаётся информация, идентифицирующая пользователя и все параметры, необходимые для выполнения операции.

В качестве платформы для веб-разработки используется Web-API ASP.NET 2 — это платформа для создания веб-API-интерфейсов на основе .NET Framework. Она позволяет создавать HTTP-сервисы поверх .NET Framework.Веб-API 2 поддерживает маршрутизацию с помощью атрибутов*.*

Для написания приложения использовалась среда разработки MicrosoftVisualStudio, основанная на платформе .NET Framework.

В MicrosoftVisualStudio интуитивно понятный интерфейс, с мощными эффективными возможностями для кодирования. Поддержка C# в VisualStudio обеспечивается с помощью полнофункционального редактора кода, компилятора, конструкторов, удобного отладчика.

В процессе разработки проекта были построены различные виды диаграмм с использованием такого программного продукта, как StarUML и MicrosoftVisio.

В качестве системы управления данными была выбрана MySQL – это простая, надежная и эффективная система управления информацией, использующая традиционный язык SQL. К серверу MySQL можно подключаться из любой точки земного шара через программу-клиент, установленную на компьютере пользователя. Но при этом СУБД MySQL снабжена развитой системой защиты от несанкционированного доступа. Среди плюсов также можно отметить большие возможности обработки. Количество строк в таблицах может достигать 50 млн.

В качестве инструмента для визуального проектирования БД используется MySQLWorkbench, так как она более удобна в использовании, чем, например, DBDesigner и другие. Этот продукт позволяет создавать новую БД, вносить изменения в уже существующую, выполнять запросы в реальном времени и наглядном виде. Также все связи между таблицами показываются на ER-диаграмме, что тоже очень удобно.

### **4.2 Реализация кода**

Приложение создавалось на основе представленной на рисунке 16 диаграмме классов.

В состав приложения входят такие классы:

1. Queries. Класс, в котором реализованы все запросы, посылаемые к базе данных.

2. Connection. Класс, отвечающий за соединение с базой данных.

3.ConnectionInstance. Класс, отвечающий за взаимодействие между Controllers и Connection.

4.Controllers. Включаетвсебяклассы: DayInWeekController, DisciplineController, DisciplineGroupController, DisciplineLecturerController, GroupController, LecturerController, LessonController, RingController, RoomController, TypeOfDisciplineController, TypeOfRoomController. Эти классы взаимодействуют с запросами.

5. WebApiApplication. Класс, для запуска сервера.

6. MainFrom. Данный класс представляет собой класс основной формы взаимодействия пользователя с программой. В нем содержится список всех занятий. Каждое занятие сочетает в себе преподавателя, дисциплину, которую данный преподаватель ведет, аудиторию, в которой проходит занятие, группу, которая присутствует на занятии и время проведения.

7. Application. Класс, содержащий в себе всю логику приложения.

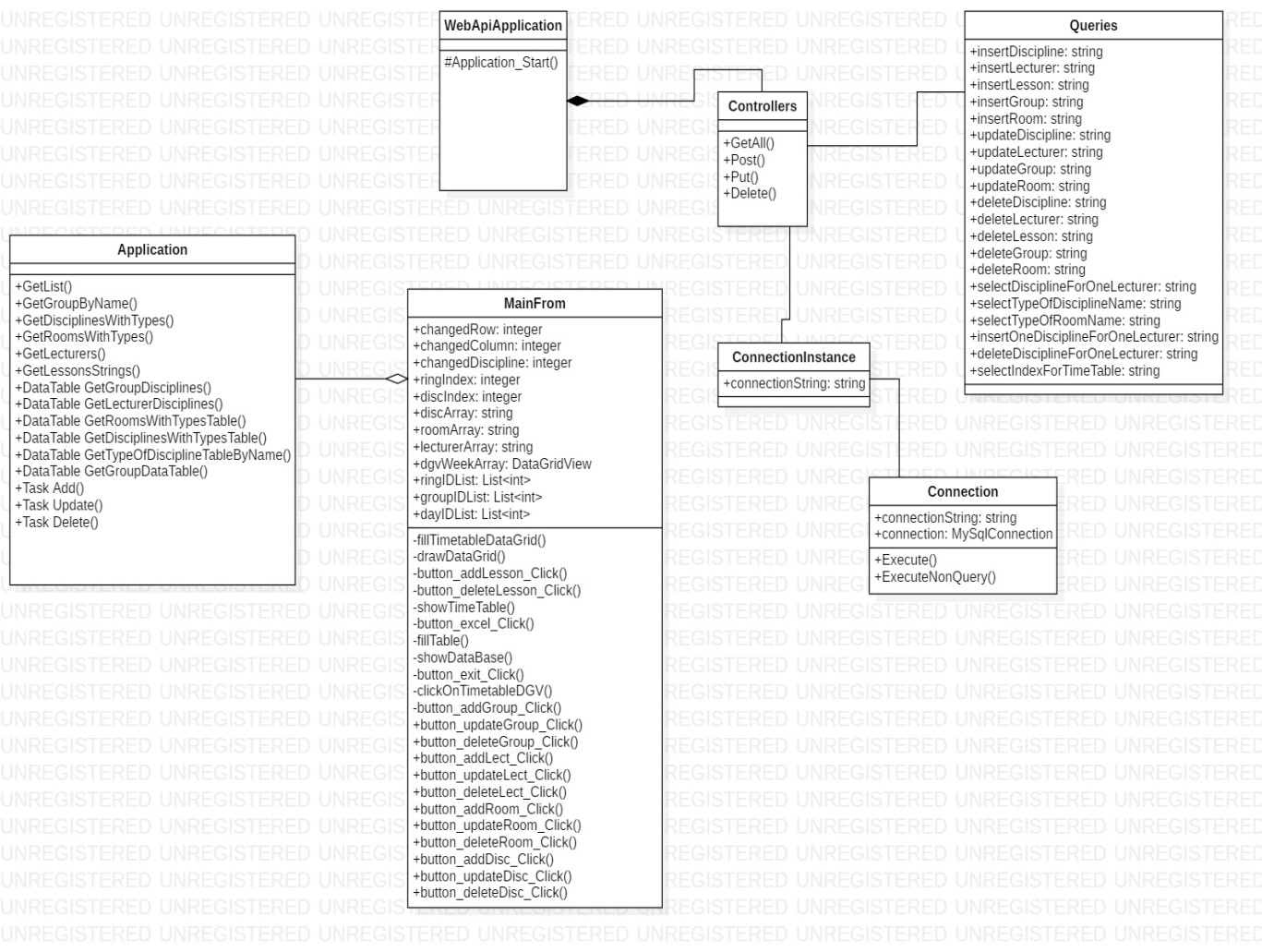
****

Рисунок 19 - Диаграмма классов

На первом этапе разработки приложения была создана база данных в СУБД MySQL. В дальнейшем, база данных была перенесена с локального сервера на den1.mysql5.gear.host. Для подключения базы данных к приложению использовался код, представленный на листинге 1.

Листинг 1 – Подключение БД.

stringconnectionString = "server=den1.mysql5.gear.host;user=facultytimetabl1;database=facultytimetabl1;password=olya123.";

publicMySqlConnection connection;

public Connection()

{

connection = newMySqlConnection(connectionString);

connection.Open()

}

В реализации программы был выделен класс (Queries), который содержит текст всех запросов к базе данных. Они необходимы для корректного вывода данных и дальнейшего их использования, представлены на листинге 2.

Листинг 2 - Запросы.

publicclassQueries {

#region select queries

publicstaticstringselectdayinweeks = "select \* from dayinweek";

publicstaticstringselectdisciplines = "select \* from discipline";

publicstaticstringselectdisciplinegroups = "select \* from discipline\_group";

publicstaticstringselectdisciplinelecturers = "select \* from discipline\_lecturer";

publicstaticstringselectgroups = "select \* from grouper";

publicstaticstringselectlecturers = "select \* from lecturer";

publicstaticstringselectlessons = "select \* from lesson";

publicstaticstringselectrings = "select \* from ring";

publicstaticstringselectrooms = "select \* from room";

publicstaticstringselecttypeofdisciplines = "select \* from typeofdiscipline";

publicstaticstringselecttypeofrooms = "select \* from typeofroom";

#endregion

#region insert queries

publicstaticstringinsertlecturer = "insert into lecturer (lecturer\_surname,lecturer\_name,lecturer\_patronymic) values (@lectsurname,@lectname,@lectpatro)";

publicstaticstringinsertroom = "insert into room (room\_number, type\_id) values (@roomnumber,@typeroomid)";

publicstaticstringinsertdiscipline = "insert into discipline (discipline\_name, type\_id) values (@disciplinename,@typediscid)";

publicstaticstringinsertgroup = "insert into groups (group\_name) values (@groupname)";

publicstaticstringinsertdisciplinelecturer = "insert into discipline\_lecturer (discipline\_id, lecturer\_id) values (@discid,@lectid)";

publicstaticstringinsertdisciplinegroup = "insert into discipline\_group (discipline\_id, group\_id) values (@discid,@groupid)";

publicstaticstringinsertlesson = "insert into lesson (discipline\_id,lecturer\_id,ring\_id,room\_id,group\_id,day\_id) values (@discid,@lectid,@ringid,@roomid,@groupid,@dayid)";

#endregion

#region update queries

publicstaticstringupdatelecturer = "update lecturer set lecturer\_surname = @lectsurname,lecturer\_name = @lectname,lecturer\_patronymic = @lectpatro where lecturer\_id = @lectid";

publicstaticstringupdateroom = "update room set room\_number = @roomnumber,type\_id = @typeroomid where room\_id = @roomid";

publicstaticstringupdatediscipline = "update discipline set discipline\_name = @disciplinename,type\_id = @typediscid where discipline\_id = @discid";

publicstaticstringupdategroup = "update groups set group\_name = @groupname where group\_id = @groupid";

publicstaticstringupdatedisciplinelecturer = "update discipline\_lecturer set discipline\_id = @newdiscid where lecturer\_id = @lectid and discipline\_id =@olddiscid";

publicstaticstringupdatedisciplinegroup = "update discipline\_group set discipline\_id = @newdiscid where group\_id = @groupid and discipline\_id =@olddiscid";

#endregion

#region delete queries

publicstaticstringdeletelecturer = "delete from lecturer where lecturer\_id = @lectid";

publicstaticstringdeleteroom = "delete from room where room\_id = @roomid";

publicstaticstringdeletediscipline = "delete from discipline where discipline\_id = @discid";

publicstaticstringdeletegroup = "delete from groups where group\_id = @groupid";

publicstaticstringdeletedisciplinelecturer = "delete from discipline\_lecturer where discipline\_id = @discid and lecturer\_id = @lectid";

publicstaticstringdeletedisciplinegroup = "delete from discipline\_group where discipline\_id = @discid and group\_id = @groupid";

publicstaticstringdeletelesson = "delete from lesson where ring\_id=@ringid and day\_id=@dayid and group\_id=@groupid";

#endregion

}

}

Ранее были рассмотрены диаграммы последовательностей для трех видов деятельности. На рисунке 8 речь шла об изменении преподавателя. Этот механизм реализован следующим образом, как показано на листинге 3. В этой реализации используется метод Update(), который посылает HTTP-запросы с методом PUT.

Листинг 3 – Изменение преподавателя.

privateasyncvoidbutton\_updateLect\_Click(object sender, EventArgs e)

{

try

{

if (tb\_lectName.Text != ""&&tB\_lectSurname.Text != ""&&tB\_lectPatro.Text != "")

{

try

{

awaitapp.Update(

new Lecturer

{

Lecturer\_name = tb\_lectName.Text, Lecturer\_surname = tB\_lectSurname.Text,

Lecturer\_patronymic = tB\_lectPatro.Text

}, "lecturer", myIndex);

lecturerArray = app.GetLecturers();

refreshArrayComboBox(lecturerArray, comboBox\_Lecturer);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

showDataBase();

tb\_lectName.Clear();

tB\_lectSurname.Clear();

tB\_lectPatro.Clear();

}

else

{

MessageBox.Show(@"Выберитепреподавателя!");

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

На рисунке 10 речь шла об изменении расписания. Этот механизм реализован следующим образом и показан на листинге 4.

Листинг 4 - Удалениезанятия.

privateasyncvoidbutton\_deleteLesson\_Click(object sender, EventArgs e)

{

try

{

if (label\_selectedGroup.Text != ""&&label\_selectedRing.Text != "")

{

intchangedDay = tabControl\_Days.SelectedIndex + 1;

changedRow = dgvWeekArray[changedDay - 1].SelectedCells[0].RowIndex;

ringIndex = Convert.ToInt32(dgvWeekArray[changedDay - 1].Rows[changedRow].HeaderCell.Value);

changedColumn = dgvWeekArray[changedDay - 1].SelectedCells[0].ColumnIndex;

stringgroupName = (dgvWeekArray[changedDay - 1].Columns[changedColumn].HeaderCell.Value).ToString();

intgroupID = app.GetGroupByName(groupName).Group\_ID;

awaitapp.Delete("lesson", app.GetLessonByDesc(ringIndex, groupID, changedDay).Lesson\_ID);

dgvWeekArray[changedDay - 1].Rows[ringIndex - 1].Cells[groupID - 1].Value = "";

showTimeTable();

}

else

{

MessageBox.Show(@"Заполнитегруппуиномерурока!");

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

Этот сложный процесс можно разобрать на более мелкие составляющие. То есть выбираем необходимую ячейку в представленной таблице. Проверяем на корректность выбора. Применяем метод Delete, отправляющий HTTP-запросы с методом DELETE. Этот метод позволяет в базе данных найти по идентификатору данное занятие и удалить. Метод showTimeTable() заполняет datagridview измененными данными. Если пользователь указал неверные данные, то программа выведет ошибку и попросит повторить действие еще раз.

Вся имеющаяся логика находится в классе Application.cs. На листинге 5 представлен процесс получения списка преподавателей, путем вызова HTTP-запросы с методом GET, который используется при изменении в 3 листинге.

Лисниг 5- Получение списка преподавателей.

publicstring[] GetLecturers()

{

vartmp = GetList<Lecturer>();

var result = new List<string>();

foreach (var lecturer intmp)

{

result.Add(lecturer.Lecturer\_surname + " " + lecturer.Lecturer\_name + " " + lecturer.Lecturer\_patronymic);

}

returnresult.ToArray();

}

Переходя непосредственно к внешнему виду программного продукта, показаны следующие рисунки. Интерфейс приложения сделан в строгом стиле, но понятном любому пользователю.

### **4.3 Реализация интерфейса**

На рисунке 20 представлено изображение расписания для каждого дня недели для каждой группы. Здесь можно добавлять и удалять занятие у учебной группы. Выбирать для занятия лектора, подбирать аудиторию.

Важно заметить наличие кнопки «Экспорт», позволяющей экспортировать данные из таблицы в формат приложения MicrosoftExcel.

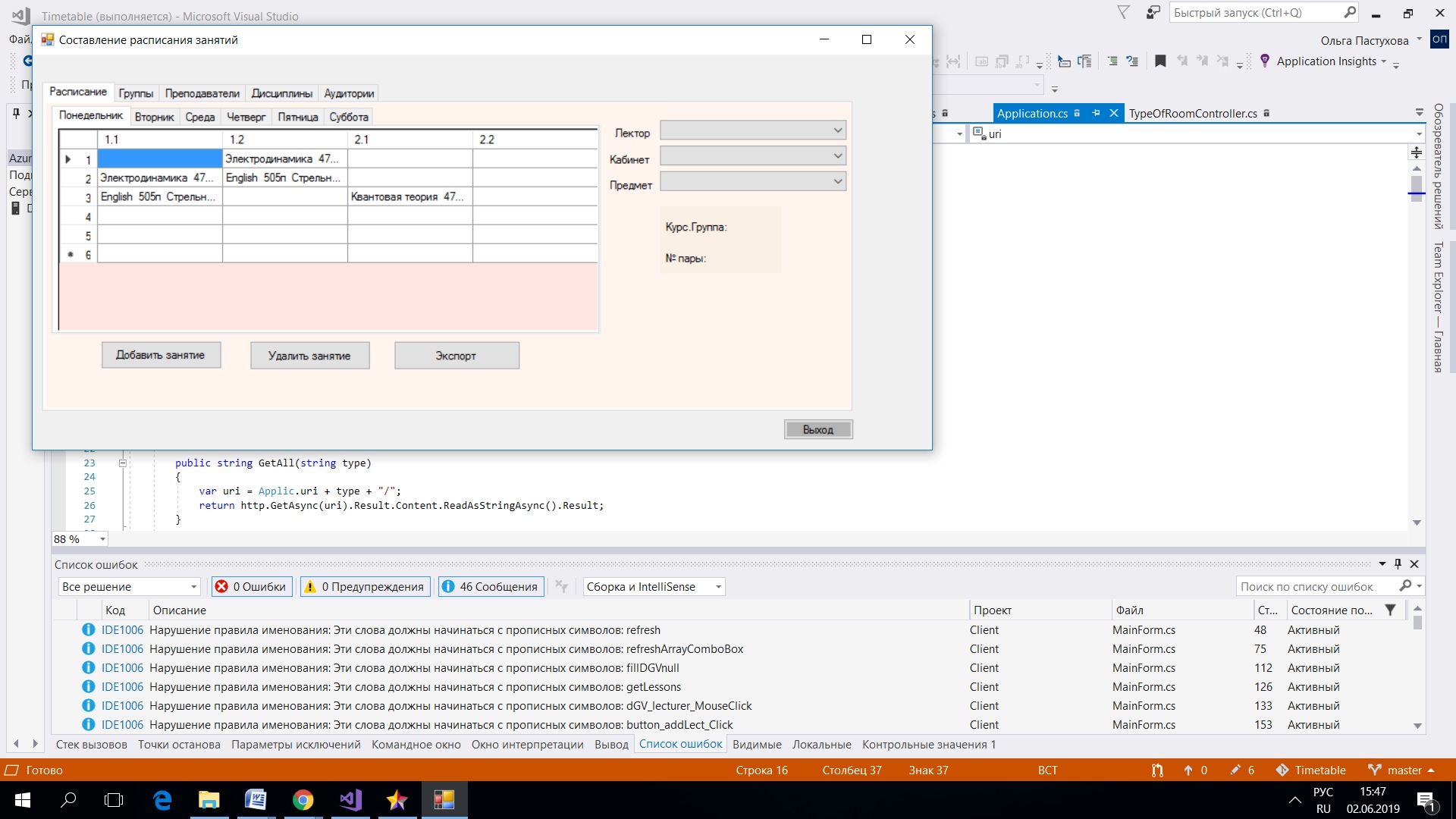


Рисунок 20- Расписание

На рисунке 21 представлено изображение экрана, когда запрашивается список групп. Если мы выбираем в таблице слева любую группу, то в таблице справа отображаются те предметы, которые у них есть. Ниже есть форма для добавления группы, также там можно изменять уже существующее данные. Справа внизу есть область для добавления предмета, который будет преподаваться у выбранной группы. Предметы можно выбрать из уже существующих, которые имеются в базе данных.

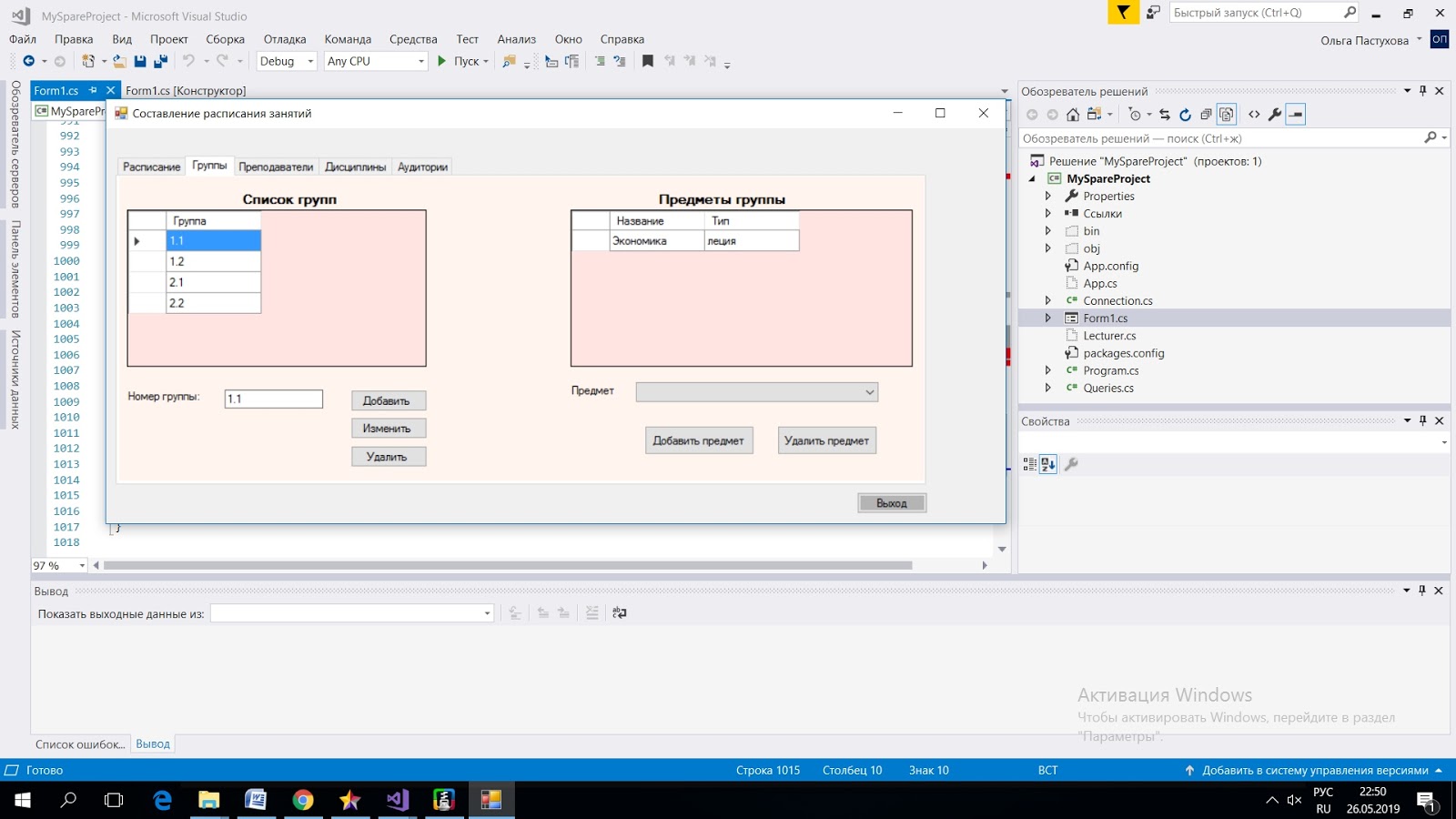
****

Рисунок 21- Группы.

На рисунке 22 представлено изображение экрана, когда запрашивается список преподавателей. Если мы выбираем в таблице слева любого преподавателя, то в таблице справа отображаются те предметы, которые он ведет. Ниже есть форма для добавления преподавателя, также там можно изменять уже существующее данные через форму. Справа внизу есть область для добавления предмета, который будет вести выбранные преподаватель. Предметы можно выбрать из уже существующих, которые имеются в базе данных.

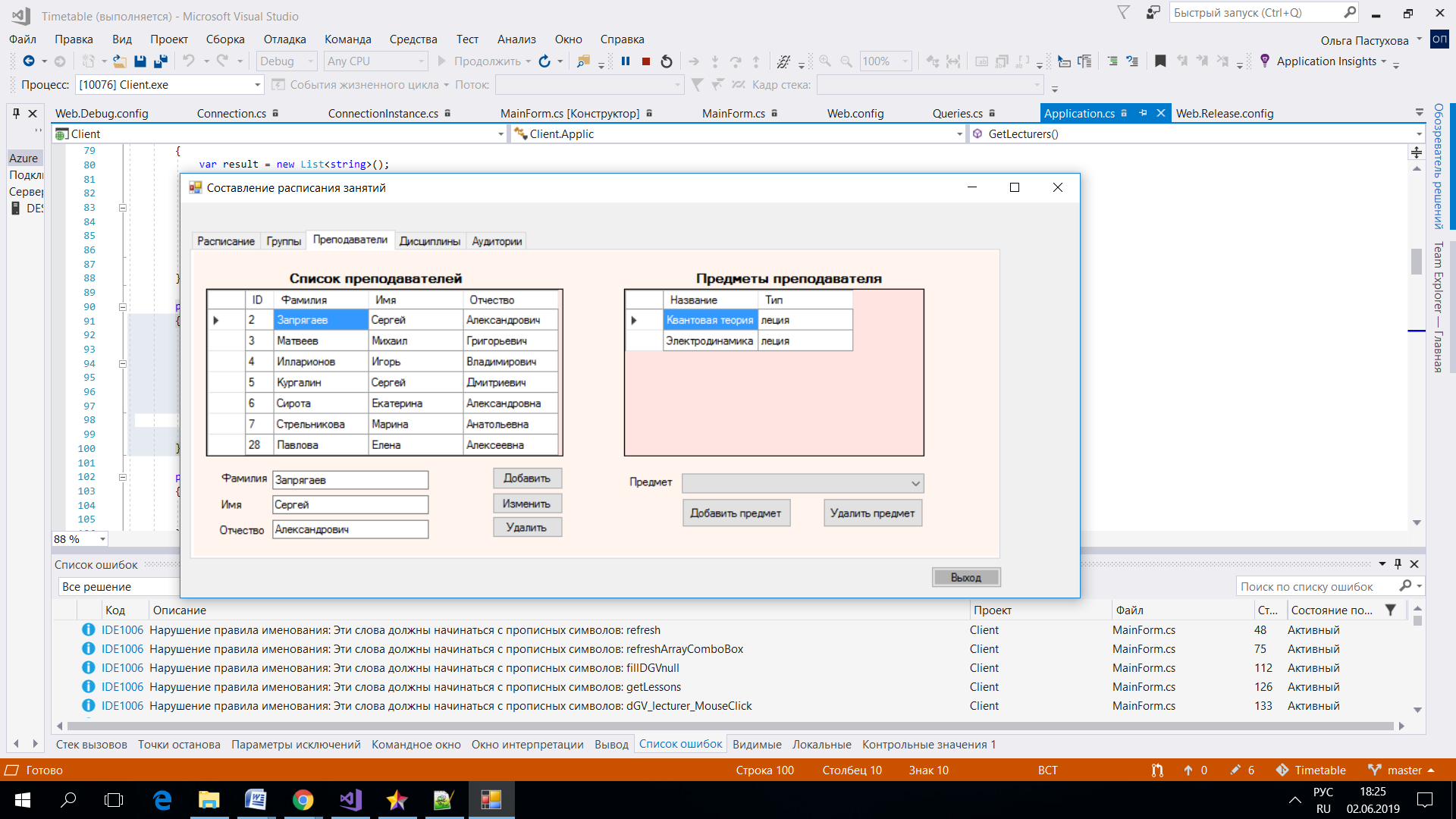
****

Рисунок 22- Преподаватели

На рисунке 23 показан снимок экрана приложения, когда необходимо посмотреть уже существующие предметы. Добавить новый, удалить или изменить предмет можно через соответствующие кнопки.

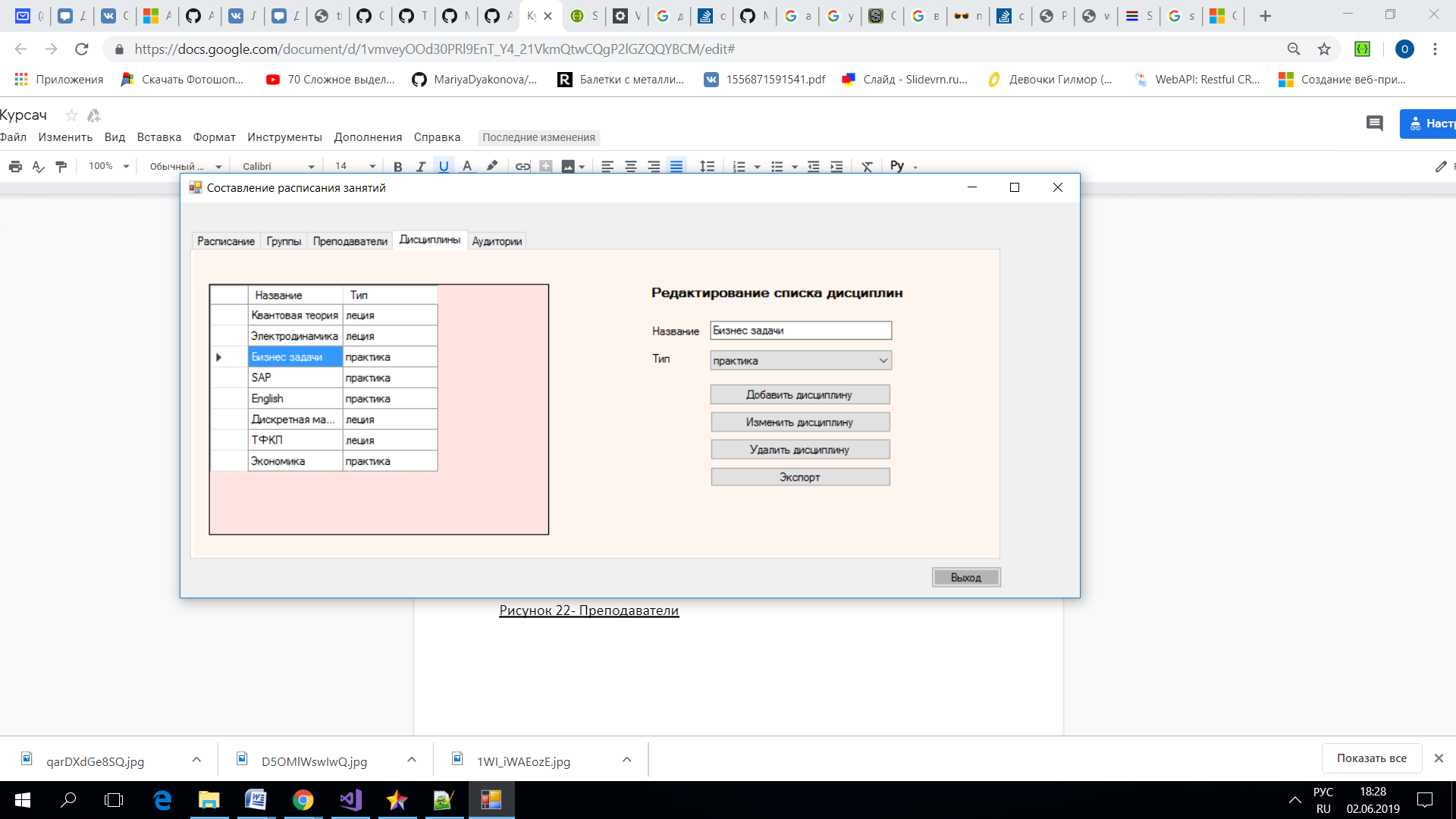


Рисунок 23- Дисциплины

На рисунке 24 показан снимок экрана приложения, когда необходимо посмотреть уже существующие аудитории. Добавить новую, удалить или изменить аудиторию можно через соответствующие кнопки.

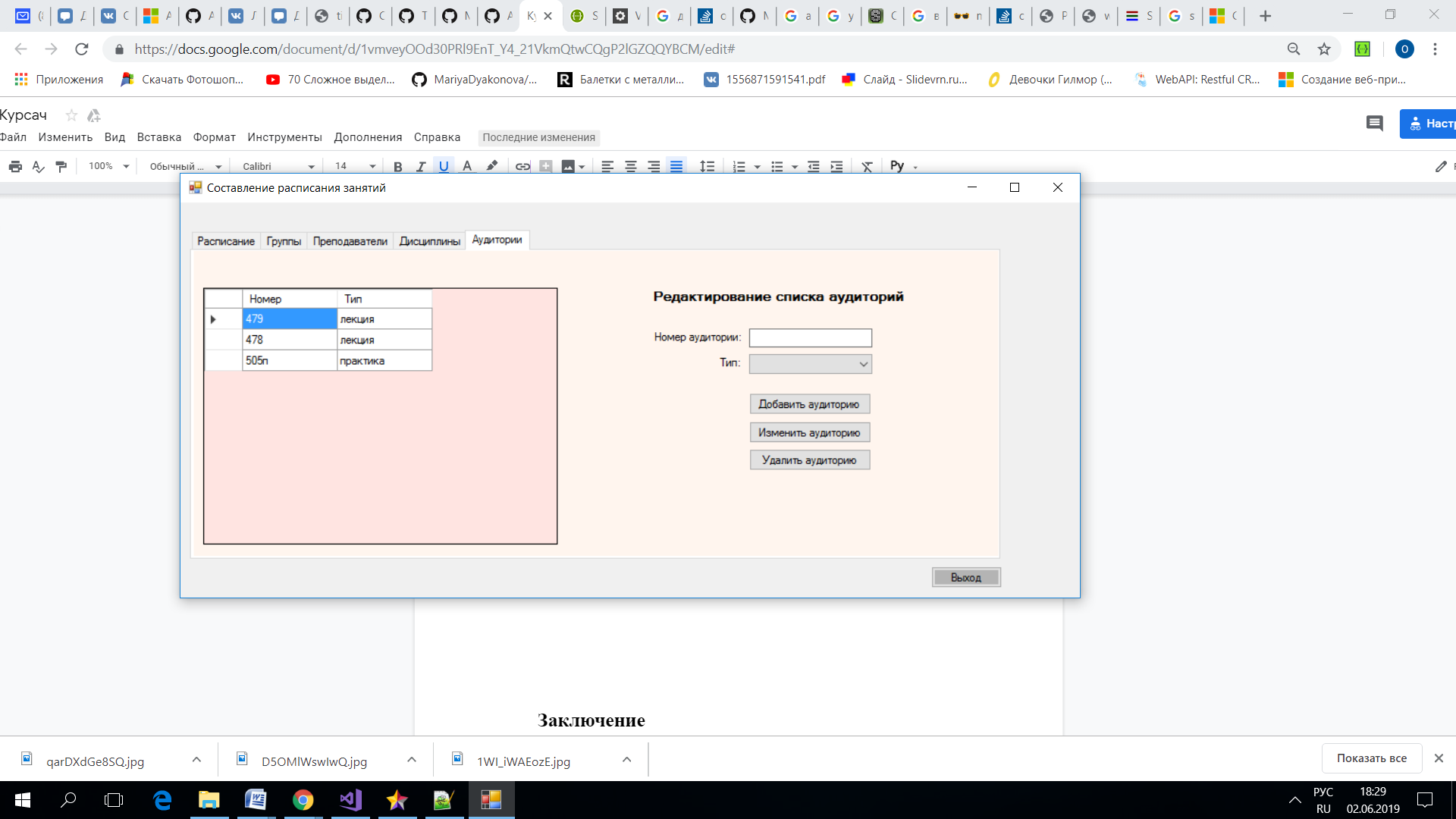
****

Рисунок 24 -Аудитории

# **Глава 5. Тестирование**

Тестирование было проведено в соответствии с документом «План тестирования», который прилагается к данной курсовой работе.

Основные виды тестирования, которые были проведены:

1. Ручное тестирование – тестовые циклы были произведены вручную, с последующей ручной фиксацией результатов по каждому тесту.

2. Smoketesting – поверхностное тестирование всех модулей приложения на предмет их работоспособности. Касаемо данной системы, было проверено соединения с базой данных при запуске приложение.

3. Sanitytesting – тестирование работы конкретных функций системы для доказательства того, что система работает согласно требованиям в Техническом Задании. В рамках данного теста были проверены возможности внесения изменения в базу данных, удаление из нее и добавление новых данных в базу. Важно отметить, что данное тестирование было еще и позитивным, так как была проверена возможность внесения изменений при корректном вводе данных. Негативное тестирование было проведено отдельно.

4. Usabilitytesting – тестирование того, насколько легко конечный пользователь сможет освоить систему. Здесь проверялось наличие сообщений-подсказок для пользователя, при неправильных действиях с его стороны.

5. Negativetesting – тестирование поведении системы, при попытке ввода некорректных данных. Было протестировано поведение системы, при попытке ввода в соответствующие поля для ФИО преподавателя или названия дисциплины цифр или иных символов, кроме букв русского или английского алфавита.

6. E2E testing – тестирование системы от начала до конца так, как с ней должен работать пользователь.

Следует отметить, что в процессе разработки было проведено еще и регрессионное тестирование. Необходимость его проведения обусловлена изменениями в коде во время разработки.

Тест - кейсы представлены отдельным документом. В результате тестирования оказалось более 95 % пройденных тест-кейсов. Не пройденные тест-кейсы не являются критическими и могут быть исправлены в случае необходимости доработки или расширения системы.

Таким образом, ИС готова к ее презентации заказчику.

## **Заключение**

Подводя итог проделанной работе, можно сказать, что разработан новый продукт, готовый к эксплуатации и облегчающий работу пользователя. Теперь любой администратор может без труда составить расписание на учебный год, либо изменить его. Также не вызывает никаких проблем действия, связанные с переносом занятий во времени или преобразованием документа в Книга Microsoft Excel.

Проблема, заключавшаяся в существовании многих нюансов связанных с составлением учебного расписания, была решена путем оптимизации и автоматизации деятельности.

Цель работы, поставленная в самом начале, достигнута путем выполнения ряда задач:

1. Изучен способ формирования учебного расписания.

2. Составлена база данных, которая содержит все необходимые данные.

3. Сформулированы основные методы, доступные для администратора системы.

4. Разработан программный продукт для автоматизации составления расписания.

Таким образом, была реализована программа, решающая реальные задачи, стоящие перед руководителями, университетов, колледжей и школ. Тем самым подтверждается практическая значимость работы.

## **Список источников**

1. Рихтер Дж. CLR via C#. Программирование на платформе Microsoft.NET Framework 4.5 на языке C# / Дж. Рихтер; пер. с англ. – 4-е изд. –СПб.: Питер, 2015. – 896с.
2. Аткинсон Л. MySQL. Библиотека профессионала / Л. Аткинсон. – М.: Диалектика, 2002. – 624с.
3. Гольцман ВMySQL 5.0 / В. Гольцман. – СПб.: Питер, 2010. – 253с.
4. Шилдт Г. C# 4.0 Полное руководство / Г. Шилдт; пер. с англ. И. Берштейном. – М.: Вильямс, 2015. – 1056с.
5. Проектирование баз данных. СУБД MicrosoftAccess. Учебное пособие / Н. Гринченко [и др.]. – М.: Горячая линия – Телеком, 2004. – 238с.
6. Создание API-интерфейсов RESTful с помощью веб-API ASP.NET – URL: <https://docs.microsoft.com/ru-ru/aspnet/web-api/overview/older-versions/build-restful-apis-with-aspnet-web-api> (дата обращения 20.05.2018).
7. Язык программирования C#. Классика ComputersScience / А. Хейлсберг [и др.]. -4-е изд. –СПб.: Питер, 2011. – 784с.
8. 6. Ликнесс Дж. Приложения для Windows на С# и XAML / Дж. Ликнесс. – СПб.:Питер, 2013. – 368с.
9. 7. Проектирование баз данных. СУБД MicrosoftAccess. Учебное пособие / Н. Гринченко [и др.]. – М.: Горячая линия – Телеком, 2004. – 238с.
10. 8. SQL. Полное руководство / Дж. Грофф [и др.]. – 3-е изд. – М.: Вильямс, 2015. – 959с.
11. 9. MySQL. Оптимизация производительности / Б. Шварц [и др.]. – СПб.: Символ-Плюс, 2010. – 768с.
12. RESTFulApi контроллеры в .NET MVC 4 – URL: <https://habr.com/ru/post/164945/>(дата обращения 15.05.2018)
13. Учебник. Создание веб-API с помощью MVC ASP.NET Core – URL: <https://docs.microsoft.com/ru-ru/aspnet/core/tutorials/first-web-api?view=aspnetcore-2.2&tabs=visual-studio> (дата обращения 17.05.2018)
14. C# 6.0. Справочник / Д. Албахари [и др.]. -6-е изд. –М.: Вильямс, 2016. – 1040с.
15. Работа с базой данных MySQL с использованием C#. – URL: <https://o7planning.org/ru/10517/working-with-mysql-database-using-csharp> (дата обращения 17.04.2018)