Зарегистрировано №\_\_\_\_\_\_

«\_\_\_»\_\_\_\_\_\_\_\_\_\_2020 г.

\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись (расшифровка подписи)

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ**

**ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»**

**(НИУ «БелГУ»)**

ИНСТИТУТ ИНЖЕНЕРНЫХ И ЦИФРОВЫХ ТЕХНОЛОГИЙ

**Кафедра математического и программного обеспечения информационных систем**

**РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ ОБСЛУЖИВАНИЕ РАБОТЫ КОНФЕРЕНЦИи С ИСПОЛЬЗОВАНИЕМ КЛИЕНТ-СЕРВЕРНОЙ ТЕХНОЛОГИИ**

Курсовая работа  
по дисциплине «Базы данных»  
студентки очной формы обучения  
направления подготовки 02.03.02 «Фундаментальная информатика и информационные технологии».  
Профиль подготовки: Супервычисления  
2 курса группы 12001801  
Капустина Виктора Сергеевича

| ***Допущена к защите***  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2020 г.  \_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Подпись (расшифровка подписи) | Научный руководитель:  Асс. Е.В.Петрова |
| --- | --- |
| ***Оценка\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2020 г.  \_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Подпись (расшифровка подписи) |  |

БЕЛГОРОД 2020

ПЛАН

КУРСОВОЙ РАБОТЫ

Введение.

1. Проектирование информационной системы
   1. Проектирование информационного обеспечения
      1. Системный анализ предметной области
      2. Инфологическое проектирование БД
      3. Даталогическое проектирование БД
   2. Проектирование программного обеспечения
      1. Выбор СУБД
      2. Выбор инструментальных средств для создания ПО
      3. Проектирование пользовательского интерфейса
2. Разработка информационной системы
   1. Разработка информационного обеспечения
      1. Физическое проектирование БД
      2. Программирование на стороне SQL-сервера
      3. Разработка бизнес-логики на стороне SQL-сервера
   2. Разработка программного обеспечения
      1. Создание Win-приложения
      2. Создание Web-приложения
3. Тестирование информационной системы
   1. Пользовательский интерфейс
      1. Интерфейс Win-приложения
      2. Интерфейс Web-приложения

Заключение

Список использованных источников

Приложение

Исполнитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ студент Капустин В.С.

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ассистент Петрова Е.В.

**Содержание**

[Введение 4](#_heading=h.gjdgxs)

[Глава 1. Проектирование информационной системы 6](#_heading=h.1fob9te)

[1.1 Проектирование информационного обеспечения 6](#_heading=h.2et92p0)

[1.2 Проектирование программного обеспечения 10](#_heading=h.tyjcwt)

[Глава 2. Разработка информационной системы 14](#_heading=h.3dy6vkm)

[2.1 Разработка информационного обеспечения 14](#_heading=h.1t3h5sf)

[2.2 Разработка программного обеспечения 20](#_heading=h.4d34og8)

[Глава 3. Тестирование информационной системы 24](#_heading=h.2s8eyo1)

[3.1 Пользовательский интерфейс 24](#_heading=h.17dp8vu)

[Заключение 33](#_heading=h.26in1rg)

[Список используемых источников 34](#_heading=h.35nkun2)

<http://172.23.64.64/ftp/00/1200180142/Web/index.php>.

# Введение

В наши дни информационные технологии входят в очень многие сферы деятельности человека. Они очень тесно вошли в жизнь человека помогая ему во многих делах. Одним из ключевых преимуществ, которые дают эти технологии это автоматизация, которая является несомненно важным и полезным инструментом прогресса. Она позволяет добиться более эффективного управления информацией, т.е. достигать больших высот с меньшим количеством затрат. Основные системы обработки информации являются важной категорией программ на сегодняшний день и основываются они на базах данных.

**База данных -** организованная в соответствии с некоторыми правилами поддерживаемая в памяти компьютера совокупность данных, которая отражает какую-либо конкретную область, также база данных используется для удовлетворения информационных потребностей пользователя.

На сегодняшний хранится просто немыслимое количество информации и для беспроблемного доступа к ней необходимо строить эффективные базы данных. Одной из важных черт эффективности является быстрый доступ к хранящимся данным. Для достижения этой быстроты требуется провести систематизацию данных, один из важнейших этапов в бизнес-процессах. Без этого невозможно представить такие важнейшие бизнес-процессы, как документооборот, делопроизводство и добиться экономии средств компании и рабочего времени сотрудников.

Таким образом можно сделать вывод, что базы данных весьма актуальны в наши дни и просто необходимы для любого предприятия.

В рамках курсовой работы необходимо провести системный анализ и по полученным разработать информационную систему для автоматизации обслуживания работы конференции, состоящей из серверного блока и клиентских приложений для более простого взаимодействия с хранящимися данными.  
 **Целью работы** является разработка информационной системы обслуживания конференции использованием клиент-серверной технологии и двух клиентских приложений.

Исходя из цели, были поставлены следующие **задачи**:

* Анализ предметной области.
* Проектирование информационной системы.
* Выбор подходящей СУБД
* Реализация спроектированной БД.
* Проектирование и разработка web-приложения.
* Проектирование и разработка wim-приложения.
* Тестирование

Курсовая работы состоит из введения, трех глав и параграфов и заключения.

Во введении описаны цели курсовой работы, а также её задачи.

Первая глава работы посвящена проектированию информационной системы, в том числе проектированию информационного и программного обеспечения.

Во второй главе описывается процесс разработки информационной системы, описанной в первой главе.

В третьей главе тестируется созданная информационная система. В том числе отражен пользовательский интерфейс Win и Web приложения. Приведены копии экрана, показывающие, функционирование системы

# Глава 1. Проектирование информационной системы

Информационная система — это взаимосвязанная совокупность методов, средств и персонала, используемых для хранения, обработки, поиска обеспечение и распространение информации для достижения цели управления. Другими словами — это позволит выполнить конкретные поступающие запросы на получение информации в рамках разработанной базы данных.

Любой проект имеет свои цели, в данном же случае требуется разработать информационную систему для автоматизации бизнес-процессов конференций и в некоторой степени даже человеческие отношения, которой нужно обеспечить требуемую функциональность и степень адаптации к изменяющимся условиям ее функционирования, требуемое время реакции системы на запрос, готовность и доступность системы для обработки запросов пользователей и простоту эксплуатации.  
 По классификации информационные системы разделяют на настольные и распределенные. К настольным относятся те, в которых все компоненты (БД, СУБД, клиентские приложения) находятся на одном компьютере. К распределенным относятся только те, в которых компоненты распределены по нескольким компьютерам. В свою очередь, стоит отметить, что распределенные информационные системы разделяют на:

1) Файл-серверные

2) Клиент-серверные

Данная работа является примером второй из этих категорий. То есть сама база данных и СУБД находятся на сервере, а на рабочих станциях находятся клиентские приложения.

## 1.1 Проектирование информационного обеспечения

В данной главе будет рассмотрен процесс проектирования информационного обеспечения, который необходим для функционирования информационной системы.

***Системный анализ предметной области****.* Требуется разработать информационную систему для автоматизации обслуживания работы конференции. Система должна иметь режим ведения системного каталога, отражающего перечень аспектов, по которым работает конференция. Они могут иметь уникальный внутренний номер и полное наименование.   
При работе с системой Организаторы должны иметь возможность решать следующие задачи:

* Регистрировать новых спикеров и распределять их по секциям
* Распределять секции по аудиториям
* Назначать модераторов на секции
* Просматривать системный каталог, т.е. перечень всех секций, докладов, модераторов и спикеров.
* Для выбранного спикера получить его: Фамилию, Имя, Отчество, Название доклада, Организация
* Для выбранного спикера получить список всех секций, в которых он принимает участие

В таблицу базы данных заносятся следующие сведения о спикере:

* Уникальный код спикера
* Фамилия
* Имя
* Отчество
* Название доклада
* Организация

Каждый спикер может участвовать в нескольких секциях. Также для помощи спикерам и организаторам в проведении самой конференции работают модераторы. Она имеют следующие характеристики:

* Уникальный код модератора
* Фамилия
* Имя
* Отчество

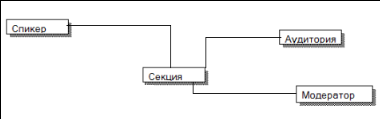
Каждый модератор прикреплен к своей секции и модерирует одну или несколько аудиторий. Аудитория состоит из следующих сведений:

* Уникальный код аудитории
* Номер
* Количество мест
* Код спикера, моделирующего данную аудиторию

Каждая секция конференции может содержать сведения из нескольких аспектов и характеризуется следующими параметрами:

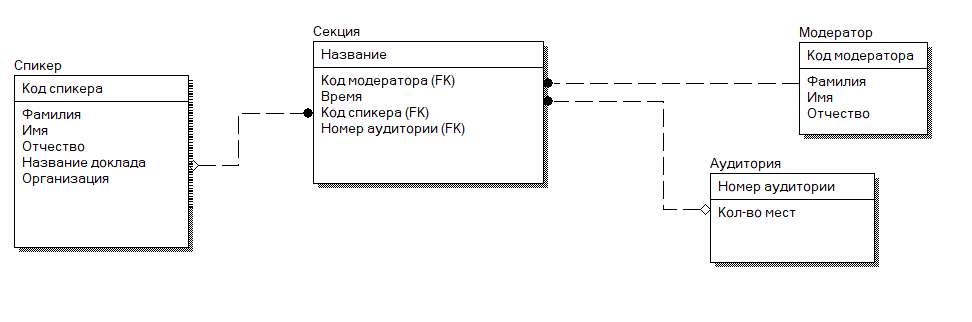
* Название
* Предметной область секции
* Уникальный код Спикера выступающий на данной секции
* Уникальный код Аудитории в которой расположилась секция
* Уникальный код Модератора моделирующего данную секцию
* Время работы секции

***Инфологическое (концептуальное) проектирование БД*** — это процесс построения семантической модели предметной области. Такая модель создаётся без ориентации на какую-либо конкретную СУБД и модель данных. Подразумевается, что это будет информационная модель наиболее высокого уровня абстракции. Термины «семантическая модель», «концептуальная модель» и «инфологическая модель» являются синонимами. Более того, в этом контексте равноправно могут использоваться слова «модель базы данных» и «модель предметной области, поскольку такая модель является как образом реальности, так и образом проектируемой базы данных для этой реальности. Внешний вид и содержание концептуальной модели базы данных зависит от выбора формального аппарата. Обычно используются графические нотации, подобные ER-диаграммам. Обычно подразумевается, что концептуальная база данных будет включать в себя описание информационных объектов или понятий предметной области и связей между ними, а также описание ограничений целостности.  
 Для проектирования модели БД в рамках данной работы была выбрана программа под названием «Erwin Data Modeler».   
 В ней была составлена инфологическая модель базы данных, которая отражает основные ключевые сущности в системе и их связи между собой. (Рис 1.1)

  
Рисунок 1.1 Инфологическая модель БД

Как показано на Рисунке 1.1, для работы информационной системы необходимо четыре сущности. Далее каждая из этих сущностей будет более детализирована в даталогическую модель, из которой в конечном итоге и получится физическая БД.

***Даталогическое проектирование БД***. Даталогическим проектированием называют организацию данных, выделенных на инфологическом проектировании базы данных в форму, принятую в выбранной СУБД. В данной главе будет рассмотрен процесс даталогического проектирования который происходит путем детализации инфологической модели БД, полученной ранее.  
 При проектировании всей логической структуры базы данных происходит преобразование исходной инфологической модели в модель данных, уже поддерживаемую конкретной СУБД. На Рис. 1.2 представлена даталогическая модель БД, полученная из инфологической путем детализации, все названия на данный момент носят только смысловой характер

.  
Рисунок 1.2 Даталогическая модель БД

Для поддержания БД в устойчивом состоянии используется ряд механизмов, которые получили обобщенное название средств поддержки целостности. Приведение структуры БД в соответствие этим ограничениям – называется нормализацией. Нормализация баз данных заключается в приведении структуры хранения данных к нормальным формам. Всего таких форм существует 8, но часто достаточным является соблюдение первых трех.  
 Даталогическая модель представленная на Рис. 1.2. была приведена к третьей нормальной форме (каждый атрибут отношения должен хранить атомарное значение, каждое отношение (строка в таблице) должно содержать одинаковое количество атрибутов (столбцов); каждый неключевой атрибут зависит только от первичного ключа и не зависят друг от друга). В этом модели также отражено четыре сущности и их взаимосвязи между собой.   
 На этом можно считать процесс проектирования информационной системы успешно завершенным.

## 1.2 Проектирование программного обеспечения

В этой главе рассматривается проектирование программного обеспечения. Под этим подразумевается процесс создания проекта будущей программы. Целью проектирования является определение внутренних свойств системы и детализации её внешних свойств на основе выданных требований к ПО (исходные условия задачи). В главе поэтапно разбирается анализ и выбор СУБД, производится выбор инструментов для создания ПО.

***Выбор СУБД***. Для того чтобы осуществлять выбор, для начала стоит разобраться что представляет собой СУБД. **СУБД** (Система управления базами данных) — совокупность программных и лингвистических средств, обеспечивающих управление, созданием и использованием баз данных(операции выбора, вставки, обновления, удаления данных в ней). Такая система обеспечивает надежность и безопасность хранения, а также целостность данных и предоставляет средства для администрирования БД.

**СУБД** ***PostgreSQL*** - это мощная объектно-реляционная система управления базами данных. Она разрабатывается на протяжении более 15 лет и улучшает архитектуру, чем завоевала репутацию надежной, масштабируемой и интегрированной СУБД.

Postgre доступна к запуску на всех основных платформах, соответствует ACID, имеет поддержку ключей, представлений, объединений, триггеров, и хранимых процедур. Она имеет программный интерфейс приложения для Python, Ruby, Tcl, C/C++, Tcl, Java, ODBC и др. Поддерживает международные кодировки, хранение больших двоичных объектов, включая звук, видео или картинки.  
 Средства обеспечения целостности данной СУБД данных включают составные первичные ключи, внешние ключи с поддержкой запрета и каскадирования изменений/удалений, проверку ограничений (constraints), ограничения уникальности и ограничения на непустые значения.

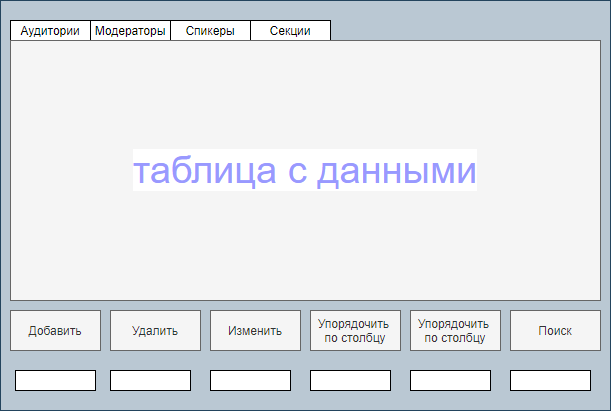
Кроме того, исходный код Postgre имеет лицензию BSD, Что позволяет свободно распространять, модифицировать и использовать СУБД в любой форме.

Таким образом мощность и доступность PostgreSQL делают эту систему одной из лучших СУБД, представленных сегодня на рынке. Она может поддерживать базы данных почти неограниченных размеров и обеспечивает деятельность организации так как её использование несложно для непрофессиональных пользователей. По этим причинам и был сделан выбор СУБД в пользу PostgreSQL.

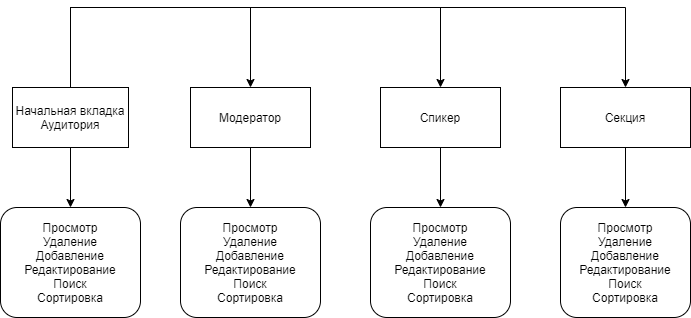
***Выбор инструментальных средств для создания ПО***. Инструментальными средствами могут быть как разные языки программирования, так и программы, с помощью которых осуществляется разработка данного ПО. В данной работе были использованы три категории инструментальных средств.

Первая категория состоит из средств для разработки серверного блока информационной системы: СУБД PostgreSQL и интерфейс для взаимодействия с ней - pgAdmin 4.  
 Вторая категория это технологии для разработки Web-приложения. Для реализации были использованы: Язык программирования PHP, HTMl, CSS.  
 Третья категория это инструментальные средства в виде: Visual Studio 2019 на языке C# и Windows forms для разработки Win-приложения.

***Проектирование пользовательского интерфейса.*** Если рассмотреть определение, то это одна из разновидностей интерфейсов, который является совокупностью средств и методов взаимодействия пользователя с программами компьютерной системы.  
 Проектирование же пользовательского интерфейса представляет собой создание макетов для приложения в графических редакторах. Рис. 1.3

  
Рисунок 1.3 Макет Win-приложения

Помимо этого необходимо составить навигационную карту приложения. Она в отличии от макета представляет не визуализацию будущего приложения, а ее функциональные возможности. На Рис. 1.4 показана карты навигации Win-приложения

  
Рисунок 1.4 Навигационная карта Win-приложения

Данный пользовательский интерфейс будет состоять из 4 вкладок, самой таблицы, кнопки и текстовые поля для взаимодействия с пользователем. Можно будет получить информацию из базы данных, добавить новые объекты, удалить их, внести в них изменения, отсортировать по 2 ключевым столбцам или воспользоваться поиском, если данных будет слишком много.

Web приложение состоит уже не из вкладок, а из Веб-страниц, и переход к таблицам осуществляется с главной страницы, макет которой отображен на Рис 1.6. Нажав на соответствующие кнопки перехода, откроется страница, на которой и появится вся информация с этой базы данных. Там же будет доступно взаимодействие с ними. С каждой страницы можно вернуться на главную, для перехода к отображению другой базы данных.

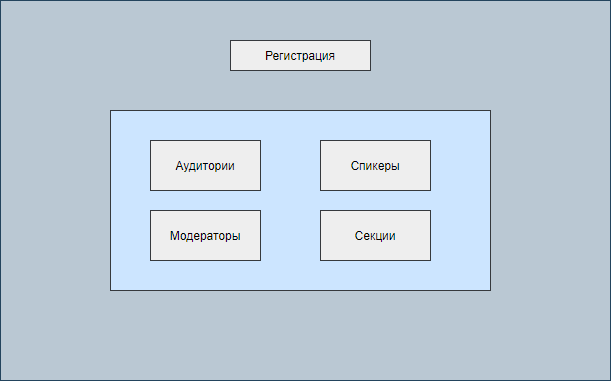
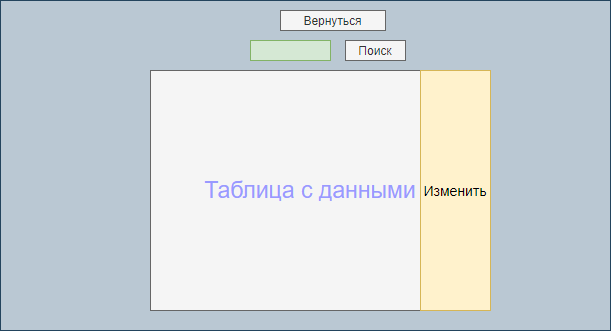
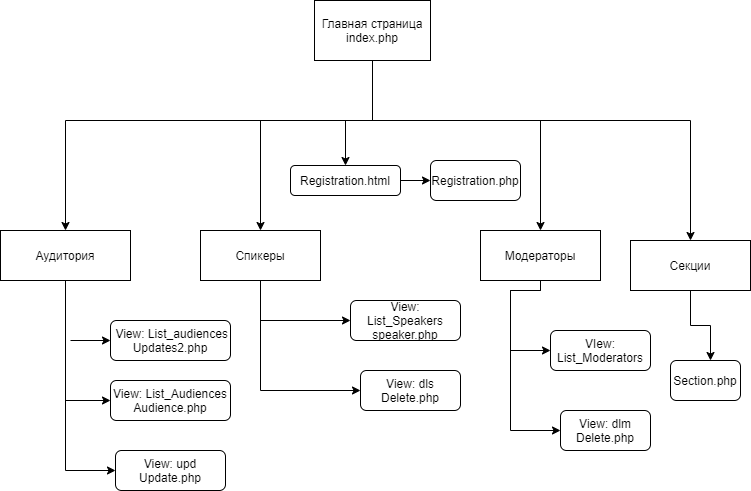
  
Рисунок 1.6 Макет главной страницы

Рисунок 1.6.1 Макет сайт с таблицей

Навигационная карта была разработана также и для web-приложения. Она отображена на рисунке 1.7

  
 Рисунок 1.7 Навигационная карта Web-приложения

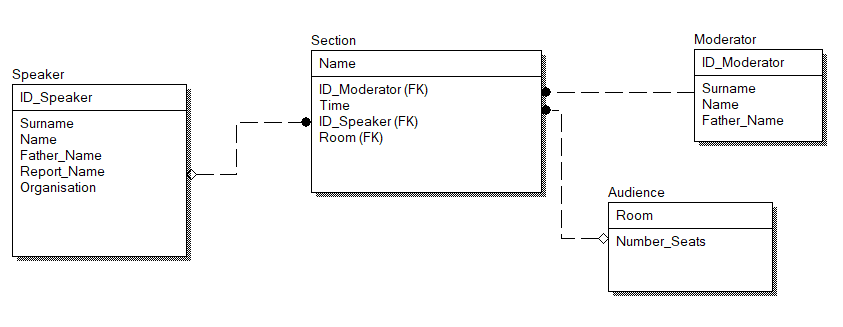
# Глава 2. Разработка информационной системы

## 2.1 Разработка информационного обеспечения

Под информационным обеспечением можно понимать нахождение или предоставление информации необходимой для осуществление какой-либо деятельности, оценки состояния или же подготовка к предупреждению нежелательных ситуаций. Его разработка происходит в два основных шага:

* Физическое проектирование базы данных
* Программирование на стороне SQL-сервера

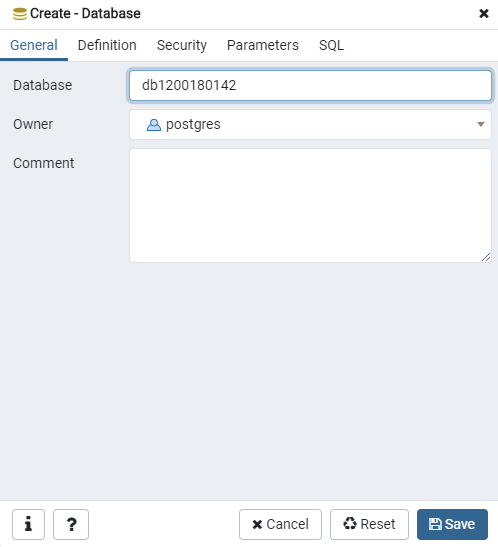
***Физическое проектирование БД***. Здесь мы вернемся к ранее описанной, еще в 1-ой главе даталогической модели будущей базы данных и составим ее физическую модель, которую затем реализуем в PostgreSQL.

  
Рисунок 2.1 Физическая модель базы данных

Здесь каждый атрибут уже имеет свой тип данных. Например «Имя» имеет тип *varchar(20)* (символьный тип с длиной в двадцать символов), а количество мест соответственно имеет тип *integer* (числовой тип)

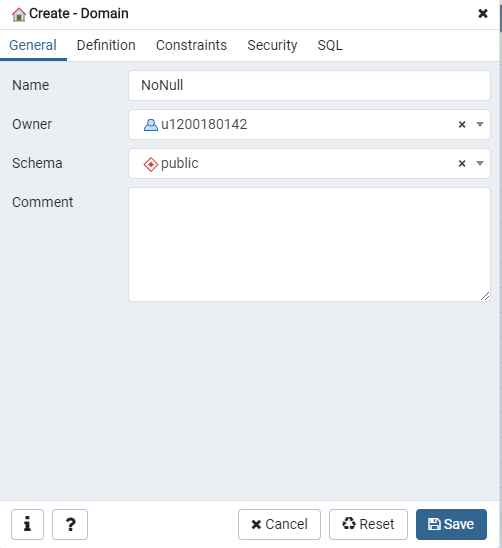
***Программирование на стороне SQL-сервера***. Программирование на стороне SQL-сервера —реализация физической модели базы данных при помощи средств СУБД, а также прописывание различного функционала информационной системы для данной базы данных.

Начнем с создания самой базы данных в клиенте pgAdmin 4, как это показано на рисунке 2.2

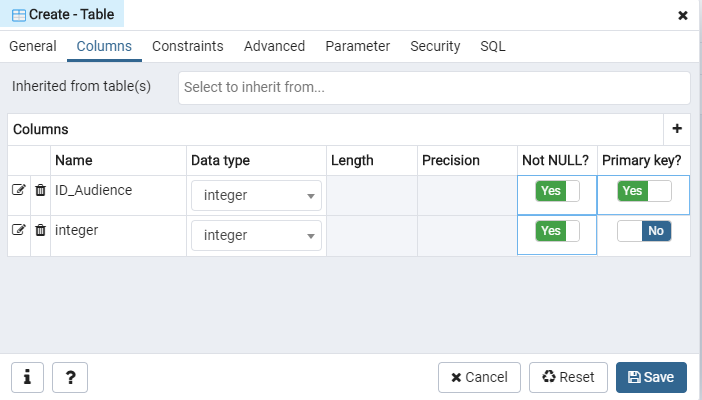
  
Рисунок 2.2 Создание базы данных

Теперь следует создать домены. Это пользовательские типа данных, на которые можно установить ограничение и они будут доступны для всех таблиц, в этой базе данных.

Создание домена отображено на рисунке 2.3

  
Рисунок 2.3 Создание домена NoNull

Вот теперь можно приступать непосредственно к созданию таблиц. На рисунке 2.4 представлен пример ее создания.

  
Рисунок 2.4 создание таблицы Аудитории

Чтобы использовать домен надо при создании таблицы указать его как тип данных.

Все остальные таблицы создаются по аналогии. После их создания, настало время их заполнить. Сделать это можно через SQL запросы как показано в листинге 1

Листинг 1 – заполнение таблицы Аудитории

Insert into Audience values(1, 3 ,117 , 32 );

Insert into Audience values(2, 2 ,118 , 32 );

Insert into Audience values(3, 4 ,119 , 30 );

Insert into Audience values(4, 7 ,121 , 32 );

Конец листинга 1

***Разработка бизнес-логики на стороне SQL-сервера.***

После заполнения базы данных данными можно перейти к бизнес-логике. Представление - это заранее составленный и хранящийся в базе данных SQL запрос для выборки данных из одной или нескольких таблиц. Пример создания представления можно увидеть в листинге 1

Листинг 2 – Создание представления для просмотра секций

Create view list\_sections  
 AS SELECT section.name AS "Название",   
section.time,  
 speaker.name AS "Имя спикера",  
 speaker.surname AS "Фамилия спикера",  
audience.id\_audience,  
 moderator.name AS "Имя модератора",  
 moderator.surname AS "Фамилия модератора" from section, speaker, audience, moderator

where speaker.id\_speaker = section.id\_speaker  
 and audience.id\_audience= section.id\_audience  
 and moderator.id\_moderator = section.id\_moderator

Конец листинга 2

Данное представление выводит информацию о всех данных хранящихся в таблице Секции и более того, вместо уникальных номеров Спикеров и Модераторов, идет обращение к соответствующим таблицам по данным номерам, что позволяет вывести имена и фамилии данных объектов. По аналогии создаются другие представления.  
 Хранимая процедура — объект базы данных, представляющий собой набор SQL-инструкций, компилирующийся всего один раз и после этого она хранится на сервере, благодаря чему работа выполняется быстрее, чем обычный запрос. Пример хранимой процедуры показан в Листинге 3.

Листинг 3 – Процедура добавления новых мест в аудиториях

Create function Add\_seats(Roomm integer, Quant integer)  
returns integer  
AS  
$$  
Begin  
UPDATE public.audience  
SET Number\_Seats=Number\_Seats + $2 WHERE ID\_Audience = Roomm;  
return roomm;  
END;  
$$  
LANGUAGE plpgsql;  
   
 Конец листинга 3

Данная функция принимает на вход номер аудитории и количество мест, которых надо добавить. Остальные функции были созданы по аналогии.

Помимо обычных функций, существуют еще и триггерные. Их отличие в том, что они будут вызываться сами при прописанных условиях. Пример триггера можно рассмотреть в листинге 4.

Листинг 4 – Триггерная функция для сверки не нулевого значения

CREATE FUNCTION Check\_Room()  
RETURNS trigger  
AS  
$$  
BEGIN  
IF NEW.ID\_Audience IS NULL THEN  
RAISE EXCEPTION '% не может быть NULL', NEW.Audience;  
END IF;  
IF NEW.Number\_Seats IS NULL THEN  
RAISE EXCEPTION '% не может быть NULL', NEW.Number\_Seats;  
END IF;  
RETURN NEW;  
END;  
$$  
LANGUAGE plpgsql;  
CREATE TRIGGER Check\_Room BEFORE INSERT OR UPDATE ON Audience  
FOR EACH ROW EXECUTE PROCEDURE Check\_Room();

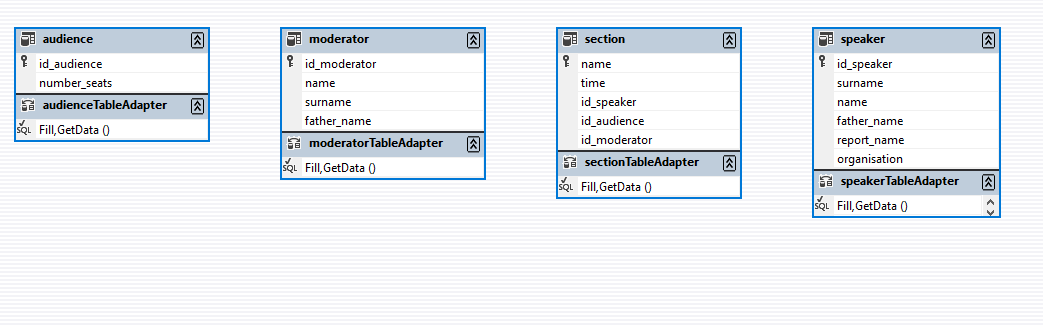
Конец листинга 4

Данный триггер проверяет заданное условие, чтобы значения таблицы Аудитории не были пустыми. Он работает прямо перед операцией *INSERT.* Если условие поля не нулевые, то триггер пропустит операцию, иначе значение не будут вставлены.

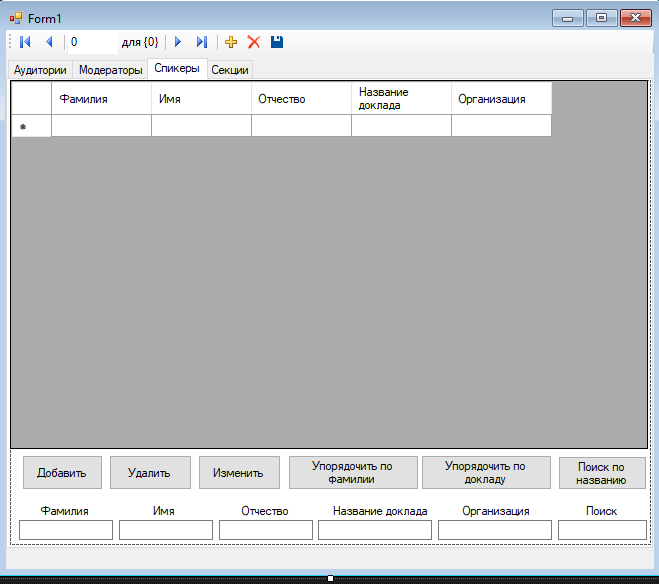
## 2.2 Разработка программного обеспечения

Под разработкой программного обеспечения подразумевается процесс реализации клиентских приложений, осуществляющих обращения к информационному обеспечению, которое было разработано заранее и было описано выше. В этом параграфе будет рассмотрен процесс создания Win- и Web-приложений.

***Создание Win-приложения.*** Как говорилось ранее, в качестве среды разработки для Windows была выбрана Visual Studio 2019 и язык программирования C# с платформой .NET.   
 Первым делом создается сама форма. На неё добавим tabcontrol для переключения между вкладками, в которых и будут таблицы. После этого на каждой вкладке расставим кнопки, текстовые поля и лейблы согласно макету. Теперь можно добавлять основы наших таблиц - элемент *DataGridView.* Также надо отмасштабировать и задать размеры нашей формы. В своем большинстве были использованы основные компоненты(Текстовые поля, кнопки, лейблы). Кроме них была использована сама форма таблицы с драйверами Odbc. Так же этот драйвер позволяет рассматривать прямо в Visual Studio модель базы данных, как это показано на рисунке 2.5

  
Рисунок 2.5 Модель базы данных в Visual Studio

Получившийся внешний вид можно посмотреть на рисунке 2.6

  
Рисунок 2.6 Пользовательский интерфейс win-приложения

Создав интерфейс, можно приступать к разработке самого приложения. Занесение данных, их изменение, а также поиск будет проходить с помощью текстовых полей, после ввода в которые и нажатие на соответствующую кнопку запустится функция добавления, указанная в листинге 5.

Листинг 5

private void button1\_Click(object sender, EventArgs e)  
 {System.Data.Odbc.OdbcConnection sqlConnection1 = audienceTableAdapter.Connection;  
System.Data.Odbc.OdbcCommand cmd = new System.Data.Odbc.OdbcCommand();  
cmd.CommandType = System.Data.CommandType.Text;  
cmd.CommandText = "SELECT addroom('" + textBox1.Text + "', ' " + textBox2.Text + "')";  
cmd.Connection = sqlConnection1;sqlConnection1.Open();  
cmd.ExecuteNonQuery();  
sqlConnection1.Close();  
audienceTableAdapter.Fill(dataSet1.audience);  
textBox1.Text = "";  
textBox2.Text = "";}

Конец листинга 5

В данном примере открывается подключение к базе данных, отправляется запрос функции на добавление объекта с указанными в текстовых полях параметрами, и заполнение обновленной таблицы. Весь код программы можно посмотреть в листинге Б. Теперь можно приступать к созданию web-приложения.

***Создание Web-приложения***. Ранее были описаны все необходимые средства для создания Web-приложения и приведены обоснования выбора применяемых технологий, потому сразу перейдем к подключению к postgreSQL из PHP. Для этого был создан отдельный файл PHP кода, который подключается к коду других страниц, обеспечивая тем самым большую эффективность работы. Сам файл содержит в себе: адрес сервера, к которому надо подключиться, логин клиента, пароль, название базы данных и порт. Код представлен в листинге 6.

Листинг 6 - Код файла settings.php

<?php

$host = '172.23.64.64';

$db = 'db1200180142';

$user = 'u1200180142';

$password = "12345";

$port = 3345;

$dsn = "pgsql:host = $host; port = $port;dbname=$db";

try {

$db = new PDO($dsn,$user,$password);

} catch (PDOException $e) {

die('Подключение не удалось: ' . $e->getMessage());

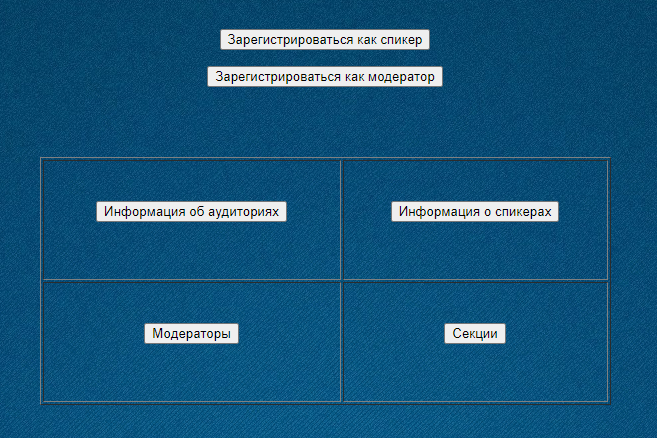
}

?>

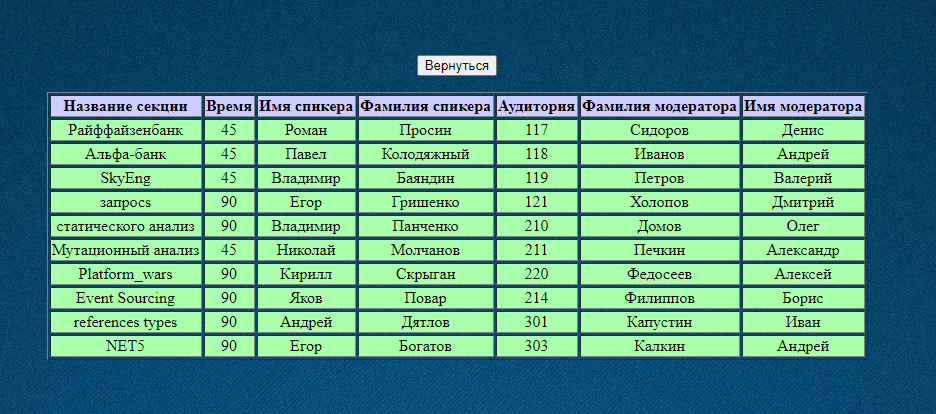
Конец листинга 6

Все остальные листинги будут представлены в приложении В.

Для работы с этим приложением был разработан интуитивно понятный пользовательский интерфейс. С главной страницы, на которую вначале попадает клиент, можно перейти к любой странице, отображающей ту или иную базу данных, нажав на одну из кнопок, представленных перед ним. Интерфейс этой странице представлен на рисунке 2.7

  
Рисунок 2.7 Главная страница web-приложения

Пример получения данных из таблицы можно увидеть на Рисунке 2.8

  
Рисунок 2.8 получение данных из таблицы Секции

Все управление базой данных строится на кнопках и некоторых текстовых полях. Таким образом готовый web-клиент позволяет пользоваться функционалом информационной системы с любого устройства, на котором имеется web-браузер и доступ к интернету.

# 

# 

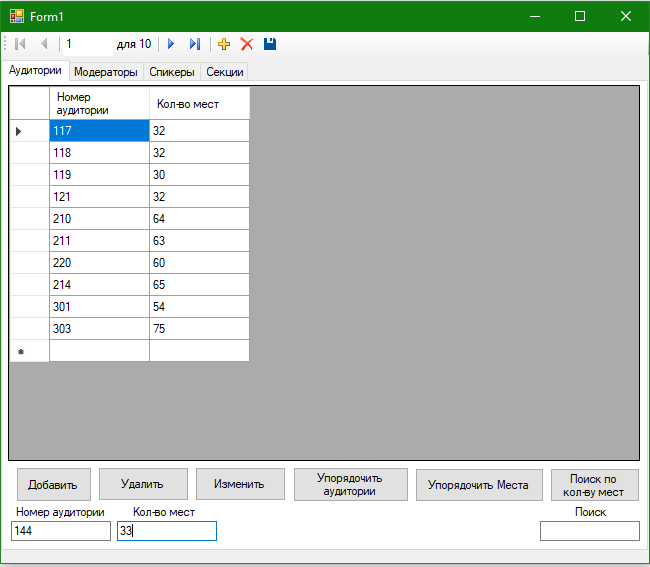
# 

# 

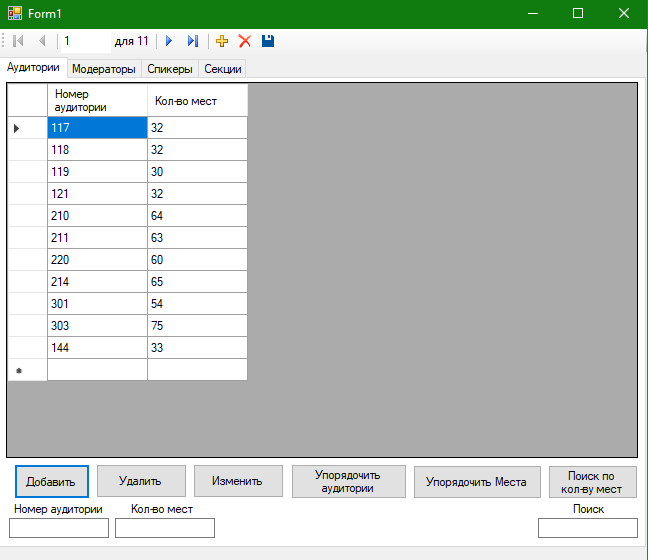
# Глава 3. Тестирование информационной системы

## 3.1 Пользовательский интерфейс

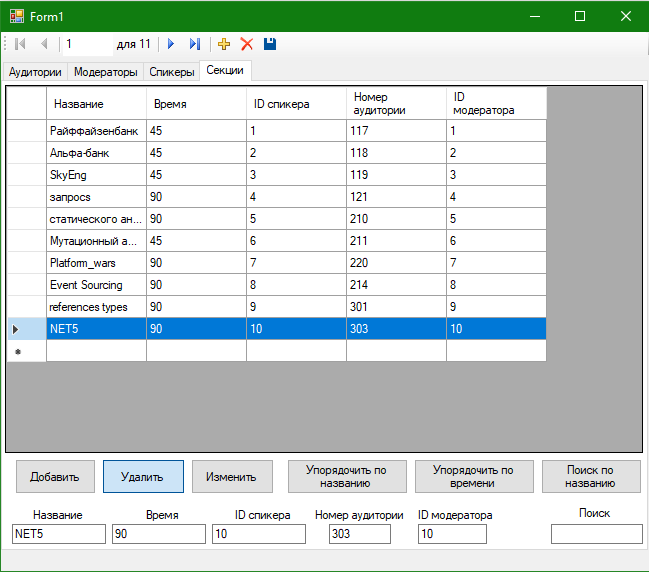
В данном параграфе мы сначала протестируем win-приложение на работоспособность, понятность и удобство, а затем протестируем web-приложение по тем же критериям.

  
Рисунок 3.1 Тестирование приложения

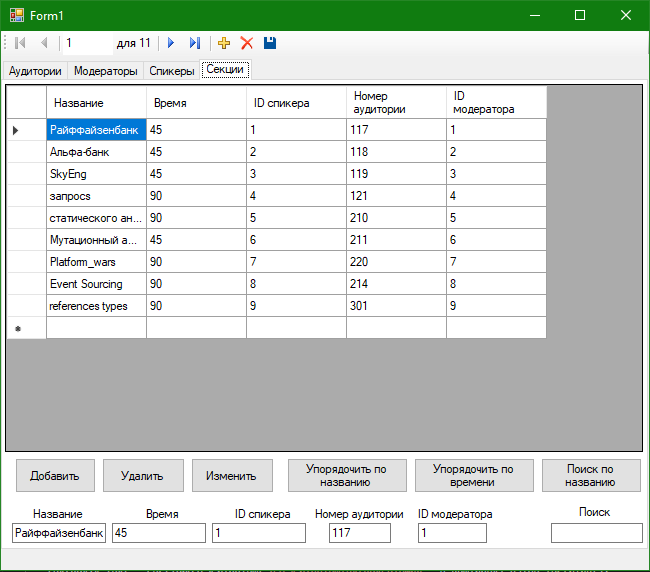
Программа успешно запустилась и вводятся данные для добавления их в таблицу.

  
Рисунок 3.2 Добавленная запись

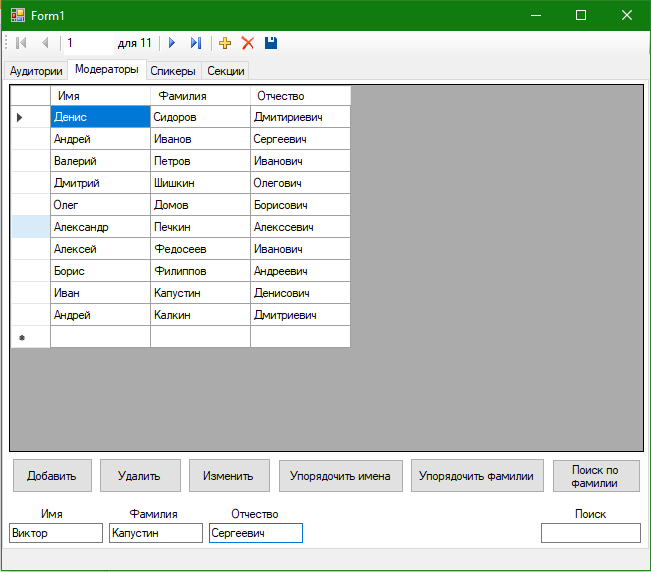
Как мы можем видеть по рисунку 3.2, запись успешно добавилась, теперь что-нибудь удалим.

  
Рисунок 3.3 Удаление записи

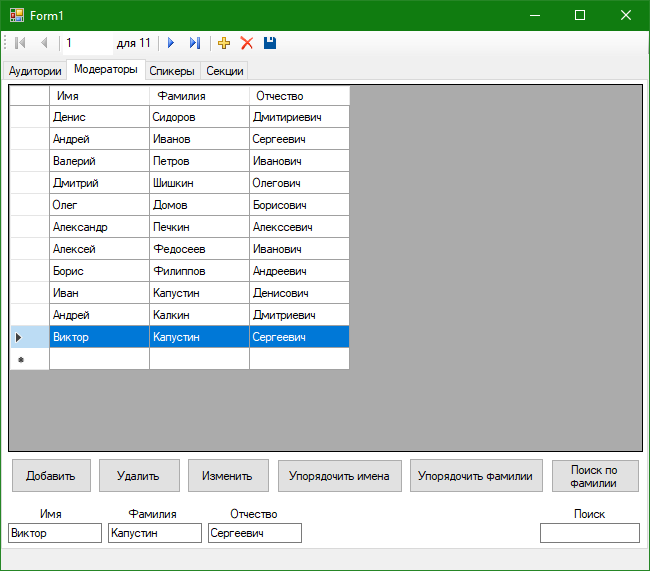
На рисунке 3.3 можно увидеть, как была выбрана запись и нажата кнопка удалить.

  
Рисунок 3.4 результат удаления

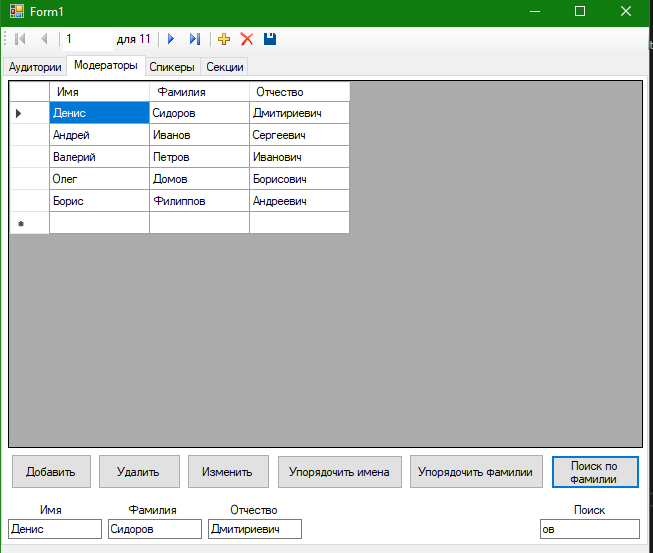
На рисунке 3.4 видно, что запись была успешно удалена. Теперь добавим запись.

  
Рисунок 3.5 Заполнение текстовых полей для ввода

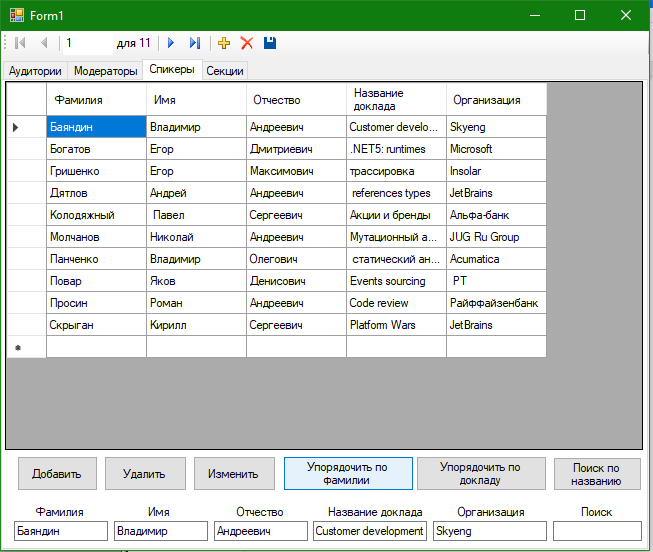
Заполним текстовые поля и нажимаем на изменить

  
Рисунок 3.6 Результат добавления

По рисунку 3.6 видно, что изменение работает корректно. Теперь попробуем применить поиск отсортировать данные.

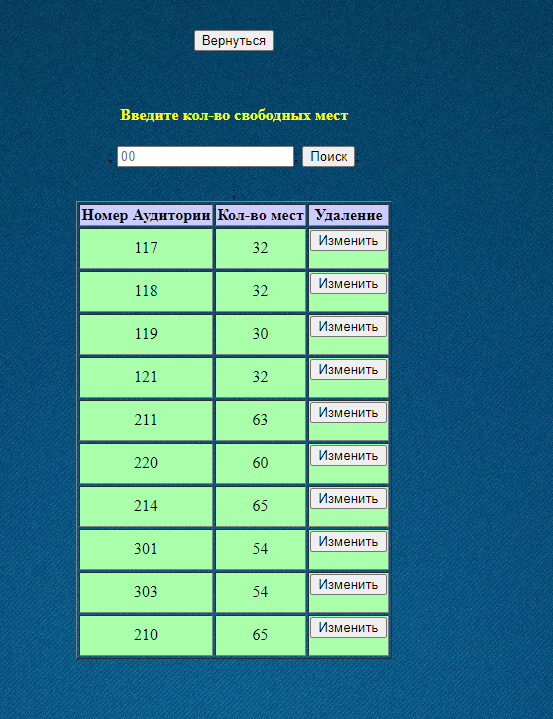
  
Рисунок 3.7 Результат поиска

По рисунку 3.7 видно, что поиск работает корректно. были найдены все люди, у которых в фамилии есть ‘ов’.

  
Рисунок 3.8 Результат упорядочивания

Упорядочивание тоже работает исправно. На этом можно сделать вывод, что win-приложение является работоспособным.

***Интерфейс Web-приложения***. Для тестирования интерфейса Web-приложения был использован браузер Google Chrome. Веб-приложение доступно по адресу[http://172.23.64.64/ftp/00/1200180142/Web/index.php](http://localhost:9998/ftp/00/1200180142/Web/index.php). Рисунками ниже приведен подробный разбор различного функционала Web-приложения.

  
Рисунок 3.9 Вывод таблицы Аудитории

По рисунку 3.9 видно, что сайт запустился, мы перешли к таблице Аудитории, которая корректно отображается. Попробуем ввести в поиск, в каких комнатах будет 50 мест

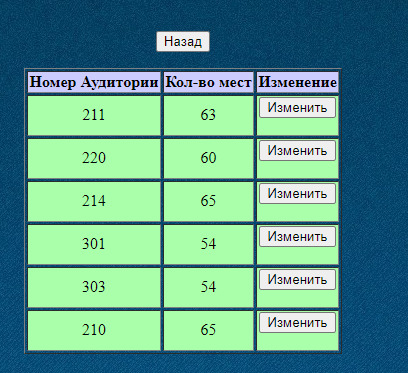
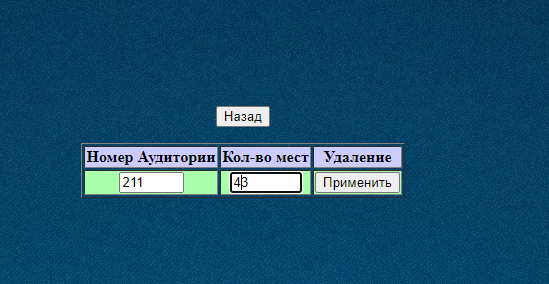
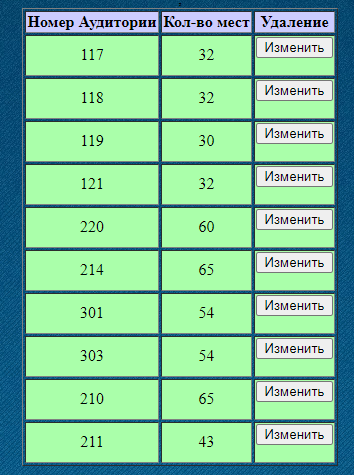


Рисунок 3.10 Получение результат поиска

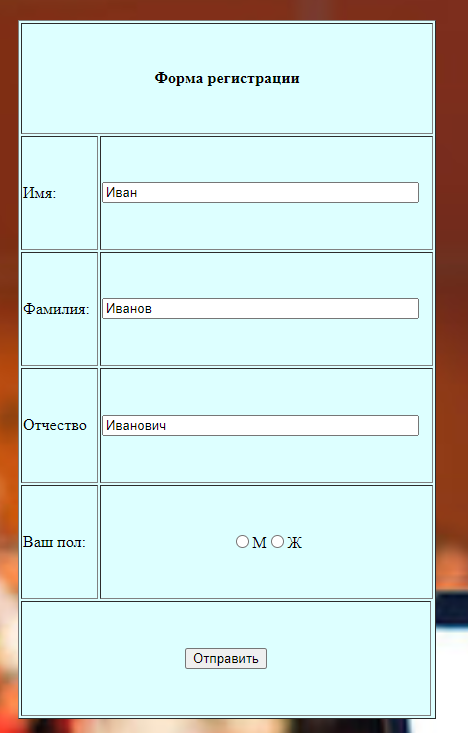
По рисунку 3.10 видно, что все найденные аудитории имеют больше 50 мест, что удовлетворяет условию.

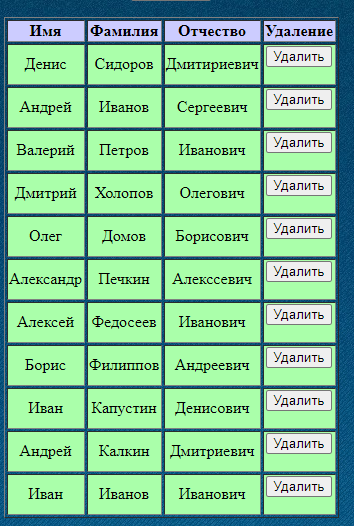
  
Рисунок 3.11 Изменение данных таблицы

Тут же попробуем изменить 211 аудиторию, уменьшив кол-во мест до 43, как показано на рисунке 3.11

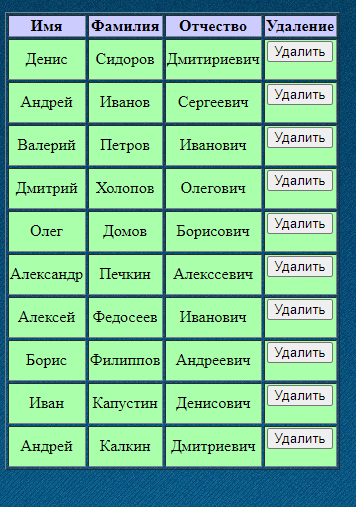
  
Рисунок 3.12 Результат изменения данных

Получив сообщение об успешном изменении, еще раз зайдем в таблицу Аудитории. Как видно на рисунке 3.12 в 211 аудитории стало 43 место, что удовлетворяет условию. Теперь попробуем добавить.

  
Рисунок 3.13 Регистрация нового модератора

Как видно из рисунка 3.13 мы отправили запрос на добавление нового модератора.  
  
Рисунок 3.15 Результат запроса на добавление

По рисунку 3.15 видно, что добавление нового объекта сработало корректно. Он занесся в таблицу. Сразу попробуем удалить его.

  
Рисунок 3.16 Результат запроса на удаление

По рисунку 3.16 видно, что объект был удален. Приложение работает корректно.

# Заключение

В процессе выполнения курсовой работы была создана информационная система обслуживания работы конференции с использованием клиент-серверной технологии. В результате были созданы серверная и два клиентских приложения,. Вся бизнес логика была реализована на SQL-сервере.  
 В ходе выполнения работы была изучена предметная область конференций, была спроектирована логическая и физическая модель данных, а также разработана информационная система. Разработанные приложения прошли этапы тестирования, были отлажены и приняты считаться корректно работающими. Поставленные цели в работе принято считать достигнутыми.

# 

# Список используемых источников

1. Vizlut/PHP [Электронный ресурс]. – Режим доступа:<https://vuzlit.ru/956664/yazyk_razrabotki_prilozheniya>
2. Мои Лекции.ру/СУБД[Электронный ресурс]. – Режим доступа:<https://mylektsii.ru/5-4870.html>
3. Васильев А.Ю., «Работа с PostgreSQL: настройка и масштабирование». -2017.
4. Котеров, Д.В. PHP 7. / Котеров Д. В, Симдянов И. В. – БХВ-Петербург, 2018. – 1088с
5. ИнфопедиЯ/Язык гипертекстовой разметки HTML [Электронный ресурс]. - Режим доступа:<https://infopedia.su/14x72af.html>
6. Лекция по информационным технологиям на тему "Автоматизированные информационные системы" [Электронный ресурс]. - Режим доступа:<https://infourok.ru/lekciya-po-informacionnim-tehnologiyam-na-temu-avtomatizirovannie-informacionnie-sistemi-2030044.html>
7. Постолит Visual Studio .NET: разработка приложений баз данных / Постолит, Анатолий. - М.: СПб: БХВ, 2003. - 544 c.
8. Фуфаев, Э.В. / Фуфаев Д.Э. Разработка и эксплуатация удаленных баз данных – М.: Издательский центр «Академия», 2008. – 256с.
9. Документация к PostgreSQL 9.6.13 [Электронный ресурс] / 1996-2019 The PostgreSQL Global Development Group. - Режим доступа: <https://postgrespro.ru/docs/postgresql/9.6/index>.
10. Современный учебник JavaScript [Электронный ресурс] / 2007–2019 Илья Кантор. - Режим доступа: https://learn.javascript.ru/.

Приложение А

Таблица соответствия доменов и таблиц с данными

| Имя таблицы | Поле | Тип | Not Null | Default | Check | Primary key | Имя домена |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Audience | ID\_Audience | integer | + |  |  | + |  |
| Number\_Seats | integer | + |  |  |  |  |
| Moderator | ID\_Moderator | integer | + |  |  | + |  |
| Surname | charactre(20) | + |  |  |  |  |
| Name | charactre(20) | + |  |  |  |  |
| Father\_Name | charactre(20) |  |  |  |  |  |
| Speaker | ID\_Speaker | integer | + |  |  | + |  |
| Name | charactre(20) | + |  |  |  |  |
| Surname | charactre(20) | + |  |  |  |  |
| Father\_Name | charactre(20) |  |  |  |  |  |
| Report\_Name | charactre(20) | + |  |  |  |  |
| Organisation | charactre(20) |  |  |  |  |  |
| Section | Name | charactre(20) | + |  |  | + |  |
| Time | integer |  |  |  |  |  |
| ID\_Speaker | integer | + |  |  |  |  |
| ID\_Audience | integer | + |  |  |  |  |
| ID\_Moderator | integer | + |  |  |  |  |

Приложение Б

Листинг win-приложения

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace blll

{

public partial class Form1 : Form

{

private int CurrantRowID = 0;

private string CurrantRow=" ";

public Form1()

{

InitializeComponent();

}

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

{

this.Validate();

this.audienceBindingSource.EndEdit();

this.tableAdapterManager.UpdateAll(this.dataSet1);

}

private void Form1\_Load(object sender, EventArgs e)

{

this.speakerTableAdapter.Fill(this.dataSet1.speaker);

// TODO: данная строка кода позволяет загрузить данные в таблицу "dataSet1.speaker". При необходимости она может быть перемещена или удалена.

// TODO: данная строка кода позволяет загрузить данные в таблицу "dataSet1.section". При необходимости она может быть перемещена или удалена.

this.sectionTableAdapter.Fill(this.dataSet1.section);

// TODO: данная строка кода позволяет загрузить данные в таблицу "dataSet1.moderator". При необходимости она может быть перемещена или удалена.

this.moderatorTableAdapter.Fill(this.dataSet1.moderator);

// TODO: данная строка кода позволяет загрузить данные в таблицу "dataSet1.audience". При необходимости она может быть перемещена или удалена.

this.audienceTableAdapter.Fill(this.dataSet1.audience);

}

private void audienceDataGridView\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

private void label1\_Click(object sender, EventArgs e)

{

}

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

{ System.Data.Odbc.OdbcConnection sqlConnection1 = audienceTableAdapter.Connection;

System.Data.Odbc.OdbcCommand cmd = new System.Data.Odbc.OdbcCommand();

cmd.CommandType = System.Data.CommandType.Text;

cmd.CommandText = "SELECT addroom('" + textBox1.Text + "', ' " + textBox2.Text + "')";

cmd.Connection = sqlConnection1;

sqlConnection1.Open();

cmd.ExecuteNonQuery();

sqlConnection1.Close();

audienceTableAdapter.Fill(dataSet1.audience);

textBox1.Text = "";

textBox2.Text = "";}

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

{

if (audienceDataGridView.SelectedRows.Count == 0)

{ return;}

System.Data.Odbc.OdbcConnection sqlConnection1 = audienceTableAdapter.Connection;

System.Data.Odbc.OdbcCommand cmd = new System.Data.Odbc.OdbcCommand();

cmd.CommandType = System.Data.CommandType.Text;

cmd.CommandText = "DELETE from public.audience where id\_audience = " + audienceDataGridView.SelectedRows[0].Cells["DataGridViewTextBoxColumn1"].Value.ToString();

cmd.Connection = sqlConnection1;

sqlConnection1.Open();

cmd.ExecuteNonQuery();

sqlConnection1.Close();

audienceTableAdapter.Fill(dataSet1.audience);

}

private void audienceDataGridView\_RowEnter(object sender, DataGridViewCellEventArgs e)

{

CurrantRowID = (int)audienceDataGridView["dataGridViewTextBoxColumn1", e.RowIndex].Value;

textBox1.Text = audienceDataGridView["dataGridViewTextBoxColumn1", e.RowIndex].Value.ToString();

textBox2.Text = audienceDataGridView["dataGridViewTextBoxColumn2", e.RowIndex].Value.ToString();

}

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

{ System.Data.Odbc.OdbcConnection sqlConnection1 = moderatorTableAdapter.Connection;

System.Data.Odbc.OdbcCommand cmd = new System.Data.Odbc.OdbcCommand();

cmd.CommandType = System.Data.CommandType.Text;

cmd.CommandText = "SELECT addModerator('"+CurrantRowID+"', '" + textBox3.Text + "', '" + textBox4.Text + "', '"+ textBox5.Text + "')";

cmd.Connection = sqlConnection1;

sqlConnection1.Open();

cmd.ExecuteNonQuery();

sqlConnection1.Close();

moderatorTableAdapter.Fill(dataSet1.moderator);textBox3.Text = "";

textBox4.Text = "";

textBox5.Text = "";

}

private void label3\_Click(object sender, EventArgs e)

{

}

private void label14\_Click(object sender, EventArgs e)

{

System.Data.Odbc.OdbcConnection sqlConnection1 = sectionTableAdapter.Connection;

System.Data.Odbc.OdbcCommand cmd = new System.Data.Odbc.OdbcCommand();

cmd.CommandType = System.Data.CommandType.Text;

cmd.CommandText = "SELECT addsection('" + textBox15.Text + "', '" + textBox14.Text + "', '" + textBox13.Text + "', '" + textBox12.Text + "', '" + textBox11.Text + "')";

cmd.Connection = sqlConnection1;

sqlConnection1.Open();

cmd.ExecuteNonQuery();

sqlConnection1.Close();

sectionTableAdapter.Fill(dataSet1.section);

textBox15.Text = "";

textBox14.Text = "";

textBox13.Text = "";

textBox12.Text = "";

textBox11.Text = "";

}

private void button12\_Click(object sender, EventArgs e)

{

System.Data.Odbc.OdbcConnection sqlConnection1 = moderatorTableAdapter.Connection;

System.Data.Odbc.OdbcCommand cmd = new System.Data.Odbc.OdbcCommand();

cmd.CommandType = System.Data.CommandType.Text;

cmd.CommandText = "SELECT addModerator('" + textBox3.Text + "', '" + textBox4.Text + "', '" + textBox5.Text + "')";

cmd.Connection = sqlConnection1;

sqlConnection1.Open();

cmd.ExecuteNonQuery();

sqlConnection1.Close();

moderatorTableAdapter.Fill(dataSet1.moderator);

textBox3.Text = "";

textBox4.Text = "";

textBox5.Text = "";

}

private void moderatorDataGridView\_RowEnter(object sender, DataGridViewCellEventArgs e)

{

CurrantRowID = (int)moderatorDataGridView["Column1", e.RowIndex].Value;

textBox3.Text = moderatorDataGridView["dataGridViewTextBoxColumn4", e.RowIndex].Value.ToString();

textBox4.Text = moderatorDataGridView["dataGridViewTextBoxColumn5", e.RowIndex].Value.ToString();

textBox5.Text = moderatorDataGridView["dataGridViewTextBoxColumn6", e.RowIndex].Value.ToString();

}

private void button5\_Click(object sender, EventArgs e)

{

if (moderatorDataGridView.SelectedRows.Count == 0)

{ return; }

System.Data.Odbc.OdbcConnection sqlConnection1 = moderatorTableAdapter.Connection;

System.Data.Odbc.OdbcCommand cmd = new System.Data.Odbc.OdbcCommand();

cmd.CommandType = System.Data.CommandType.Text;

cmd.CommandText = "DELETE from public.moderator where id\_moderator = " + moderatorDataGridView.SelectedRows[0].Cells["Column1"].Value.ToString();

cmd.Connection = sqlConnection1;

sqlConnection1.Open();

cmd.ExecuteNonQuery();

sqlConnection1.Close();

moderatorTableAdapter.Fill(dataSet1.moderator);

}

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

{

if (CurrantRowID == 0) return;

System.Data.Odbc.OdbcConnection sqlConnection1 = audienceTableAdapter.Connection;

System.Data.Odbc.OdbcCommand cmd = new System.Data.Odbc.OdbcCommand();

cmd.CommandType = System.Data.CommandType.Text;

cmd.CommandText = "SELECT editA('" + textBox1.Text + "', '" + textBox2.Text + "')";

cmd.Connection = sqlConnection1;

sqlConnection1.Open();

cmd.ExecuteNonQuery();

sqlConnection1.Close();

audienceTableAdapter.Fill(dataSet1.audience);

}

private void button6\_Click(object sender, EventArgs e)

{

if (CurrantRowID == 0) return;

System.Data.Odbc.OdbcConnection sqlConnection1 = moderatorTableAdapter.Connection;

System.Data.Odbc.OdbcCommand cmd = new System.Data.Odbc.OdbcCommand();

cmd.CommandType = System.Data.CommandType.Text;

cmd.CommandText = "SELECT editM('" + textBox1.Text + "', '" + textBox2.Text + "', '" +textBox5 + "')";

cmd.Connection = sqlConnection1;

sqlConnection1.Open();

cmd.ExecuteNonQuery();

sqlConnection1.Close();

moderatorTableAdapter.Fill(dataSet1.moderator);

}

private void button9\_Click(object sender, EventArgs e)

{

System.Data.Odbc.OdbcConnection sqlConnection1 = speakerTableAdapter.Connection;

System.Data.Odbc.OdbcCommand cmd = new System.Data.Odbc.OdbcCommand();

cmd.CommandType = System.Data.CommandType.Text;

cmd.CommandText = "SELECT addSpeaker('" + textBox8.Text + "', '" + textBox7.Text + "', '" + textBox6.Text + "', '" + textBox10.Text + "', '" + textBox9.Text + "')";

cmd.Connection = sqlConnection1;

sqlConnection1.Open();

cmd.ExecuteNonQuery();

sqlConnection1.Close();

speakerTableAdapter.Fill(dataSet1.speaker);

textBox8.Text = "";

textBox7.Text = "";

textBox6.Text = "";

textBox10.Text = "";

textBox9.Text = "";

}

private void button8\_Click(object sender, EventArgs e)

{

if (speakerDataGridView.SelectedRows.Count == 0)

{ return; }

System.Data.Odbc.OdbcConnection sqlConnection1 = speakerTableAdapter.Connection;

System.Data.Odbc.OdbcCommand cmd = new System.Data.Odbc.OdbcCommand();

cmd.CommandType = System.Data.CommandType.Text;

cmd.CommandText = "DELETE from public.speaker where id\_speaker = " + speakerDataGridView.SelectedRows[0].Cells["DataGridViewTextBoxColumn13"].Value.ToString();

cmd.Connection = sqlConnection1;

sqlConnection1.Open();

cmd.ExecuteNonQuery();

sqlConnection1.Close();

speakerTableAdapter.Fill(dataSet1.speaker);

}

private void button7\_Click(object sender, EventArgs e)

{

if (CurrantRowID == 0) return;

System.Data.Odbc.OdbcConnection sqlConnection1 = speakerTableAdapter.Connection;

System.Data.Odbc.OdbcCommand cmd = new System.Data.Odbc.OdbcCommand();

cmd.CommandType = System.Data.CommandType.Text;

cmd.CommandText = "SELECT editS('" + textBox8.Text + "', '" + textBox7.Text + "', '" + textBox6 + "', '" + textBox10.Text + "', '" + textBox9 + "')";

cmd.Connection = sqlConnection1;

sqlConnection1.Open();

cmd.ExecuteNonQuery();

sqlConnection1.Close();

speakerTableAdapter.Fill(dataSet1.speaker);

}

private void speakerDataGridView\_RowEnter(object sender, DataGridViewCellEventArgs e)

{

CurrantRowID = (int)speakerDataGridView["Column2", e.RowIndex].Value;

textBox8.Text = speakerDataGridView["dataGridViewTextBoxColumn13", e.RowIndex].Value.ToString();

textBox7.Text = speakerDataGridView["dataGridViewTextBoxColumn14", e.RowIndex].Value.ToString();

textBox6.Text = speakerDataGridView["dataGridViewTextBoxColumn15", e.RowIndex].Value.ToString();

textBox10.Text = speakerDataGridView["dataGridViewTextBoxColumn16", e.RowIndex].Value.ToString();

textBox9.Text = speakerDataGridView["dataGridViewTextBoxColumn17", e.RowIndex].Value.ToString();

}

private void button11\_Click(object sender, EventArgs e)

{

if (sectionDataGridView.SelectedRows.Count == 0)

{ return; }

System.Data.Odbc.OdbcConnection sqlConnection1 = sectionTableAdapter.Connection;

System.Data.Odbc.OdbcCommand cmd = new System.Data.Odbc.OdbcCommand();

cmd.CommandType = System.Data.CommandType.Text;

cmd.CommandText = "DELETE from public.section where Name = " + sectionDataGridView.SelectedRows[0].Cells["DataGridViewTextBoxColumn7"].Value.ToString();

cmd.Connection = sqlConnection1;

sqlConnection1.Open();

cmd.ExecuteNonQuery();

sqlConnection1.Close();

sectionTableAdapter.Fill(dataSet1.section);

}

private void sectionDataGridView\_RowEnter(object sender, DataGridViewCellEventArgs e)

{

CurrantRow = (string)sectionDataGridView["dataGridViewTextBoxColumn7", e.RowIndex].Value.ToString();

textBox15.Text = sectionDataGridView["dataGridViewTextBoxColumn7", e.RowIndex].Value.ToString();

textBox14.Text = sectionDataGridView["dataGridViewTextBoxColumn8", e.RowIndex].Value.ToString();

textBox13.Text = sectionDataGridView["dataGridViewTextBoxColumn9", e.RowIndex].Value.ToString();

textBox12.Text = sectionDataGridView["dataGridViewTextBoxColumn10", e.RowIndex].Value.ToString();

textBox11.Text = sectionDataGridView["dataGridViewTextBoxColumn11", e.RowIndex].Value.ToString();

}

private void button10\_Click(object sender, EventArgs e)

{

if (CurrantRow == "") return;

System.Data.Odbc.OdbcConnection sqlConnection1 = sectionTableAdapter.Connection;

System.Data.Odbc.OdbcCommand cmd = new System.Data.Odbc.OdbcCommand();

cmd.CommandType = System.Data.CommandType.Text;

cmd.CommandText = "SELECT editSec('" + textBox15.Text + "', '" + textBox14.Text + "', '" + textBox13 + "', '" + textBox12.Text + "', '" + textBox11 + "')";

cmd.Connection = sqlConnection1;

sqlConnection1.Open();

cmd.ExecuteNonQuery();

sqlConnection1.Close();

sectionTableAdapter.Fill(dataSet1.section);

}

private void button13\_Click(object sender, EventArgs e)

{

using (System.Data.Odbc.OdbcConnection connection = new System.Data.Odbc.OdbcConnection())

{

var select = "SELECT \* from Audience Order by ID\_Audience";

connection.ConnectionString = "DSN=postgres";

System.Data.Odbc.OdbcDataAdapter dataAdapter = new System.Data.Odbc.OdbcDataAdapter(select, connection);

System.Data.Odbc.OdbcCommandBuilder commandBuilder = new System.Data.Odbc.OdbcCommandBuilder(dataAdapter);

var ds = new DataSet();

dataAdapter.Fill(ds);

audienceDataGridView.DataSource = ds.Tables[0];

}

}

private void button14\_Click(object sender, EventArgs e)

{

using (System.Data.Odbc.OdbcConnection connection = new System.Data.Odbc.OdbcConnection())

{

var select = "SELECT \* from Audience Order by Number\_Seats ASC";

connection.ConnectionString = "DSN=postgres";

System.Data.Odbc.OdbcDataAdapter dataAdapter = new System.Data.Odbc.OdbcDataAdapter(select, connection);

System.Data.Odbc.OdbcCommandBuilder commandBuilder = new System.Data.Odbc.OdbcCommandBuilder(dataAdapter);

var ds = new DataSet();

dataAdapter.Fill(ds);

audienceDataGridView.DataSource = ds.Tables[0];

}

}

private void button15\_Click(object sender, EventArgs e)

{

using (System.Data.Odbc.OdbcConnection connection = new System.Data.Odbc.OdbcConnection())

{

var select = "SELECT \* from Moderator Order by Name ASC";

connection.ConnectionString = "DSN=postgres";

System.Data.Odbc.OdbcDataAdapter dataAdapter = new System.Data.Odbc.OdbcDataAdapter(select, connection);

System.Data.Odbc.OdbcCommandBuilder commandBuilder = new System.Data.Odbc.OdbcCommandBuilder(dataAdapter);

var ds = new DataSet();

dataAdapter.Fill(ds);

moderatorDataGridView.DataSource = ds.Tables[0];

}

}

private void button16\_Click(object sender, EventArgs e)

{

using (System.Data.Odbc.OdbcConnection connection = new System.Data.Odbc.OdbcConnection())

{

var select = "SELECT \* from Moderator Order by Surname ASC";

connection.ConnectionString = "DSN=postgres";

System.Data.Odbc.OdbcDataAdapter dataAdapter = new System.Data.Odbc.OdbcDataAdapter(select, connection);

System.Data.Odbc.OdbcCommandBuilder commandBuilder = new System.Data.Odbc.OdbcCommandBuilder(dataAdapter);

var ds = new DataSet();

dataAdapter.Fill(ds);

moderatorDataGridView.DataSource = ds.Tables[0];

}

}

private void button17\_Click(object sender, EventArgs e)

{

using (System.Data.Odbc.OdbcConnection connection = new System.Data.Odbc.OdbcConnection())

{

var select = "SELECT \* from Speaker Order by Surname ASC";

connection.ConnectionString = "DSN=postgres";

System.Data.Odbc.OdbcDataAdapter dataAdapter = new System.Data.Odbc.OdbcDataAdapter(select, connection);

System.Data.Odbc.OdbcCommandBuilder commandBuilder = new System.Data.Odbc.OdbcCommandBuilder(dataAdapter);

var ds = new DataSet();

dataAdapter.Fill(ds);

speakerDataGridView.DataSource = ds.Tables[0];

}

}

private void button18\_Click(object sender, EventArgs e)

{

using (System.Data.Odbc.OdbcConnection connection = new System.Data.Odbc.OdbcConnection())

{

var select = "SELECT \* from Speaker Order by Report\_Name ASC";

connection.ConnectionString = "DSN=postgres";

System.Data.Odbc.OdbcDataAdapter dataAdapter = new System.Data.Odbc.OdbcDataAdapter(select, connection);

System.Data.Odbc.OdbcCommandBuilder commandBuilder = new System.Data.Odbc.OdbcCommandBuilder(dataAdapter);

var ds = new DataSet();

dataAdapter.Fill(ds);

speakerDataGridView.DataSource = ds.Tables[0];

}

}

private void button19\_Click(object sender, EventArgs e)

{

using (System.Data.Odbc.OdbcConnection connection = new System.Data.Odbc.OdbcConnection())

{

var select = "SELECT \* from Section Order by Name ASC";

connection.ConnectionString = "DSN=postgres";

System.Data.Odbc.OdbcDataAdapter dataAdapter = new System.Data.Odbc.OdbcDataAdapter(select, connection);

System.Data.Odbc.OdbcCommandBuilder commandBuilder = new System.Data.Odbc.OdbcCommandBuilder(dataAdapter);

var ds = new DataSet();

dataAdapter.Fill(ds);

sectionDataGridView.DataSource = ds.Tables[0];

}

}

private void button20\_Click(object sender, EventArgs e)

{

using (System.Data.Odbc.OdbcConnection connection = new System.Data.Odbc.OdbcConnection())

{

var select = "SELECT \* from Section Order by Time ASC";

connection.ConnectionString = "DSN=postgres";

System.Data.Odbc.OdbcDataAdapter dataAdapter = new System.Data.Odbc.OdbcDataAdapter(select, connection);

System.Data.Odbc.OdbcCommandBuilder commandBuilder = new System.Data.Odbc.OdbcCommandBuilder(dataAdapter);

var ds = new DataSet();

dataAdapter.Fill(ds);

sectionDataGridView.DataSource = ds.Tables[0];

}

}

private void button21\_Click(object sender, EventArgs e)

{

using (System.Data.Odbc.OdbcConnection connection = new System.Data.Odbc.OdbcConnection())

{

var select = "SELECT \* from Section WHERE Name like '%"+textBox16.Text+"%'";

connection.ConnectionString = "DSN=postgres";

System.Data.Odbc.OdbcDataAdapter dataAdapter = new System.Data.Odbc.OdbcDataAdapter(select, connection);

System.Data.Odbc.OdbcCommandBuilder commandBuilder = new System.Data.Odbc.OdbcCommandBuilder(dataAdapter);

var ds = new DataSet();

dataAdapter.Fill(ds);

sectionDataGridView.DataSource = ds.Tables[0];

}

}

private void button22\_Click(object sender, EventArgs e)

{

using (System.Data.Odbc.OdbcConnection connection = new System.Data.Odbc.OdbcConnection())

{

var select = "SELECT \* from Speaker WHERE Report\_Name like '%" + textBox17.Text + "%'";

connection.ConnectionString = "DSN=postgres";

System.Data.Odbc.OdbcDataAdapter dataAdapter = new System.Data.Odbc.OdbcDataAdapter(select, connection);

System.Data.Odbc.OdbcCommandBuilder commandBuilder = new System.Data.Odbc.OdbcCommandBuilder(dataAdapter);

var ds = new DataSet();

dataAdapter.Fill(ds);

speakerDataGridView.DataSource = ds.Tables[0];

}

}

private void button23\_Click(object sender, EventArgs e)

{

using (System.Data.Odbc.OdbcConnection connection = new System.Data.Odbc.OdbcConnection())

{

var select = "SELECT \* from moderator WHERE Surname like '%" + textBox18.Text + "%'";

connection.ConnectionString = "DSN=postgres";

System.Data.Odbc.OdbcDataAdapter dataAdapter = new System.Data.Odbc.OdbcDataAdapter(select, connection);

System.Data.Odbc.OdbcCommandBuilder commandBuilder = new System.Data.Odbc.OdbcCommandBuilder(dataAdapter);

var ds = new DataSet();

dataAdapter.Fill(ds);

moderatorDataGridView.DataSource = ds.Tables[0];

}

}

private void button24\_Click(object sender, EventArgs e)

{

using (System.Data.Odbc.OdbcConnection connection = new System.Data.Odbc.OdbcConnection())

{

var select = "SELECT \* from audience WHERE Number\_Seats >= " + textBox19.Text + "";

connection.ConnectionString = "DSN=postgres";

System.Data.Odbc.OdbcDataAdapter dataAdapter = new System.Data.Odbc.OdbcDataAdapter(select, connection);

System.Data.Odbc.OdbcCommandBuilder commandBuilder = new System.Data.Odbc.OdbcCommandBuilder(dataAdapter);

var ds = new DataSet();

dataAdapter.Fill(ds);

audienceDataGridView.DataSource = ds.Tables[0];

}

}

}

}

Приложение В

Листинг web-приложения

**Index.php**

<html><head>

<style>

h1{

color:red;

}

a{

color:red;

}

</style>

</head>

<body background="Pict.jpg">

<br><br><br><br><br>

<center>

<h1> Конференция Светила науки - 2020 </h1>

<br><br><br><br>

<form action="registration.html" method="post">

<input type="Submit" value="Зарегистрироваться как спикер">

</form>

<form action="registration1.html" method="post">

<input type="Submit" value="Зарегистрироваться как модератор">

</form>

<br><br><br>

<table border=1 width ="30%" height= "27%" align="center">

<tr>

<th colspan="3">

<form name="f1" action="Audience.php" >

<input type="submit" name="s1" size ="44" value="Информация об аудиториях">

</form>

</th>

<th colspan="3" >

<form name="f1" action="Speakers.php" >

<input type="submit" name="s1" size ="44" value="Информация о спикерах">

</form>

</th> </tr>

<tr><th colspan="3" >

<form name="f1" action="Moderator.php" >

<input type="submit" name="s1" size ="44" value="Модераторы">

</form>

</th>

<th colspan="3" >

<form name="f1" action="Section.php" >

<input type="submit" name="s1" size ="44" value="Секции">

</form>

</tr>

**Registration.html**

<html>

<head>

<title>JavaScript</title>

<link rel="stylesheet" href="teset.css">

<style>

body{

background: url(fon.jpg) no-repeat;

background-size: 100%;

}

table {

background-color: #dff; /\* Цвет фона \*/

/\* Цвет текста \*/

}

</style>

</head>

<body>

<meta charset="UTF-8" />

<br><br><br><br><br><br>

<table backgroundcolor="Green" border align="center" width="22%" height="76%" id="color">

<form action='registration.php' method="POST" >

<tr>

<th colspan="4" class="cl">Форма регистрации</th>

</th>

<tr>

<td colspan="1"width=19% class="cl">Имя:</td>

<td colspan="3" class="cl"> <INPUT type="text" name="name" value="Иван" size=40 maxlength=25> </td>

</tr>

<tr>

<td colspan="1"class="cl"> Фамилия:</td>

<td colspan="3"><INPUT type="text" name="surname" value="Иванов" size=40 maxlength=30> </td>

</tr>

<tr>

<td colspan="1" class="cl">Отчество</td>

<td colspan="3"><INPUT type="text" name="father\_name" value="Иванович" size=40 maxlength=32> </td>

</tr>

<tr>

<td colspan="1" class="cl">Название доклада</td>

<td colspan="3"><INPUT type="text" name="report\_name" value="Иванович" size=40 maxlength=32> </td>

</tr>

<tr>

<td colspan="1" class="cl">Орагнизация</td>

<td colspan="3"><INPUT type="text" name="organisation" value="Иванович" size=40 maxlength=32> </td>

</tr>

<tr>

<td colspan="1" class="cl"> Ваш пол:</td>

<td colspan="3" align="center" class="cl"> <INPUT type="radio" name="sex" value="OK">М<INPUT type="radio" name="sex" value="NOK">Ж</td>

</tr>

<tr>

<td colspan="3" align="center"><input type="submit" name="sub"> </td>

</table>

<br><br>

<center>

</body>

</html>

**Registration.php**

<html>

<head>

<style>

h2{

color:yellow;

}

</style>

</head>

<body background="Pict.jpg">

<?PHP include('settings.php');

$Name=$\_POST['name'];

$Surname=$\_POST['surname'];

$Father\_Name=$\_POST['father\_name'];

$Report\_Name=$\_POST['report\_name'];

$Organisation=$\_POST['organisation'];

$sql = "SELECT \* from Speaker";

$res = $db->query($sql);

$num\_results=$res->rowCount();

$result = $res->fetchAll(PDO::FETCH\_NUM);

$num\_results+=1;

$sql = "INSERT INTO Speaker values('$num\_results','$Name','$Surname','$Father\_Name','$Report\_Name','$Organisation')";

$res = $db->query($sql);

$num\_results=$res->rowCount();

$result = $res->fetchAll(PDO::FETCH\_NUM);

echo "<center>";

echo "<h2>Спасибо за ваше участие!</h2>";

echo "<h2>Теперь можно вернуться на главную страницу.</h2>";

echo "<form action=\"index.php\" method=\"post\"> ";

echo "<input type=\"Submit\" value=\"Вернуться\">";

echo "</form>";

?>

</body>

</html>

**Registration1.html**

<html>

<head>

<title>JavaScript</title>

<link rel="stylesheet" href="teset.css">

<style>

body{

background: url(fon.jpg) no-repeat;

background-size: 100%;

}

table {

background-color: #dff; /\* Цвет фона \*/

/\* Цвет текста \*/

}

</style>

</head>

<body>

<meta charset="UTF-8" />

<br><br><br><br><br><br>

<table backgroundcolor="Green" border align="center" width="22%" height="76%" id="color">

<form action='registration1.php' method="POST" >

<tr>

<th colspan="4" class="cl">Форма регистрации</th>

</th>

<tr>

<td colspan="1"width=19% class="cl">Имя:</td>

<td colspan="3" class="cl"> <INPUT type="text" name="name" value="Иван" size=40 maxlength=25> </td>

</tr>

<tr>

<td colspan="1"class="cl"> Фамилия:</td>

<td colspan="3"><INPUT type="text" name="surname" value="Иванов" size=40 maxlength=30> </td>

</tr>

<tr>

<td colspan="1" class="cl">Отчество</td>

<td colspan="3"><INPUT type="text" name="father\_name" value="Иванович" size=40 maxlength=32> </td>

</tr>

<tr>

<td colspan="1" class="cl"> Ваш пол:</td>

<td colspan="3" align="center" class="cl"> <INPUT type="radio" name="sex" value="OK">М<INPUT type="radio" name="sex" value="NOK">Ж</td>

</tr>

<tr>

<td colspan="3" align="center"><input type="submit" name="sub"> </td>

</table>

<br><br>

<center>

</body>

</html>

**Registration.php**

<html>

<head>

<style>

h2{

color:yellow;

}

</style>

</head>

<body background="Pict.jpg">

<?PHP include('settings.php');

$Name=$\_POST['name'];

$Surname=$\_POST['surname'];

$Father\_Name=$\_POST['father\_name'];

$sql = "SELECT \* from Speaker";

$res = $db->query($sql);

$num\_results=$res->rowCount();

$result = $res->fetchAll(PDO::FETCH\_NUM);

$num\_results+=1;

$sql = "INSERT INTO moderator values('$num\_results','$Name','$Surname','$Father\_Name')";

$res = $db->query($sql);

$num\_results=$res->rowCount();

$result = $res->fetchAll(PDO::FETCH\_NUM);

echo "<center>";

echo "<h2>Спасибо за ваше участие!</h2>";

echo "<h2>Теперь можно вернуться на главную страницу.</h2>";

echo "<form action=\"index.php\" method=\"post\"> ";

echo "<input type=\"Submit\" value=\"Вернуться\">";

echo "</form>";

?>

</body>

</html>

**Audience.php**

<html><style>

h4{

color:yellow;

}

</style>

</head>

<body background="Pict.jpg">

<?PHP include('settings.php');

$sql = "SELECT \* FROM public.list\_audiences";

$res = $db->query($sql);

$num\_results=$res->rowCount();

$result = $res->fetchAll(PDO::FETCH\_NUM);

echo '<center>';

echo '<br><br><br><br><br><br>';

echo "<form action=\"index.php\" method=\"post\"> ";

echo "<input type=\"Submit\" value=\"Вернуться\">";

echo "</form>";

echo "<br>";

?>

<h4>Введите кол-во свободных мест</h4>

<form name="f2" method="POST" action="Updated2.php">;

<input type="text" name="quant" placeholder ="00">;

<input type="submit" value="Поиск">;

</form>;

<?PHP

echo '<table border=1>';

echo '<tr>'.'<th bgcolor="#CCCCFF" >Номер Аудитории</th>'.'<th bgcolor="#CCCCFF">Кол-во мест</th>'.'<th bgcolor="#CCCCFF">Удаление</th>'.'</tr>';

foreach($result as $row){

$ID=$row[0];

echo "<tr><td align='center' bgcolor='#AAFFAA'>$row[0]</td><td

align='center' bgcolor='#AAFFAA'>$row[1]</td><td align='center' bgcolor='#AAFFAA'><form name=\"f1\" method=\"POST\" action=\"Update.php\"><input type=\"text\" name=\"ID\" value=\"$ID\" hidden><input type=\"submit\" name=\"s1\" value=\"Изменить\">

</form></td></tr>";

}

?>

</body>

</html>

**Delete.php**

<html>

<head>

<style>

h2{

color:yellow;

}

</style>

</head>

<body background="Pict.jpg">

<?PHP include('settings.php');

$flag2=0;

$flag=$\_POST['flag'];

$ID=$\_POST['ID'];

switch ($flag)

{

case 1:

$sql = "SELECT section.ID\_Speaker from public.section where section.id\_speaker = 'ID'";

$res = $db->query($sql);

$num\_results=$res->rowCount();

$result = $res->fetchAll(PDO::FETCH\_NUM);

if($num\_results==0){

$sql = "SELECT from public.dels('$ID')";

$res = $db->query($sql);

$flag2=1;

}

break;

case 2:

$sql = "SELECT section.ID\_Moderator from public.section where section.ID\_moderator='$ID'";

$res = $db->query($sql);

$num\_results=$res->rowCount();

$result = $res->fetchAll(PDO::FETCH\_NUM);

if($num\_results==0){

$sql = "SELECT from public.delm('$ID')";

$res = $db->query($sql);

$flag2=1;

}

break;

}

if($flag2==1){

echo "<center>";

echo "<h2>Удаление прошло успешно</h2>";

echo "<h2>Теперь можно вернуться на главную страницу.</h2>";

echo "<form action=\"index.php\" method=\"post\"> ";

echo "<input type=\"Submit\" value=\"Вернуться\">";

echo "</form>";

}

else{

echo "<center>";

echo "<h2>Упсс... произошла ошибка</h2>";

echo "<h2>Возможно вы патетесь удалить человека, который задействованв секции.</h2>";

echo "<form action=\"index.php\" method=\"post\"> ";

echo "<input type=\"Submit\" value=\"Вернуться\">";

echo "</form>";

}

?>

</body>

</html>

**Moderator.php**

<html>

<body background="Pict.jpg">

<?PHP include('settings.php');

$flag=2;

$sql = "SELECT \* FROM public.list\_moderators";

$res = $db->query($sql);

$num\_results=$res->rowCount();

$result = $res->fetchAll(PDO::FETCH\_NUM);

echo '<center>';

echo '<br><br><br><br><br><br>';

echo "<form action=\"index.php\" method=\"post\"> ";

echo "<input type=\"Submit\" value=\"Вернуться\">";

echo "</form>";

echo '<table border=1>';

echo '<tr>'.'<th bgcolor="#CCCCFF">Имя</th>'.'<th bgcolor="#CCCCFF" >Фамилия</th>'.'<th bgcolor="#CCCCFF"

>Отчество</th>'.'<th bgcolor="#CCCCFF" >Удаление</th>'.'</tr>';

foreach($result as $row){

$ID=$row[0];

echo "<tr><td align='center' bgcolor='#AAFFAA'>$row[1]</td><td

align='center' bgcolor='#AAFFAA'>$row[2]</td><td align='center' bgcolor='#AAFFAA'>$row[3]</td><td align='center' bgcolor='#AAFFAA'><form name=\"f111\" method =\"POST\" action=\"Delete.php\"><input type=\"text\" name=\"ID\" value=\"$ID\" hidden><input type=\"text\" name=\"flag\" value=\"$flag\" hidden><input type=\"submit\" name=\"s1\" value=\"Удалить\">

</form></td></tr>";

}

?>

</body>

</html>

**Section.php**

<html>

<body background="Pict.jpg">

<?PHP include('settings.php');

$sql = "SELECT \* FROM public.List\_Sections";

$res = $db->query($sql);

$num\_results=$res->rowCount();

$result = $res->fetchAll(PDO::FETCH\_NUM);

echo '<center>';

echo '<br><br><br><br><br><br>';

echo "<form action=\"index.php\" method=\"post\"> ";

echo "<input type=\"Submit\" value=\"Вернуться\">";

echo "</form>";

echo '<table border=1>';

echo '<tr><th bgcolor="#CCCCFF" >Название секции</th>'.'<th bgcolor="#CCCCFF" >Время </th>'.'<th bgcolor="#CCCCFF">Имя спикера</th>'.'<th bgcolor="#CCCCFF" >Фамилия спикера</th>'.'<th bgcolor="#CCCCFF" >Аудитория</th>'.'<th bgcolor="#CCCCFF" >Фамилия модератора</th>'.'<th bgcolor="#CCCCFF" >Имя модератора</th>'.'</tr>';

foreach($result as $row){

echo "<tr><td bgcolor='#AAFFAA' align='center' >$row[0]</td><td align='center' bgcolor='#AAFFAA'>$row[1]</td><td

align='center' bgcolor='#AAFFAA'>$row[2]</td><td align='center' bgcolor='#AAFFAA'>$row[3]</td><td align='center' bgcolor='#AAFFAA'>$row[4]</td><td align='center' bgcolor='#AAFFAA'>$row[5]</td><td align='center' bgcolor='#AAFFAA'>$row[6]</td></tr>";

}

?>

</body>

</html>

**Settings.php**

<?php

$host = '172.23.64.64';

$db = 'db1200180142';

$user = 'u1200180142';

$password = "12345";

$port = 3345;

$dsn = "pgsql:host = $host; port = $port;dbname=$db";

try {

$db = new PDO($dsn,$user,$password);

} catch (PDOException $e) {

die('Подключение не удалось: ' . $e->getMessage());

}

?>

**Speakers.php**

<html>

<body background="Pict.jpg">

<?PHP include('settings.php');

$flag=1;

if(!empty($\_POST['cond'])){

$cond=$\_POST['cond'];

switch ($cond) {

case 1:

$sql = "SELECT \* FROM public.list\_speakers order by Surname ";

$res = $db->query($sql);

$num\_results=$res->rowCount();

$result = $res->fetchAll(PDO::FETCH\_NUM);

break;

case 2:

$sql = "SELECT \* FROM public.list\_speakers order by Name ";

$res = $db->query($sql);

$num\_results=$res->rowCount();

$result = $res->fetchAll(PDO::FETCH\_NUM);

break;

case 3:

$sql = "SELECT \* FROM public.list\_speakers order by Father\_Name ";

$res = $db->query($sql);

$num\_results=$res->rowCount();

$result = $res->fetchAll(PDO::FETCH\_NUM);

break;

case 4:

$sql = "SELECT \* FROM public.list\_speakers order by Report\_Name ";

$res = $db->query($sql);

$num\_results=$res->rowCount();

$result = $res->fetchAll(PDO::FETCH\_NUM);

break;

case 5:

$sql = "SELECT \* FROM public.list\_speakers order by Organisation ";

$res = $db->query($sql);

$num\_results=$res->rowCount();

$result = $res->fetchAll(PDO::FETCH\_NUM);

break;

default:

break;

}

}

else{

$sql = "SELECT \* FROM public.list\_speakers order by Surname ";

$res = $db->query($sql);

$num\_results=$res->rowCount();

$result = $res->fetchAll(PDO::FETCH\_NUM);

}

echo '<center>';

echo '<br><br><br><br><br><br>';

echo "<form action=\"index.php\" method=\"post\"> ";

echo "<input type=\"Submit\" value=\"Вернуться\">";

echo "</form>";

echo '<table border=1>';

echo "

<tr>

<td align='center' bgcolor='#AAFFAA'>

<form name=\"f11\" method =\"POST\" action=\"Speakers.php\">

<input type=\"text\" name=\"cond\" value=\"1\" hidden>

<input type=\"submit\" name=\"s1\" value=\"Упорядочить\">

</form></td>

<td align='center' bgcolor='#AAFFAA'>

<form name=\"f12\" method =\"POST\" action=\"Speakers.php\">

<input type=\"text\" name=\"cond\" value=\"2\" hidden>

<input type=\"submit\" name=\"s1\" value=\"Упорядочить\">

</form></td>

<td align='center' bgcolor='#AAFFAA'>

<form name=\"f13\" method =\"POST\" action=\"Speakers.php\">

<input type=\"text\" name=\"cond\" value=\"3\" hidden>

<input type=\"submit\" name=\"s1\" value=\"Упорядочить\">

</form></td>

<td align='center' bgcolor='#AAFFAA'>

<form name=\"f14\" method =\"POST\" action=\"Speakers.php\">

<input type=\"text\" name=\"cond\" value=\"4\" hidden>

<input type=\"submit\" name=\"s1\" value=\"Упорядочить\">

</form></td>

<td align='center' bgcolor='#AAFFAA'>

<form name=\"f15\" method =\"POST\" action=\"Speakers.php\">

<input type=\"text\" name=\"cond\" value=\"5\" hidden>

<input type=\"submit\" name=\"s1\" value=\"Упорядочить\">

</form></td><td align='center' bgcolor='#AAFFAA'>

</td></tr>

";

echo '<tr>'.'<th bgcolor="#CCCCFF">Фамилия</th><th bgcolor="#CCCCFF" >Имя</th>'.'<th bgcolor="#CCCCFF"

>Отчество</th>'.'<th bgcolor="#CCCCFF" >Доклад</th>'.'<th bgcolor="#CCCCFF" >Организация</th>'.'<th bgcolor="#CCCCFF" >Удаление</th>'.'</tr>';

foreach($result as $row){

$ID=$row[0];

echo "<tr><td align='center' bgcolor='#AAFFAA'>$row[1]</td><td

align='center' bgcolor='#AAFFAA'>$row[2]</td><td align='center' bgcolor='#AAFFAA'>$row[3]</td><td align='center' bgcolor='#AAFFAA'>$row[4]</td><td align='center' bgcolor='#AAFFAA'>$row[5]</td><td align='center' bgcolor='#AAFFAA'><form name=\"f1\" method =\"POST\"action=\"Delete.php\"><input type=\"text\" name=\"ID\" value=\"$ID\" hidden><input type=\"text\" name=\"flag\" value=\"$flag\" hidden><input type=\"submit\" name=\"s1\" value=\"Удалить\">

</form></td></tr>";

}

?>

</body>

</html>

**Update.php**

<html>

<body background="Pict.jpg">

<?PHP include('settings.php');

$ID=$\_POST['ID'];

$sql = "SELECT \* FROM public.list\_audiences where ID\_Audience = '$ID'";

$res = $db->query($sql);

$num\_results=$res->rowCount();

$result = $res->fetchAll(PDO::FETCH\_NUM);

echo '<center>';

echo '<br><br><br><br><br><br>';

echo "<form action=\"Audience.php\" method=\"post\"> ";

echo "<input type=\"Submit\" value=\"Назад\">";

echo "</form>";

echo '<table border=1>';

echo '<tr>'.'<th bgcolor="#CCCCFF" >Номер Аудитории</th>'.'<th bgcolor="#CCCCFF"

>Кол-во мест</th>'.'<th bgcolor="#CCCCFF"

>Удаление</th>'.'</tr>';

foreach($result as $row){

$ID=$row[0];

echo "<tr><form name=\"f1\"method =\"POST\" action=\"Updated1.php\"> <td align='center' bgcolor='#AAFFAA'><input type \"text\" name=\"row1\" value=\"$row[0]\" size =\"4\"></td><td

align='center' bgcolor='#AAFFAA'><input type \"text\" name=\"row2\" value=\"$row[1]\" size =\"5\"></td><td align='center' bgcolor='#AAFFAA'><input type=\"text\" name=\"ID\" value=\"$ID\" hidden><input type=\"submit\" name=\"s1\" value=\"Применить\">

</form></td></tr>";

}

?>

</body>

</html>

**Updated1.php**

<html>

<head>

<style>

h2{

color:yellow;

}

</style>

</head>

<body background="Pict.jpg">

<?PHP include('settings.php');

$ID=$\_POST['ID'];

$room = $\_POST['row2'];

$sql = "SELECT \* from public.upd('$ID','$room');";

$res = $db->query($sql);

$num\_results=$res->rowCount();

$result = $res->fetchAll(PDO::FETCH\_NUM);

echo "<center>";

echo "<h2>Данные изменены</h2>";

echo "<h2>Теперь можно вернуться на главную страницу.</h2>";

echo "<form action=\"index.php\" method=\"post\"> ";

echo "<input type=\"Submit\" value=\"Вернуться\">";

echo "</form>";

?>

</body>

</html>

**Updated2.php**

<html>

<body background="Pict.jpg">

<?PHP include('settings.php');

$quant=$\_POST['quant'];

$sql = "SELECT \* FROM public.list\_audiences where Number\_Seats >= '$quant'";

$res = $db->query($sql);

$num\_results=$res->rowCount();

$result = $res->fetchAll(PDO::FETCH\_NUM);

echo '<center>';

echo '<br><br><br><br><br><br>';

echo "<form action=\"Audience.php\" method=\"post\"> ";

echo "<input type=\"Submit\" value=\"Назад\">";

echo "</form>";

echo '<table border=1>';

echo '<tr>'.'<th bgcolor="#CCCCFF" >Номер Аудитории</th>'.'<th bgcolor="#CCCCFF">Кол-во мест</th>'.'<th bgcolor="#CCCCFF">Изменение</th>'.'</tr>';

foreach($result as $row){

$ID=$row[0];

echo "<tr><td align='center' bgcolor='#AAFFAA'>$row[0]</td><td

align='center' bgcolor='#AAFFAA'>$row[1]</td><td align='center' bgcolor='#AAFFAA'><form name=\"f1\" method=\"POST\" action=\"Update.php\"><input type=\"text\" name=\"ID\" value=\"$ID\" hidden><input type=\"submit\" name=\"s1\" value=\"Изменить\">

</form></td></tr>";

}

?>

</body>

</html>

