Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Кафедра электронных вычислительных машин

Лабораторная работа №6

«Создание приложения для базы данных»

Вариант № 15 – Кафе (бар)

Выполнила Проверила:

студент группы 150503: Игнатович А. О.

Петрович Е. А.

МИНСК 2024

1. **Цель работы**

Создание прикладной программы для работы с базой данных, которая выполняет заданные транзакции. Реализация механизма работы с базой данных (добавление новых данных в таблицу, удаление, обновление). Можно использовать любую среду и язык программирования.

1. **Выполнение работы**
   1. **Описание приложения**

Используемые технологии:

* JavaScript – серверная часть
* JavaScript, CSS, HTML – клиентская часть
* PostgreSQL – база данных

Программа представляет собой веб-приложение, состоящее из нескольких страниц. Элементы переключения между страницами – кнопки. На главной странице приложения с помощью кнопок пользователь может вывести содержание таблиц базы данных, ввести свой запрос или же перейти на другие страницы. На двух других страницах можно вывести результат запросов, которые были сделаны в 4 и 5 лабораторных работах.

Пользователь с помощью кнопок выбирает запрос, который нужно отправить на сервер, или же вводит запрос сам. Запрос отправляется на сервер, где и обрабатывается. Сервер пытается подключиться к базе данных, выполняет запрос, результат возвращается серверу в формате JSON. Далее сервер отправляет полученные данные клиентской части, после чего все выводится на экран пользователя в виде таблицы.

* 1. **Разработка серверной части приложения**

Серверная часть написана на языке JavaScript. Как уже было сказано, сервер получает запросы от клиента, обрабатывает их, подключается к базе данных и выполняет запрос. Код подключения к базе данных приведен ниже.

const pool = new Pool({

user: 'postgres',

host: 'localhost',

database: 'cafe-bar',

password: '2223',

port: 5432,

});

Далее в серверной части прописаны обработчики запросов для вывода всех таблиц и для пользовательских запросов. Ниже приведен обработчик запроса на примере таблицы “посетители”.

app.get('/visitors', async (req, res) => {

try {

const client = await pool.connect();

const result = await client.query('SELECT \* FROM visitor');

const visitors = result.rows;

res.json(visitors);

client.release();

} catch (err) {

console.error(err);

res.status(500).send('Internal server error');

}

});

Обработчик пользовательских запросов:

app.get('/custom-query', async (req, res) => {

const { query } = req.query;

try {

const client = await pool.connect();

const result = await client.query(query);

const data = result.rows;

res.json(data);

client.release();

} catch (err) {

console.error(err);

res.status(500).send('Internal server error');

}

});

Все данные приходят на сервер в формате JSON. Просмотреть результаты выполнения запросов можно уже, не написав клиентскую часть по адресу <http://localhost:3001>. Таблица посетителей в формате JSON будет приведена на рисунке 2.1.

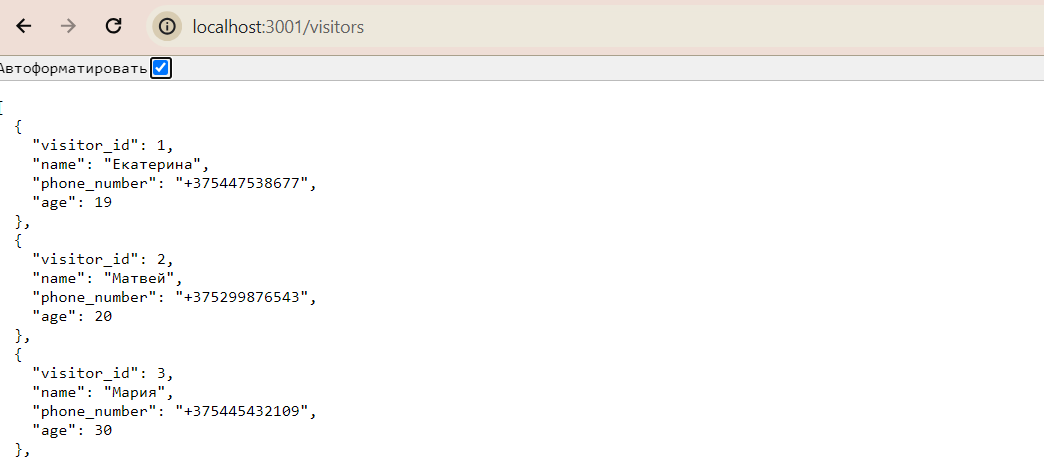


Рисунок 2.1 – Таблица посетителей

* 1. **Разработка клиентской части**

Клиентская часть позволяет создать полноценное веб-приложение. Как уже говорилось, на главном экране есть кнопки для вывода данных из таблиц, поле для ввода пользовательского запроса и кнопки для перехода на другие страницы.

Сначала в коде объявляются переменные, которые отвечают за хранение данных из таблиц и за состояние видимости таблицы (то есть, пока пользователь не захочет просмотреть таблицу, она будет скрыта). Например, переменные для таблицы посетителей:

const [visitors, setVisitors] = useState([]);

const [visitorsVisible, setVisitorsVisible] = useState(false);

Далее, если пользователь захочет просмотреть таблицу посетителей, переменная visitorsVisible будет установлена в соответствующее значение и выполнится участок кода для получения данных о посетителях:

if (visitorsVisible) {

fetch('/visitors')

.then(response => response.json())

.then(data => setVisitors(data))

.catch(error => console.error('Ошибка:', error));

}

Функция toggleVisitors() меняет значение переменной для видимости таблицы. Она вызывается, когда пользователь нажмет на кнопку “показать таблицу”.

В функции return прописан HTML код для вывода и структурирования данных на странице. Объектам присвоены классы, чтобы сделать минимальный дизайн в файле .css. Для примера будет приведен код вывода поля для ввода пользовательского запроса, кнопки для его отправки на сервер и вывода таблицы.

<h2>Введите пользовательский запрос:</h2>

<div className="request">

<input type="text" value={query} onChange={handleInputChange} className="input" />

<button onClick={executeQuery} className="myButton">Выполнить запрос</button>

<table className="myTable">

<tbody>

{data0.map((row, rowIndex) => (

<tr key={rowIndex}>

{Object.values(row).map((cell, cellIndex) => (

<td key={cellIndex}>{cell}</td>

))}

</tr>

))}

</tbody>

</table>

</div>

Функции для всех таблиц аналогичные, на страницах с запросами из 4 и 5 лабораторных работ используются те же участки кода. Полный листинг кода с серверной и клиентской частью приведен в репозитории:

<https://github.com/kattymwk/cafe-bar>

* 1. **Демонстрация работы приложения**

На рисунке 2.2 приведена основная страница приложения. На рисунке 2.3 – вывод таблицы посетителей.

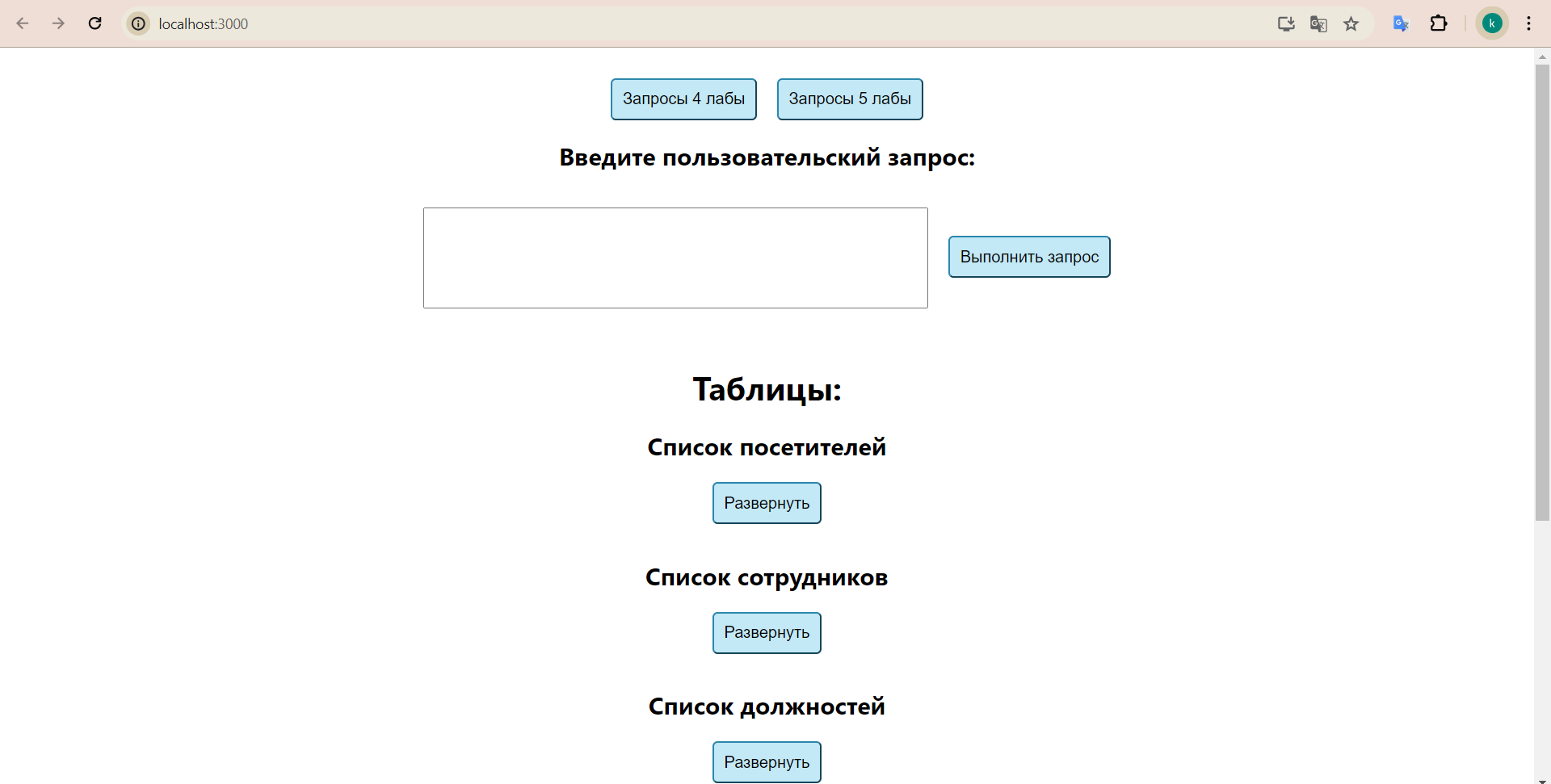


Рисунок 2.2 – Главная страница приложения

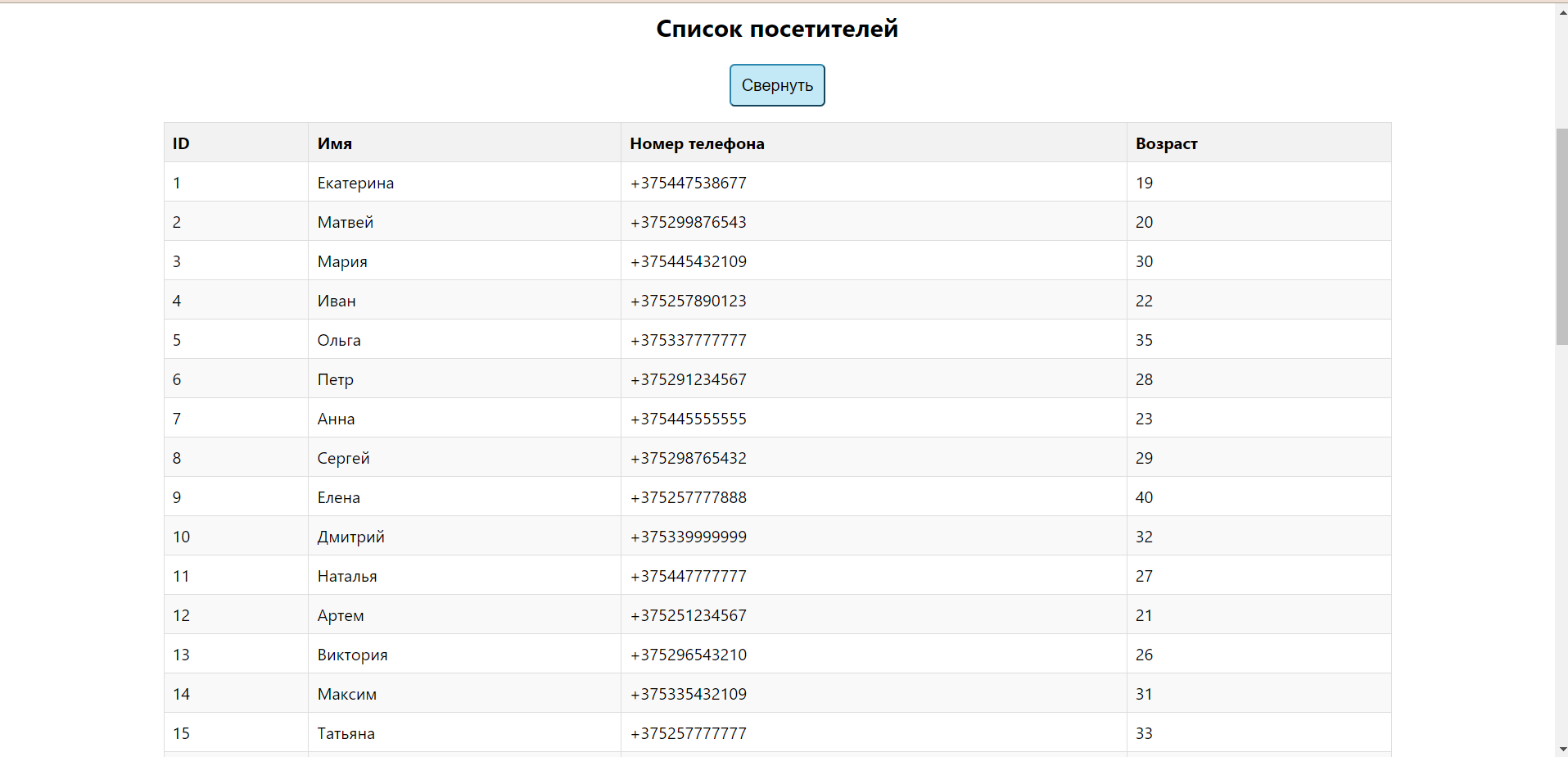


Рисунок 2.3 – Таблица посетителей

Далее показана работа поля для ввода пользовательского запроса на примере добавления строки в таблицу посетителей. Результат запроса будет показан на рисунке 2.4. Сам запрос приведен ниже.

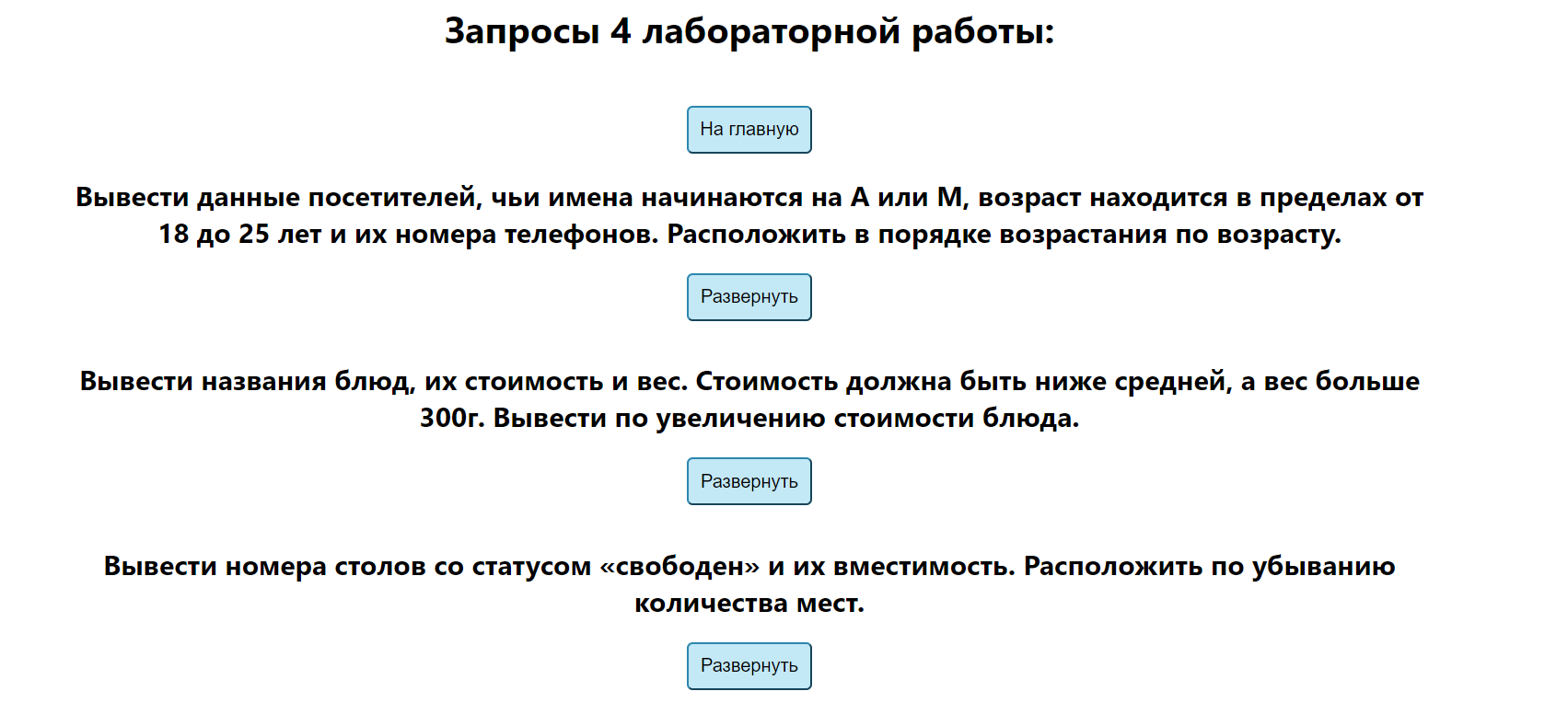
INSERT INTO visitor (visitor\_id, name, phone\_number, age)

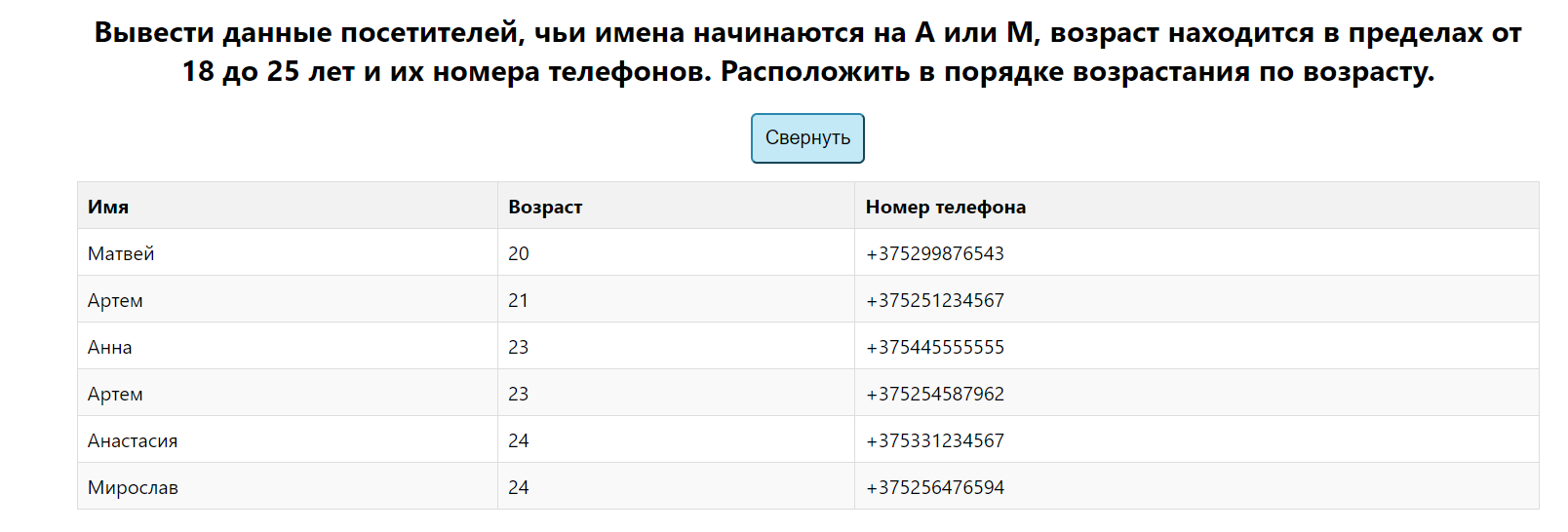
VALUES (DEFAULT, 'Лиана', '+375440102030', 33)



Рисунок 2.4 – Результат выполнения пользовательского запроса

На рисунках 2.5 и 2.7 приведены страницы с запросами из 4 и5 лабораторных работ соответственно. На рисунках 2.6 и 2.8 – вывод таблицы для одного из запросов.

Рисунок 2.5 – Страница с запросами 4 лабораторной

Рисунок 2.6 – Запрос 4 лабораторной

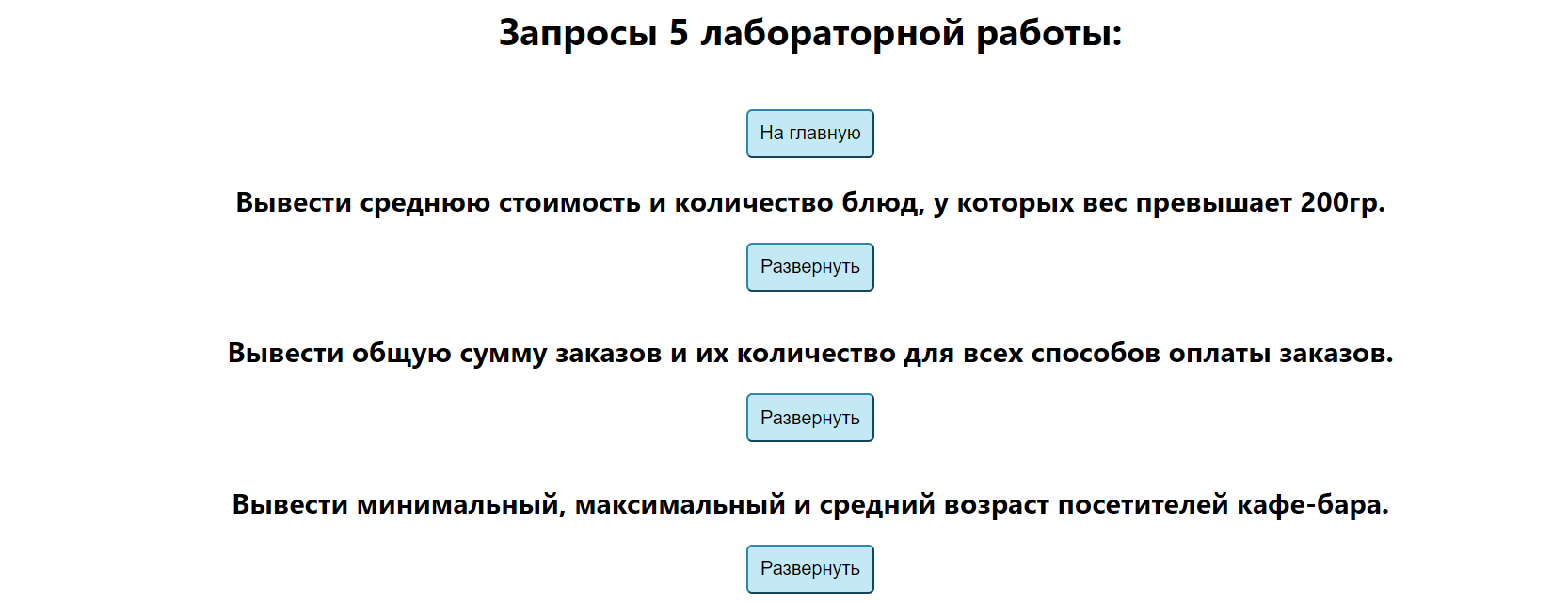


Рисунок 2.7 – Страница с запросами 5 лабораторной

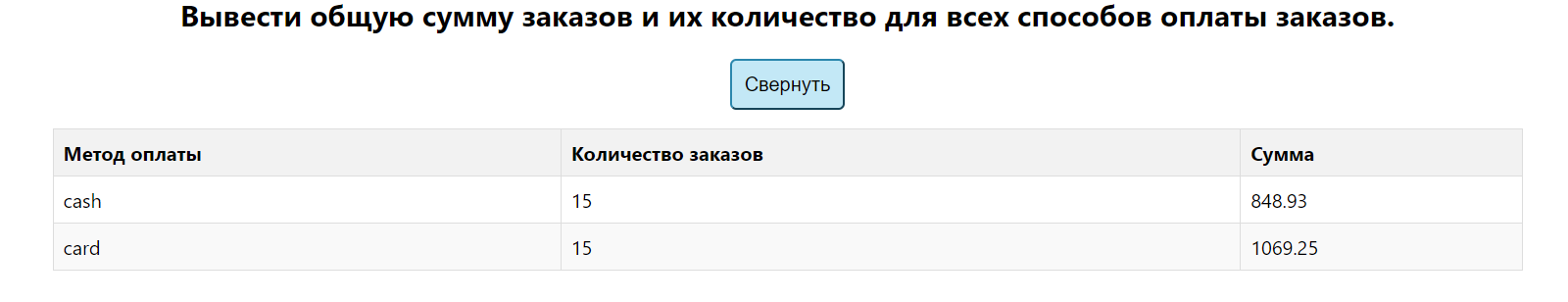


Рисунок 2.8 – Запрос 5 лабораторной