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 № 24 – прокат видеодисков

Выполнил
студент группы 150503:
Ходосевич М. А.

Проверила:
Игнатович А. О.

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.

2 Выполнение работы

2.1 Описание приложения

Используемые технологии:

- Node.js – серверная часть
- React.js – 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.

2.2 Разработка серверной части приложения

Серверная часть написана JavaScript фреймворке React. 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: 'prokat_disks',
  password: '',
  port: 5433,
});
```

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

```
const getTableData = async (req, res, table) => {
  try {
    const client = await pool.connect();
    const result = await client.query(`SELECT * FROM
${table}`);
    const data = result.rows;
    res.json(data);
    client.release();
  } catch (err) {
    console.error(err);
    res.status(500).send('Internal server error');
  }
};

app.get('/client', async (req, res) => {
  await getTableData(req, res, 'client');
});
```

Обработчик пользовательских запросов:

```
app.get('/query', async (req, res) => {
  const { query } = req.query;
  try {
    const result = await pool.query(query);
    res.json(result.rows);
  } 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 клиентов в формате JSON будет приведена на рисунке 2.1.

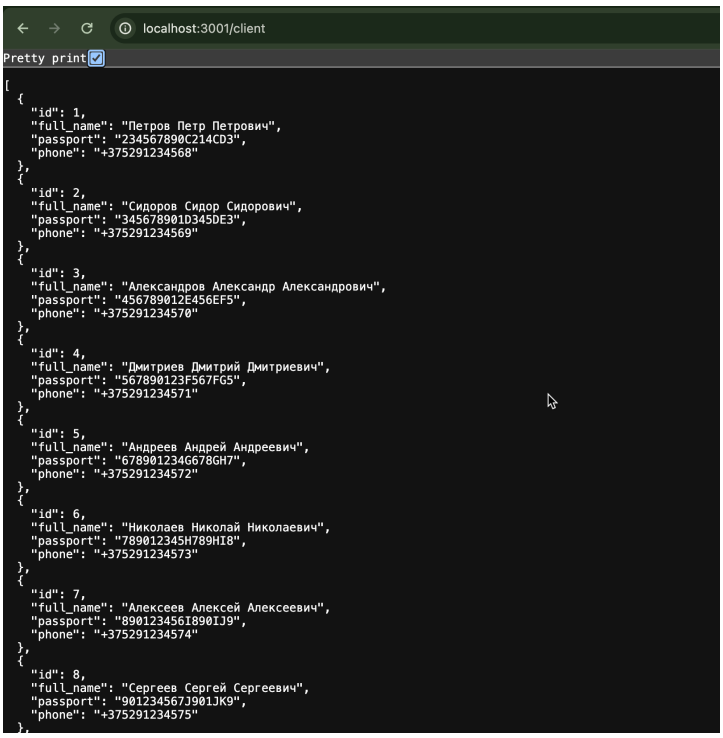


Рисунок 2.1 – Таблица клиентов

2.3 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.

В коде есть несколько компонент, которые отвечают за корректную работы веб-приложения.

Компонента `App.jsx` отвечает за роутинг программы:

```jsx
import React from 'react';
import { BrowserRouter as Router, Route, Routes } from 'react-router-dom';
import Home from './Home';
import Laba4 from "./components/Laba4";
import Laba5 from "./components/Laba5";

function App() {
    return (
        <Router>
            <Routes>
                <Route path="/" element={<Home/>} />
                <Route path="/laba4" element={<Laba4/>} />
                <Route path="/laba5" element={<Laba5/>} />
            </Routes>
        </Router>
    );
}

export default App;
```

Компонента `App.jsx`

Компонента `Home.jsx` является самой главной и в ней просходит отрисовка таблиц и кнопок:

```
import React, {useState} from 'react';
import {Box,Button,Container,CssBaseline,Paper, Table, TableBody, TableCell,
TableContainer,TableRow,TextField,
} from "@mui/material";
import MyTable from "./Table";
import MyNavLink from "./components/MyNavLink";

function Home() {

    const clientHeaders = ["id", "full_name", "passport", "phone"];
    const diskHeaders = ["id", "rental_cost", "quantity", "state"];
    const filmHeaders = ["id", "name", "genre", "year","director_id"];
    const orderHeaders = ["id", "time_get", "time_out", "sum","client_id"];
    const reviewHeaders = ["id", "mark", "comment", "client_id","film_id"];

    const [searchInput, setSearchInput] = useState("")
    const [data, setData] = useState([]);

    const queryRequest = () => {
        fetch(`/query?query=${encodeURIComponent(searchInput)}`)
            .then(response => response.json())
            .then(data => setData(data))
            .catch(error => console.error('Ошибка:', error));
    }

    return (
        <Box>
            <Container>
                <CssBaseline/>
                <Box sx={{
                    display:"flex",
                    gap:"20px",
                    width:"100%",
                    margin:"30px 0",
                }}>
                    <MyNavLink link={"/laba4"} name={"lb4"}/>
                    <MyNavLink link={"/laba5"} name={"154"}/>
                </Box>

                <Box
                    sx={{margin: "50px 0", display: "flex", gap: "20px", alignItems: "start",
flexDirection: "column"}}>
                    <TextField sx={{width: 400}} onChange={(e) => setSearchInput(e.target.value)}
fullWidth
                               label="Свой запрос" id="fullWidth"/>
                    <Button onClick={queryRequest} sx={{height: "40px", width: 100}}
variant="contained"
                            color="success">
                        клац
                    </Button>
                    <TableContainer sx={{margin:"20px 0"}} component={Paper}>
                        <Table sx={{minWidth: 650}} aria-label="simple table">
                            <TableBody>
                                {data.map((row, rowIndex) => (
                                    <TableRow key={rowIndex}>
                                        {Object.values(row).map((cell, cellIndex) => (
                                            <TableCell key={cellIndex}>{cell}</TableCell>
                                        ))}
                                    </TableRow>
                                ))}
                            </TableBody>
                        </Table>
                    </TableContainer>
                </Box>
                <Box sx={{display: "flex", flexDirection: "column", gap: "20px"}}>
                    <MyTable title={"Список клиентов"} request={"/client"} headers={clientHeaders}/>
                    <MyTable title={"Список дисков"} request={"/disks"} headers={diskHeaders}/>
                    <MyTable title={"Список фильмов"} request={"/film"} headers={filmHeaders}/>
                    <MyTable title={"Список отзывов"} request={"/reviews"} headers={reviewHeaders}/>
                    <MyTable title={"Список заказов"} request={"/orders"} headers={orderHeaders}/>
                </Box>
            </Container>
        </Box>
    );
}
export default Home;
```

Компонента `Home.jsx`

Компонента `Table.jsx` отвечает за отрисовку таблицы.

```
import React, {useEffect, useState} from 'react';
import {Box, Button, Table, Paper, TableBody, TableCell, TableContainer, TableHead, TableRow,
Typography} from "@mui/material";

const MyTable = ({title, request, headers}) => {

    const [tableData, setTableData] = useState([]);
    const [tableVisible, setTableVisible] = useState(false);

    const fetchTable = () => {
        fetch(request)
            .then(response => response.json())
            .then(data => setTableData(data))
            .catch(error => console.error('Ошибка:', error));
    }

    const toggleOpen = () => {
        setTableVisible(!tableVisible);
    };

    useEffect(() => {
        if (tableVisible)
            fetchTable();
    },[tableVisible])

    return (
        <Box sx={{width:"100%"}}>
            <Typography>{title}</Typography>
            <Button sx={{marginTop:"10px"}} variant="contained"
onClick={toggleOpen}>{tableVisible ? 'Свернуть' : 'Развернуть'}</Button>
            {
                tableVisible &&
                <TableContainer sx={{margin:"20px 0"}} component={Paper}>
                    <Table sx={{minWidth: 650}} aria-label="simple table">
                        <TableHead>
                            <TableRow>
                                {headers.map(header => (
                                    <TableCell key={header}>{header}</TableCell>
                                ))}
                            </TableRow>
                        </TableHead>
                        <TableBody>
                            {
                                tableData.map(item => (
                                    <TableRow
                                        key={item.id}
                                    >
                                        {headers.map(header => (
                                            <TableCell key={header}>{item[header]}</TableCell>
                                        ))}
                                    </TableRow>
                                ))
                            }
                        </TableBody>
                    </Table>
                </TableContainer>
            }
        </Box>
    );
};

export default MyTable;
```

Компонента `Table.jsx`

2.4 Демонстрация работы приложения

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

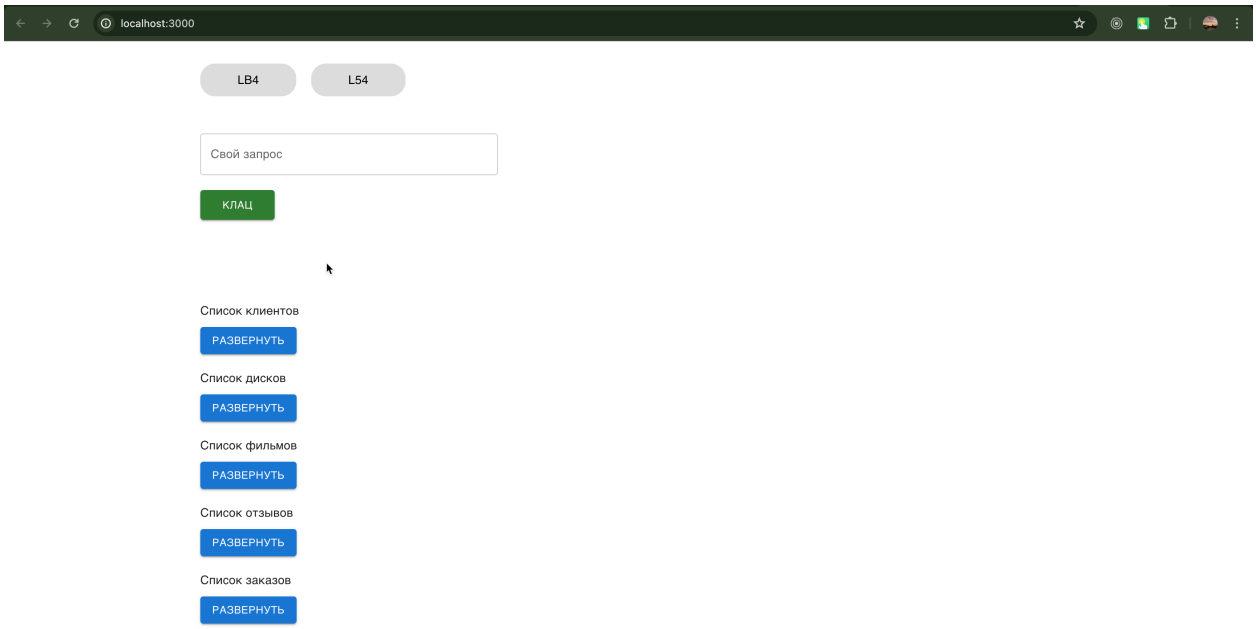
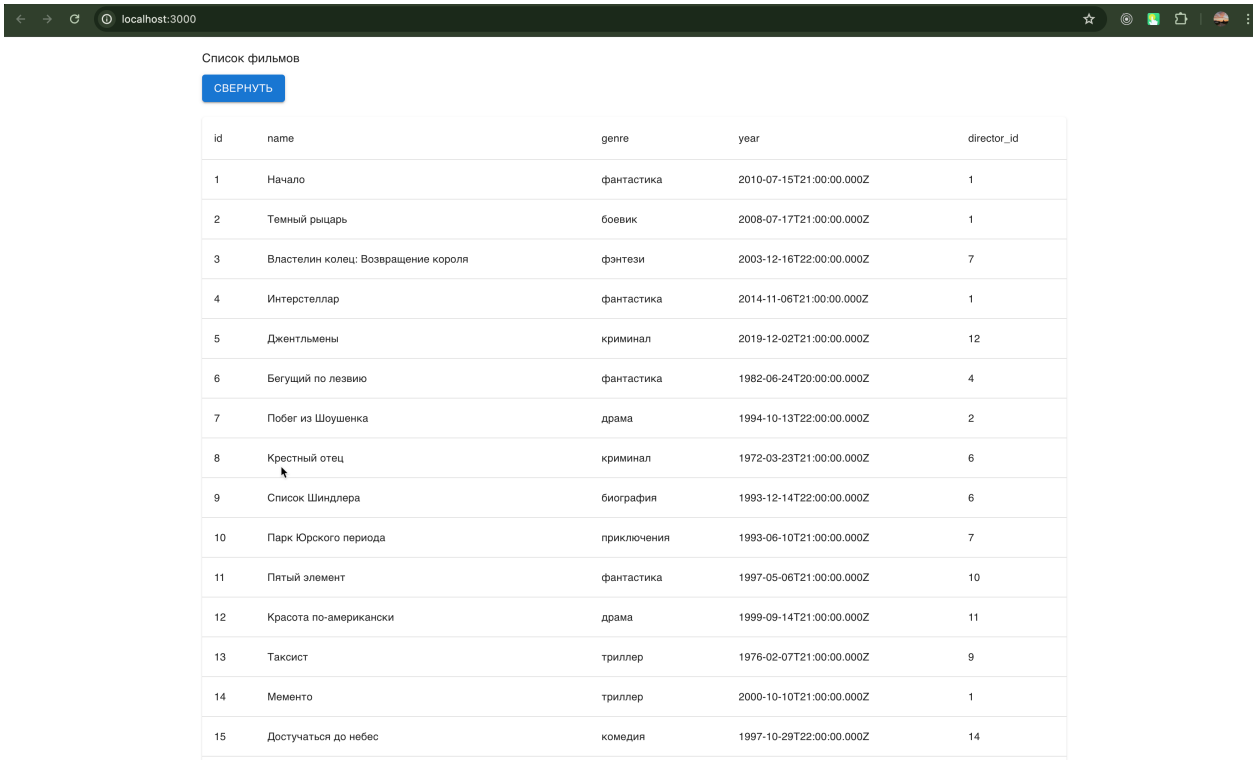


Рисунок 2.2 – Главная страница приложения



id	name	genre	year	director_id
1	Начало	фантастика	2010-07-15T21:00:00.000Z	1
2	Темный рыцарь	боевик	2008-07-17T21:00:00.000Z	1
3	Властелин колец: Возвращение короля	фэнтези	2003-12-16T22:00:00.000Z	7
4	Интерстеллар	фантастика	2014-11-06T21:00:00.000Z	1
5	Джентльмены	криминал	2019-12-02T21:00:00.000Z	12
6	Бегущий по лезвию	фантастика	1982-06-24T20:00:00.000Z	4
7	Побег из Шоушенка	драма	1994-10-13T22:00:00.000Z	2
8	Крестный отец	криминал	1972-03-23T21:00:00.000Z	6
9	Список Шиндлера	биография	1993-12-14T22:00:00.000Z	6
10	Парк Юрского периода	приключения	1993-06-10T21:00:00.000Z	7
11	Пятый элемент	фантастика	1997-05-06T21:00:00.000Z	10
12	Красота по-американски	драма	1999-09-14T21:00:00.000Z	11
13	Таксист	триллер	1976-02-07T21:00:00.000Z	9
14	Мементо	триллер	2000-10-10T21:00:00.000Z	1
15	Достучаться до небес	комедия	1997-10-29T22:00:00.000Z	14

Рисунок 2.3 – Таблица фильмов

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.

```
SELECT full_name FROM client where id IN( SELECT id FROM client
INTERSECT SELECT client_id from review);
```

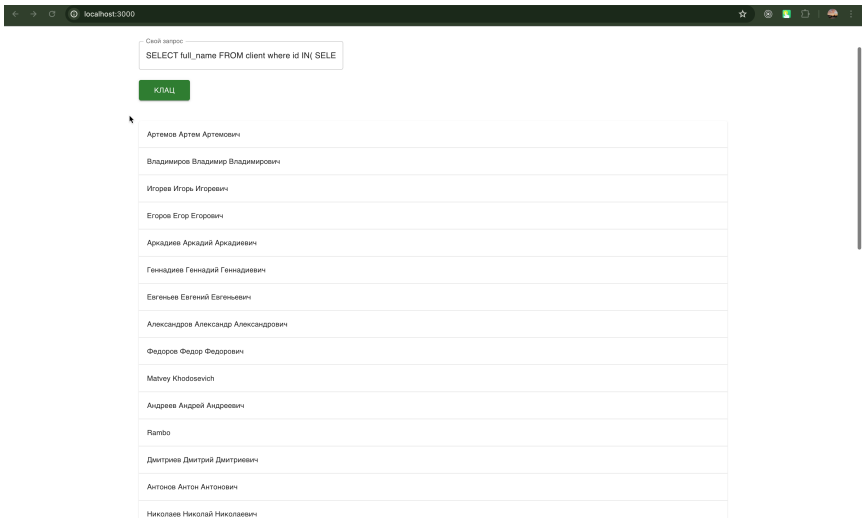


Рисунок 2.4 – Результат выполнения пользовательского запроса

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.

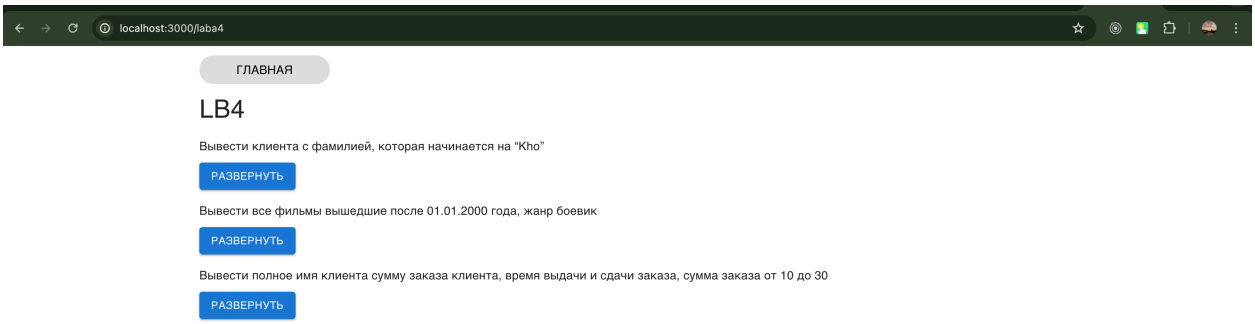


Рисунок 2.5 – Страница с запросами 4 лабораторной

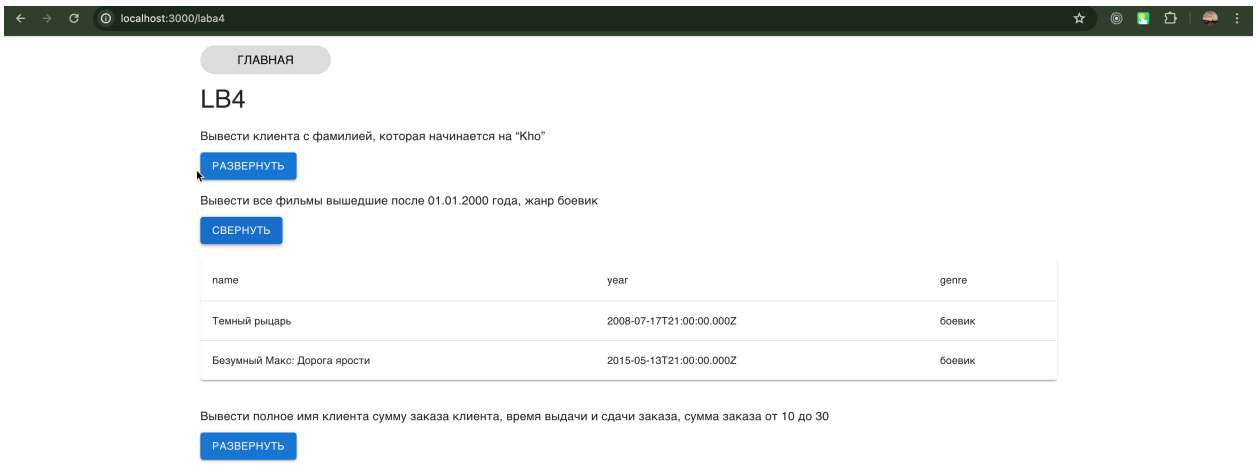


Рисунок 2.6 – Запрос 4 лабораторной

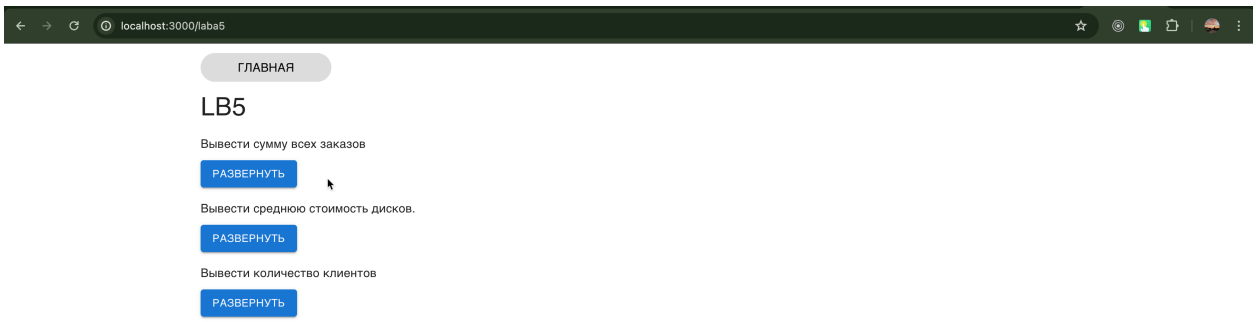


Рисунок 2.7 – Страница с запросами 5 лабораторной

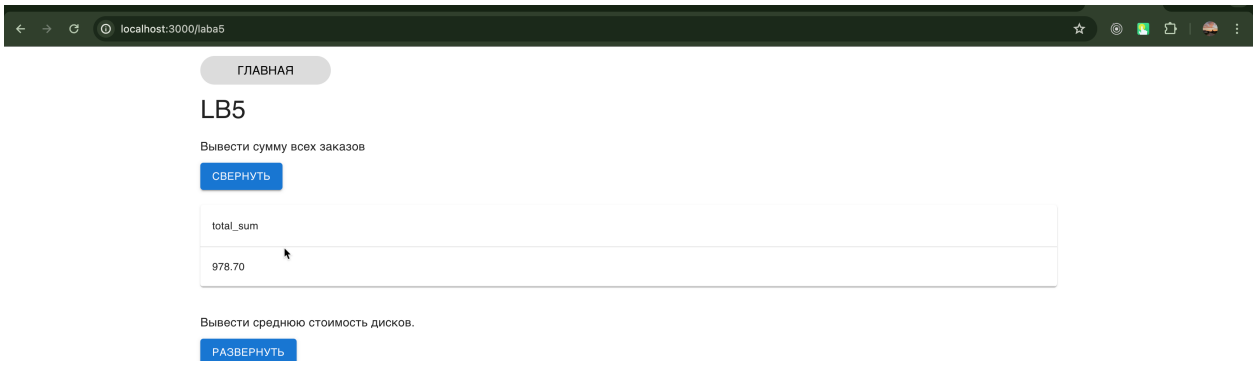


Рисунок 2.8 – Запрос 5 лабораторной