Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное
учреждение высшего образования
«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Инженерно-технологическая академия
Институт компьютерных технологий и информационной безопасности

Отчет по практической работе № 4
на тему «**Работа с SQL БД в Web-приложениях Node.js**»
по дисциплине «Технологии разработки серверной части веб-приложений»

Выполнила
студентка группы КТбо3-4 _____ Н. И. Селевцова

Принял
доцент кафедры МОП ЭВМ _____ А. Н. Шкурко

Таганрог 2023

СОДЕРЖАНИЕ

ЦЕЛЬ РАБОТЫ

Данная практическая работа направлена на изучение и практическое использование базовых методов доступа к БД использующих SQL в web-приложениях Node.js.

Задание

В рамках выполнения практической работы требуется выполнить следующие задачи и продемонстрировать полученные навыки:

– Создать новое node.js приложение;

– Подключить к приложению библиотеки express.js и liquid;

– Разработать HTML-шаблоны для страниц приложения;

– Реализовать функциональность доступа к БД, необходимую для выполнения задания;

– Реализовать логику обработки на сервере согласно варианту задания.

Вариант задачи: 5

Необходимо реализовать простое приложение доски объявлений. На главной странице приложения пользователю отображается список добавленных объявлений. Отдельно доступна страница добавления объявления. Также на отдельной странице выводятся все объявления и доступно удаление отдельных объявлений по кнопке.

ХОД РАБОТЫ

Проект разрабатываемого веб-приложения имеет следующую файловую структуру – рисунок 1. В приложение были подключены библиотеки express.js и liquid, установлено соединение с локальным сервером и базой данных PostgreSQL.

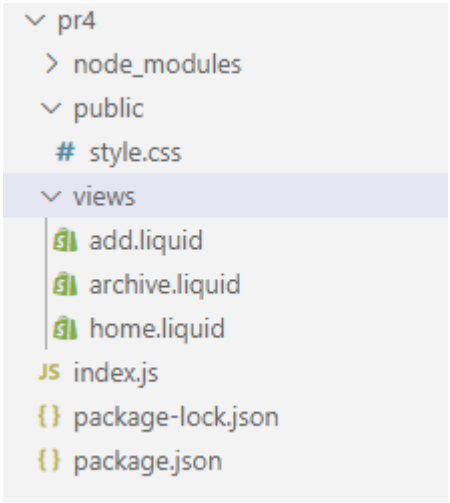


Рисунок 1 – Файловая структура приложения

```
const express = require('express');
const bodyParser = require('body-parser');
const { Sequelize, DataTypes } = require('sequelize');

const app = express();
app.use(express.static('public'));
const port = 3000;

var { Liquid } = require('liquidjs');
var engine = new Liquid();
app.use(bodyParser.urlencoded());
app.engine('liquid', engine.express());
app.set('views', './views');            // specify the views directory
app.set('view engine', 'liquid');       // set liquid to default


const sequelize = new Sequelize('postgres://postgres:1@localhost:5432/postgres');
```

Рисунок 2 – Начальные настройки

На языке HTML был создан шаблон страницы приложения. Был использован шаблонизатор Liquid вместе с промежуточным обработчиком bodyParser. Во время выполнения механизм шаблонов заменяет переменные в файле шаблона фактическими значениями и преобразует шаблон в HTML-

файл, отправляемый клиенту. Такой подход упрощает разработку HTML-страницы.

На рисунках 3-5 представлено содержание HTML-страниц с доской объявлений, страницей добавления и удаления объявлений.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Объявления</title>
    <link href="style.css" rel="stylesheet">
  </head>
  <body>
   <div> <button><a href="/add">Добавить объявление</a></button>
   <button><a href="/arc">Удалить объявление</a></button></div>
    <div>
      <br>
      {% for a in active %}
        <div id="col">
          {{ a.text }}<br>
          {{ a.name }}<br>
          {{ a.phone }}<br>
          </div>
          <br>
    {% endfor %}
      </div>
    </body>
  </body>
```

Рисунок 3 – Страница с добавленными объявлениями

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Добавить</title>
    <link href="style.css" rel="stylesheet">
  </head>
  <body>
    <h2>Добавьте свое объявление!</h2>
<form action="/add" method="POST">
    <p>Текст вашего объявления:</p> <input  type="text" name="text" /><br>
    <p>Как к вам обращаться?</p><input type="text" name="name" /><br>
    <p>Номер телефона:</p><input type="text" name="phone" /><br><br>
     <input type="submit" value="Добавить объявление" />
    </form>

    </body>
    </body>
</html>
```

Рисунок 4 – Страница добавления объявления

```
<!DOCTYPE html>
<html
  lang="en">
  <head>
    <meta
      charset="UTF-8">
    <meta
      http-equiv="X-UA-Compatible"
      content="IE=edge">
    <meta
      name="viewport"
      content="width=device-width, initial-scale=1.0">
    <title>Удалить</title>
    <link
      href="style.css"
      rel="stylesheet">
  </head>
  <body>
    <h2>Удалить объявление</h2>
    <form
      action="/delete"
      method="POST">
      <div>
        <br>
        {% for a in active %}
        <div
          id="col">
          {{ a.text }}<br>
          {{ a.name }}<br>
          {{ a.phone }}<br>

        </div>
        <br>
        <input
          type="submit"
          name="delete_{{a.id}}"
          value="Удалить" />
        <br><br>
        {% endfor %}
      </form>
    </body>
  </body>
</html>
```

Рисунок 5 – Страница удаления объявления

На рисунках 6-7 представлен код для функциональности доступа к БД. Далее представлен ход работы с приложением, отображенный в БД.

```
const Ads = sequelize.define('Ads', {
    text: {
      type: DataTypes.STRING,
      allowNull: false
    },
    name:{
      type: DataTypes.STRING,
      allowNull: false
    },
    phone:{
      type: DataTypes.STRING,
      allowNull: false
    }
  });

  app.get('/', async (req, res) => {
    const a = await Ads.findAll();
    const model = {
      active: a.map(x => x.get({plain:true})),
    }
    res.render('home', model);
});

app.get('/add', (req, res) => res.render('add'));

app.post('/add', async (req, res) => {
  const ad = new Ads();
  ad.text = req.body.text;
  ad.name = req.body.name;
  ad.phone = req.body.phone;
  await ad.save();
  res.redirect('/');
});
```

Рисунок 6 – Реализация работы БД в приложении

```
app.get('/arc', async (req, res) => {
  const a = await Ads.findAll();
    const model = {
      active: a.map(x => x.get({plain:true})),
    }
    res.render('archive', model);
});

app.post('/delete', async (req, res) => {

  const deletedKeys = Object.keys(req.body)
      .filter(x=>x.startsWith('delete_'))
      .map(x => x.split('_')[1]);

  for (let id of deletedKeys) {
      const ad = await Ads.findByPk(id);
      await ad.destroy();
  }
  res.redirect('/');
});


Ads.sync();

app.listen(port, () => console.log('Server is running on port 3000'));
```

Рисунок 7 - Реализация работы БД в приложении [2]

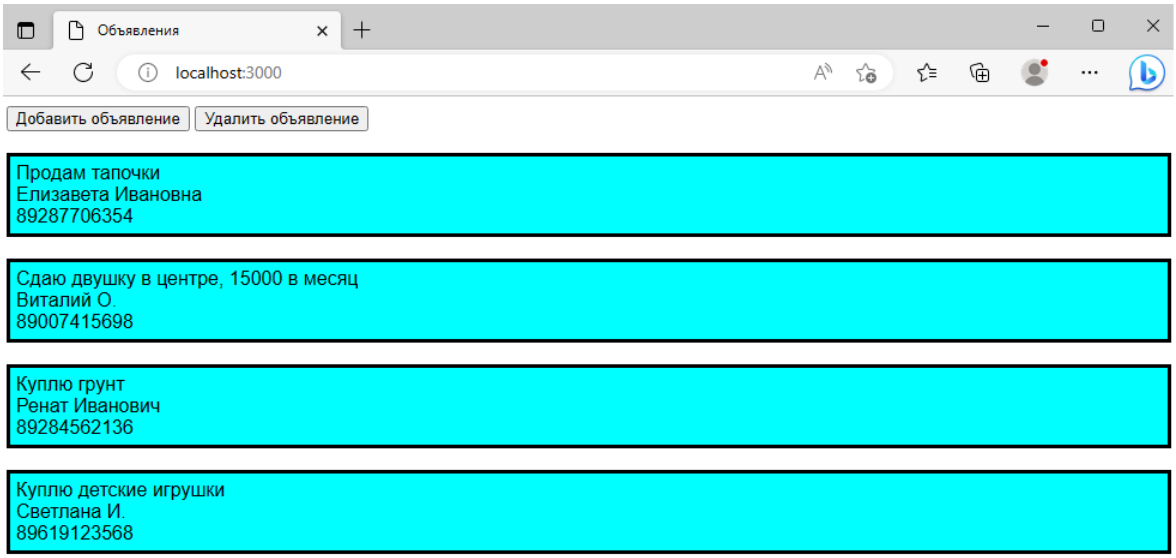


Рисунок 8 – Страница с объявлениями

Data Output | Messages | Notifications

	id [PK] integer	text character varying (255)	name character varying (255)	phone character varying (255)	createdAt timestamp with time zone	up
1	6	Продам тапочки	Елизавета Ивановна	89287706354	2023-04-26 21:55:26.752+03	20
2	8	Сдаю двушку в центре, 15000 в месяц	Виталий О.	89007415698	2023-04-26 22:03:56.403+03	20
3	14	Куплю грунт	Ренат Иванович	89284562136	2023-04-26 22:24:22.904+03	20
4	15	Куплю детские игрушки	Светлана И.	89619123568	2023-04-26 22:27:52.593+03	20

Рисунок 9 – Данные в БД

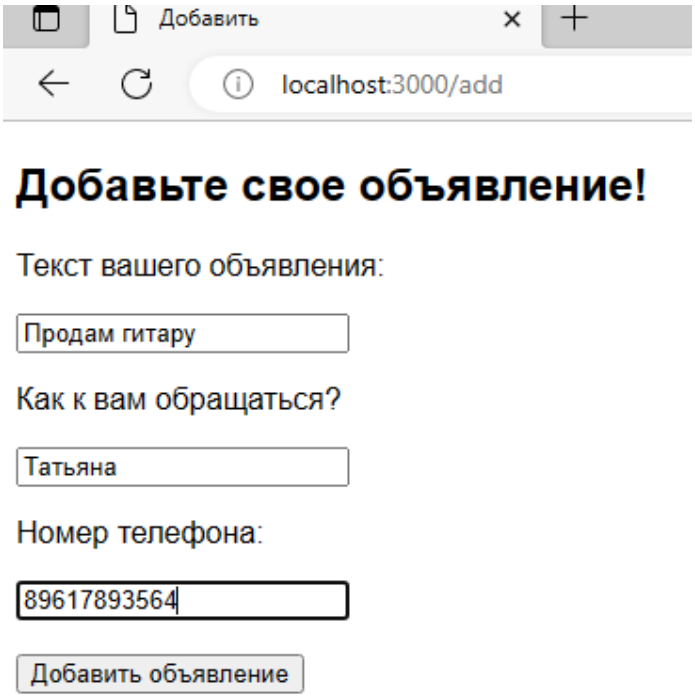


Рисунок 10 – Добавление объявления

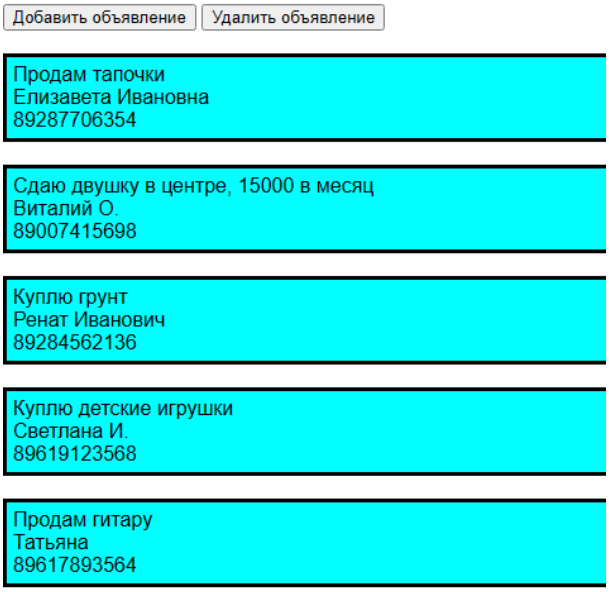
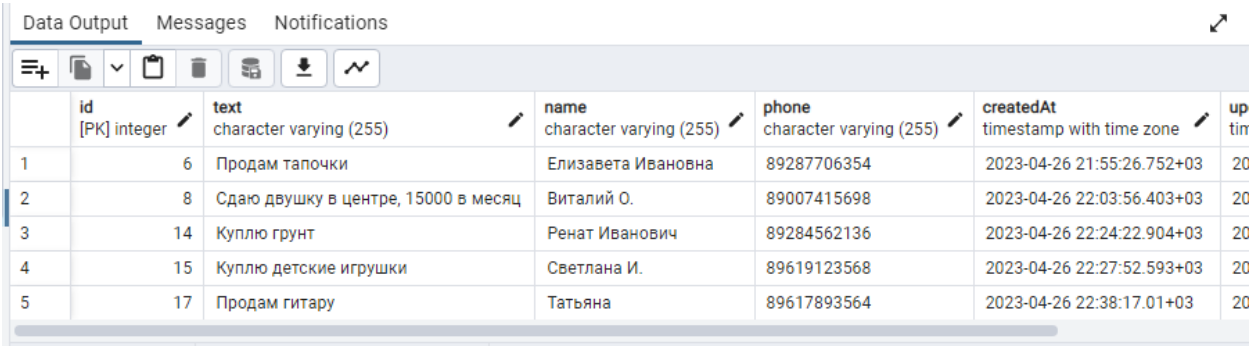


Рисунок 11 – Появление объявления на странице



	id [PK] integer	text character varying (255)	name character varying (255)	phone character varying (255)	createdAt timestamp with time zone	up tim
1	6	Продам тапочки	Елизавета Ивановна	89287706354	2023-04-26 21:55:26.752+03	20
2	8	Сдаю двушку в центре, 15000 в месяц	Виталий О.	89007415698	2023-04-26 22:03:56.403+03	20
3	14	Куплю грунт	Ренат Иванович	89284562136	2023-04-26 22:24:22.904+03	20
4	15	Куплю детские игрушки	Светлана И.	89619123568	2023-04-26 22:27:52.593+03	20
5	17	Продам гитару	Татьяна	89617893564	2023-04-26 22:38:17.01+03	20

Рисунок 12 – Обновление в БД

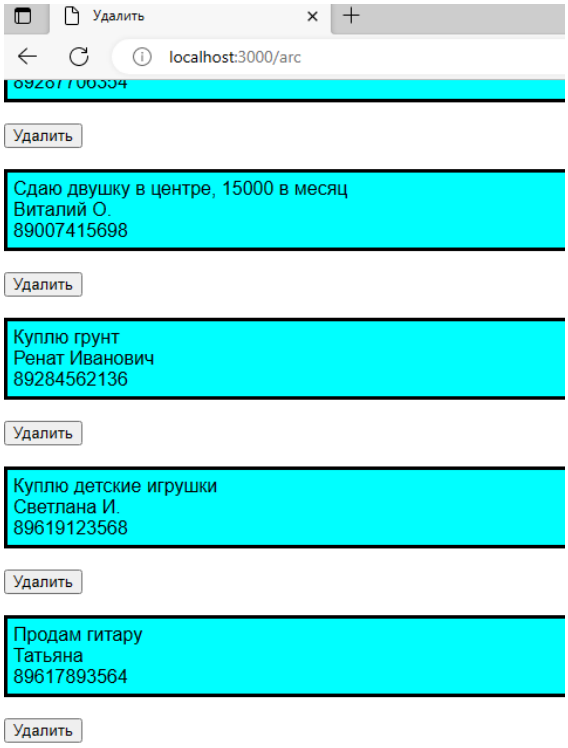


Рисунок 13 – Страница удаления объявлений

Data Output | Messages | Notifications

	id [PK] integer	text character varying (255)	name character varying (255)	phone character varying (255)	createdAt timestamp with time zone	up
1	6	Продам тапочки	Елизавета Ивановна	89287706354	2023-04-26 21:55:26.752+03	20
2	8	Сдаю двушку в центре, 15000 в месяц	Виталий О.	89007415698	2023-04-26 22:03:56.403+03	20
3	14	Куплю грунт	Ренат Иванович	89284562136	2023-04-26 22:24:22.904+03	20
4	15	Куплю детские игрушки	Светлана И.	89619123568	2023-04-26 22:27:52.593+03	20

Total rows: 4 of 4 | Query complete 00:00:00.406 | Successfully run. Total query runtime: 406 msec. 4 rows affected.

Рисунок 14 – Удаление записи из БД

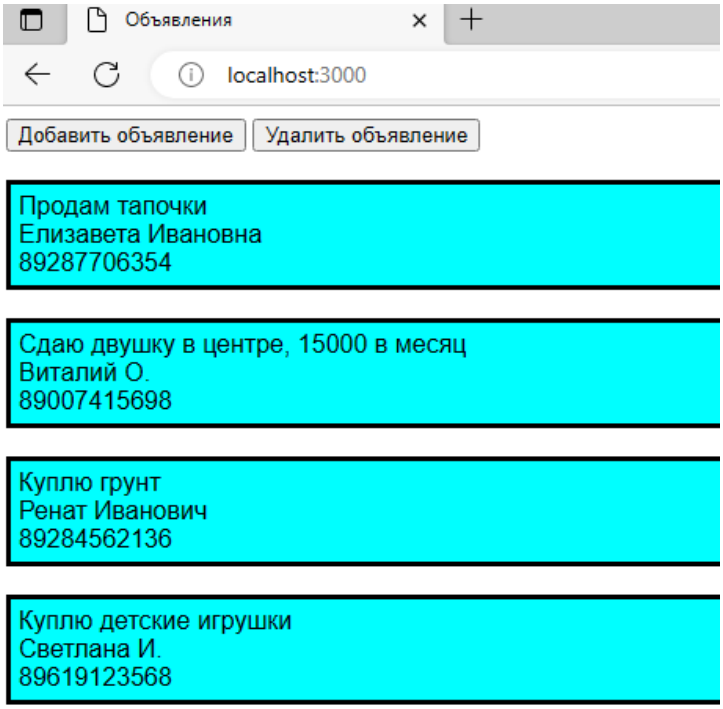


Рисунок 15 – Обновленная информация в приложении

ЗАКЛЮЧЕНИЕ

В ходе практической работы были изучены и использованы базовые методы доступа к БД, использующих SQL в web-приложениях Node.js. Было создано новое node.js приложение, подключены библиотеки express.js и liquid, разработаны HTML-шаблоны для страниц приложения, реализована функциональность доступа к БД, необходимая для выполнения задания, и логика обработки на сервере согласно варианту задания.

ЛИСТИНГ ПРОГРАММЫ

Файл **index.js**

```
const express = require('express');
const bodyParser = require('body-parser');
const { Sequelize, DataTypes } = require('sequelize');

const app = express();
app.use(express.static('public'));
const port = 3000;

var { Liquid } = require('liquidjs');
var engine = new Liquid();
app.use(bodyParser.urlencoded());
app.engine('liquid', engine.express());
app.set('views', './views');            // specify the views directory
app.set('view engine', 'liquid');       // set liquid to default


const sequelize = new
Sequelize('postgres://postgres:1@localhost:5432/postgres');

const Ads = sequelize.define('Ads', {
    text: {
      type: DataTypes.STRING,
      allowNull: false
    },
    name:{
      type: DataTypes.STRING,
      allowNull: false
    },
    phone:{
      type: DataTypes.STRING,
      allowNull: false
    }
  });

  app.get('/', async (req, res) => {
    const a = await Ads.findAll();
    const model = {
      active: a.map(x => x.get({plain:true})),
    }
    res.render('home', model);
});

app.get('/add', (req, res) => res.render('add'));

app.post('/add', async (req, res) => {
  const ad = new Ads();
```

```
  ad.text = req.body.text;
  ad.name = req.body.name;
  ad.phone = req.body.phone;
  await ad.save();
  res.redirect('/');
});

app.get('/arc', async (req, res) => {
  const a = await Ads.findAll();
    const model = {
      active: a.map(x => x.get({plain:true})),
    }
    res.render('archive', model);
});

app.post('/delete', async (req, res) => {

  const deletedKeys = Object.keys(req.body)
      .filter(x=>x.startsWith('delete_'))
      .map(x => x.split('_')[1]);

  for (let id of deletedKeys) {
      const ad = await Ads.findByPk(id);
      await ad.destroy();
  }
  res.redirect('/');
});


Ads.sync();

app.listen(port, () => console.log('Server is running on port 3000'));
```

Файл **home.liquid**

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Объявления</title>
    <link href="style.css" rel="stylesheet">
  </head>
  <body>
   <div> <button><a href="/add">Добавить объявление</a></button>
   <button><a href="/arc">Удалить объявление</a></button></div>
    <div>
      <br>
      {% for a in active %}
        <div id="col">
```

```
            {{ a.text }}<br>
            {{ a.name }}<br>
            {{ a.phone }}<br>
            </div>
            <br>
    {% endfor %}
      </div>
    </body>
  </body>
</html>
```

Файл **add.liquid**

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Добавить</title>
    <link href="style.css" rel="stylesheet">
  </head>
  <body>
    <h2>Добавьте свое объявление!</h2>
<form action="/add" method="POST">
     <p>Текст вашего объявления:</p> <input  type="text" name="text"
/><br>
     <p>Как к вам обращаться?</p><input type="text" name="name" /><br>
     <p>Номер телефона:</p><input type="text" name="phone" /><br><br>
      <input type="submit" value="Добавить объявление" />
    </form>

    </body>
    </body>
  </html>
```

Файл **archive.liquid**

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Удалить</title>
    <link href="style.css" rel="stylesheet">
  </head>
  <body>
    <h2>Удалить объявление</h2>
    <form action="/delete" method="POST">
      <div>
        <br>
```

```
        {% for a in active %}
          <div id="col">
            {{ a.text }}<br>
            {{ a.name }}<br>
            {{ a.phone }}<br>

          </div>
          <br>
          <input
            type="submit"
            name="delete_{{a.id}}"
            value="Удалить" />
          <br><br>
        {% endfor %}
      </form>
    </body>
  </body>
</html>
```

Файл **style.css**

```
#col{
    border: 3px solid black;
    background-color: aqua;
    padding: 5px;
}
*{
    font-family: Arial;
}
a{
    text-decoration: none;
    color: black;
}
```