**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информатика»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 4

по дисциплине **«Разработка приложений для Интернет»**

на тему: **«Http/Https»**

Выполнил: студент гр. ИП-31

Кузнецова Е. А.

Принял: преподаватель-стажёр

Фролова Е. В.

Дата сдачи отчета: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата допуска к защите: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата защиты: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Гомель 2024

**Цель работы:** освоить возможности NodeJs по созданию локальных серверов с выводом статической информации, ознакомиться с фреймворком Express.

**Задание 1.** Создать сервер на NodeJs без использования фреймворка express. На базе сервера разработать статический web-сайт, состоящий из нескольких страниц. Тема выбирается студентом самостоятельно. К страницам сайта применить стилизацию. Предусмотреть переходы между станицами. Страницы хранятся как статические файлы.

**Код программы:**

Server.js

const http = require('http');

const fs = require('fs');

const path = require('path');

const server = http.createServer((req, res) => {

let filePath = '.' + req.url;

if (filePath === './') {

filePath = './index.html';

}

const extname = path.extname(filePath);

let contentType = 'text/html';

switch (extname) {

case '.js':

contentType = 'text/javascript';

break;

case '.css':

contentType = 'text/css';

break;

case '.json':

contentType = 'application/json';

break;

case '.png':

contentType = 'image/png';

break;

case '.jpg':

contentType = 'image/jpg';

break;

case '.ico':

contentType = 'image/x-icon';

break;

}

fs.readFile(filePath, (err, content) => {

if (err) {

if (err.code === 'ENOENT') {

// Если файл не найден, отобразить страницу 404

fs.readFile('./404.html', (err, content) => {

res.writeHead(404, { 'Content-Type': 'text/html' });

res.end(content, 'utf-8');

});

} else {

// Если возникла ошибка сервера, отобразить ошибку 500

res.writeHead(500);

res.end(`Server Error: ${err.code}`);

}

} else {

// Успешный ответ

res.writeHead(200, { 'Content-Type': contentType });

res.end(content, 'utf-8');

}

});

});

const PORT = process.env.PORT || 3000;

server.listen(PORT, () => {

console.log(`Server is running on port ${PORT}`);

});

Index.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Vet Clinic</title>

<link href="style2.css" rel="stylesheet">

</head>

<body>

<header>

<h1 class="h\_index">Vet Clinic</h1>

</header>

<nav>

<a href="home.html">Home</a>

<a href="services.html">Services</a>

<a href="contact.html">Contact</a>

</nav>

<section>

<div class="container">

<h2 class="h\_index">Make an Appointment</h2>

<div class="appointment-form">

<form action="#">

<div class="form-group">

<label for="name">Name:</label>

<input type="text" id="name" name="name" required>

</div>

<div class="form-group">

<label for="email">Email:</label>

<input type="email" id="email" name="email" required>

</div>

<div class="form-group">

<label for="phone">Phone:</label>

<input type="text" id="phone" name="phone">

</div>

<div class="form-group">

<label for="message">Message:</label>

<textarea id="message" name="message" rows="4" required></textarea>

</div>

<div class="form-group">

<input type="submit" value="Submit">

</div>

</form>

</div>

</div>

</section>

</body>

</html>

Home.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Vet Clinic - Home</title>

<link href="style2.css" rel="stylesheet">

</style>

</head>

<body>

<header>

<h1>Welcome to Our Vet Clinic</h1>

</header>

<nav>

<a href="index.html">Make an appointment</a>

<a href="contact.html">Contact</a>

<div style="clear:both"></div>

</nav>

<div class="content">

<h2>Our Mission</h2>

<p>At our vet clinic, we are dedicated to providing compassionate and comprehensive care for your beloved pets. We strive to promote the health and well-being of animals through preventive medicine, advanced diagnostics, and personalized treatment plans.</p>

<h2>Our Services</h2>

<ul>

<li>Routine check-ups and vaccinations</li>

<li>Surgical procedures</li>

<li>Dental care</li>

<li>Diagnostic imaging</li>

<li>Emergency care</li>

</ul>

</div>

<footer>

<p>&copy; 2024 Vet Clinic. All rights reserved.</p>

</footer>

</body>

</html>

Contact.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Vet Clinic - Home</title>

<link href="style2.css" rel="stylesheet">

</head>

<body>

<header>

<h1>Contact Us</h1>

</header>

<nav>

<a href="index.html">Make an appointment</a>

<a href="home.html">Home</a>

<div style="clear:both"></div>

</nav>

<div class="content">

<div class="contact-info">

<h2>Our Location</h2>

<p>123 Main Street</p>

<p>City, State ZIP</p>

<p>Country</p>

</div>

<div class="contact-info">

<h2>Contact Information</h2>

<p>Phone: 123-456-7890</p>

<p>Email: info@example.com</p>

</div>

</div>

<footer>

<p>&copy; 2024 Vet Clinic. All rights reserved.</p>

</footer>

</body>

</html>

Services.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Vet Clinic - Services</title>

<link href="style2.css" rel="stylesheet">

</head>

<body>

<header>

<h1>Our Services</h1>

</header>

<nav>

<a href="index.html">Make an appointment</a>

<a href="contact.html">Contact</a>

<div style="clear:both"></div>

</nav>

<div class="content">

<div class="service-list">

<div class="service">

<h2>Preventive Care</h2>

<p>Regular check-ups and vaccinations to keep your pet healthy.</p>

</div>

<div class="service">

<h2>Surgical Procedures</h2>

<p>Comprehensive surgical care for your pet, including spaying, neutering, and more.</p>

</div>

<div class="service">

<h2>Dental Care</h2>

<p>Professional dental cleanings, extractions, and dental health assessments.</p>

</div>

<div class="service">

<h2>Diagnostic Imaging</h2>

<p>Advanced imaging services, such as X-rays and ultrasound, to diagnose and treat your pet's conditions.</p>

</div>

<div class="service">

<h2>Emergency Care</h2>

<p>24/7 emergency services for your pet's urgent medical needs.</p>

</div>

</div>

</div>

<footer class="footer\_special">

<p>&copy; 2024 Vet Clinic. All rights reserved.</p>

</footer>

</body>

</html>

Style.css

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

background-color: #ffffff;

}

header {

background-color: #9bcf47;

color: white;

padding: 20px;

text-align: center;

}

nav {

display: flex;

justify-content: center;

background-color: #333;

padding: 10px;

}

nav a {

color: white;

text-decoration: none;

padding: 10px;

margin: 0 10px;

}

section {

padding: 20px;

}

.h\_index {

text-align: center;

}

.container {

max-width: 800px;

margin: 0 auto;

}

.appointment-form {

background-color: #fff;

padding: 20px;

border-radius: 5px;

}

.form-group {

margin-bottom: 20px;

}

.form-group label {

display: block;

font-weight: bold;

}

.form-group input[type="text"],

.form-group input[type="email"],

.form-group textarea {

width: 100%;

padding: 10px;

border: 1px solid #ccc;

border-radius: 5px;

}

.form-group input[type="submit"] {

background-color: #9bcf47;

color: white;

border: none;

padding: 10px 20px;

cursor: pointer;

}

.form-group input[type="submit"]:hover {

background-color: #9bcf47;

}

.content {

padding: 20px;

}

footer {

background-color: #9bcf47;

color: white;

text-align: center;

padding: 10px;

position: fixed;

bottom: 0;

width: 100%;

}

.footer\_special {

background-color: #9bcf47;

color: white;

text-align: center;

padding: 10px;

position: relative;

bottom: 0;

width: 100%;

}

.service-list {

margin-top: 20px;

}

.service {

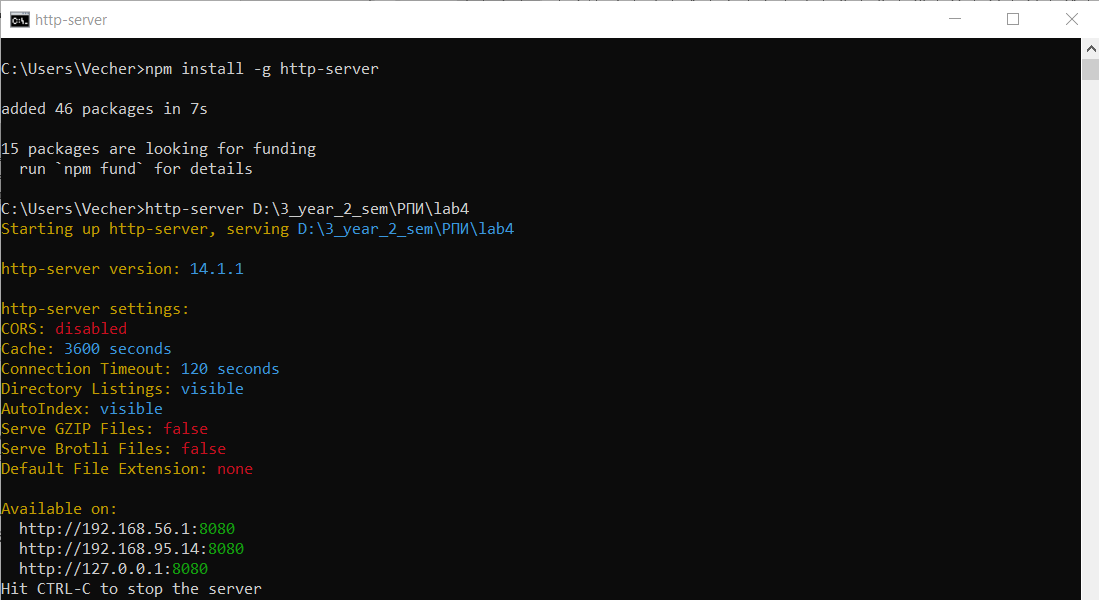
margin-bottom: 10px;

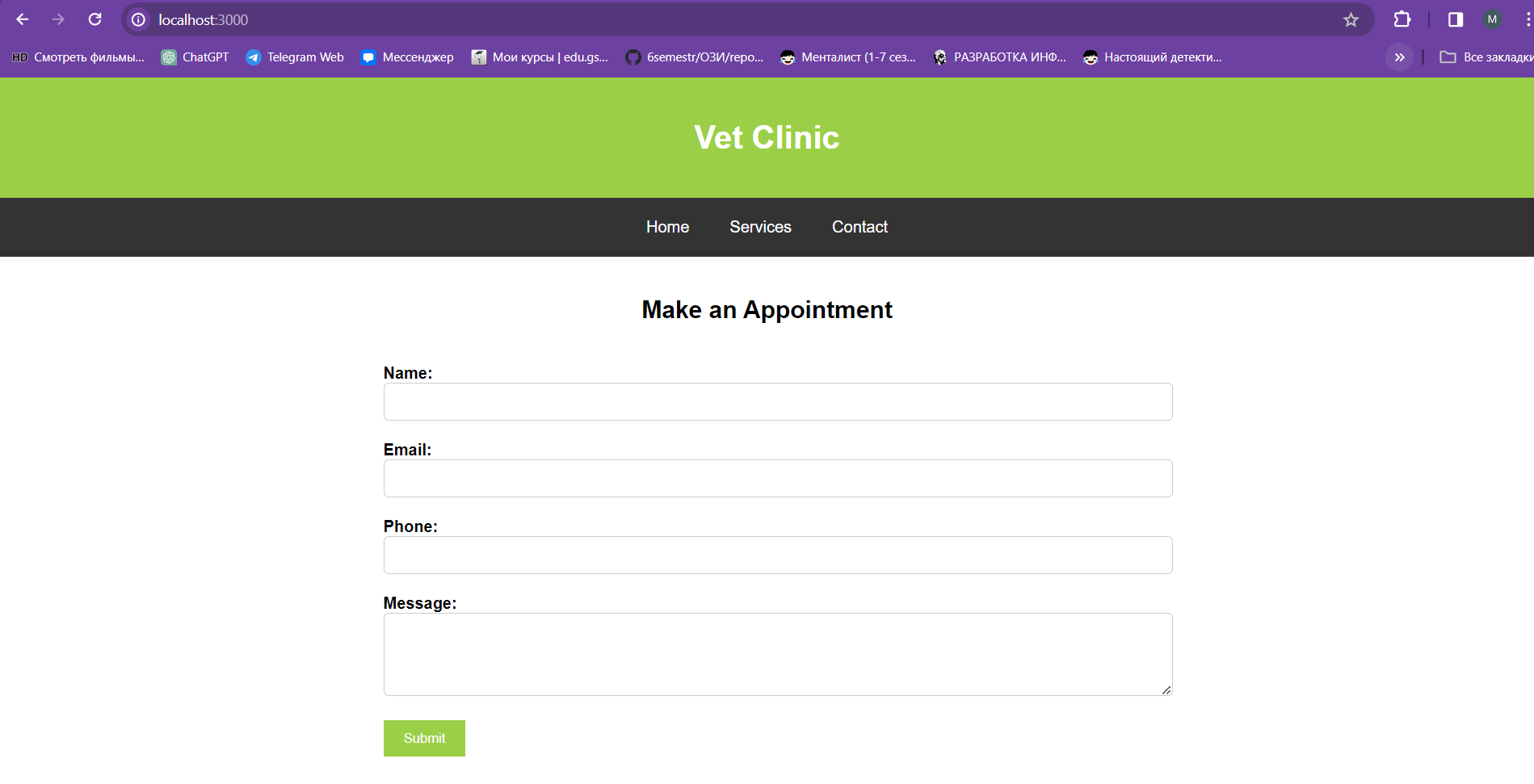
padding: 10px;

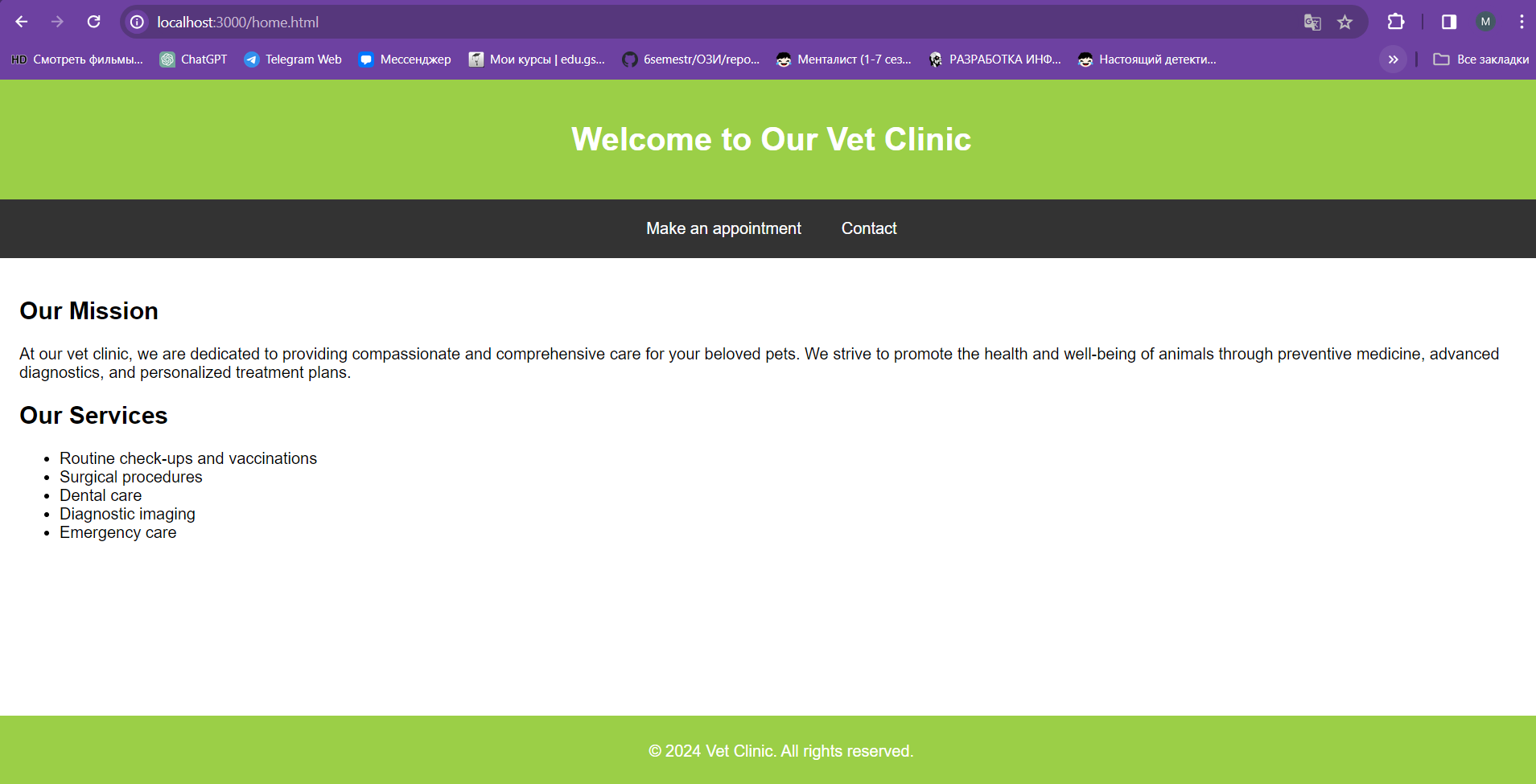
border: 1px solid #ccc;

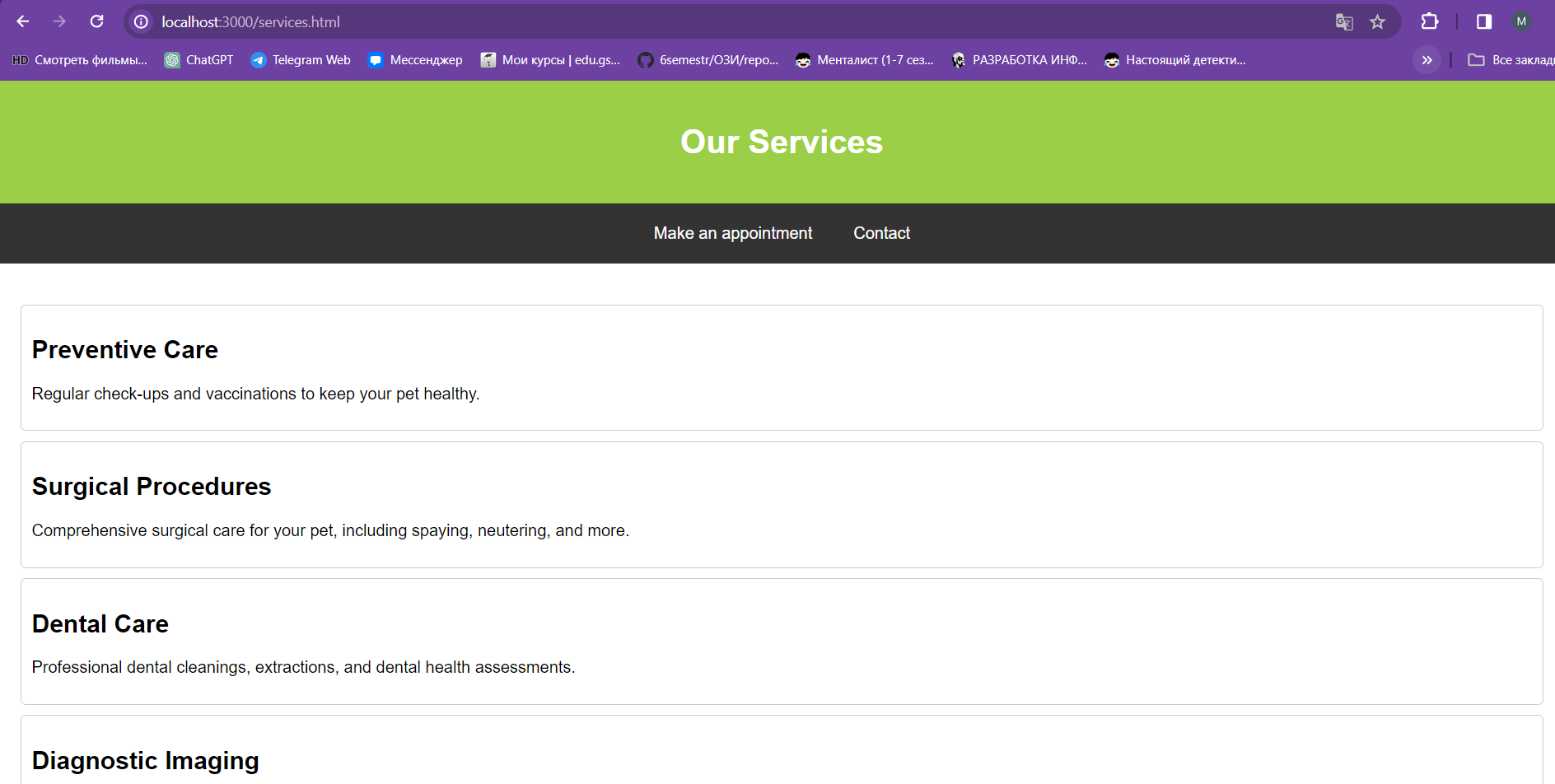
border-radius: 5px;}

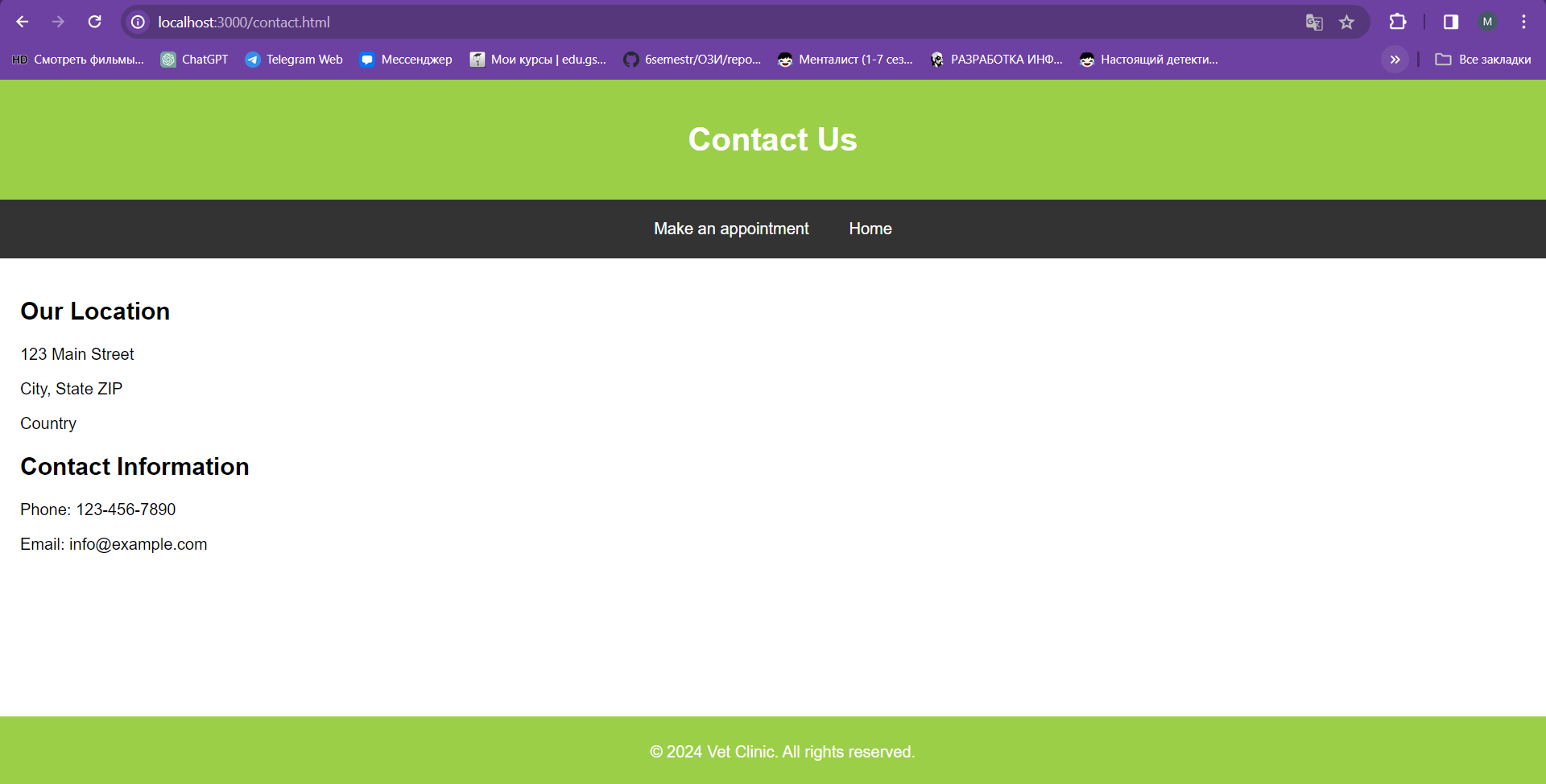
**Результат выполнения:**

****

****

****

****

****

**Задание 2.**

Выполнить задание1 с использованием фреймворка Express. Дополнительно: все переходы логгировать в отдельный файл server.log.

**Код программы:**

Server.js

const express = require('./expressapp/node\_modules/express');

const fs = require('fs');

const path = require('path');

const app = express();

const PORT = process.env.PORT || 3000;

app.use((req, res, next) => {

const logData = `${new Date().toISOString()} - ${req.method} ${req.url}\n`;

fs.appendFile('server.log', logData, (err) => {

if (err) console.error('Error writing to server.log:', err);

});

next();

});

app.use(express.static(path.join(\_\_dirname, 'pages')));

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

res.sendFile(path.join(\_\_dirname, 'pages', '\*.html'));

});

// Запуск сервера

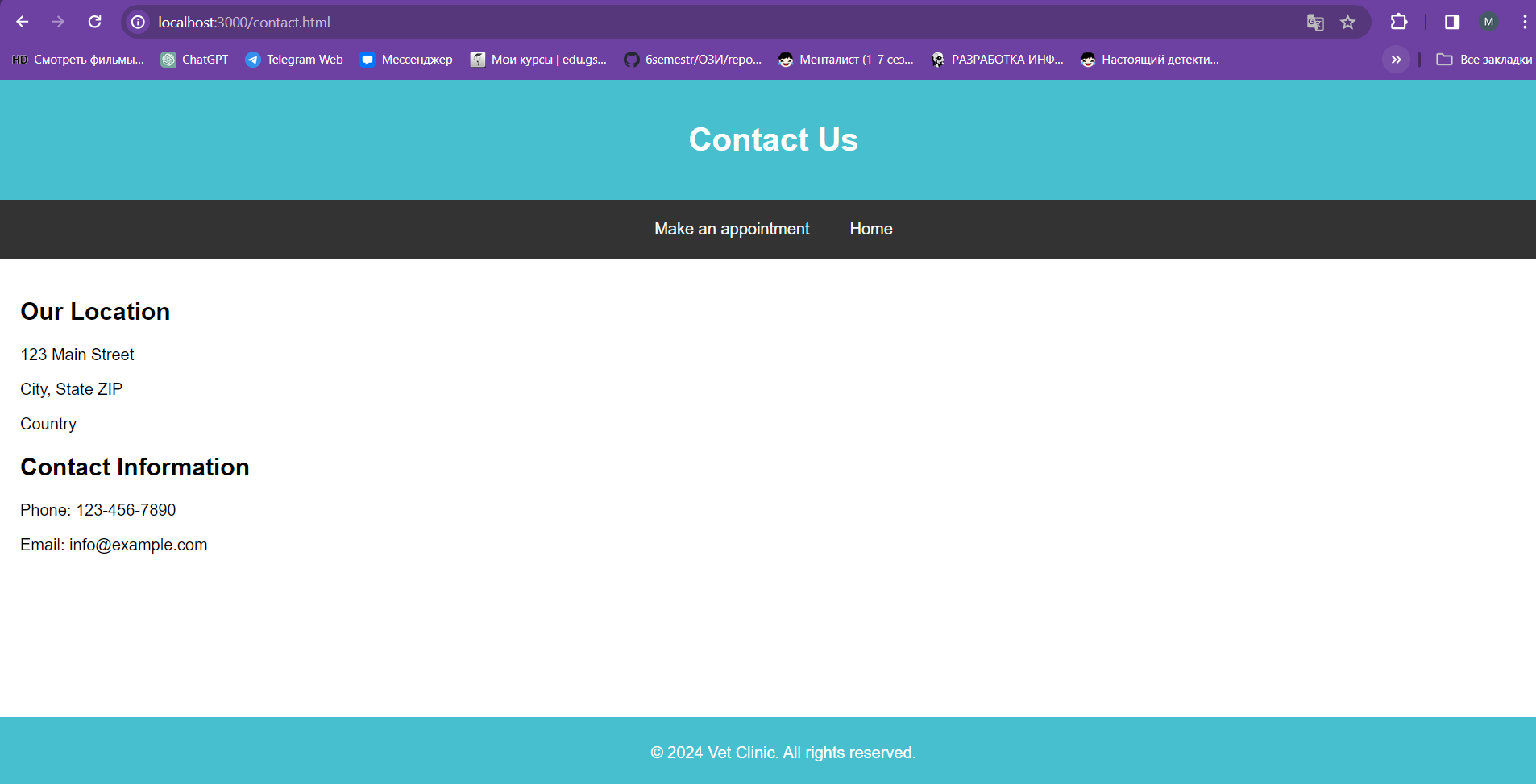
app.listen(PORT, () => {

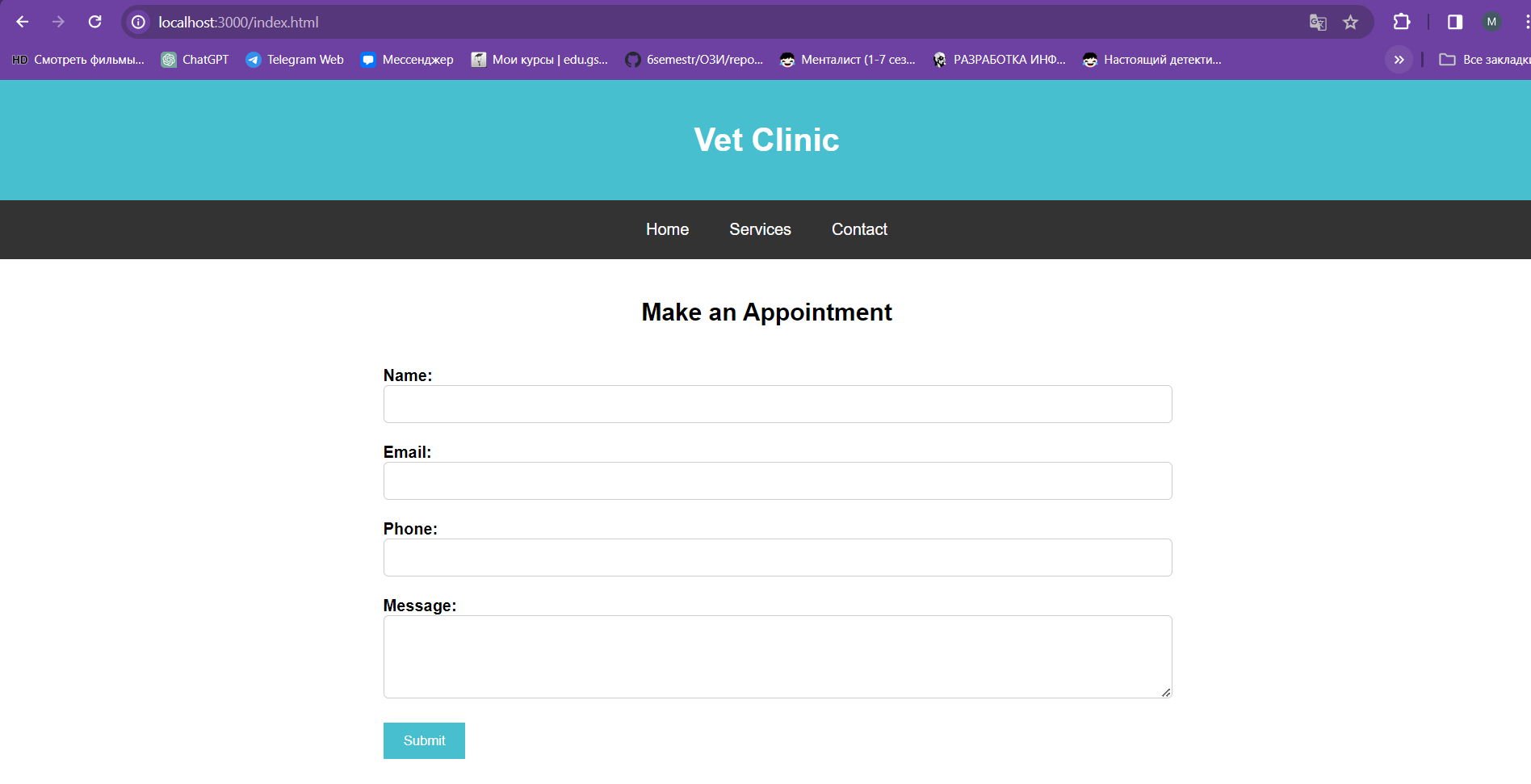
console.log(`Сервер запущен на порту ${PORT}`);

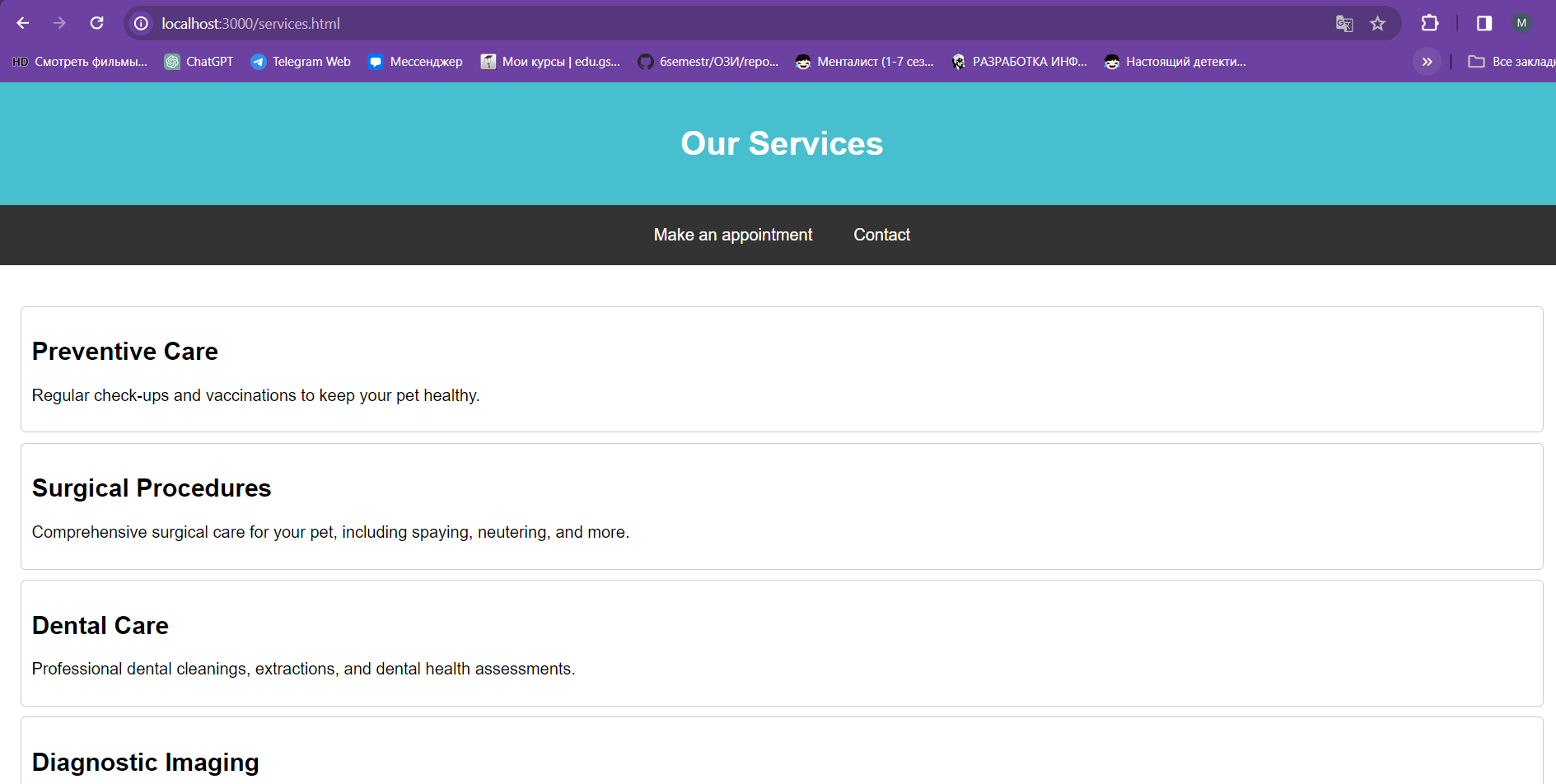
});

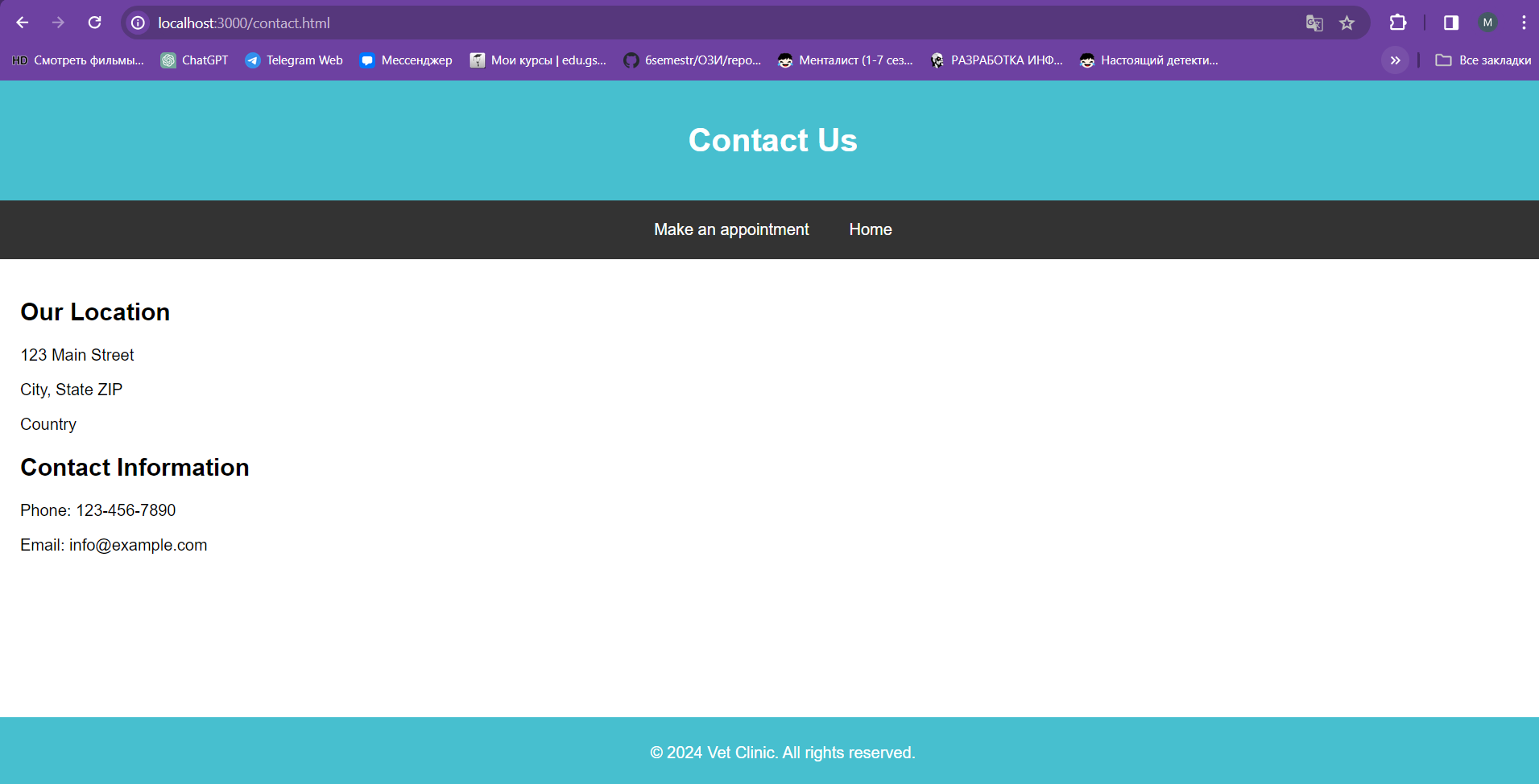
**Результат выполнения:**

****









**Задание 3.**

Используя фреймворк Express, разработать REST API. В качестве предметной области взять задание на курсовую работу. В качестве источника данных создать файл в формате JSON. Установить Postman и протестировать все разработанный REST API.

**Код программы:**

App3.js

const express = require('./expressapp/node\_modules/express');

const fs = require('fs');

const bodyParser = require('./expressapp/node\_modules/body-parser');

const app = express();

const PORT = 3003;

app.use(bodyParser.json());

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

fs.readFile('./static/users.json', 'utf8', (err, data) => {

if (err) {

console.error('Error reading users.json:', err);

return res.status(500).json({ error: 'Internal Server Error' });

}

const users = JSON.parse(data).users;

res.json(users);

});

});

app.get('/users/:id', (req, res) => {

const userId = parseInt(req.params.id);

fs.readFile('./static/users.json', 'utf8', (err, data) => {

if (err) {

console.error('Error reading users.json:', err);

return res.status(500).json({ error: 'Internal Server Error' });

}

const users = JSON.parse(data).users;

const user = users.find(u => u.id === userId);

if (!user) {

return res.status(404).json({ error: 'User not found' });

}

res.json(user);

});

});

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

const newUser = req.body;

console.log(newUser)

fs.readFile('./static/users.json', 'utf8', (err, data) => {

if (err) {

console.error('Error reading users.json:', err);

return res.status(500).json({ error: 'Internal Server Error' });

}

const users = JSON.parse(data).users;

newUser.id = users.length + 1;

users.push(newUser);

fs.writeFile('./static/users.json', JSON.stringify({ users }), (err) => {

if (err) {

console.error('Error writing to users.json:', err);

return res.status(500).json({ error: 'Internal Server Error' });

}

res.json(newUser);

});

});

});

app.put('/users/:id', (req, res) => {

const userId = parseInt(req.params.id);

const updatedUser = req.body;

fs.readFile('./static/users.json', 'utf8', (err, data) => {

if (err) {

console.error('Error reading users.json:', err);

return res.status(500).json({ error: 'Internal Server Error' });

}

let users = JSON.parse(data).users;

const index = users.findIndex(u => u.id === userId);

if (index === -1) {

return res.status(404).json({ error: 'User not found' });

}

users[index] = { ...users[index], ...updatedUser };

fs.writeFile('./static/users.json', JSON.stringify({ users }), (err) => {

if (err) {

console.error('Error writing to users.json:', err);

return res.status(500).json({ error: 'Internal Server Error' });

}

res.json(users[index]);

});

});

});

app.delete('/users/:id', (req, res) => {

const userId = parseInt(req.params.id);

fs.readFile('./static/users.json', 'utf8', (err, data) => {

if (err) {

console.error('Error reading users.json:', err);

return res.status(500).json({ error: 'Internal Server Error' });

}

let users = JSON.parse(data).users;

const index = users.findIndex(u => u.id === userId);

if (index === -1) {

return res.status(404).json({ error: 'User not found' });

}

const deletedUser = users.splice(index, 1)[0];

fs.writeFile('./static/users.json', JSON.stringify({ users }), (err) => {

if (err) {

console.error('Error writing to users.json:', err);

return res.status(500).json({ error: 'Internal Server Error' });

}

res.json(deletedUser);

});

});

});

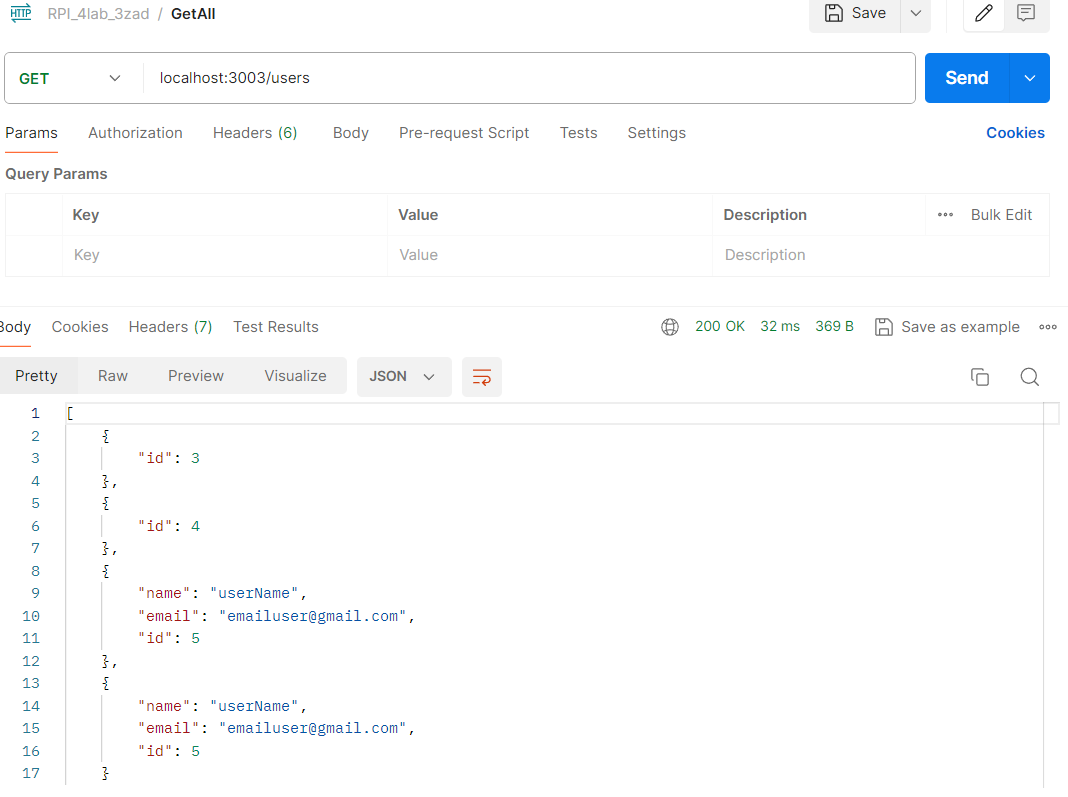
app.listen(PORT, () => {

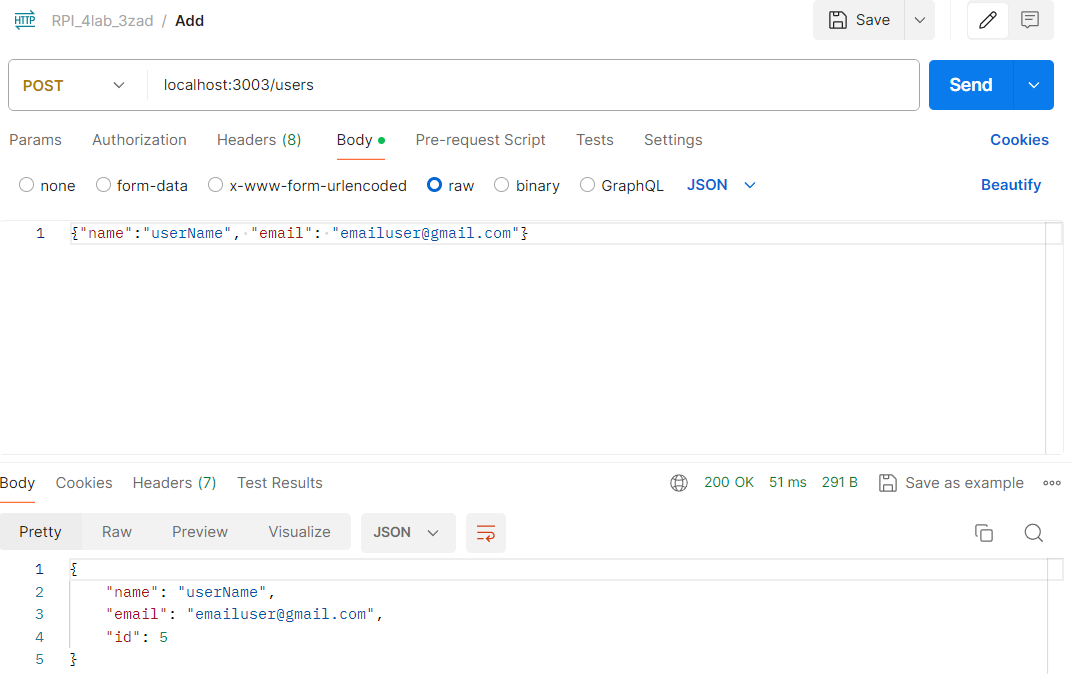
console.log(`Server running at http://localhost:${PORT}/`);

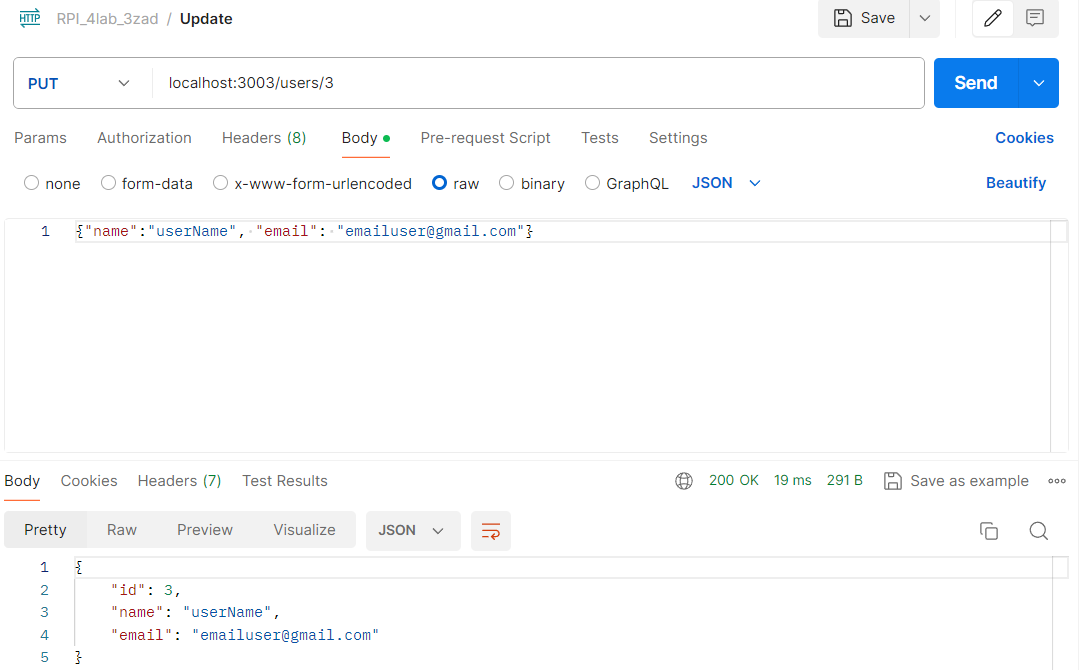
});

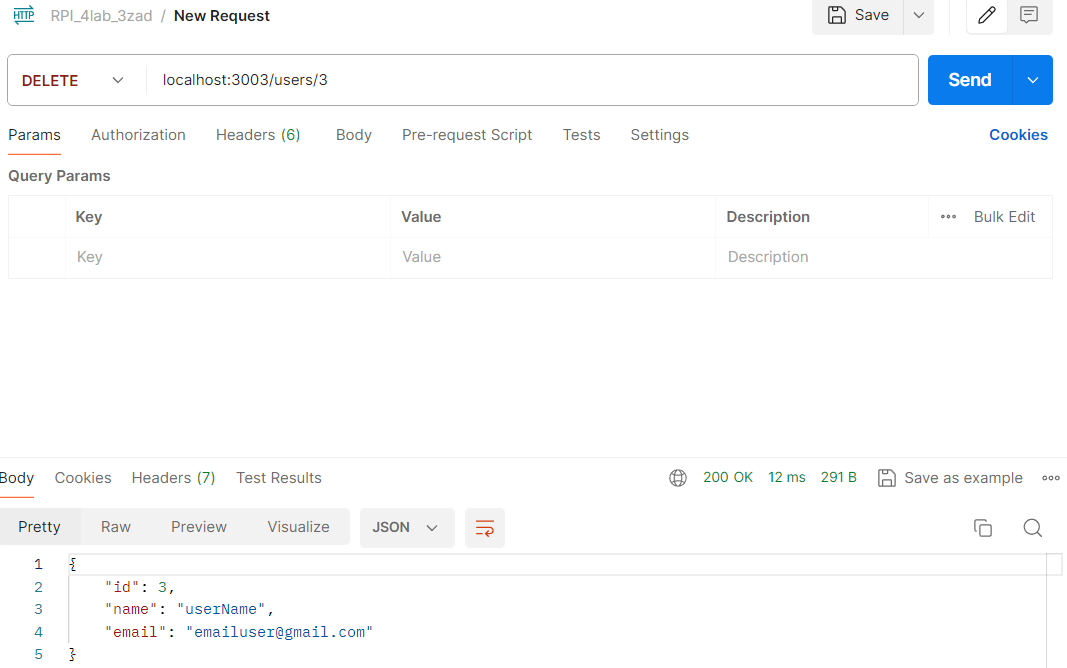
**Результат выполнения:**











**Вывод:** в результате выполнения лабораторной работы мноюосвоены возможности NodeJs по созданию локальных серверов с выводом статической информации, я ознакомилась с фреймворком Express.