|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

**ФАКУЛЬТЕТ** ***ИУК «Информатика и управление»***

**КАФЕДРА** \_\_***ИУК4 «Программное обеспечение ЭВМ, информационные технологии»***

**ДОМАШНЯЯ РАБОТА**

**«Разработка микросервисов»**

**ДИСЦИПЛИНА: «Перспективные языки программирования»**

|  |  |  |
| --- | --- | --- |
| Выполнил: студент гр. ИУК4-52Б | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( Карельский М.К. )  (Подпись) |
| Проверил: | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( Гришунов С.С. )  (Подпись) |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | | |
|  | - Балльная оценка:  - Оценка: | |

Калуга, 2022

**Цель:** получить навык разработки веб-приложений на основе микросервисов.

**Задачи:**

1. Разработать ReactJS приложение
2. Разработать микросервис для получения данных приложением с помощью микрофреймворка Flask

**Предметная область** – сайт фитнес центра

Сайт может быть произвольной тематики.

В качестве сервера приложений должен выступать nginx (apache или IIS). Сервер приложений должен обрабатывать только запросы по загрузке статичных файлов. Для загрузки данных использовать микросервис (можно использовать несколько микросервисов), разработанный с помощью микрофреймворка Flask. Микросервис должен быть логически разделен на blueprint’ы.

Содержимое сайта должно быть доступно только авторизированным пользователям. Должна быть предусмотрена возможность регистрации новых пользователей.

Для хранения данных использовать реляционную СУБД (MySQL, PostgreSQL или Firebird).

Сайт должен состоять минимум из 2-х смысловых частей с возможностью перехода между ними (заданием переменной в state элемента). Как минимум на одной из них должна быть таблица с данными, получаемыми от сервера, должна быть предусмотрена возможность добавления информации на сервер.

Для работы с формами использовать библиотеку Formik.

Разрешается использовать готовые компоненты: react-bootstrap, reactstrap или др.

**Листинг:**

***server.py***

from flask import Flask

from flask\_migrate import Migrate

from models import db

from reviews import reviews

from users import users

app = Flask(\_\_name\_\_)

app.config['SQLALCHEMY\_DATABASE\_URI'] = "postgresql://postgres:2458173671@localhost:5432/LifeformDB"

app.config['SQLALCHEMY\_TRACK\_MODIFICATIONS'] = False

db.init\_app(app)

migrate = Migrate(app, db)

app.register\_blueprint(reviews, url\_prefix='/reviews')

app.register\_blueprint(users, url\_prefix='/users')

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

***models.py***

from flask\_sqlalchemy import SQLAlchemy

from dataclasses import dataclass

db = SQLAlchemy()

@dataclass

class ReviewModel(db.Model):

\_\_tablename\_\_ = 'reviews'

id: int

name: str

comment: str

id = db.Column(db.Integer, primary\_key = True)

name = db.Column(db.String())

comment = db.Column(db.String())

def \_\_init\_\_(self, name, comment):

self.name = name

self.comment = comment

def \_\_repr\_\_(self):

return f"{self.name}:{self.comment}"

@dataclass

class UserModel(db.Model):

\_\_tablename\_\_ = 'users'

id: int

name: str

password: str

id = db.Column(db.Integer, primary\_key = True)

name = db.Column(db.String())

password = db.Column(db.String())

def \_\_init\_\_(self, name, password):

self.name = name

self.password = password

def \_\_repr\_\_(self):

return f"{self.name}:{self.password}"

***reviews.py***

from flask import Blueprint

from flask import Flask, request, jsonify

from models import db, ReviewModel

reviews = Blueprint('reviews', \_\_name\_\_)

@reviews.route('/', methods = ['POST', 'GET'])

def Review():

if request.method == 'POST':

name = request.form['name']

comment = request.form['comment']

review = ReviewModel(name=name, comment=comment)

db.session.add(review)

db.session.commit()

return f"Done!"

response = jsonify(ReviewModel.query.all())

response.headers.add("Access-Control-Allow-Origin", "\*")

return response

***users.py***

from flask import Blueprint

from flask import Flask, request, jsonify

from models import db, UserModel

users = Blueprint('users', \_\_name\_\_)

@users.route('/login', methods = ['POST'])

def Login():

if request.method == 'POST':

res = UserModel.query.filter\_by(name=request.form['login'],

password=request.form['password']).first()

response = jsonify(res)

response.headers.add("Access-Control-Allow-Origin", "\*")

return response

return jsonify(None)

@users.route('/signup', methods = ['POST'])

def Signup():

if request.method == 'POST':

res = UserModel.query.filter\_by(name=request.form['login']).first()

if res == None:

login = request.form['login']

password = request.form['password']

user = UserModel(name=login, password=password)

db.session.add(user)

db.session.commit()

response = jsonify(user)

response.headers.add("Access-Control-Allow-Origin", "\*")

return response

return jsonify(None)

***index.html***

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

<link rel="icon" href="%PUBLIC\_URL%/icon.png" />

<meta name="viewport" content="width=device-width, initial-scale=1" />

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous">

<link rel="stylesheet" type="text/css" href="styles.css">

<title>Fitness center Lifeform</title>

</head>

<body>

<div id="root"></div>

</body>

</html>

***styles.css***

body {

background-color: #ffa31a;

padding-top: 80px;

padding-bottom: 40px;

}

h1 {

font-family: 'Arial Rounded MT';

}

h2 {

font-family: 'OCR A';

}

a {

color: #1b1b1b;

}

.table-borderedless td {

border: none;

}

.text {

font-family: 'Berlin Sans FB';

}

#reviews {

border: solid;

border-collapse: collapse;

}

.review-cell {

border: solid;

}

img {

border: 4px solid #ffa31a;

}

img:hover {

border: 4px solid #1b1b1b;

}

***index.js***

import React from 'react';

import ReactDOM from 'react-dom/client';

import Main from './Main';

import Entry from './Entry';

import reportWebVitals from './reportWebVitals';

import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

<Router>

<div>

<Routes>

<Route path="\*" element={<Main />} />

<Route path="/LogIn" element={<Entry mode="login" />} />

<Route path="/SignUp" element={<Entry mode="signup" />} />

</Routes>

</div>

</Router>

);

reportWebVitals();

***Main.js***

import React from "react";

import Navbar from './Navbar';

import Pictures from './Pictures';

import Reviews from './Reviews';

function Main() {

return (

<div>

<Navbar />

<Pictures />

<Reviews />

</div>

);

}

export default Main;

***Navbar.js***

import React, { useState, useEffect } from "react";

function Navbar() {

const [nickname, setNickname] = useState("");

useEffect(() => {

const data = window.localStorage.getItem('nickname');

if ( data !== JSON.stringify("") ) setNickname(JSON.parse(data));

}, []);

const Quit = () => {

window.localStorage.setItem('nickname', JSON.stringify(""));

setNickname("");

window.location = '/main';

}

return (

<nav className="navbar fixed-top navbar-expand-lg navbar-dark bg-dark">

<a className="navbar-brand" href="/">Fitness center Lifeform</a>

<div className="collapse navbar-collapse d-flex justify-content-end">

<ul className="navbar-nav">

<li className="nav-item active">

{ nickname == "" ? <a className="nav-link" href="/SignUp">Sign up</a> : <div className="nav-link">{ nickname }</div>}

</li>

<li className="nav-item active">

{ nickname == "" ? <a className="nav-link" href="/LogIn">Login</a> : <button type="button" className="nav-link btn btn-link" onClick={ Quit }>Quit</button>}

</li>

</ul>

</div>

</nav>

);

}

export default Navbar;

***Pictures.js***

import React from "react";

function Pictures() {

return (

<div>

<div className="container">

<div className="row justify-content-lg-center">

<h2>News and events</h2>

</div>

</div>

<div className="container">

<div className="row justify-content-lg-center">

<div className="col-lg-4">

<img src="2.png" className="img-fluid"/>

</div>

<div className="col-lg-4">

<img src="3.png" className="img-fluid"/>

</div>

<div className="col-lg-4">

<img src="4.png" className="img-fluid"/>

</div>

</div>

<div className="row justify-content-lg-center">

<div className="col-lg-4">

<img src="1.png" className="img-fluid"/>

</div>

<div className="col-lg-4">

<img src="5.png" className="img-fluid"/>

</div>

<div className="col-lg-4">

<img src="6.png" className="img-fluid"/>

</div>

</div>

</div>

</div>

);

}

export default Pictures;

***Reviews.js***

import React from "react";

import { useFormik } from "formik";

import ReviewTable from './ReviewTable';

const Reviews = () => {

const refTable = React.useRef(null);

const formik = useFormik({

initialValues: { name: "", comment: "" },

onSubmit: async values => {

let formData = new FormData();

formData.append('name', JSON.parse(window.localStorage.getItem('nickname')));

formData.append('comment', values['comment']);

let response = await fetch('http://localhost:5000/reviews/', {

method: 'POST',

body: formData

});

refTable.current.forceUpdate();

}

});

return (

<div>

<div className="container">

<div className="row justify-content-lg-center">

<h2>Reviews</h2>

</div>

</div>

<ReviewTable />

<div className="container">

<div className="row justify-content-lg-center text">

{ window.localStorage.getItem('nickname') == JSON.stringify("") ?

<h2>Login to send review</h2>

:

<form onSubmit={formik.handleSubmit}>

<label htmlFor="comment">Comment</label><br/>

<input

id="comment"

name="comment"

type="text"

onChange={formik.handleChange}

value={formik.values.comment}

/><br/>

<div className="row justify-content-lg-center text mt-2">

<button type="submit">Submit</button>

</div>

</form>

}

</div>

</div>

</div>

);

};

export default Reviews;

***ReviewTable.js***

import React from "react";

class ReviewTable extends React.Component {

constructor(props){

super(props);

this.state = {

reviews: []

}

}

render(){

fetch('http://localhost:5000/reviews/').then((res) =>

res.json().then((data) => {

let rows = [];

for (let i = 0; i < data.length; i++) {

rows.push(

<tr>

<td className="review-cell">{ data[i].name }</td>

<td className="review-cell">{ data[i].comment }</td>

</tr>

);

}

if (rows.length != this.state.reviews.length){

this.setState({

reviews: rows

});

}

})

);

return (

<div className="container">

<div className="row justify-content-lg-center">

<div className="col-lg-4">

<table id="reviews" className="table text">

<tr>

<td className="review-cell"><b>Name</b></td>

<td className="review-cell"><b>Comment</b></td>

</tr>

{ this.state.reviews }

</table>

</div>

</div>

</div>

);

}

}

export default ReviewTable;

***Entry.js***

import React from "react";

import Navbar from './Navbar';

import EntryForm from './EntryForm';

function Entry(props) {

return (

<div>

<Navbar />

<EntryForm mode={props.mode} />

</div>

);

}

export default Entry;

***EntryForm.js***

import React, { useState } from "react";

import { useFormik } from "formik";

import { useNavigate } from 'react-router-dom';

const EntryForm = (props) => {

const [error, setError] = useState("");

const navigate = useNavigate();

const formik = useFormik({

initialValues: { login: "", password: "" },

onSubmit: async values => {

let formData = new FormData();

formData.append('login', values['login']);

formData.append('password', values['password']);

let response = await fetch(props.mode == "login" ? 'http://localhost:5000/users/login' : 'http://localhost:5000/users/signup', {

method: 'POST',

body: formData

});

let result = await response.json();

if (result != null) {

window.localStorage.setItem('nickname', JSON.stringify(result.name));

navigate('/main');

}

else{

setError(props.mode == "login" ? "User wasn't found" : "User already exist");

}

}

});

return (

<div className="container">

<div className="row justify-content-lg-center">

{ error }

</div>

<div className="row justify-content-lg-center">

{ props.mode == "login" ? <h2>Login</h2> : <h2>Sign up</h2>}

</div>

<div className="row justify-content-lg-center text">

<form onSubmit={formik.handleSubmit}>

<label htmlFor="login">Login</label><br/>

<input

id="login"

name="login"

type="text"

onChange={formik.handleChange}

value={formik.values.login}

/><br/>

<label htmlFor="password">Password</label><br/>

<input

id="password"

name="password"

type="password"

onChange={formik.handleChange}

value={formik.values.password}

/><br/>

<div className="row justify-content-lg-center text mt-2">

<button type="submit">Submit</button>

</div>

</form>

</div>

</div>

);

};

export default EntryForm;

**Результат:**

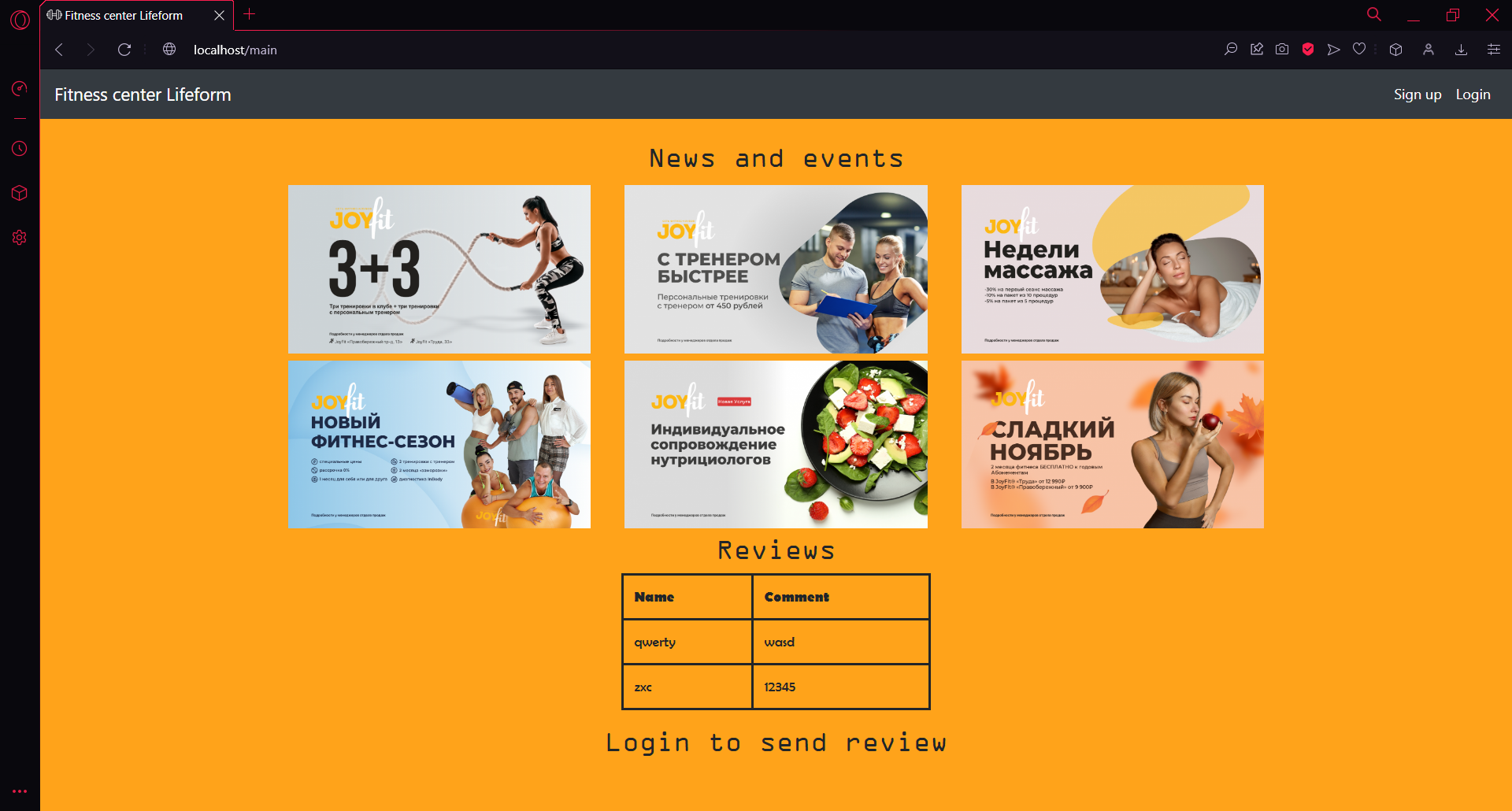


Рис. 1. Главная страница без авторизации

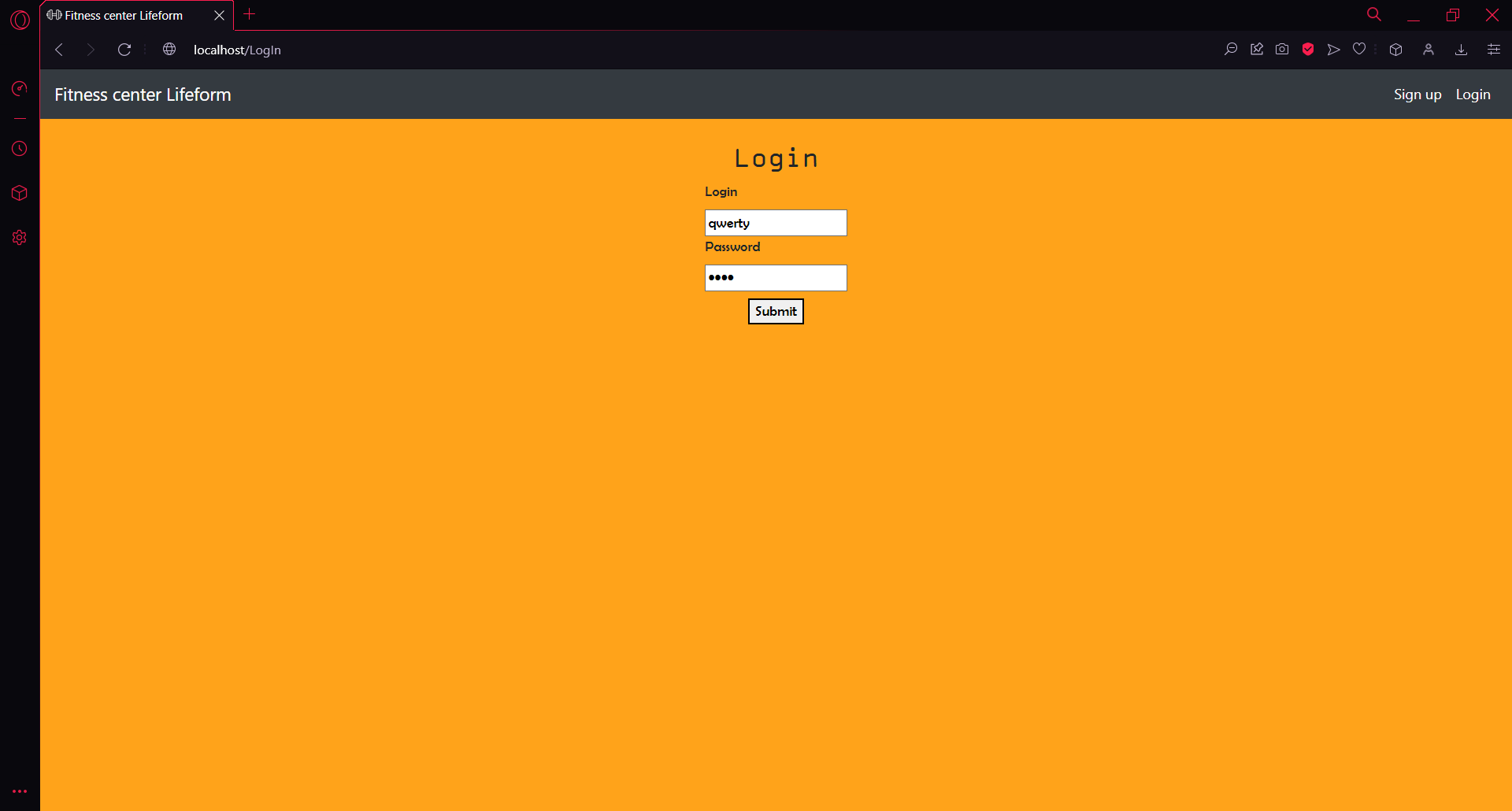


Рис. 2. Вход

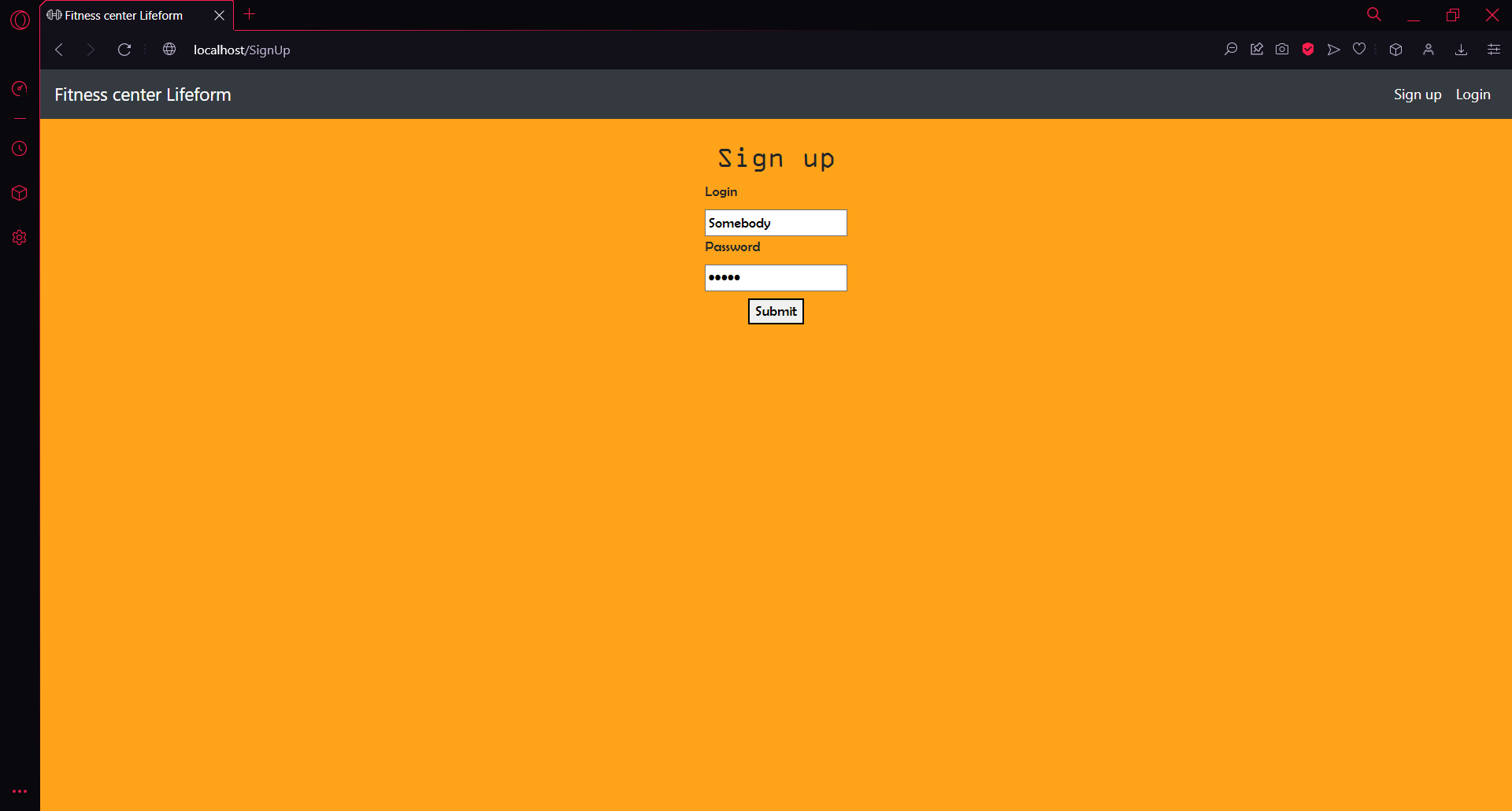


Рис. 3. Регистрация

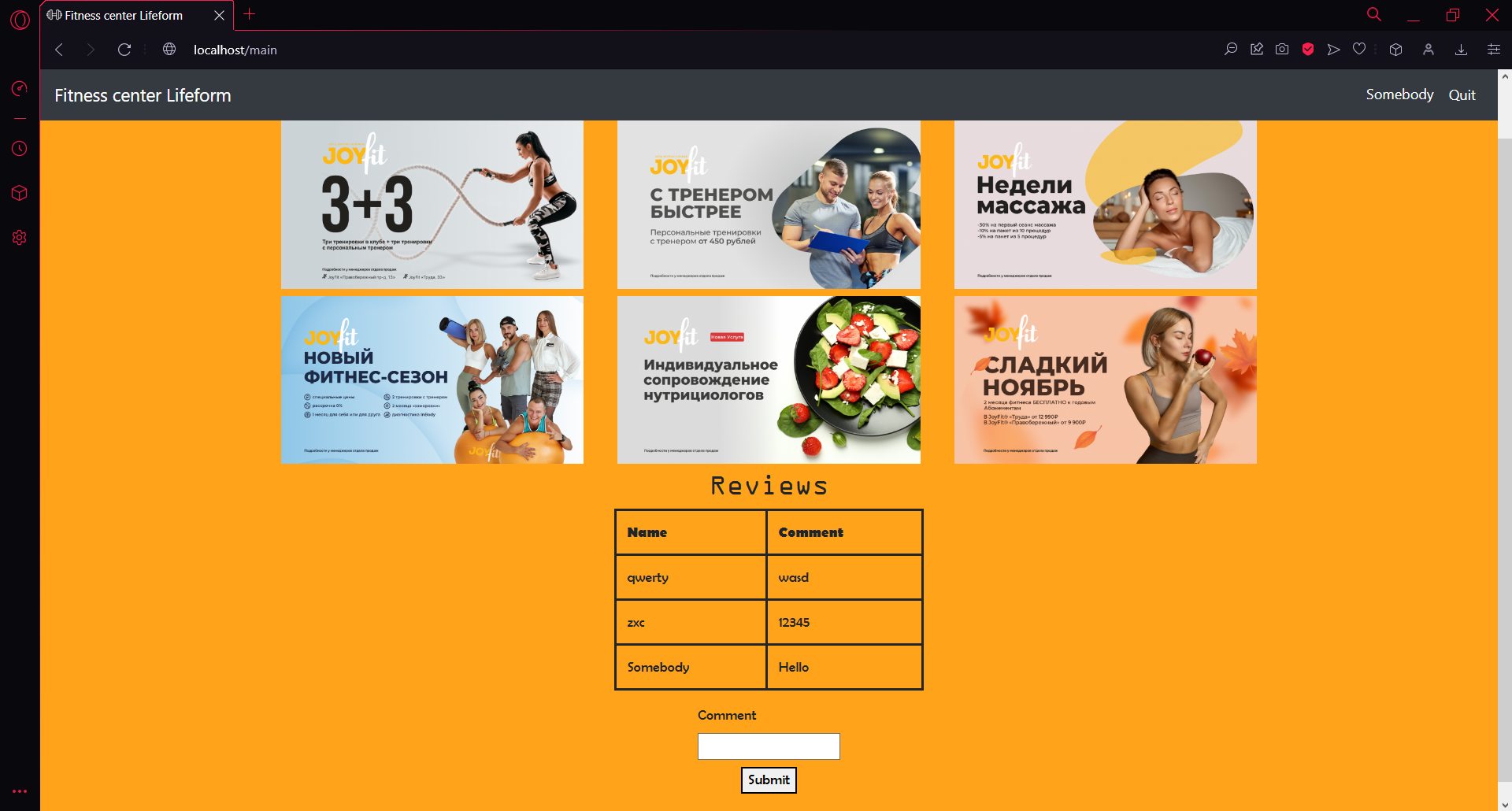


Рис. 4. Главная страница с авторизацией и добавление рецензии

**Вывод:** в ходе выполнения лабораторной работы были получены практические навыки разработки веб-приложений на основе микросервисов.