МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ «ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра Вычислительной техники

ОТЧЕТ

по лабораторной работе № 9

по дисциплине «Web программирование»

Тема: «Модульное тестирование приложения»

Студент гр. 3311 Баймухамедов Р. Р. _____

Преподаватель Калмыков М.А. _____

Санкт-Петербург

2025

Цель работы

Знакомство и использование Jtest для модульного тестирования web-приложений

Выполнение лабораторной работы

Установим jtest и сделаем тесты нашего web-приложения

login.test.js – проверяющий авторизацию пользователя

```
const request = require('supertest');
const app = require('../server');

describe('POST /login', () => {
  test('успешный логин -> 200, текст и 2 cookie (user, pwd)', async () => {
    const res = await request(app)
      .post('/login')
      .send({ user: '12345', password: 'passw0rd' });

    expect(res.statusCode).toBe(200);
    expect(res.text.trim()).toBe('Login successfull...');

    const setCookie = res.headers['set-cookie'] || [];
    // Должны быть оба куки
    const hasUser = setCookie.some(c => /^user=12345;/.test(c));
    const hasPwd  = setCookie.some(c => /^pwd=passw0rd;/.test(c));
    expect(hasUser).toBe(true);
    expect(hasPwd).toBe(true);
  });

  test('неверный пароль -> 401 и "ERROR"', async () => {
    const res = await request(app)
      .post('/login')
      .send({ user: '12345', password: 'wrong' });

    expect(res.statusCode).toBe(401);
    expect(res.text.trim()).toBe('ERROR');
    // Куки ставиться не должны
    const setCookie = res.headers['set-cookie'] || [];
    expect(setCookie.length).toBe(0);
  });
});
```

movies.test.js – проверяющий работу с выводом и добавлением фильмов

```
// __tests__/movies.test.js
```

```
const fs = require('fs');
const path = require('path');

describe('Movies API', () => {
  const realRead = fs.readFileSync;
  const realWrite = fs.writeFileSync;

  // Фикстуры фильмов
  const dataRU = [
    { id: 1, title: 'Фильм', year: 2001, genres: ['Драма'] },
    { id: 3, title: 'Кино',  year: 2005, genres: ['Комедия'] }
  ];
  const dataEN = [
    { id: 2, title: 'Movie',  year: 2002, genres: ['Drama'] },
    { id: 5, title: 'Cinema', year: 2008, genres: ['Comedy'] }
  ];

  beforeAll(() => {
    // Мокаем readFileSync: отдаём JSON по имени файла
    jest.spyOn(fs, 'readFileSync').mockImplementation((fname, enc) => {
      const f = String(fname);
      if (f.endsWith('movies.ru.json'))  return JSON.stringify(dataRU);
      if (f.endsWith('movies.en.json'))  return JSON.stringify(dataEN);
      return realRead(fname, enc);
    });
    // Мокаем writeFileSync, чтобы не писать на диск
    jest.spyOn(fs, 'writeFileSync').mockImplementation(() => {});
  });

  afterAll(() => {
    fs.readFileSync.mockRestore();
    fs.writeFileSync.mockRestore();
  });

  // Импортим app только после установки шпионов,
  // чтобы server.js взял замоканный fs
  const app = require('../server');

  test('GET /api/movies/all (en по умолчанию) -> lang cookie + JSON', async () =>
{
    const request = require('supertest');
    const res = await request(app).get('/api/movies/all');

    expect(res.statusCode).toBe(200);
    expect(res.body.meta.lang).toBe('en');
    expect(res.body.items.length).toBe(dataEN.length);

    const setCookie = res.headers['set-cookie'] || [];
    const hasLang = setCookie.some(c => /^lang=en;/.test(c));
    expect(hasLang).toBe(true);
  });
```

```
  test('GET /api/movies/all?lang=ru -> ru данные и lang=ru cookie', async () => {
    const request = require('supertest');
    const res = await request(app).get('/api/movies/all?lang=ru');

    expect(res.statusCode).toBe(200);
    expect(res.body.meta.lang).toBe('ru');
    expect(res.body.items.length).toBe(dataRU.length);

    const setCookie = res.headers['set-cookie'] || [];
    const hasLang = setCookie.some(c => /^lang=ru;/.test(c));
    expect(hasLang).toBe(true);
  });

  test('GET /api/movies?title=cin -> фильтр по title (en)', async () => {
    const request = require('supertest');
    const res = await request(app).get('/api/movies').query({ title: 'cin' });
    expect(res.statusCode).toBe(200);
    expect(res.body.meta.mode).toBe('title');
    // В наших EN фикстурах 'Cinema' матчится
    expect(res.body.items.map(x => x.title)).toEqual(['Cinema']);
  });

  test('GET /api/movies без title и без ids -> 400', async () => {
    const request = require('supertest');
    const res = await request(app).get('/api/movies');
    expect(res.statusCode).toBe(400);
    expect(res.body.error).toMatch(/title/i);
  });

  test('GET /api/movies?ids=1,5,999 -> выборка по id (из en по умолчанию)', async
() => {
    const request = require('supertest');
    const res = await request(app).get('/api/movies').query({ ids: '1,5,999' });
    expect(res.statusCode).toBe(200);
    expect(res.body.meta.mode).toBe('ids');
    // Т.к. язык по умолчанию en, id=5 присутствует, id=1 (есть в ru) — не
попадёт
    expect(res.body.items.map(x => x.id)).toEqual([5]);
  });

  test('POST /api/movies -> создаёт запись в ru+en, id = max(ru,en)+1', async ()
=> {
    const request = require('supertest');
    const body = {
      title_ru: 'Новый',
      title_en: 'New',
      year: 2020,
      genres: 'drama,comedy',
      director: 'John D.',
      rating: 7.5
```

```
    };
    const res = await request(app).post('/api/movies').send(body);
    expect(res.statusCode).toBe(201);

    // max id среди фикстур = 5 => ожидаем 6
    expect(res.body.item_ru.id).toBe(6);
    expect(res.body.item_en.id).toBe(6);
    expect(fs.writeFileSync).toHaveBeenCalledTimes(2);
    const calls = fs.writeFileSync.mock.calls.map(c => path.basename(c[0]));
    expect(calls.sort()).toEqual(['movies.en.json', 'movies.ru.json'].sort());
  });
});
```

prefs.test.js – проверяющий установку и корректное сохранение Cookie-файлов

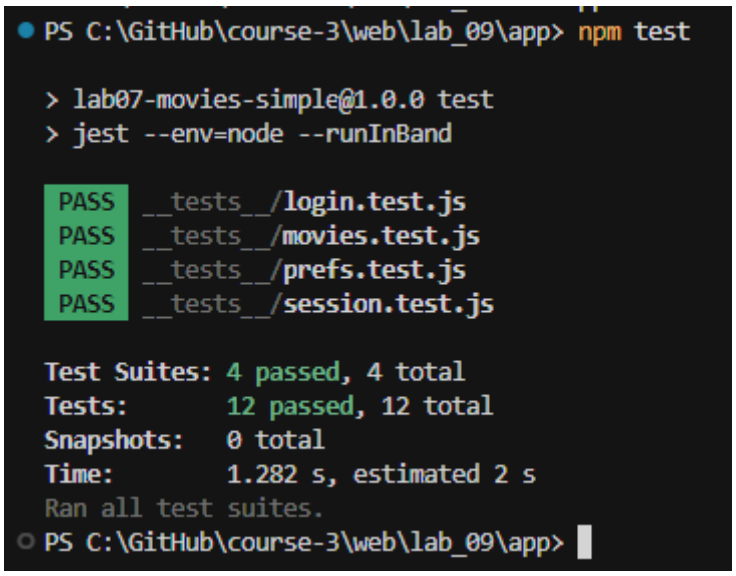
```
// __tests__/prefs.test.js
const request = require('supertest');
const app = require('../server');

describe('Prefs (cookies)', () => {
  test('POST /api/prefs без username -> 400', async () => {
    const res = await request(app).post('/api/prefs').send({ color: '#ff00ff' });
    expect(res.statusCode).toBe(400);
    expect(res.body.error).toMatch(/username/i);
  });

  test('POST /api/prefs с persistDays -> Set-Cookie с Max-Age', async () => {
    const res = await request(app)
      .post('/api/prefs')
      .set('X-Forwarded-Proto', 'https') // чтобы secure=true
      .send({ username: 'Rafael', color: '00ffaa', persistDays: 7 });

    expect(res.statusCode).toBe(200);
    expect(res.body.ok).toBe(true);

    const raw = res.headers['set-cookie'];
    const setCookie = Array.isArray(raw) ? raw : (raw ? [raw] : []);

    // username, page.color, lang
    expect(setCookie.length).toBeGreaterThanOrEqual(3);

    const hasUser = setCookie.some(c =>
/username=Rafael/.test(decodeURIComponent(c)) && /Max-Age=/.test(c));
    const hasColor = setCookie.some(c => /page\.color=%2300ffaa/.test(c) && /Max-
Age=/.test(c));
    const hasLang = setCookie.some(c => /lang=/.test(c) && /Max-Age=/.test(c));
    expect(hasUser && hasColor && hasLang).toBe(true);
  });
```

```
  test('GET /api/prefs возвращает значения из кук', async () => {
    const agent = request.agent(app);

    // сначала установим куки
    const res1 = await agent
      .post('/api/prefs')
      .send({ username: 'UserX', color: '#123456' });
    expect(res1.statusCode).toBe(200);

    // теперь читаем их тем же агентом (cookie-jar)
    const res2 = await agent.get('/api/prefs');
    expect(res2.statusCode).toBe(200);
    expect(res2.body.prefs).toEqual(
      expect.objectContaining({ username: 'UserX', color: '#123456' })
    );
  });
});
```

session.test.js – проверяющий корректную информацию о сессиях

```
// __tests__/session.test.js
const request = require('supertest');
const app = require('../server');

describe('Session counters', () => {
  test('GET /api/session инкрементирует visits в рамках одной сессии', async ()
=> {
    const agent = request.agent(app);

    const r1 = await agent.get('/api/session');
    expect(r1.statusCode).toBe(200);
    const v1 = r1.body.session.visits;

    const r2 = await agent.get('/api/session');
    expect(r2.statusCode).toBe(200);
    const v2 = r2.body.session.visits;

    expect(typeof v1).toBe('number');
    expect(v2).toBe(v1 + 1);

    // lastVisit должен обновляться
    expect(r2.body.session.lastVisit).toBeTruthy();
  });
});
```

Пример работоспособности программы



Все тесты успешно прошли проверку

Заключение

В данной лабораторной работе мы подвели итоги нашего веб-приложения, познакомились и использовали Jtest для модульного тестирования получившегося итогового web-приложения