МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”
ІПСА
Кафедра Системного проектування

Лабораторна робота 5

з дисципліни: «ПРОЕКТУВАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ»

Модульне тестуваня (Unit-тести) та рефакторинг.

Виконав:

студент групи ДА-71

Павлюк Вадим

Мета роботи: оволодіти навичками створення програмного забезпечення за метолологією TDD та ознайомитися з процедурами рефакторинга.

Код до рефакторингу

```
const axios = require('axios');

module.exports = class Api {
  constructor(url) {
    if (typeof url !== 'string') {
      throw new Error('Api constructor() url expected to be typeof String');
    }
    Object.defineProperties(this, {
      url: { get: () => url },
      status: {
        get: () => (obj) => {
          console.log(`Status code: ${obj.status}`);
          console.log(`Status text: ${obj.statusText}`);
          console.log(`Request method: ${obj.request.method}`);
          console.log(`Path: ${obj.request.path}`);
          console.log(`Date: ${obj.headers.date}`);
        }
      }
    })
    Object.freeze(this)
  }

  async makeGet(params='') {
    try {
      this.url += params;
      const response = await axios.get(this.url);
      this.status(response);
      return response.data
    }
    catch (error) {
      console.error(error);
    }
  }

  async makePost(params={}) {
    try {
      const response = await axios.post(this.url, params);
      this.status(response);
    }
    catch (error) {
      console.error(error);
```

```
    }
  }

  async makeDelete(params='') {
    try {
      this.url += params;
      const response = await axios.delete(this.url);
      this.status(response);
    }
    catch (error) {
      console.error(error);
    }
  }

  async makeUpdate(params={}) {
    try {
      this.url += params.path;
      const response = await axios.patch(link, params.config);
      this.status(response);
    }
    catch (error) {
      console.error(error);
    }
  }
}
```

Код після рефакторингу

```
const axios = require('axios');

module.exports = class Api {
  constructor(url) {
    if (typeof url !== 'string') {
      throw new Error('Api constructor() url expected to be typeof String');
    }
    Object.defineProperties(this, {
      url: { get: () => url },
      status: {
        get: () => (obj) => {
          console.log(`Status code: ${obj.status}`);
          console.log(`Status text: ${obj.statusText}`);
          console.log(`Request method: ${obj.request.method}`);
          console.log(`Path: ${obj.request.path}`);
          console.log(`Date: ${obj.headers.date}`);
        }
      }
    })
    Object.freeze(this)
```

```
}

async makeGet(params='', isStatus=false) {
  try {
    const link = this.url + params;
    const response = await axios.get(link);
    if (isStatus) {
      this.status(response);
    }
    return response.data
  }
  catch (error) {
    console.error(error);
  }
}

async makePost(params={}, isStatus=false) {
  try {
    const response = await axios.post(this.url, params);
    if (isStatus) {
      this.status(response);
    }
  }
  catch (error) {
    console.error(error);
  }
}

async makeDelete(params='', isStatus=false) {
  try {
    const link = this.url + params;
    const response = await axios.delete(link);
    if (isStatus) {
      this.status(response);
    }
  }
  catch (error) {
    console.error(error);
  }
}

async makeUpdate(params={}, isStatus=false) {
  try {
    const link = this.url + params.path;
    const response = await axios.patch(link, params.config);
    if (isStatus) {
      this.status(response);
```

```
      }
    }
    catch (error) {
      console.error(error);
    }
  }
}
```

Що змінилося?

1) Для того, щоб можна було відключити вивід статусів запиту(this.status) в методах(makeUpdate, makeDelete, makePost, makeGet) було добавлено новий аргумент(isStatus) за замовчуванням він false. Також змінився виклик властивості this.status в методах:

Було

this.status()

Стало

```
if(isStatus) {
        this.status()
}
```

2) Для того, щоб не змінювати this.url в методах(makeUpdate, makeDelete, makePost, makeGet), була добавлена нова змінна link

Було

this.url += params;

Стало

const link = this.url + params

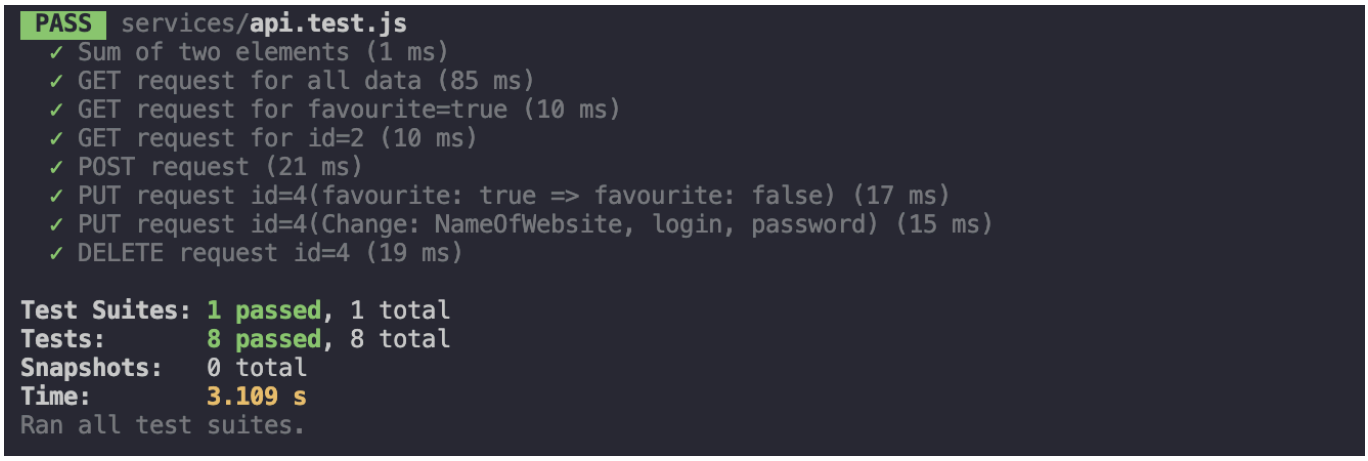
Unit-Тести

```
const Api = require('./api');
import * as dataForTest from './dataForTest.json';
const url = "http://localhost:3000/Storage";
const api = new Api(url)

function sum(a, b) {
  return  a + b
}
```

```
test('Sum of two elements', () => {
  const result = sum(1,2);
  expect(result).toBe(3);
})

test('GET request for all data', async () => {
  const data = await api.makeGet();
  expect(data).toEqual(dataForTest.data1GET);
})

test('GET request for favourite=true', async () => {
  const data = await api.makeGet('/?favourite=true');
  expect(data).toEqual(dataForTest.data2GET);
})

test('GET request for id=2', async () => {
  const data = await api.makeGet('/?id=2');
  expect(data).toEqual(dataForTest.data3GET);
})

test('POST request', async () => {
  await api.makePost(dataForTest.optionPOST);
  const data = await api.makeGet('/?id=4');
  expect(data).toEqual(dataForTest.data1POST);
})

test('PUT request id=4(favourite: true => favourite: false)', async () => {
  await api.makeUpdate(dataForTest.paramsPUT);
  const data = await api.makeGet('/?id=4');
  expect(data).toEqual(dataForTest.data1PUT);
})

test('PUT request id=4(Change: NameOfWebsite, login, password)', async () => {
  await api.makeUpdate(dataForTest.paramsPUT2);
  const data = await api.makeGet('/?id=4');
  expect(data).toEqual(dataForTest.data2PUT);
})

test('DELETE request id=4', async () => {
  await api.makeDelete('/4');
  const data = await api.makeGet();
  expect(data).toEqual(dataForTest.data1GET);
})
```

Для тестів використовувалась бібліотека Jest.

Unit-Тести результати



Висновки

У даній лабораторній роботі оволодів навичками створення програмного забезпечення за метолологією TDD та ознайомитися з процедурами рефакторинга. На мові програмування JS ознайомився з бібліотекою Jest та отримав навички в тестуванні коду.