



МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

Лабораторна робота № 4
з дисципліни “ Основи веб програмування ”
тема “Обробка даних веб-форм”

Виконав(ла)

студент(ка) II курсу

групи КП-____

Кривенко Віталій Ігорович

(прізвище, ім'я, по батькові)

Перевірів

“ ____ ” “ _____ ” 20__ р.

викладач

Гадиняк Руслан Анатолійович

(прізвище, ім'я, по батькові)

варіант № 5

Київ 2018

Мета роботи

Навчитись створювати HTML-форми для взаємодії користувача із веб-сайтом та обробляти на сервері внесені користувачем дані.

Вивчити основні принципи асинхронного програмування за допомогою callback-функцій.

Постановка завдання

1. Пошук сутностей по назві

1. Модифікувати веб-сторінку `/entities` з сутностями так, щоби можна було шукати сутності по частковому співпадінню строки назви за допомогою форми із `GET` запитом.
2. Сторінка має містити спеціальні компоненти для інформування користувача про те, для якого пошукового запиту відображаються результати.
3. Якщо в результаті пошуку не було знайдено жодних об'єктів, додавати на сторінку спеціальне повідомлення про це.

2. Створення нових сутностей

1. Додати на сторінку `/entities/new` HTML форму для створення нової сутності (за варіантом) за допомогою `POST` запиту.
2. Дана форма обов'язково повинна мати одне поле для завантаження на сервер файлу (зображення чи ін.), що пов'язаний із типом об'єкта.
3. Завантажені файли розміщувати у директорії `data/fs/`. Додати у веб-сервер обробник `GET` запитів на отримання файлів із

data/fs/. У поля сутностей зберігати URL для отримання цих файлів через GET запити до веб-сервера.

4. Після створення нової сутності і присвоєння їй ідентифікатора, перенаправити на сторінку нової сутності (`{entities}/:id`).

3. Видалення сутностей

1. На сторінці перегляду інформації про окрему сутність (`{entities}/:id`) додати форму (яка виглядатиме як кнопка, посилання чи зображення) для видалення поточної сутності за допомогою POST запиту.

Тексти коду програм

Завдання 1.

routes/attachments.ts

```
import * as express      from 'express';
import * as HttpCodes    from 'http-codes';
import { Errors, Strings } from 'veetaha-web';
import * as path         from 'path';
import * as Config       from '../config';

export const router = express.Router();

router.get('/:filename', ({ params: { filename } }, res, next) => {
    filename = filename.trim();
    if (!filename || Strings.containsSlashes(filename)) {
        return next(new Errors.StatusError(
            'invalid attachment file name', HttpCodes.NOT_ACCEPTABLE
        ));
    }
    res.sendFile(path.join(Config.FormFilesUploadDir, path.parse(filename).base), next);
});
```

public/src/paginator.ts

```
import 'jquery';
import * as Types from "vee-type-safe";

export interface PaginatorOptions {
    buttonPrev: JQuery<HTMLButtonElement>;
    buttonNext: JQuery<HTMLButtonElement>;
    searchButton: JQuery<HTMLButtonElement>;
    searchInputElement: JQuery<HTMLInputElement>;
    objPerPage?: number;
    currentPage?: number;
}

export abstract class Paginator<T> {
    protected abstract getRequestIdUrl(
        page: number, objPerPage: number, searchInput: string
    ): string;
    protected abstract loadNewPageData(data: T): void;
    public abstract getAvailablePages(data: T): number;

    // Fields
    protected static readonly DefaultAmountPerPage = 5;

    readonly buttonPrev: JQuery<HTMLButtonElement>;
    readonly buttonNext: JQuery<HTMLButtonElement>;
    readonly searchButton: JQuery<HTMLButtonElement>;
    readonly searchInputElement: JQuery<HTMLInputElement>;
    private _objPerPage: number;
    private _currentPage: number;
    private _availablePages: number;
    private _lastQuery: string;
    // -----

    get objPerPage() { return this._objPerPage; }
    get currentPage() { return this._currentPage; }
    get lastQuery() { return this._lastQuery; }
    get currentInput() { return (this.searchInputElement.val() as string).trim(); }
    get availablePages() { return this._availablePages; }
```

```

protected constructor(options: PaginatorOptions) {
    this.buttonPrev      = options.buttonPrev;
    this.buttonNext      = options.buttonNext;
    this.searchButton    = options.searchButton;
    this.searchInputElement = options.searchInputElement;
    this._objPerPage     = Types.conforms<number>(options.objPerPage,
Types.isPositiveInteger)
        ? options.objPerPage
        : Paginator.DefaultAmountPerPage;
    this._currentPage = Types.conforms<number>(
        options.currentPage, Types.isZeroOrPositiveInteger
    )
        ? options.currentPage
        : 0;
    this._availablePages = 1;
    this._lastQuery = '';
    this.setupEvents();
}

async launch() {
    await this.doSearch(0, '');
    this.updateButtons();
}

private updateButtons() {
    this.buttonPrev.prop('disabled', !this._currentPage);
    this.buttonNext.prop(
        'disabled',
        this._currentPage >= this._availablePages - 1
    );
}

private async loadPrevPage() {
    if (!this._currentPage) {
        throw new Error('Undefined behaviour');
    }
    if (!(this._currentPage - 1)) {
        this.buttonPrev.prop('disabled', true);
    }
    await this.doSearch(this._currentPage - 1, this._lastQuery);
}

private async loadNextPage() {
    if (this._currentPage >= this._availablePages - 1) {
        throw new Error('Undefined behaviour');
    }
    if (this._currentPage + 1 >= this._availablePages - 1) {
        this.buttonNext.prop('disabled', true);
    }
    await this.doSearch(this._currentPage + 1, this._lastQuery);
}

private async loadNewQuery() {
    await this.doSearch(0, this.currentInput);
}

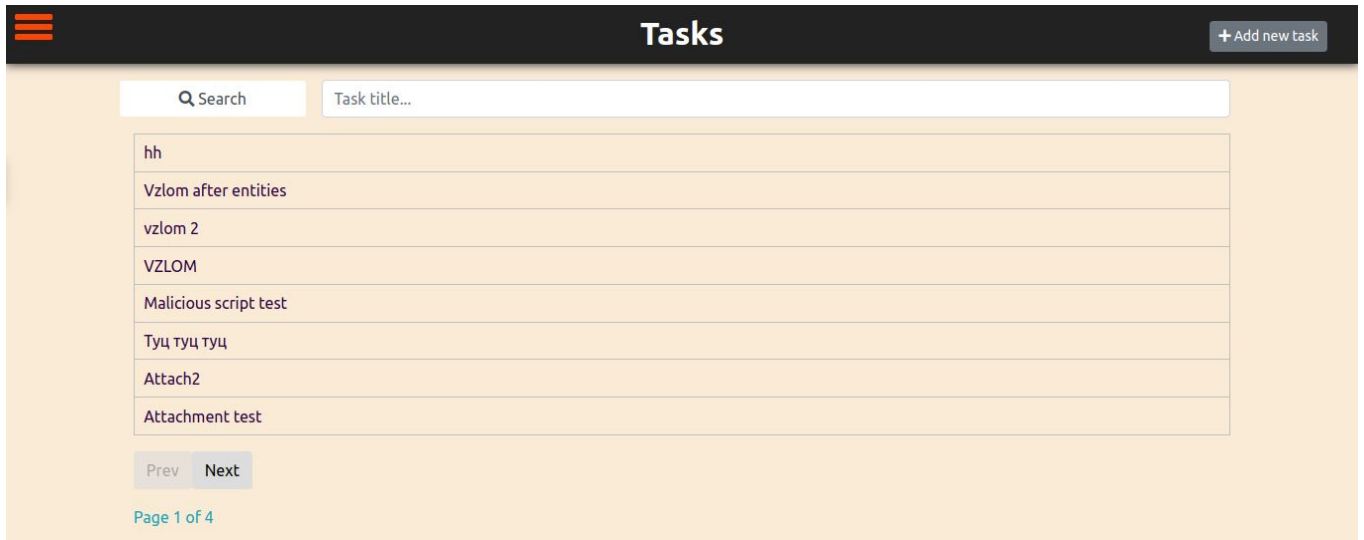
private async doSearch(page: number, query: string) {
    try {
        const data: T = await $.get(this.getRequestUrl(
            page, this._objPerPage, query
        ));
        console.dir(data);
        this._availablePages = this.getAvailablePages(data);
        this._currentPage = page;
        this._lastQuery = query;
        this.updateButtons();
        this.loadNewPageData(data);
    }
}

```

```
    } catch (err) {  
      console.error(`Paginator failed to reload page: ${err.message}`);  
    }  
  }  
  
  private setupEvents() {  
    this.buttonPrev.on('click', () => this.loadPrevPage());  
    this.buttonNext.on('click', () => this.loadNextPage());  
    this.searchInputElement.on('keydown', (event) => {  
      if (event.key === 'Enter') {  
        this.loadNewQuery();  
      }  
    });  
    this.searchButton.on('click', () => this.loadNewQuery());  
  }  
}
```

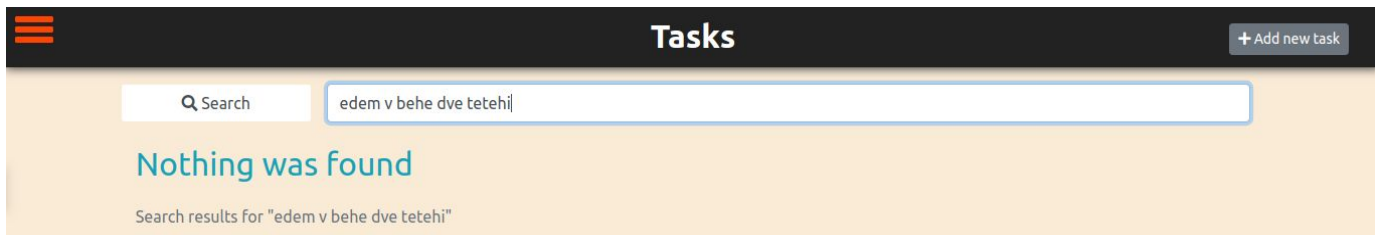
Приклади результатів

Сторінка зі списком сутностей з пагінацією.



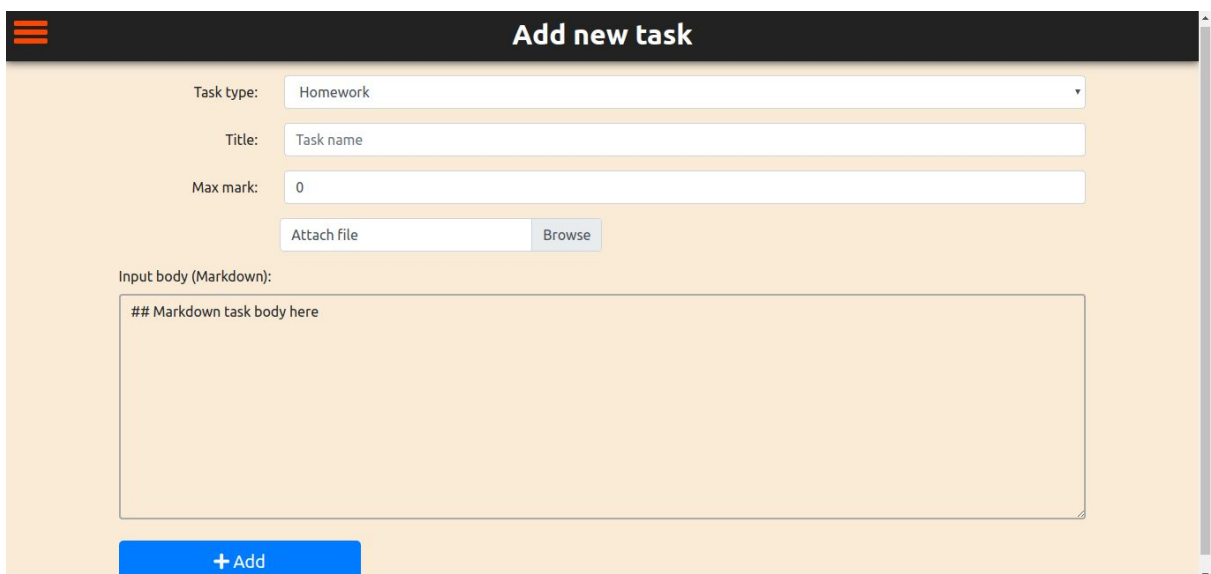
The screenshot shows a web application interface for managing tasks. At the top, there is a dark header with a hamburger menu icon on the left, the title "Tasks" in the center, and a "+ Add new task" button on the right. Below the header, there is a search bar with a magnifying glass icon and the placeholder text "Search". To the right of the search bar is a text input field with the placeholder text "Task title...". Below these elements is a table with a light orange background. The table has one visible row with the following content: "hh", "Vzлом after entities", "vzлом 2", "VZLOM", "Malicious script test", "Туц туц туц", "Attach2", and "Attachment test". Below the table, there are two buttons: "Prev" and "Next". At the bottom left, there is a text label "Page 1 of 4".

Повідомлення про невдалий пошук сутностей.



The screenshot shows the same "Tasks" page as before, but with a search result message. The search bar now contains the text "edem v behe dve teteht". Below the search bar, there is a large text message in blue: "Nothing was found". Below this message, there is a smaller text label: "Search results for 'edem v behe dve teteht'".

Сторінка додавання нової сутності за допомогою форми:



The screenshot shows a web application interface for adding a new task. At the top, there is a dark header with a hamburger menu icon on the left, the title "Add new task" in the center, and a "+ Add new task" button on the right. Below the header, there is a form with the following fields: "Task type:" with a dropdown menu showing "Homework"; "Title:" with a text input field showing "Task name"; "Max mark:" with a text input field showing "0"; and "Attach file" with a text input field and a "Browse" button. Below these fields, there is a text label "Input body (Markdown):" and a large text area with the placeholder text "## Markdown task body here". At the bottom left, there is a blue button with a white plus sign and the text "Add".

Висновки

Під час лабораторної роботи я вдосконалив свої навички роботи з фронтендом та використав бібліотеку JQuery для пагінації за допомогою AJAX запитів на сервер. Створив свій npm пакет для зручного використання парсера multipart/form-data з бібліотеки formidable у якості фабрики middleware для ExpressJS. Вивчив основні принципи асинхронного програмування за допомогою колбек функцій.