**Національний технічний університет України**

**“Київський політехнічний інститут імені Ігоря Сікорського”**

**Факультет прикладної математики**

**Кафедра системного програмування і спеціалізованих комп’ютерних систем**

**ЛАБОРАТОРНА РОБОТА №2**

***з дисципліни “Web-дизайн”***

**ТЕМА: “Розробка функціональності Web-додатка мовою Javascript”**

**Виконала: студентка 3 курсу**

**Групи КВ-12**

**Бурда Є. А.**

**Оцінка:**

**Київ – 2024**

**Загальна інформація**

*Завдання на лабораторну роботу:*

розробити функціональність для статичних сторінок Web-додатку першої лабораторної роботи із використанням шаблону MVC

*Контакт студента в Telegram:*

[https://t.me/night\_fool](https://t.me/night_fool1)

*Назва додатку:*

Contact Book

*Короткий опис його функціональності:*

Телефонна книга, що дає можливість додавати, вилучати та редагувати

контакти. Можна сортувати абонентів за алфавітом.

*Схема переходу:*

З будь-якої сторінки сайту можна перейти на будь-яку іншу сайту. Всі сторінки з’єднані між собою.

**Діаграма модулів Javascript додатку**

Зображення, що містить ряд, знімок екрана, схема, текст

Автоматично згенерований опис

Стрілочка веде від модуля, що імпортує, до модуля, що експортується.

model.js – зберігає дані про користувачів (імена, паролі, електронні адреси) та їх контакти. Методи для додавання, редагування та видалення користувачів та їх контакти. Перевірка правильності даних перед збереженням.

view.js – відображає сторінки веб додатку, оновлює інтерфейс при змінні даних користувачів або контактів. Відображає повідомлення про помилки.

controller.js – взаємодія з користувачем, через реакцію на взаємодію з елементами інтерфейсу. Виклик для збереження або оновлення даних. Оновлення через виклик методів відображення.

main.js – ініцілізація та запуск додатку. Підключення модулів моделі, виду і контролера. Обробка та завантаження даних при запуску додатку.

**Скріншоти усіх сторінок Web-додатка**

*Робоча сторінка*

Зображення, що містить текст, знімок екрана, фіалка

Автоматично згенерований опис

*Профіль*

Зображення, що містить текст, знімок екрана

Автоматично згенерований опис

*Про додаток*

*Зображення, що містить текст, знімок екрана, Шрифт, програмне забезпечення

Автоматично згенерований опис*

*Зображення, що містить текст, знімок екрана, програмне забезпечення, Мультимедійне програмне забезпечення

Автоматично згенерований опис*

*Сторінка реєстрації*

***Зображення, що містить текст, знімок екрана, фіолетовий

Автоматично згенерований опис***

*Сторінка входу*

*Зображення, що містить текст, знімок екрана, Мультимедійне програмне забезпечення, програмне забезпечення

Автоматично згенерований опис*

**Код Javascript-файлів**

*main.js*

1. // Підключення модулів
2. **import** **Model** **from** './model.js';
3. **import** **View** **from** './view.js';
4. **import** **Controller** **from** './controller.js';
5. // Ініціалізація моделі, виду і контролера
6. **const** model = **new** **Model**();
7. **const** view = **new** **View**();
8. **const** controller = **new** **Controller**(model, view);
9. // Функція для запуску додатку
10. **function** **initApp**() {
11. // Завантаження даних при запуску
12. model.**loadInitialData**();
13. // Відображення веб-інтерфейсу
14. view.**render**();
15. }
16. // Обробка події завантаження сторінки
17. document.**addEventListener**('DOMContentLoaded', initApp);

*controller.js*

1. **export** **default** **class** **Controller** {
2. **constructor**(model, view) {
3. this.model = model;
4. this.view = view;
5. // Додавання обробників подій
6. this.view.**bindCreateContact**(this.handleCreateContact);
7. this.view.**bindUpdateContact**(this.handleUpdateContact);
8. this.view.**bindDeleteContact**(this.handleDeleteContact);
9. this.view.**bindSortContacts**(this.handleSortContacts);
10. }
11. // Обробник події створення контакту
12. handleCreateContact = (contactData) => {
13. this.model.**createContact**(contactData);
14. this.view.**renderContacts**(this.model.contacts);
15. }
16. // Обробник події оновлення контакту
17. handleUpdateContact = (contactId, updatedData) => {
18. this.model.**updateContact**(contactId, updatedData);
19. this.view.**renderContacts**(this.model.contacts);
20. }
21. // Обробник події видалення контакту
22. handleDeleteContact = (contactId) => {
23. this.model.**deleteContact**(contactId);
24. this.view.**renderContacts**(this.model.contacts);
25. }
26. // Обробник події сортування контактів
27. handleSortContacts = (sortBy) => {
28. this.model.**sortContacts**(sortBy);
29. this.view.**renderContacts**(this.model.contacts);
30. }
31. }

*model.js*

1. **export** **default** **class** **Model** {
2. **constructor**() {
3. // Ініціалізуємо базу даних контактів
4. this.contacts = [];
5. }
6. // Метод для додавання нового контакту
7. **addContact**(contact) {
8. this.contacts.**push**(contact);
9. }
10. // Метод для оновлення інформації про контакт
11. **updateContact**(id, updatedContact) {
12. **const** index = this.contacts.**findIndex**(contact => contact.id === id);
13. **if** (index !== -1) {
14. this.contacts[index] = { ...this.contacts[index], ...updatedContact };
15. }
16. }
17. // Метод для видалення контакту
18. **deleteContact**(id) {
19. this.contacts = this.contacts.**filter**(contact => contact.id !== id);
20. }
21. // Метод для сортування контактів за вказаним параметром
22. **sortContacts**(sortBy) {
23. **switch** (sortBy) {
24. **case** 'name':
25. this.contacts.**sort**((a, b) => a.name.**localeCompare**(b.name));
26. **break**;
27. **case** 'email':
28. this.contacts.**sort**((a, b) => a.email.**localeCompare**(b.email));
29. **break**;
30. **case** 'number':
31. this.contacts.**sort**((a, b) => a.number.**localeCompare**(b.number));
32. **break**;
33. default:
34. **break**;
35. }
36. }
37. }

*view.js*

1. **export** **default** **class** **View** {
2. **constructor**() {
3. this.createContactForm = document.**getElementById**('createContactForm');
4. this.updateContactForm = document.**getElementById**('updateContactForm');
5. this.deleteContactForm = document.**getElementById**('deleteContactForm');
6. this.sortContactsForm = document.**getElementById**('sortContactsForm');
7. this.contactsContainer = document.**getElementById**('contactsContainer');
8. this.createContactForm.**addEventListener**('submit', this.handleCreateContact);
9. this.updateContactForm.**addEventListener**('submit', this.handleUpdateContact);
10. this.deleteContactForm.**addEventListener**('submit', this.handleDeleteContact);
11. this.sortContactsForm.**addEventListener**('submit', this.handleSortContacts);
12. }
13. // Метод для відображення контактів
14. **renderContacts**(contacts) {
15. // Очищення контейнера перед оновленням
16. this.contactsContainer.innerHTML = '';
17. // Відображення кожного контакту в контейнері
18. contacts.**forEach**(contact => {
19. **const** contactElement = document.**createElement**('div');
20. contactElement.textContent = `${contact.name} - ${contact.email} - ${contact.number}`;
21. this.contactsContainer.**appendChild**(contactElement);
22. });
23. }
24. // Метод для отримання даних форми створення контакту
25. **getCreateContactData**() {
26. **const** name = this.createContactForm.**querySelector**('#name').value;
27. **const** email = this.createContactForm.**querySelector**('#email').value;
28. **const** number = this.createContactForm.**querySelector**('#number').value;
29. **return** { name, email, number };
30. }
31. // Метод для отримання даних форми оновлення контакту
32. **getUpdateContactData**() {
33. **const** id = this.updateContactForm.**querySelector**('#contactId').value;
34. **const** name = this.updateContactForm.**querySelector**('#name').value;
35. **const** email = this.updateContactForm.**querySelector**('#email').value;
36. **const** number = this.updateContactForm.**querySelector**('#number').value;
37. **return** { id, name, email, number };
38. }
39. // Метод для отримання id контакту для видалення
40. **getDeleteContactId**() {
41. **return** this.deleteContactForm.**querySelector**('#contactId').value;
42. }
43. // Метод для отримання параметрів сортування контактів
44. **getSortBy**() {
45. **return** this.sortContactsForm.**querySelector**('input[name="sortBy"]:checked').value;
46. }
47. // Метод для прив'язки обробників подій до форм
48. **bindCreateContact**(handler) {
49. this.handleCreateContact = (event) => {
50. event.**preventDefault**();
51. **handler**(this.**getCreateContactData**());
52. }
53. }
54. **bindUpdateContact**(handler) {
55. this.handleUpdateContact = (event) => {
56. event.**preventDefault**();
57. **handler**(this.**getUpdateContactData**());
58. }
59. }
60. **bindDeleteContact**(handler) {
61. this.handleDeleteContact = (event) => {
62. event.**preventDefault**();
63. **handler**(this.**getDeleteContactId**());
64. }
65. }
66. **bindSortContacts**(handler) {
67. this.handleSortContacts = (event) => {
68. event.**preventDefault**();
69. **handler**(this.**getSortBy**());
70. }
71. }
72. }