ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«УЖГОРОДСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ»
ФАКУЛЬТЕТ МАТЕМАТИКИ ТА ЦИФРОВИХ ТЕХНОЛОГІЙ

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ ДО ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ З СУЧАСНИХ ТЕХНОЛОГІЙ WEB-ПРОГРАМУВАННЯ. Ч.1

(для студентів спеціальностей 124 «Системний аналіз» та 113 «Прикладна математика»)

Ужгород – 2021

Методичні рекомендації до виконання лабораторних робіт із сучасних технології web-програмування. Частина 1 (для студентів спеціальностей 124 «Системний аналіз» та 113 «Прикладна математика») / Упоряд.: Ю.В. Андрашко, Т.Ю. Жабур. Ужгород: УжНУ, 2021. 32 с

Упорядники:

Андрашко Юрій Васильович, к.т.н., доцент кафедри системного аналізу та теорії оптимізації;

Жабур Тетяна Юріївна, вчитель методист, вчитель інформатики та технологій Ужгородської загальноосвітньої спеціалізованої школи-інтернату з поглибленим вивченням окремих предметів Закарпатської обласної ради;

Рецензент: Млавець Юрій Юрійович, кандидат фіз.-мат. наук,
доцент кафедри кібернетики і прикладної математики.

Рекомендовано до друку Науково-методичною комісією факультету математики та цифрових технологій (протокол №7 від 24 травня 2021 року).

Вступ

Історично склалося, що всі браузери, мобільні і десктопні, на всіх платформах розуміють всього лише три речі: HTML, CSS і JavaScript. Були спроби запровадити в браузери інші технології: Visual Basic, Java, ActiveX,–- але всі вони провалилися, тому що виробники заліза і браузерів не змогли домовитися про відкриті стандарти. Залишилися тільки відкриті стандарти, що розробляються публічними консорціумами і робочими групами. Наприклад, W3C розробляє HTML і CSS. Отже, у нас є три технології.

- HTML відповідає за структуру сторінки.
- CSS - за оформлення сторінки.
- JavaScript - за взаємодію сторінки з користувачем.

Кожна технологія розвивається незалежно, у кожної є кілька версій.

Кожен раз писати код на JavaScript, щоб зробити одні і ті ж дії не ефективно, тому з'явилися бібліотеки – набори готових функцій на JavaScript, що виконують типові операції з HTML-кодом сторінки. Приклад такої бібліотеки – jQuery.

Для HTML та CSS теж стали з'являтися подібні «напівфабрикати» – заготовки з шматків коду, які вирішують типові проблеми верстки. Наприклад, багатоколонкові верстка, закріплений на сторінці хедер або футер і інші типові завдання, які вигідніше вирішити один раз, а потім застосовувати в нових проектах.

Так з'явилися перші фронтенд-каркаси розробки, або фреймворки. Чому вони не бібліотеки? Тому що це не набір готових функцій, які можна додати до проекту і використовувати точково на окремих сторінках. Каркас (фреймворк) передбачає, що весь проект буде слідувати заданої ними структурі. Тобто він задає обмеження, яких потрібно дотримуватися, щоб прискорити розробку, точніше слідувати стандартам. Найвідоміший зразок HTML та CSS-фреймворка – Bootstrap.

Кілька років тому з'явилася ідея – оскільки більшість сайтів і мобільних додатків оперують обмеженим набором шаблонів, розумно відокремити подання

даних від власне даних. Нехай бекенд займається тільки зберіганням, обробкою і безпекою даних, а все інше доручимо клієнту (браузеру). Дамо йому набір даних і шаблон – набір інструкцій по перетворенню даних в верстку. Фреймворк на клієнті буде підставляти дані в шаблон, реалізовувати бізнес-логіку та маніпулювати сторінкою.

Такі фреймворки називаються реактивними, тому що в них стан інтерфейсу автоматично реагує на зміну даних. Якщо сказати «ось X змінна управляє кольором кнопки», а потім змінити значення змінної то колір кнопки зміниться сам, перефарбовувати її вручну не треба.

Приклади реактивних фреймворків: React, Angular, Vue і ще десятки менш відомих.

Отже, на бекенд можна використовувати будь-яку мову програмування, добувати дані, упаковувати їх в JSON, XML або інший формат і віддавати на фронт.

А на фронтенді JS-фреймворк робить всю чистову роботу: відображає елементи управління, анімує їх, перевіряє дані, представляє їх відповідно до локальних налаштуквань користувача і реалізує бізнес-логіку.

1. Фреймворк Vue

Vue – це прогресивний фреймворк для створення користувацьких інтерфейсів. На відміну від фреймворків-монолітів, Vue створений придатним для поступового впровадження. Його ядро в першу чергу вирішує завдання рівня уявлення (view), що спрощує інтеграцію з іншими бібліотеками та існуючими проектами. З іншого боку, Vue повністю підходить і для створення складних односторінкових додатків (SPA, Single-Page Applications), якщо використовувати його спільно з сучасними інструментами та додатковими бібліотеками.

Vue CLI - це повна система для швидкої розробки Vue.js, що забезпечує:

- Інтерактивне редагування проекту через `@vue/cli`.
- Початкова конфігурація швидкого прототипування за допомогою `@vue/cli`
- Залежність від середовища виконання (`@vue/cli-service`), яке:
  - Підтримує оновлення;
  - Вбудоване поверх веб-пакета з розумними значеннями за замовчуванням;
  - Налаштовується за допомогою конфігураційного файлу в проекті;
  - Розширюється за допомогою плагінів;
  - Містить офіційні плагіни що інтегрують найкращі інструменти у зовнішній екосистемі;
  - Повноцінний графічний інтерфейс для створення та управління проектами Vue.js.

Vue CLI є стандартною базою інструментів для екосистеми Vue. Це забезпечує безперебійну роботу різних інструментів побудови разом із розумними значеннями за замовчуванням, завдяки чому можна зосередитись на написанні програми, а не витрачати час на налаштування конфігурації. У той же час, `@vue/cli` пропонує гнучкість налаштування конфігурації кожного інструменту без необхідності виштовхування.

Щоб встановити новий пакет, використовуйте наступну команду.

```
npm install -g @vue/cli
```

Вам потрібні привілеї адміністратора для їх виконання, якщо npm не було встановлено у вашій системі через менеджер версій Node.js.

Після встановлення ви отримаєте доступ до двійкового файлу vue у вашому командному рядку. Ви можете переконатися, що він правильно встановлений, просто запустивши vue, який повинен представити вам довідкове повідомлення із переліком усіх доступних команд.

Ви можете перевірити, чи маєте ви правильну версію, за допомогою цієї команди:

```
vue -version
```

2. Лабораторна робота №1. Створення проекту Vue

Завдання 1.

1. Сворити проєкт Vue.

2. Створити шаблон компонента Vue, який містить інформацію про групу, ПІБ студета та номер варіанту.

3. Спеціальність, курс, номер варіанту та ПІБ зберегти як дані (data)

4. Застосувати стилі для оформлення сторінки (на розсуд студента)

Щоб створити новий проект, запустіть:

```
vue create project-name
```

Вам буде запропоновано вибрати пресет. Ви можете вибрати стандартний пресет, який постачається з базовою настройкою Babel + ESLint (рис. 1)

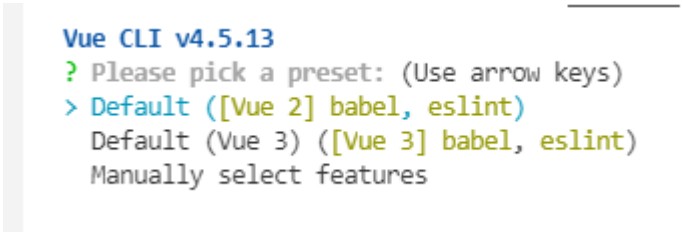


Рис. 1. Діалог творення нового проєкту

Це те, що ви побачите в package.json проекту за допомогою пресету, встановленого за замовчуванням:

```
{
  "scripts": {
    "serve": "vue-cli-service serve",
    "build": "vue-cli-service build"
  }
}
```

Ви можете викликати ці скрипти за допомогою команди

```
npm run serve
```

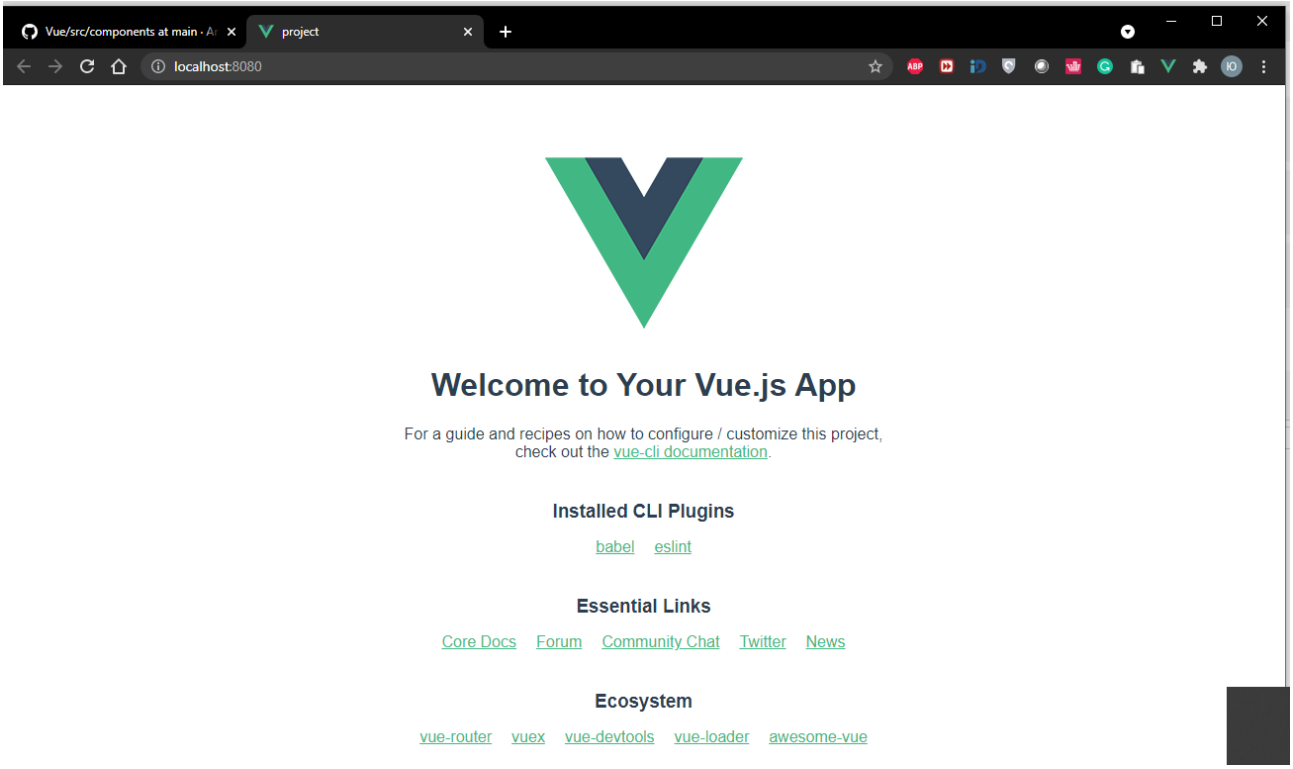


Рис.2. Приклад запущеного проєкту Vue.

Кожна програма починається зі створення нового екземпляра Vue за допомогою функції Vue:

```
import Vue from 'vue'
import App from './App.vue'

new Vue({
  render: h => h(App),
}).$mount('#app')
```

Хоч Vue і не реалізує патерн MVVM в повній мірі, архітектура фреймворка їм багато в чому надихнула. Тому змінну з екземпляром Vue традиційно називають vm (скорочено від ViewModel).

При створенні екземпляра Vue необхідно передати об'єкт опцій.

Кожен екземпляр Vue при створенні проходить через послідовність кроків ініціалізації – наприклад, налаштовує спостереження за даними, компілює шаблон, монтує екземпляр в DOM, оновлює DOM при зміні даних. Між цими кроками викликаються функції, звані хуками життєвого циклу, за допомогою яких можна виконувати свій код на певних етапах.

Наприклад, хук `created` можна використовувати для виконання коду після створення екземпляра:

Існують і інші хуки, що викликаються на різних стадіях життєвого циклу примірника, наприклад `mounted`, `updated` і `destroyed` (Рис. 3.). Все хуки викликаються з контекстної змінної `this`, що посилається на викликає екземпляр Vue.

Для зв'язування DOM з даними примірника Vue використовує синтаксис, заснований на HTML. Всі шаблони Vue є дійсним HTML-кодом, який можуть розпарсити все HTML-парсери і браузери.

Для роботи Vue компілює шаблони в render-функції віртуального DOM. У поєднанні з системою реактивності, Vue вміє визначати мінімальне число компонентів для повторної відтворення і застосовує мінімальну кількість маніпуляцій до DOM при зміні стану програми.

Для виконання 2 завдання замінимо стандартний шаблон:

```
<template>
  <div id="app">
    <img alt="Vue logo" src="./assets/logo.png">
    <HelloWorld msg="Welcome to Your Vue.js App"/>
  </div>
</template>
```

На шаблон компонента Vue, який містить інформацію про групу, ПІБ студета та номер варіанту (Рис 4.):

```
<template>
  <div id="app">
    <p> <b>Група</b>: ПМ4</p>
    <p> <b>Студент</b>: Андрашко Юрій Васильович</p>
    <p> <b>Варіант</b>: 1</p>
  </div>
</template>
```

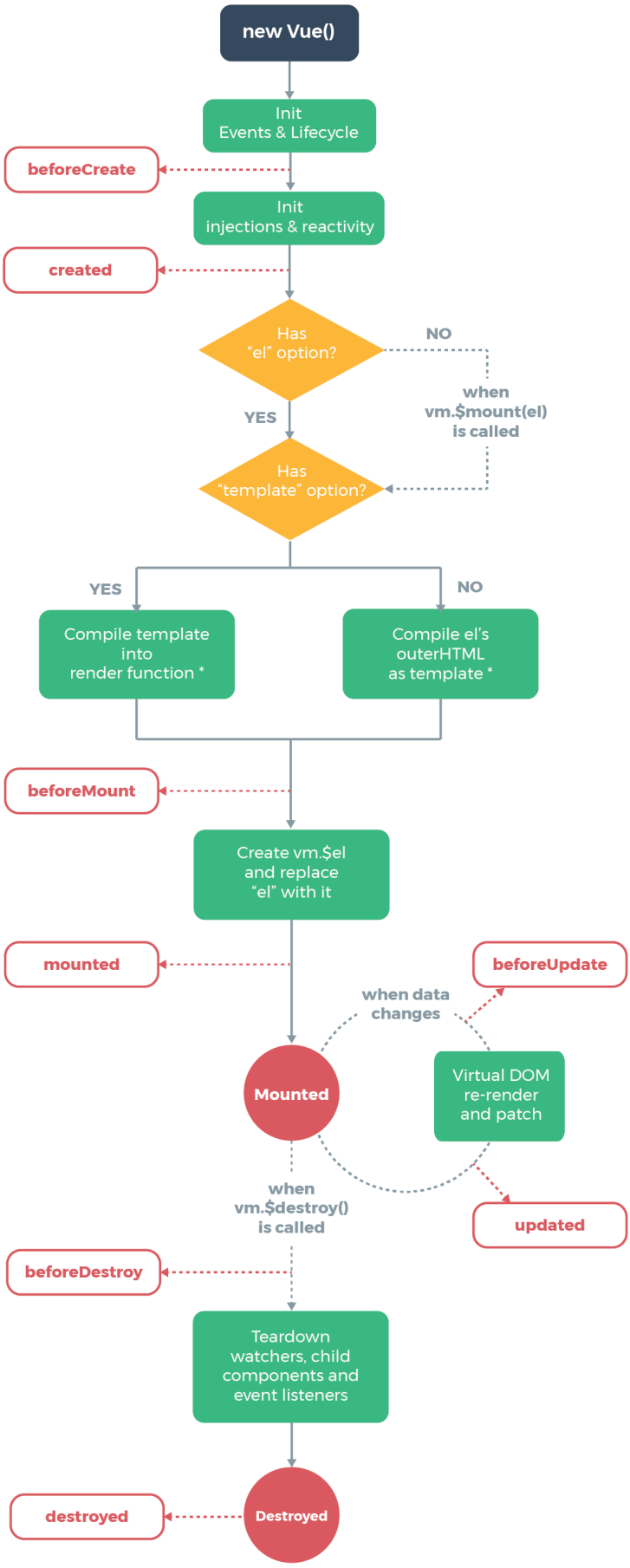


Рис. 3. Життєвий цикл Vue

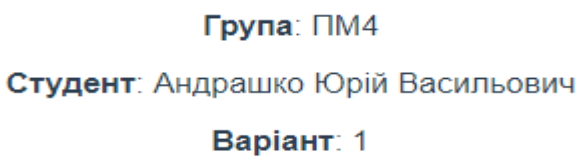


Рис. 4. Результати рендеру шаблону.

Коли екземпляр Vue створений, він додає всі властивості, знайдені в опції data, в систему реактивності Vue. Тому уявлення буде «реагувати» на їх зміни, оновлюючись відповідно до нових значень. Найбільш простий спосіб зв'язування даних – це текстова інтерполяція з використанням синтаксису Mustache (подвійних фігурних дужок):

```
<template>
  <div id="app">
    <p><b>Група</b>: {{ group }}</p>
    <p><b>Студент</b>: {{ name }}</p>
    <p><b>Варіант</b>: {{ course }}</p>
  </div>
</template>

<script>
export default {
  name: "App",
  data() {
    return {
      group: "ПМ4",
      name: "Андрашко Юрій Васильович",
      course: 1,
    };
  },
};
</script>
```

Для виконання 4 завдання можна застосувати довільні стилі, наприклад.

```
<style>
#app {
  font-family: Avenir, Helvetica, Arial, sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  text-align: center;
  color: #2c3e50;
  margin-top: 60px;
}
</style>
```

3. Лабораторна робота №2. SPA

Завдання. Використовуючи фреймворк Vue розробити SAP для відображення колекції об’єктів. Реалізувати функції

• додавання одного об’єкту

• редагування інформації об’єкту

• видалення інформації про вказаний об’єкт.

• отримання вибірки з колекції згідно з вказаним запитом

Об’єкт “Книги” (код; прізвище автора; назва книги; рік видання; кількість сторінок; ціна). Запит книг які було виданно у році X.

Односторінковий додаток (SPA) – це веб-додаток або веб-сайт, який взаємодіє з користувачем шляхом динамічного переписування поточної веб-сторінки з новими даними з веб-сервера, замість того, щоб веб-браузер за завантажував цілі нові сторінки. Мета – швидший перехід, завдяки якому веб-сайт відчуває себе як при роботі зі звичайними програмами.

У SPA ніколи не відбувається оновлення сторінки; натомість весь необхідний код HTML, JavaScript та CSS браузер отримує або за допомогою одного завантаження сторінки, або відповідні ресурси динамічно завантажуються та додаються на сторінку за необхідності, як правило, у відповідь на дії користувача. Сторінка не перезавантажується в будь-який момент процесу, а також не передає управління на іншу сторінку, хоча хеш розташування або API історії HTML5 можуть бути використані для забезпечення сприйняття та навігації окремих логічних сторінок у програмі.

Директиви - це спеціальні атрибути з префіксом v-. Як значення вони приймають один вислів JavaScript (за винятком `v-for`). Директива реактивно застосовує до DOM зміни при оновленні значення цього виразу.

Директива `v-if` використовується для рендеринга блоку за умовою. Блок буде відображатися тільки в тому випадку, якщо вираз директиви повертає значення, що приводиться до `true`.

Використовуйте директиву `v-for` для відтворення списку елементів на основі масиву даних. У директиви `v-for` особливий синтаксис запису: `item in items`, де `items` - вихідний масив, а `item` - посилання на поточний елемент масиву.

Усередині блоку `v-for` нам доступні властивості з області видимості батька. У `v-for` також є другий опціональний параметр з індексом поточного елемента. Замість `in` роздільником можна використовувати `of`, як в Ітераторах JavaScript.

При оновленні Vue списку елементів, що відображається директивою v-for, за замовчуванням використовується стратегія оновлення «на місці». Якщо порядок елементів масиву змінився, Vue не стане переміщати елементи DOM, а просто оновить кожен елемент «на місці», щоб він відображав нові дані по відповідному індексу.

Режим за замовчуванням ефективний, але застосовується лише у випадках, коли результат відтворення вашого списку не покладається на стан дочірніх компонентів або тимчасові стану DOM (наприклад, значення полів форм).

Щоб підказати Vue, як відстежувати ідентичність кожного елемента, що дозволить перевикористати і переміщати існуючі елементи, вкажіть унікальний атрибут key для кожного елемента. Не вказуйте в якості ключів v-for непрімітівние значення. Замість цього використовуйте рядкові або числові значення.

Деякі директиви можуть приймати аргумент, відокремлений від назви директиви двокрапкою. Наприклад, директива `v-bind` використовується для реактивного поновлення атрибутів HTML:

```
<a v-bind:href="url"> ... </a>
```

В даному випадку `href` - аргумент, який вказує директиві v-bind зв'язати атрибут `href` елемента зі значенням вирази url.

Іншим прикладом буде директива `v-on`, яка відстежує події DOM:

```
<a v-on:click="doSomething"> ... </a>
```

В даному випадку аргументом є тип події.

Префікс v- служить для візуального визначення Vue-специфічних атрибутів в шаблонах. Це зручно, коли Vue використовується для додавання динамічного поведінки в існуючій розмітці, але для часто використовуваних директив може здатися багатослівним. У той же час необхідність в v- менш значима при створенні односторінкових додатків, де Vue контролює кожен шаблон. Тому є скорочений запис для двох найбільш часто використовуваних директив, v-bind і v-on:

Скорочення v-bind

```
<! - повний синтаксис ->
<a v-bind:href="url"> ... </a>
<! - скорочений запис ->
<a :href="url"> ... </a>
```

Скорочення v-on

```
<! - повний синтаксис ->
<a v-on:click="doSomething"> ... </a>
<! - скорочений запис ->
<a @click="doSomething"> ... </a>
```

Логіка обробника події може бути досить складною, тому залишати JavaScript-код в значенні атрибута v-on не завжди можливо. В цьому випадку v-on може прийняти і назва методу, який потрібно викликати.

Методи, які будуть підмішані до примірника Vue описуються в опції methods. Можна запускати ці методи безпосередньо з примірника VM, або використовувати їх в директивах. this методів вказує на екземпляр Vue.

Не використовуйте стрілочні функції при визначенні методів (наприклад, `plus: () => this.a ++)`. Стрілочні функції зв'язуються з батьківським контекстом, тому `this` буде вказувати не на екземпляр Vue, і `this.a` виявиться невизначеним.

Вбудовувані в шаблони вираження зручні, але можуть призначатися тільки для простих операцій. При ускладненні логіки їх важче підтримувати:

```
{{ message.split('').reverse().join('') }}
```

Такий шаблон вже не виглядає простим і декларативним. З першого погляду і не скажеш, що він всього лише відображає message задом наперед. Ситуація стане ще гірше, якщо цю логіку потрібно використовувати в декількох місцях шаблону. На допомогу тут приходять обчислювані властивості.

```
computed: {
    reversedMessage: function () {
      return this.message.split('').reverse().join('')
    }
  }
```

Ми визначили як обчислюється властивість reversedMessage. Написана нами функція буде використовуватися як геттер властивості `vm.reversedMessage`

Використовуючи ці знання реалізуємо завдання лабораторної роботи (рис. 5-8)

Рис 5. Вивід списку книг

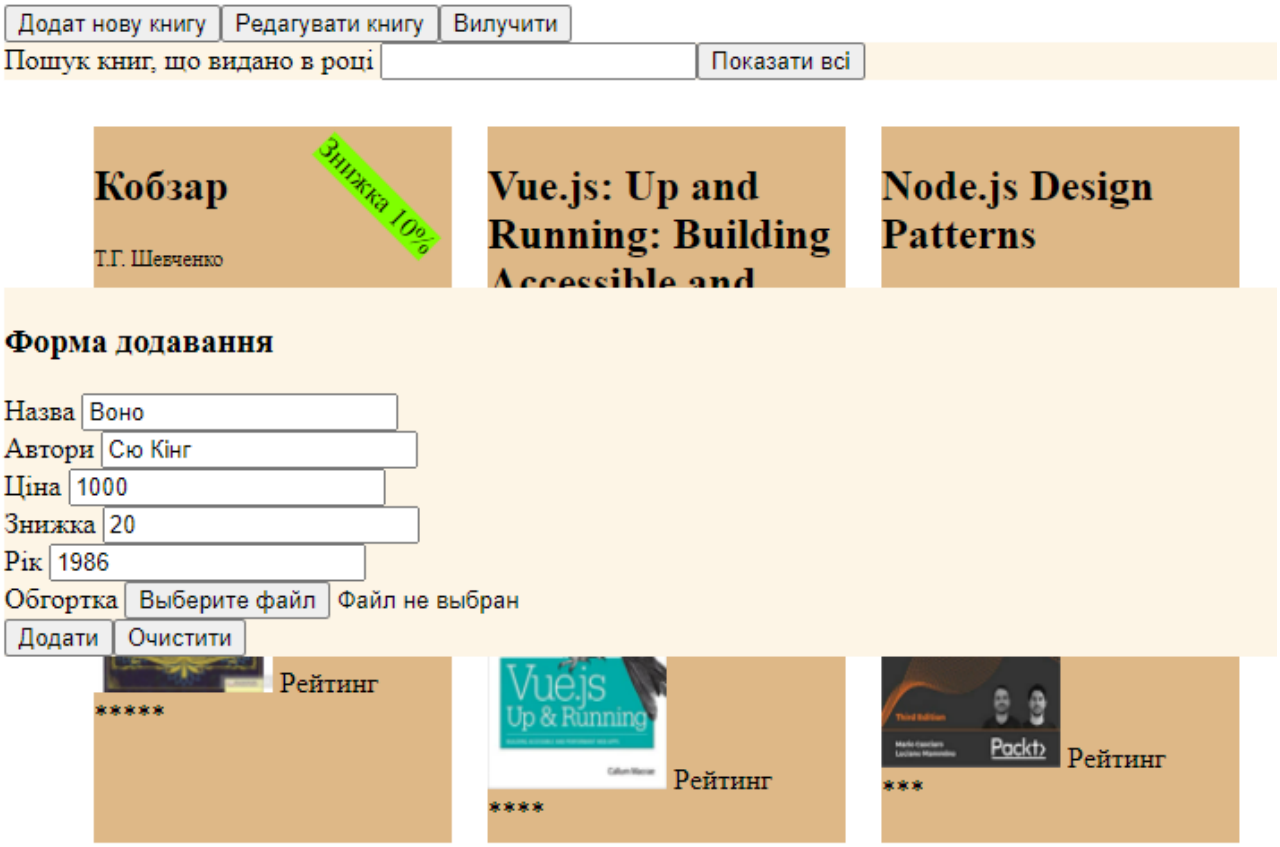


Рис 6. Додавання книги

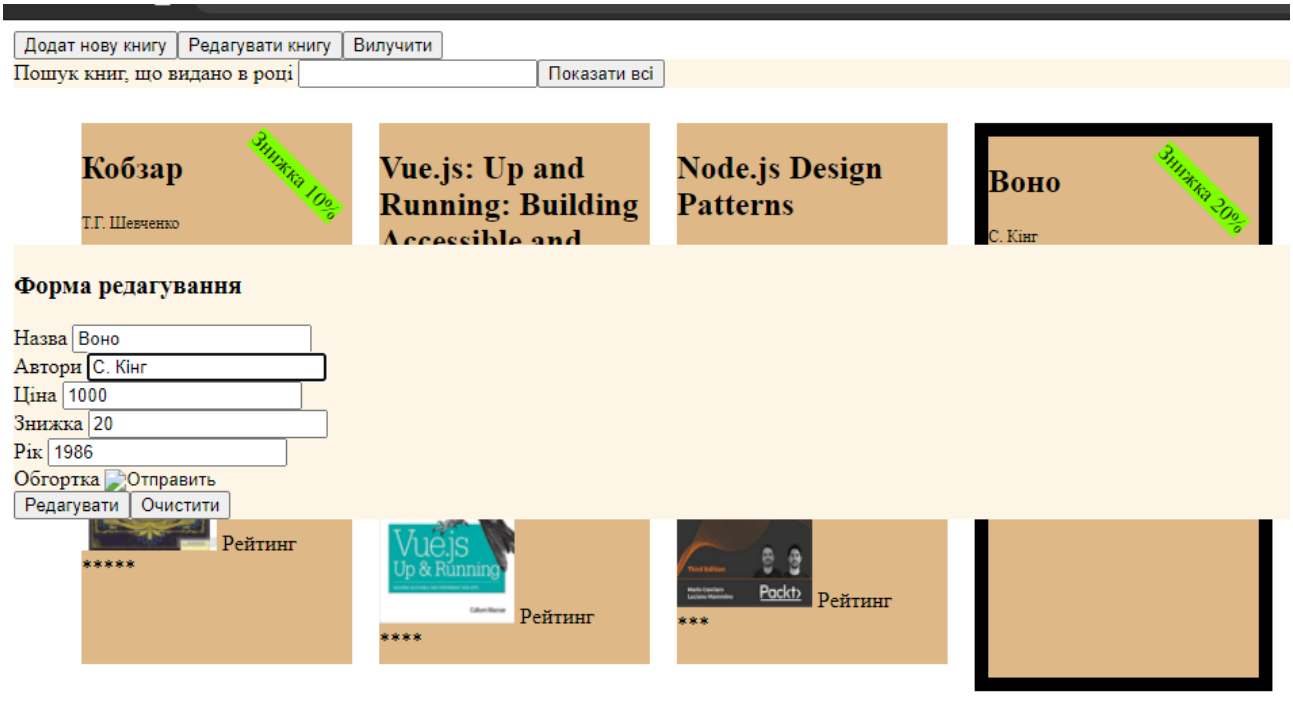


Рис 7. Редагування книги

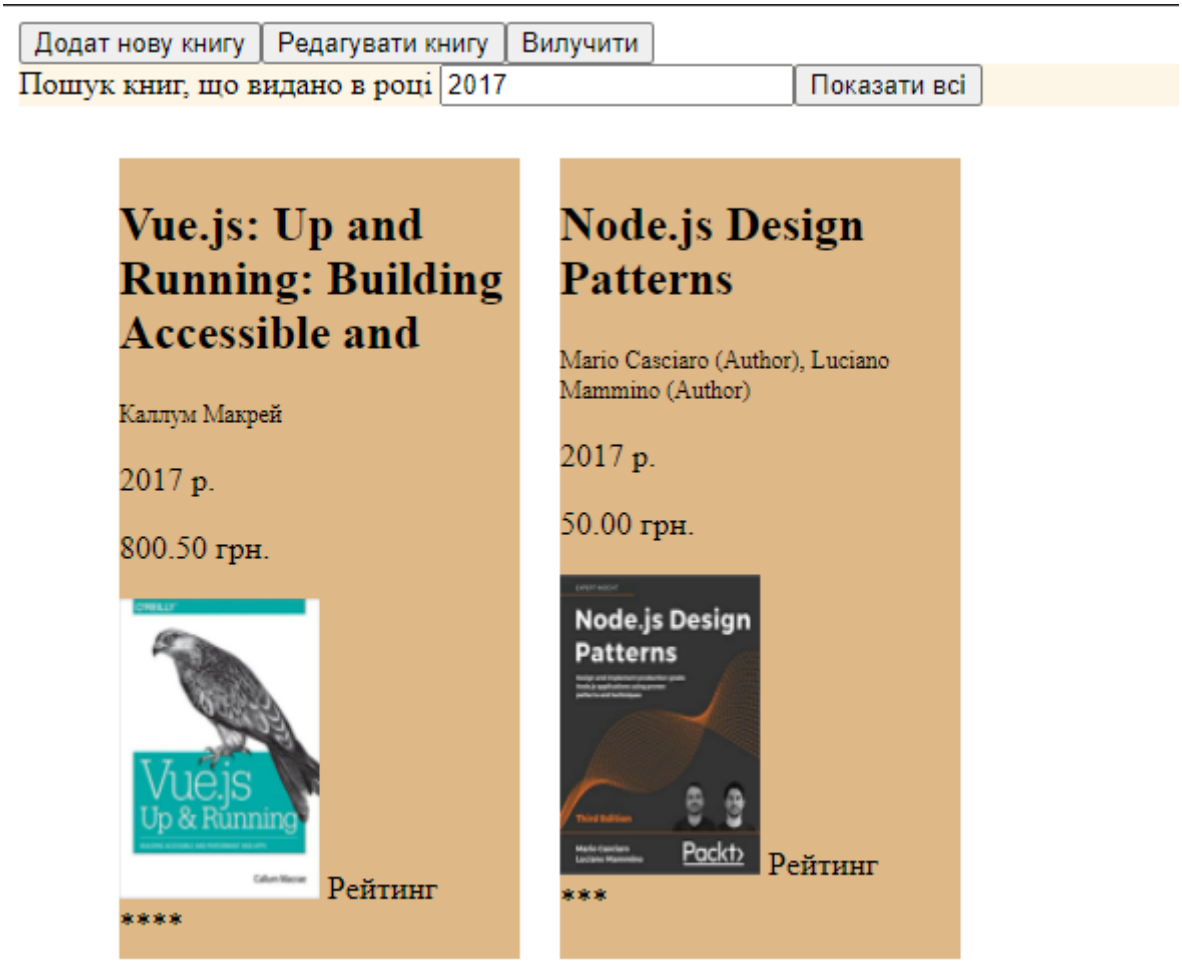


Рис 8. Пошук книг що видано в 2017 році

Текст проєкту, що реалізує завдання лабораторної роботи 2 наведено нижче:

```
<template>
  <div>
    <input type="button" value="Додат нову книгу" @click="showForm" />
    <input type="button" value="Редагувати книгу" @click="showEditForm" />
    <input type="button" value="Вилучити" @click="deleteBook" />
    <form id="search">
      <label>
        Пошук книг, що видано в році
        <input type="number" v-model.number="searchYear" />
      </label>
      <button type="reset" @click="searchYear = ''">Показати всі</button>
    </form>
    <div class="wrap">
      <form
        v-on:submit.prevent="addNewBook"
        class="newForm"
        v-if="showNewBookForm"
      >
        <h3>Форма додавання</h3>
```

```
  Назва <input v-model="newBook.Title" /><br />
  Автори <input v-model="newBook.Author" /> <br />
  Ціна <input type="number" v-model.number="newBook.Price" /> <br />
  Знижка <input type="number" v-model.number="newBook.Disount" /> <br />
  Рік <input type="number" v-model.number="newBook.Year" /> <br />
  Обгортка <input type="file" /> <br />
  <button type="submit">Додати</button>
  <button type="reset">Очистити</button>
</form>

<form
  v-on:submit.prevent="editSelectedBook"
  class="newForm"
  v-if="showEditBookForm"
>
  <h3>Форма редагування</h3>
  Назва <input v-model="editBook.Title" /><br />
  Автори <input v-model="editBook.Author" /> <br />
  Ціна <input type="number" v-model.number="editBook.Price" /> <br />
  Знижка <input type="number" v-model.number="editBook.Disount" /> <br />
  Рік <input type="number" v-model.number="newBook.Year" /> <br />
  Обгортка <input type="image" v-model="editBook.Cover" /> <br />
  <button type="submit">Редагувати</button>
  <button type="reset">Очистити</button>
</form>

<ul v-if="searchedBooks.length > 0">
  <li
    v-for="(book, i) in searchedBooks"
    :key="book.id"
    class="bookvie"
    @click="selectBook(i)"
    :style="i == selected ? 'border:10px solid black' : ''"
  >
    <h2>{{ book.Title }}</h2>
    <p class="authors">{{ book.Author }}</p>
    <p>{{ book.Year }} р.</p>
    <p>{{ book.Price.toFixed(2) }} грн.</p>
    <img alt="Cover" :src="book.Cover" class="cover" />
    <div v-if="book.Disount > 0" class="discount">
      Знижка {{ book.Disount }}%
    </div>
    Рейтинг
    <div class="stars">
      <div v-if="book.Stars > 0" class="star">*</div>
      <div v-if="book.Stars > 1" class="star">*</div>
      <div v-if="book.Stars > 2" class="star">*</div>
      <div v-if="book.Stars > 3" class="star">*</div>
      <div v-if="book.Stars > 4" class="star">*</div>
    </div>
  </li>
```

```
      </ul>
      <p v-else>Дані для показу відсутні</p>
    </div>
  </div>
</template>

<script>
export default {
  name: "App",
  data() {
    return {
      selected: -1,
      newComment: "",
      showComments: true,
      showNewBookForm: false,
      showEditBookForm: false,
      searchYear: "",
      books: [
        {
          Id: 54564,
          Stars: 5,
          Disount: 10,
          Title: "Кобзар",
          Author: "Т.Г. Шевченко",
          Price: 450.0,
          Year: 2020,
          Cover:
            "https://img.yakaboo.ua/media/catalog/product/cache/1/image/398x565/234
c7c011ba026e66d29567e1be1d1f7/1/8/18291_26236_13.jpg",
          Comments: ["Найкраща книга", "Дуже патріотично", "Супер"],
        },
        {
          Id: 1322222,
          Stars: 4,
          Disount: 0,
          Title: "Vue.js: Up and Running: Building Accessible and",
          Author: "Каллум Макрей",
          Price: 800.5,
          Year: 2017,
          Cover:
            "https://images-na.ssl-images-
amazon.com/images/I/51ooMi3IrCL._SX379_BO1,204,203,200_.jpg",
          Comments: ["Найкраща книга", "Дуже патріотично", "Супер"],
        },
        {
          Id: 13136346,
          Stars: 3,
          Disount: 0,
          Title: "Node.js Design Patterns",
          Author: "Mario Casciaro  (Author), Luciano Mammino  (Author)",
          Price: 50.0,
```

```
          Year: 2017,
          Cover:
            "https://images-na.ssl-images-
amazon.com/images/I/515wgVc7S9L._SX404_BO1,204,203,200_.jpg",
          Comments: ["Найкраща книга", "Дуже патріотично", "Супер"],
        },
      ],
      newBook: {
        Disount: 0,
        Title: "",
        Author: "",
        Price: 0,
        Cover: "",
      },
      editBook: null,
    };
  },
  methods: {
    addComment() {
      this.book.Comments.push(this.newComment);
    },
    showHideComments() {
      this.showComments = !this.showComments;
    },
    addNewBook() {
      let newBookCopy = Object.assign({}, this.newBook);
      this.books.push(newBookCopy);
      this.showNewBookForm = false;
    },
    showForm() {
      this.showNewBookForm = true;
    },
    selectBook(index) {
      this.selected = index;
    },
    showEditForm() {
      if (this.selected >= 0) {
        this.editBook = this.books[this.selected];
        this.showEditBookForm = true;
      } else alert("Виберіть книгу");
    },
    editSelectedBook() {
      this.showEditBookForm = false;
    },
    deleteBook() {
      if (this.selected >= 0) this.books.splice(this.selected, 1);
    },
  },
  computed: {
    searchedBooks() {
      if (this.searchYear) {
```

```
        return this.books.filter((book) => book.Year == this.searchYear);
      }
      return this.books;
    },
  },
};
</script>

<style scoped>
ul {
  list-style: none;
  position: relative;
}
.authors {
  font: 12px grey;
}
.bookvie {
  background: burlywood;
  width: 200px;
  position: relative;
  float: left;
  margin: 10px;
  height: 400px;
}
.discount {
  position: absolute;
  right: 0px;
  top: 30px;
  background: chartreuse;
  transform: rotate(45deg);
}
.star {
  float: left;
}
.cover {
  position: relative;
  width: 100px;
  height: 150px;
}
form {
  position: absolute;
  z-index: 10;
  background: oldlace;
  margin: 100px auto;
  width: 50%;
}
.wrap {
  position: relative;
}
#search {
  position: relative;
```

```
  margin: 0;
}
</style>
```

Перелік варіантів

1. Об’єкт“Бухгалтерія” (ПІБ; посада; заробітна плата; кількість дітей; стаж). Запит працюючих, які обіймають посаду X і мають не більше, ніж Y дітей.

2. Об’єкт“ДАІ” (ПІБ власника машини; марка, номер машини; колір. Запит автомобілів марки X, номери яких починаються із вказаного шаблону.

3. Об’єкт“Студентська поліклініка” (ПІБ студента, курс, коефіцієнт здоров’я (1 – здоровий; 2 – стоїть на обліку по якомусь захворюванню; 3 – має якісь відхилення від норми). Запит студентів з відповідним станом здоров’я.

4. Об’єкт“Підприємство” (назва підприємства; кількість співробітників; назва продукції; назва країни, з якою має зв’язки СП). Запит підприємств де кількість співробітників в заданих межах [X;Y].

5. Об’єкт“Турнір” (ПІБ; стать; вік; назва країни, за яку він виступає; оцінки по трьох видах змагань). Запит учасників-жінок з країни X, вік яких не менший за Y.

6. Об’єкт“Проект” (автор проекту, кошторис проекту у грн., оцінки проекту у трьох номінаціях (цілі числа від 1 до 5). Запит проектів, які у трьох номінаціях у сумі набрали не менше, ніж X балів.

7. Об’єкт“Абонент” (номер телефону, домашня адреса, власник; сумарна тривалість розмов). Запит абоненті телефонів із тривалістю розмов більше ніж вказана.

8. Об’єкт“АЗС” (адреса, фірма-власник, запаси бензину марок А-80, А-92, А-95 (у літрах) та ціни одного літру пального кожної марки. Запит пального на всіх АЗС фірми X.

9. Об’єкт“Центр зайнятості” (прізвище і ініціали, стать, рік народження, освіта, спеціальність, дата прийняття на облік). Запит про безробітних пенсійного віку (для чоловіків пенсійний вік складає 60 років, для жінок – 55) та вказаної статті.

10. Об’єкт“Постачання” (номер складу, на якому зберігається товар, назва магазина, в який здійснено поставку, код товару, кількість одиниць товару, ціна одиниці товару. Запит поставок у магазин А зі складу В.

11. Об’єкт“Каса” (номер каси, номер операції, грошова сума, на яку було здійснено операцію, тип операції (0 – видача коштів, 1 – отримання коштів). Запит операцій вказаного типу, які перевищують суму Х.

12. Об’єкт“Аукціон” (назва лота, дата початку торгів, дата завершення торгів, стартова ціна, кінцева ціна). Запит торгів за вказану дату.

13. Об’єкт“Новини” (заголовок новини, короткий зміст новини, текст новини, дата публікації, кількість переглядів). Запит на новини з кількістю переглядів не меше Х.

14. Об’єкт“Форум” (тема форуму, короткий зміст форуму, дата публікації, кількість обговорень по темі форуму). Вивести теми з кількістю обговорень меше заданої теми з кількістю обговорень меше заданої..

15. Об’єкт“Конкурс краси” (прізвище, вік, спеціальність, оцінки по 3-х номінаціях (від 1 до 10 балів). Запит учасниць конкурсу серед усіх студенток, які навчаються на спеціальності Х

16. .Об’єкт“Пасажир” (прізвище, ім'я, по-батькові; пункт призначення; кількість місць багажу; загальна вага даного багажу). Запит пасажирів, які слідують у пункту призначення Х.

17. Об’єкт“Завод” (прізвище, ім'я, по-батькові; номер цеху; посада; стаж роботи; заробітна плата). Запит працівника за заданим прізвищем.

18. Об’єкт“Галерея” (код; прізвище художника; назва картини; ціна; ознака: 1 – картина в експозиції; 2 – картина в запаснику; 3 – картина на “виїзді”). Запит картин вказаного художника що містяться в запаснику.

19. Об’єкт“Автомобіль” (прізвище; номер машини; марка; ціна; домашня адреса). Запит власників машин марки Х.

20. Об’єкт“Школа”, (ПІБ учня; стать; назва класу (рік навчання і літера), оцінка1; оцінка2. Запит учнів класу Х.

21. Об’єкт“Кадри” (код, ПІБ; цех; посада; стать; стаж; заробітна плата). Запит працівників-жінок цеху X.

22. Об’єкт“Вступ” (ПІБ; стать; рік закінчення школи; результати ЗНО). Запит абітурієнтів, які закінчили школу в році X і набрали на ЗНО не менше, ніж Y балів.

23. Об’єкт“Аеропорт” (номер рейсу, місто прибуття; кількість місць; час перельоту; ціна білету). Запит пасажирів, які летять в місто Y рейсами, час перельоту яких не перевищує 5 год.

24. Об’єкт“Міні-АВС” (назва товару; кількість товару в одиницях; ціна за одиницю товару; термін зберігання (в днях). Запит товару, що закінчується в магазині (кількість менше заданої).

25. Об’єкт“Книги” (код; прізвище автора; назва книги; рік видання; кількість сторінок; ціна). Запит книг які мають від 100 до 200 сторінок і були виданні у році X.

4. Лабораторна робота №3. Компонентний підхід.

Завдання. Використовуючи фреймворк Vue розробити SAP для роботи з колекцією об’єктів. Реалізувати компоненти:

- виводу всієї колекції;
- форму для додавання одного об’єкту;
- форму для редагування інформації об’єкту;
- сторінку виводу інформації про вказаний об’єкт;
- компонент пошуку для отримання вибірки з колекції згідно з вказаним запитом;

Об’єкт та запит вибрати згідно з варіантом лабораторної роботи 1.

Компонент - це екземпляр Vue зі своїм ім'ям який можна використовувати неодноразово. Його можна використовувати як для користувача тег всередині кореневого примірника Vue, створеного за допомогою new Vue:

```
Vue.component('button-counter', {
  data: function () {
    return {
      count: 0
    }
  },
  template: '<button v-on:click="count++">
                К-сть кліків {{ count }}
              </button>'
})

<div id="components-demo">
  <button-counter></button-counter>
</div>
```

Компоненти можна перевикористати стільки раз, скільки захочете. Зверніть увагу, при натисканні на кнопки, кожна змінює свій власний, окремий

count. Це тому, що кожен раз коли ви використовуєте компонент буде створений його новий екземпляр.

Зазвичай додаток організовується у вигляді дерева вкладених компонентів (рис 9).

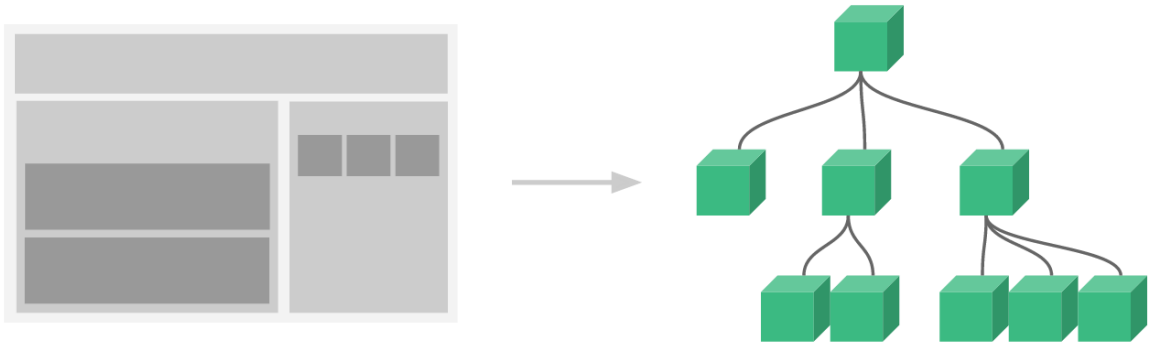


Рис 9. Дерево компонентів.

Вхідні параметри - це атрибути, які ви можете встановити на компоненті. Коли значення передається в атрибут вхідного параметра, воно стає властивістю примірника компонента. Щоб передати заголовок в компонент нашої записи блогу, ми можемо включити його в список вхідних параметрів, які приймає компонент, за допомогою опції props. компонент може приймати стільки вхідних параметрів, скільки захочете, і за замовчуванням будь-яке значення може бути передано в будь-який вхідний параметр. У шаблоні ми можемо отримати доступ до цього значення в екземплярі компонента, як і до будь-якого властивості data. Після оголошення вхідного параметра ви можете передавати дані в нього через призначений для користувача атрибут.

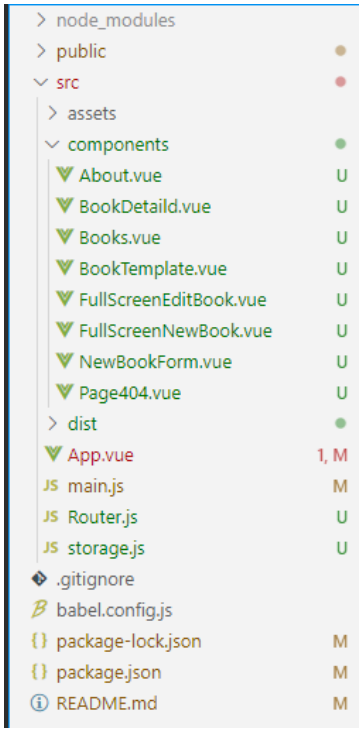
Для деяких можливостей може знадобитися передавати дані назад в батьківський компонент. Батьківський компонент може прослуховувати будь-які події на екземплярі дочірнього компонента за допомогою v-on, як ми це робимо з нативними подіями DOM. Дочірній компонент може згенерувати подія за допомогою вбудованого методу $ emit, передаючи йому ім'я події.

У багатьох проектах, глобальні компоненти визначаються за допомогою Vue.component, з подальшим new Vue ({el: '#container'}) для вказівки елемента-контейнера в тілі кожної сторінки.

Для проектів малого і середнього розміру, в яких JavaScript використовується лише для деяких сторінок, цей підхід може прекрасно працювати. У більш же складних проектах або у випадках, коли весь ваш фронтенд управляється JavaScript, явними стають такі недоліки:

- Глобальне визначення змушує давати унікальне ім'я кожному компоненту.
- Строковим шаблонами не вистачає підсвічування синтаксису. Крім того, доводиться використовувати потворні Слеш для багаторядкового HTML.
- Немає модульної підтримки CSS - в той час як HTML і JavaScript розбиваються на модулі-компоненти, CSS опиняється за бортом.
- Відсутність кроку збірки обмежує нас тільки HTML і ES5 JavaScript, не дозволяючи використовувати препроцесори Pug і Babel

Всі ці недоліки вирішуються однофайловий компонентами з розширенням .vue, використання яких дозволяють такі інструменти як Webpack.



Розділимо проєкт лабораторної роботи 2 на компоненти (рис 10.)

Тоді компонент – плитка

Компонент BookTemplate містить одну плитку книги

Компонент Books виводить всі плитки книг.

Компонент BookDetaild містить сторінку з детальною інформацією про одну книгу.

Компонент FullScreenNewBook містить форму для додавання нової книги.

Компонент FullScreenEditBook містить форму для редагування книги.

Компонент About містить дані про автора.

Рис 10. Структура проєкту.

Розглянемо детальніше BookTemplate та Books.

```
<template>
  <li>
    <h2 @click="showDetailPage">{{ truncTitle(book.Title, 20) }}</h2>
    <p class="authors">{{ book.Author }}</p>
    <p>{{ fromatMoney(book.Price) }}</p>
    <img alt="Cover" :src="book.Cover" class="cover" />
    <div v-if="book.Disount > 0" class="discount">
      Знижка {{ book.Disount }}%
    </div>
    <br />
    Рейтинг <br />
    <p>{{ stars(book.Stars) }}</p>
  </li>
</template>

<script>
export default {
  name: "BookTempate",
  props: {
    book: Object,
  },
  data() {
    return {};
  },
  methods: {
    truncTitle(title, len) {
      if (title.length > len) return title.substring(0, len) + "...";
      else return title;
    },
    fromatMoney(value) {
      return `${value.toFixed(2)} грн.`;
    },
    stars(count) {
      return "★".repeat(count);
    },
    showDetailPage(){
      this.$router.push(`/book/${this.book.Id}`);
    }

  },
  computed:{
    shortTitle (){
      const len = 20;
       if (this.book.Title.length > len) return this.book.Title.substring(0,
len) + "...";
      else return this.book.Title;
    }
  }
};
</script>
```

```
      <style scoped>
      .authors {
        font: 12px grey;
      }
      </style>
<template>
  <section class="control">
     <router-link to="/book/new"> Додати книгу </router-link>
    <input type="button" value="Додат нову книгу" v-on:click="showForm" />
    <input type="button" value="Редагувати книгу" v-on:click="showEditForm" />
    <input type="button" value="Вилучити" v-on:click="deleteBook" />
    <input type="button" value="Сортувати" @click="sortBooksByPrice"/>
    <input type="text" placeholder="Шукати по назві" v-model="searchTitleString">
  </section>
  <div class="wrap">
    <new-book-form
      v-model = "newBook"
      @submit.prevent="addNewBook"
      ref="newBookForm"
    > </new-book-form>
    <new-book-form
      v-model = "editBook"
      ref="editBookForm"
    >
    </new-book-form>

    <ul>
      <book-template
        v-for="b in filtredBooks"
        :key="b.Id"
        class="bookvie"
        v-on:click="selectBook(b.Id)"
        v-bind:book="b"
      >
      </book-template>
    </ul>
  </div>
</template>
```

Список рекомендованих джерел

1. What is Vue.js? URL: https://vuejs.org /v2/guide/
2. Пасічник В. В., Пасічник О. В., Угрин Д. І. Веб-технології Підручник. Львів: Магнолія, 2017. 336 с.
3. Мельник Р.А. Програмування веб-застосувань (фронт-енд та бек-енд). Львів: Львівська політехніка, 2018. 248 с.
4. Русскін В.М. Технології WEB-програмування: курс лекцій. Харків, 2019. 130.
5. Macrae C. Vue.js: Up and Running. Building Accessible and Performant Web Apps. O'Reilly Media. 2018. 173 p.

Навчально-методичне видання

Андрашко Юрій Васильович
Жабур Тетяна Юріївна

Методичні рекомендації до виконання лабораторних робіт із сучасних технології web-програмування. Ч.1 (для студентів спеціальностей 124 «Системний аналіз» та 113 «Прикладна математика»)

В авторській редакції

Рекомендовано до друку Науково-методичною комісією факультету математики та цифрових технологій (протокол №7 від 24 травня 2021 року)