DOI: https://doi.org/10.36910/6775-2524-0560-2021-42-15
УДК 004.4'22
Трофімук Андрій Олександрович, студент
Булатецька Леся Віталіївна, канд. фіз.-мат. наук, доцент
https://orcid.org/0000-0002-7202-826X
Павленко Юлія Степанівна, старший викладач
https://orcid.org/0000-0002-4065-045X
Гришанович Тетяна Олександрівна, канд. фіз.-мат. наук, старший викладач
https://orcid.org/0000-0002-3595-6964
Волинський національний університет імені Лесі Українки, м. Луцьк, Україна

РОЗРОБКА ІНТЕРАКТИВНОЇ КАРТИ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЇ SPA

А. О. Трофімук, Л. В. Булатецька, Ю. С. Павленко Т. О. Гришанович. Розробка інтерактивної карти з використанням технології SPA. В роботі розглянуто основні питання, що стосуються процесу розробки та проектування інформаційної системи із використанням технології Single Page Application. Дана технологія надає можливість зміни контенту без перезавантаження сторінки, що забезпечує користувачам миттєвий зворотній зв'язок. Розроблено веб-сайт з візуально відображеною картою місцевості, на якій користувач має можливість створювати маркери, які містять детальну інформацію про дане місцезнаходження. При розробці використані HTML, CSS, JavaScript.

Ключові слова: Single Page Application, веб-сайт, база даних, фреймворк, кросбраузерність, контролер, географічні карти.

А. А. Трофимук, Л. В. Булатецкая, Ю. С. Павленко Т. А. Гришанович. Разработка интерактивной карты с использованием технологии SPA. В работе рассмотрены основные вопросы, касающиеся процесса разработки и проектирования информационной системы с использованием технологии Single Page Application. Данная технология предоставляет возможность изменения контента без перезагрузки страницы, обеспечивает пользователям мгновенную обратную связь. Разработан сайт с визуально отраженной картой местности, на которой пользователь имеет возможность создавать маркеры. Эти маркеры содержат подробную информацию о локации. При разработке использованы HTML, CSS, JavaScript.

Ключевые слова: Single Page Application, веб-сайт, база данных, фреймворк, кроссбраузерность, контроллер, географические карты.

A.O. Trofimuk, L. V. Bulatetska, Y. S. Pavlenko, T. O. Hryshanovych Development of the interactive map using SPA technology. The paper considers the main issues related to the process of developing and designing the information system using Single Page Application. This technology allows users to change the content of a web-page without reloading. It provides users with instant feedback. The website that displays a map of an area, where users are able to create markers, was developed. These markers include detailed information about the location. HTML, CSS, JavaScript were used in website development.

Key words:: Single Page Application, website, database, framework, cross browser, controller, geography maps.

Постановка проблеми та аналіз досліджень. Враховуючи те, що користувачі мережі Інтернет стають більш вимогливими до інтерактивності веб-сторінок, то одним із актуальних завдань для розробки інформаційних систем з веб-інтерфейсом є покращення адаптивності сайту, використання ресурсів для перегляду й переходу до інформації без використання перезавантажень сторінок. Тобто будь-яка реалізована на сторінці компонента взаємодії з користувачем не обов'язково призводить до перезавантаження всієї сторінки, а всі візуальні елементи будуються просто в браузері. Така технологія, що надає користувачам миттєвий зворотній зв'язок у порівнянні зі звичайними сайтами, називається Single Page Application (SPA). SPA – це веб-застосунок, розміщений на одній сторінці, яка для забезпечення роботи завантажує всі JavaScript-файли, а також файли CSS разом із завантаженням самої сторінки [1]. Оскільки технологія SPA надає можливість зміни контенту без перезавантаження сторінки, то цю технологію актуально використовувати на сайтах із великою кількістю зовнішньої інформації, наприклад, при розробці інформаційної системи з веб-інтерфейсом для зберігання даних про об'єкти, які містяться на географічних картах. Саме в таких системах використання технології SPA дає можливість користувачу працювати з даними без використання перемальовування карти.

Метою роботи є дослідження процесу розробки інтерактивної інформаційної карти у вигляді веб-додатку з використанням технології SPA, призначеного для збереження, накопичення та відображення інформації про визначні місця регіону та для комунікації користувача із картою. Для цього потрібно підібрати карту, яка є безкоштовною і оптимальною у використанні, обрати інструментарій розробки, реалізувати зручний інтерфейс. Таку розробку можна буде використовувати як, наприклад, туристичну мапу місцевості з постійним доповнення бази даних визначних локацій регіону.

Виклад основного матеріалу й обґрунтування отриманих результатів дослідження. Основою реалізації веб-сайту є HTML5, яка забезпечує створення його структури у вигляді блоків для взаємодії. Для зовнішнього оформлення сайту обрано CSS3, що дозволяє визначати стилі для елементів HTML. Динамічність та адаптивність сайту забезпечено з допомогою фреймворку Bootstrap4, за допомогою якого можна контролювати стилізацію елементів під різні екрани пристроїв. Даний фреймворк бере на себе й забезпечення кросбраузерності веб-сайту [2]. Для реалізації різних дій та функціоналу розробки вибрано мову JavaScript, а для розширення та комфортності використано бібліотеку jQuery.

Розробку веб-інтерфейсу виконано за допомогою фреймворку Vue, який повністю підходить для створення складних односторінкових додатків (SPA). Оскільки Vue вирішує завдання рівня уявлення (view), то він легко інтегрується з іншими проектами та бібліотеками [3, 4].

В якості сервера вибрано OpenServer із базою даних MySQL. Для реалізації збереження інформації у базі даних інформаційної системи було використано мову програмування загального призначення з відкритим вихідним кодом PHP (Hypertext Preprocessor). PHP спеціально сконструйована для веб-розробок та її код може впроваджуватися безпосередньо в HTML.

Об'єднання всіх технологій здійснюється з допомогою фреймворку Laravel, що надає зручну можливість використання запитів до бази даних та шаблони для розробки авторизації користувачів. В даний фрейморк включений Webpack, який використовується для підключення нових бібліотек та контролю за версіями [5].

В якості IDE (Інтегроване середовище розробки) було вибрано VSCode. Крім IDE, необхідна наявність менеджера залежностей Composer, через який можна встановлювати та оновлювати Laravel.

Після створення веб-проекту і його тестування необхідно підготувати проєкт до розміщення на сервері. Laravel надає інструмент Laravel Mix, який використовує Webpack і вміє працювати з CSS, JavaScript, Less, Saas, Stylus, PostCSS. Цей інструмент, використовуючи спеціальний складальник модулів Webpack, збирає разом всі JS і CSS-файли та вміє створювати версії цих файлів. Тобто, кожна збірка нашого проекту дозволяє мати різні назви JS і CSS-файлів в HTML-коді, що вирішує проблему з кешуванням при зміні вмісту файлу [5].

В Laravel є посередник для перевірки аутентифікації користувача. Посередники надають зручний механізм для фільтрації HTTP-запитів програми. Якщо користувач не аутентифікований, посередник перенаправить його на сторінку входу в систему. Якщо ж користувач аутентифікований, посередник дозволить запитом пройти далі в додаток. Але посередники потрібні не тільки для авторизації. CORS-посередник може стати в нагоді для додавання особливих заголовків для всіх відповідей у додатку, а посередник логів може зареєструвати всі вхідні запити. У Laravel є кілька стандартних посередників, включаючи посередники для аутентифікації і CSRF-захисту. Всі вони розташовані в директорії app/Http/Middleware [5].

Для організації аутентифікації в Laravel використовується конфігураційний файл, який розташований в config/auth.php, де містяться опції для тонкого налагодження поведінки служб аутентифікації. Laravel забезпечує швидкий спосіб створення заготовок всіх необхідних для аутентифікації Pауса і шаблонів за допомогою однієї команди [5]:

php artisan make:auth.

Підключення фреймворку Laravel відбувається за допомогою команди в консолі OpenServer: Composer [6]. Для розробки інформаційної системи потрібно підключити додаткові пакети, які надають різні функції, зокрема:

- npm – компіляція JS та CSS коду;
- laravel/ui – отримання базового забезпечення, а саме використання Bootstrap, React чи Vue;
- vue vue-router – пакети бібліотеки маршрутизації vue-router;
- vuex – пакети бібліотеки сховища Vuex;
- fontawesome free – пакет іконок;
- sass – компілятор препроцесора;
- admin-lte – адміністративна панель;
- vueperslider – слайдер;
- nprogress – стрічка завантаження.

Для більш комфортної роботи редактор коду Virtual Studio Code надає свої пакети. Для підключення до бази даних нам потрібно заповнити поля у файлі .env:

- DB_DATABASE – назва бази створеної у OpenServer;
- DB_USERNAME та DB_PASSWORD – доступ до неї.

Після підключення бази даних можна здійснювати міграції до неї та отримувати з неї інформацію.

При створенні міграцій було створено наступні таблиці: користувачів, локацій місцезнаходження, картинок та коментарів. Ці таблиці обов'язково повинні бути пов'язані для забезпечення каскадного оновлення чи видалення відповідних даних з декількох таблиць. Наприклад, при видаленні із таблиці users користувача, повинні бути видалені всі записи в таблиці locations за ключем user_id (за яким ці таблиці пов'язані).

Для міграції даних використовувались команди php artisan: php artisan migrate.

Заповнення контентом здійснюється через використання функції seeds фреймворку Laravel [5]. У Laravel наповнення БД початковими даними виконується за допомогою спеціальних класів, які зберігаються в директорії database/seeds. За замовчуванням визначений клас DatabaseSeeder. З цього класу можна викликати метод call для підключення інших класів з даними, що дозволить контролювати порядок їх виконання. Стандартні дані для запитів описуються у директорії seeds у файлі DatabaseSeeds. За допомогою командної стрічки та коду php artisan можна заповнити таблиці [5]:

php artisan db:seed

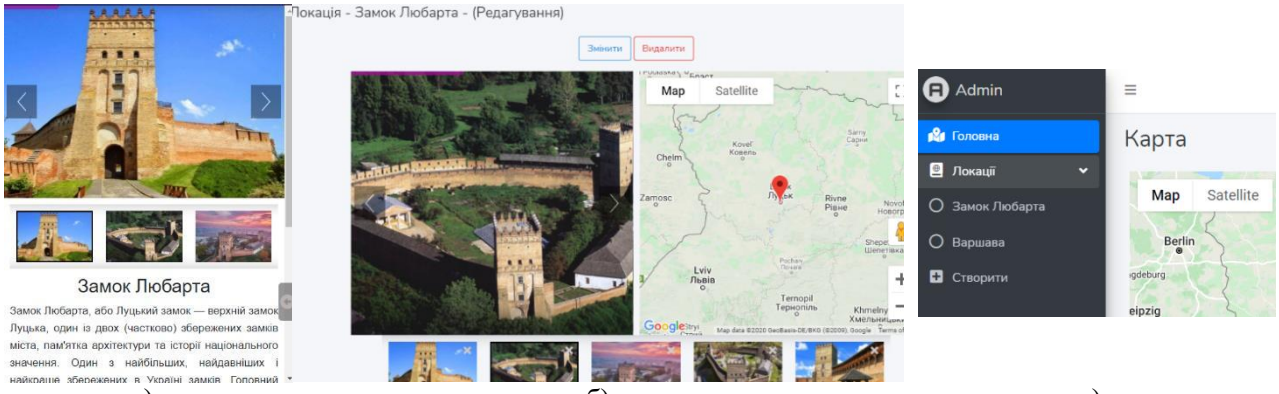
php artisan db:seed -- class=UsersTableSeeder.

При створенні інформаційної системи з веб-інтерфейсом, в якій реалізована взаємодія користувача з картою ми обрали карту від компанії Google, що надана у безкоштовне використання [7]. Була розроблена сторінка користувача та сторінка адміністратора, в якій не використовується footer, тому що Google-карта буде розширена на весь екран і сайт не буде прокручуватись. У шапці сайту розміщуються логотип, меню авторизації та інтерфейс буде адаптований під мобільні пристрої.

Однією із основних частин інтерфейсу, що може бачити користувач, є слайдер. Наявність слайдера робить сучасні сайти більш живими і анімованими. В роботі слайдер реалізований з допомогою VueperSlider. За допомогою тегу <vueper-slides /> в ньому розміщені картинки, які показують, як виглядає місце, із яким користувач взаємодіє через карту, та виводиться коротка інформація про нього. На рис. 1(а) подано вигляд зовнішнього оформлення слайдера при задіюванні та компіляції коду.

Для сторінки адміністратора використано плагін AdminLTE 3, який надає шаблон адміністративної панелі. Для підключення різних пакетів та взаємодії із ними, вони повинні бути завантажені за допомогою команди npm і підключені до файлу app.js за допомогою збирача коду Webpack та бібліотек стилів [8].

Для проектування сторінки користувачів та гостей використовувались шаблонізатори фреймворку Vue: LocationCreateComponent.vue та LocationUpdateComponent.vue для створення та редагування локацій; LocationListComponent.vue для реалізації списку всіх локацій та роботи з ними. Основу сайту займає компонент MapComponent.vue – мапа сайту, через яку можна взаємодіяти із різними локаціями. Інформаційне поле для виведення інформації про локацію та взаємодію із нею реалізоване з допомогою шаблону LocationComponent.vue [2, 3] (рис. 1).



а) б) в)

Рис. 1. Знімки екрану інформаційного поля (а), адміністративної сторінки оновлення локації (б), адміністративного sidebar (в)

Авторизовані користувачі можуть також залишати коментарі у відведеній для цього області, гості мають право лише перегляду. Було створено функцію onMarkerClick(), яка отримує параметри:

- e – інформаційний об'єкт, має в собі повну інформацію про події, що відбулись;

- marker – масив всіх маркерів із бази даних;
- id – ідентифікатор локації, використовується для отримання інформації із бази даних.

Для проектування сторінки адміністратора використовувались такі самі шаблонізатори фреймворку Vue, що і для проектування сторінки користувача, але вони знаходяться у різних папках. В адміністративній частині використовуються компоненти MapComponent.vue, де реалізація для взаємодії з картою дещо видозмінена.

Фрейморк Laravel надає можливість самому налагоджувати маршрутизацію веб-сторінок у файлі routes/web.php [5]. Стандартну сторінку напишемо такою командою:

```
Route::get('/', function () { return redirect()->route('userMap');});
Route::get('/InfoMap.com/map', function() { return  view('layouts.user.userPage');})->name('userMap');
```

При стандартній URL-навігації "/", ми переходимо до маршруту із ім'ям userMap та відкриваємо сторінку за маршрутом layouts/user/userPage.php. Це стандартний маршрут для гостей. Для користувачів та адміністратора маршрути будуть розширені та зв'язані із контролерами та посередниками. Ми використовуємо груповий маршрут для того, щоб задіяти наступні функції:

- задати початкове ім'я для маршруту;
- використовувати посередників для перевірки авторизації та перевірки на наявність адміністративних повноважень.

Всі функції, які задіяні для роботи із базою даних, були виконані в контролері UserController. Ми використовували ресурсний контролер, який надає вже готові стандартні функції для взаємодії із вхідними даними. Адміністративний груповий маршрут схожий до користувацького, за винятком контролера AdminController та URL, які відрізняються шляхом та посередником.

Інші контролери також задіяні. Вони потрібні для адміністрування та авторизації, зокрема, для Ajax-запитів і інше. Фреймворком автоматично створені два контролера: для авторизації та стандартний контролер. Також було використано анонімний контролер, який виконує функції переходу від невизначеного шляху. Тобто при будь якому URL, невідомому програмі, буде відбуватися перехід на сторінку гостей чи користувачів, що вже визначає посередник redirectUrl.

Для взаємодії із базою даних використовувались два контролери UserController та AdminController.

Контролер для користувача:

```
class UserController extends Controller {
    public function index() {
    return view('layouts.user.userPage');}}
```

Контролер адміністратора:

```
class AdminController extends Controller {
    public function index() {
    return redirect()->route('admin.map');}}
```

Контролери користувача та адміністратора подібні але відрізняються виконанням функції. На прикладі публічної функції index(), контролер звертається до маршруту із іменем admin.map:

```
route::get('/InfoMap.com/admin/map','AdminController@indexMap')->name('admin.map);
```

Посередники (middleware) призначені для фільтрації маршрутів. В нашому випадку посередники використовувались для перевірки авторизації користувачів та перевірки наявності в них прав адміністратора. Для перевірки на права адміністратора використовується посередник RedirectUrl. Також було створено ще два посередники на два маршрути, користувацького та адміністративного, задля перевірки користувача при перезавантаженні сторінки [5].

Посередник UserCheck:

```
public function handle($request, Closure $next) {
if(Auth::user() and Auth::user()->is_admin == 1) {
    return redirect()->route('admin.index');}
else if(Auth::user() and Auth::user()->is_admin == 0) {
    return $next($request);}
    return redirect()->guest('/')}
```

Посередник AdminCheck:

```
public function handle($request, Closure $next) {
if(Auth::user() and Auth::user()->is_admin == 0) {
    return redirect()->route('user.index');}
else if(Auth::user() and Auth::user()->is_admin == 1) {
    return $next($request);}
```

return redirect()->guest('/')}

```
/**
 * Route get location information on id
 */
Route::get('/location/{id}',function($id) {
    return [
        Location::where('id',$id)->select('id','user_id','title','text','marker')->first(),
        Location::where('id',$id)->first()->reviews()->select('id','review','surname','upda
        Location::where('id',$id)->first()->images()->select('id','image_url')->get()
    ];
});
/**
```

Рис. 2. Маршрут отримання інформації про локацію

На рис. 2 описано маршрут отримання інформації про локацію. Ми бачимо маршрут для Ajax-запиту, в якому реалізується отримання інформації. За допомогою моделі Location відбувається звернення до даних інших таблиць, а саме, використовуючи, як у прикладі, функції reviews() та images().

Для взаємодії із Google-картою потрібно підключити плагін vue2-google.maps та вставити API-ключ:

import * VueGoogleMaps from 'vue2-google-maps'
Vue.use(VueGoogleMaps, {
load{key: 'API_KEY', libraries: 'places'}})

Оскільки картою користуються як користувачі, так і адміністратор, то вона задіяна у шаблонах Vue. Також за допомогою функції Vue відбувається робота з масивом для визначення декількох маркерів. Ajax-запити потрібні для того, щоб отримувати інформацію із бази даних не використовуючи перезавантаження сторінки, тобто не використовувати напряму PHP-функції.

В фреймворку Laravel функція, яка використовується для Ajax-запитів, має вигляд:

$.ajax({
Type: "method",
url: "url",
data: "data",
success: function (response) {…}})

Для звернень до сервера у вигляді HTTP-запитів у Vue використовують axios.put або axios.get:

axios.put(payload.url + payload.id, {…})
.then(data => {…})
.catch(error => {…})

Вони є асинхронними функціями і не будуть стояти в черзі на виконання коду [3].

Для налаштування маршрутизації по компонентам Vue використовували Vue-router. Для початку потрібно його підключити до файлу app.js. Ми робимо це окремим файлом:

import router from './routes.js';

Також там же прописуються маршрути компонентів:

const routes = {
{path:'InfoMap.com/map', name: 'map'},
{path:'InfoMap.com/map/location/:id', name: 'location', component: location, props: true},
{path:'InfoMap.com/user/map/location/:id', name: 'userLocation' component: location, props: true}}

Для задання місцезнаходження відображення компонент задіюємо тег <router-view />. Створення посилань на компоненти реалізовується тегом <router-link /> із характеристиками про URL. Для використання переходу між компонентами у функціях Vue використовується команда:

this.$router.push({}): this.$router.push({name: "adminUpdateLocation",params: {id:id}})

Тобто ми звернулись до команди Vue-router та вказали здійснити перехід до компоненти adminUpdateLocation та передати параметр ідентифікатора [3, 4].

Для передавання інформації між компонентами раніше застосовувалась функція $emit: this.$emit('deleteloc',this.location.id). Дана функція дозволяє спілкуватися із батьківським класом та передавати зміни. Для передавання та збереження даних у Vue використовують сховище Vuex. Сховище реалізовано окремим файлом задля простішого використання та реалізації в інших проєктах.

Висновки та перспективи подальшого дослідження. В роботі було спроектовано та розроблено інформаційну систему з прив'язкою до карти з використанням технологій SPA. Створена нами інформаційна система направлена на полегшення пошуку інформації про певні місця на Google-

картах. При реалізації були використані HTML, CSS, Bootstrap, JavaScript, бібліотеки jQuery і Vue, Laravel та технологія SPA, яка надала можливість переходу між сторінками сайту без використання перезавантаження сторінок. Дану інформаційну систему можна буде використовувати як для туристичних цілей, з постійним доповненням бази даних визначних місць регіону, так і для підприємницьких та рекламних цілей.

Список бібліографічного опису.

1. SPA-архитектура для CRM-систем: часть 1 [Електронний ресурс] – Режим доступу : https://habr.com/ru/company/qbs/blog/243545/
2. Bootstrap 4 [електронний ресурс] — Режим доступу : https://bootstrap-4.ru/docs/4.0/getting-started/introduction/
3. Intro to Vue2 [Електронний ресурс] — Режим доступу : https://www.vuemastery.com/courses/intro-to-vue-js/vue-instance/
4. Введение. Что такое Vue.js? [Електронний ресурс] — Режим доступу : https://ru.vuejs.org/v2/guide/index.html.
5. Laravel основи [Електронний ресурс] — Режим доступу : https://laravel.su/docs/5.4.
6. Установка Laravel 5.7-7.x через OpenServer [Електронний ресурс] — Режим доступу : https://offgalaxy.com/install-laravel-5-7/
7. Google Maps Platform [Електронний ресурс] — Режим доступу : https://developers.google.com/maps/documentation
8. AdminLTE 3 Docs [Електронний ресурс] — Режим доступу : https://adminlte.io/docs/3.0/index.html

References

1. SPA architecture for CRM systems: part 1 [online] – Available from : https://habr.com/ru/company/qbs/blog/243545/
2. Bootstrap 4 [online] — Available from : https://bootstrap-4.ru/docs/4.0/getting-started/introduction/
3. Intro to Vue2 [online] — Available from : https://www.vuemastery.com/courses/intro-to-vue-js/vue-instance/
4. Introduction. What is Vue.js? [online] — Available from : https://vuejs.org/v2/guide/index.html
5. Laravel. documentation [Електронний ресурс] — Available from : https://laravel.su/docs/5.4
6. Installing Laravel 5.7-7.x via OpenServer [online] — Available from : https://offgalaxy.com/install-laravel-5-7/
7. Google Maps Platform [online] — Available from : https://developers.google.com/maps/documentation
8. AdminLTE 3 Docs [online] — Available from : https://adminlte.io/docs/3.0/index.html