Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2023 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти

на тему

“ Програмне забезпечення системи керування відносинами з клієнтами**”**

Виконав здобувач вищої освіти
IV курсу, групи КМ-19
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____Бондаревський С.О.
«____» _____ 2023 р.

Керівник проекту
кандидат технічних наук, доцент
Пархоменко Ю.М.
«____» _____ 2023 р.
Рецензент _____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*
Кафедра *Кібербезпеки та програмного забезпечення*
Рівень вищої освіти *бакалавр*
Галузь знань. *12 "Інформаційні технології"*
Спеціальність *123 "Комп'ютерна інженерія"*
Освітньо-професійна (освітньо-наукова) програма *"Комп'ютерна інженерія"*

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
"____" _____2023 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Бондаревському Сергію Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи керування відносинами з клієнтами*
2. Керівник роботи *Пархоменко Юрій Михайлович, канд. техн. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від "5" січня 2023року № 10-02

3. Строк подання роботи до захисту *22.05.2023 р.*
4. Мета та завдання випускної кваліфікаційної роботи. *Метою роботи є розробка програмного забезпечення системи керування відносинами з клієнтами*
5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
   1. Призначення та область використання.
   2.Перегляд аналогічних існуючих систем.
   3. Опис і обґрунтування проектних рішень.
   4. Етапи програмування системи.
   5. Впровадження системи в промислову експлуатацію
   6. Висновки.
6. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

7. Дата видачі завдання « » 20 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2023 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2023 р.	
3.	Розробка моделі компонента	20.03.2023 р.	
4.	Розробка структур даних	25.03.2023 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2023 р.	
6.	Програмування алгоритмів	10.04.2023 р.	
7.	Оформлення ПЗ	17.05.2023 р.	
8.	Попередній захист роботи	22.05.2023 р.	

Дата видачі завдання
«__»_____2023р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
«__»_____2023р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Бондаревський С.О. Програмне забезпечення системи керування відносинами з клієнтами. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

Метою даної бакалаврської роботи є розробка програмного забезпечення, який буде відповідати вимогам з технічного завдання.

Система реалізована на базі фреймворку Laravel та за допомогою The Maps JavaScript API.

Бакалаврська робота містить такі розділи: призначення і область застосування системи; огляд існуючих програмних систем та рішень; обґрунтування вибору принципу розробки і методики побудови системи; опис і обґрунтування проектних рішень з проектування веб-додатку; розрахунки і експериментальні матеріали, що підтверджують вірність проектних та програмних рішень; впровадження системи в експлуатацію.

Спочатку було проаналізовано інші схожі за основною темою системи, охарактеризовано їх плюси та недоліки. В процесі роботи було описано всі кроки розробки з поясненнями до них – розробка серверної частини, проектування бази даних, розробка веб-сторінок, робота з картою з JavaScript. Зроблено висновки в результаті реалізованої системи.

Робота містить структурну схему системи, функціональну схему системи, 3 діаграми процесів та блок-схему алгоритмів роботи додатку.

Ключові слова: комп'ютерна інженерія, веб-додаток, програмно-апаратні рішення, фреймворк, серверна частина.

ABSTRACT

Bondarevsky S.O. Customer relationship management system software. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

The purpose of this bachelor thesis is to develop software that will meet the requirements of the technical task.

The system is implemented on the basis of the Laravel framework and using The Maps JavaScript API.

The bachelor thesis contains the following sections: the purpose and scope of the system; review of existing software systems and solutions; justification of the choice of the development principle and method of building the system; description and substantiation of project solutions for web application design; calculations and experimental materials confirming the validity of design and program solutions; putting the system into operation.

First, other systems similar to the main theme were analyzed, their advantages and disadvantages were characterized. In the course of work, all development steps were described with explanations for them - development of the server part, database design, development of web pages, work with a JavaScript map. Conclusions are made as a result of the implemented system.

The work contains a structural diagram of the system, a functional diagram of the system, 3 process diagrams and a block diagram of the algorithms of the application.

Keywords: computer engineering, web application, hardware and software solutions, framework, server part.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

СЧ	–	Серверна частина
TMJSAPI	–	The Maps JavaScript API
PHP	–	Personal Home Page
API	–	Application Programming Interface
CSS	–	Cascading Style Sheets
HTML	–	HyperText Markup Language
XML	–	eXtensible Markup Language

ВСТУП

Актуальність роботи. Використання веб-технологій швидко зростає в сучасному світі, охоплюючи всі сфери людського життя. Тому підприємства у своїй діяльності активно застосовують не декстопні, а веб-додатки. Їхнє використання полегшується тим, що для функціонування потрібний тільки браузер, тому вони працюють і з телефонів.

Головна задача – відмічати об’єкти зі складськими приміщеннями, називати їх, зберігати дані про них у таблицях Excel, які можна прикріпити до будь-якої місцевості на карті.

Для досягнення мети буде розв’язано такі задачі:

- розробка бази даних;

- розробка структури інтерфейсу додатка;

- дослідження та опрацювання документації Google Maps для реалізації задачі (виведення та нанесення об’єктів).

- розробка функціоналу:

а) веб-сайт – відображення карти, форми примітивного типу (без дизайну, а тільки для огляду функціоналу);

б) СЧ – авторизація, збереження об’єктів і файлів, редагування, віддача даних, API;

- реалізація всіх задач.

Об’єктом дослідження є розробка системи керування відносинами з клієнтами на базі web-додатку.

Предметом дослідження роботи є методи і засоби створення веб-продукту для підприємницької діяльності.

Актуальність теми роботи полягає у тому, що створений додаток має практичне застосування: допомагає систематизувати та маркувати об’єкти складських приміщень. Таким чином, дана бакалаврська дипломна

робота націлена на розвиток інфраструктури підприємницької діяльності, може буде застосована у багатьох сферах: зберігання та упорядкування товарів, створення рієлторської бази даних, відміток для кур'єрської служби тощо.

1. ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Розробка веб-додатку – це процес створення веб-сторінок або сайтів. Веб-сторінки створюються з використанням HTML, CSS і JavaScript і т.д. Сторінки можуть статичними (текст або інформація не змінюються протягом перегляду) або динамічними (дані можуть оновлюватися після відправки форми, довантажувати інформацію по типу пагінації; це здійснюється за допомогою request з JavaScript).

Перед створенням конкретного веб-продукту відбувається комплексний аналіз, що визначає критерії, яким він має відповідати. Він включає такі етапи:

- визначення цілей та задач;
- розробка структури;
- HTML-верстка;
- програмування та контроль якості;
- тестування окремих частин і в цілому;
- запуск та оптимізація.

Продукт надає можливість занесення об’єктів до бази даних (для зберігання та подальшого відображення), виділяючи їхнє місцерозташування на карті з’єднаними точками. Для кожного об’єкта потрібно ввести назву і прикріпити файл в інтерфейсі. Файл потім можна завантажити чи замінити.

Усі об’єкти будуть відображатися на одній карті. У середині виділеної місцевості буде маркер, при натисканні на який показуватимуться назва і можливість вибору дій над файлами.

1.2 Область застосування

Додаток може використовуватися як для потенційних покупців будь-яких товарів, так і для внутрішніх бізнес-потреб.

Після того, як об’єкти були занесені в базу даних й відображаються, клієнти можуть переглядати та читати інформацію. На сайті можна побачити карту з усіма об’єктами, є пошук за назвою міста для швидкого знаходження населених пунктів. Наприклад, коли людина шукає нерухомість із метою купівлі, а на карті виділений якийсь житловий комплекс, то, натиснувши на маркер, можна завантажити файл із інформацією про квартири, що є у продажу. Або ж, якщо об’єкти це складські приміщення чи магазини, то можна завантажити інформацію про них.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським рівнем вищої освіти)

Так як система, що буде розроблятися, може використовуватися для підприємств різних сфер – рієлторських контор, виробництв із складськими приміщеннями, тощо – тому й вибрані існуючі системи для огляду матимуть різну специфіку, але всі будуть з головним елементом – картою з виділеними об’єктами на ній.

Розглянемо ЛУН - найбільший в Україні сайт із пошуку квартир. Проект пропонує співпрацю агенціям нерухомості, порталам та дошкам оголошень. Щомісяця на цей сайт приходить до 2-х млн користувачів, які хочуть купити або орендувати нерухомість. Для розміщення оголошення потрібно експортувати його в пошукову систему порталу. Аби оголошення з сайту потрапили у пошукові результати, потрібно автоматично експортувати вашу базу в спеціальному XML-форматі. Тоді ці оголошення з'являться на Flatfy від ЛУН з посиланням на ваш сайт. Також вас додадуть до списку сайтів-партнерів. Для підключення до пошуковика Flatfy від ЛУН вам необхідно автоматично генерувати фід у спеціальному XML-форматі. Фід повинен містити всі актуальні оголошення з вашого сайту та оновлюватися мінімум 2 рази на день. Рекомендована частота оновлення — щогодини. Якщо ви не знаєте, як налаштувати фіди, ознайомтеся з інструкцією, або передайте її системному адміністратору/розробнику

вашого сайту. Але це не на безкоштовній основі й для цього потрібно, щоб ваш портал відповідав певним умовам. Далі наведені умови:

- сайту має бути більше ніж 6 місяців, він повинен знаходитися на власному домені та працювати стабільно;

- для кожного оголошення повинна існувати окрема сторінка на вашому сайті. Саме на неї ми приведемо користувача з наших результатів пошуку;

- співпраця з сайтами з нерухомості проходить на платній основі, за схемою поклікового продажу трафіку на сайт (CPC). Мінімальний обсяг замовлення - 12000 грн на місяць;

- підключення порталів та дошок оголошень розглядається індивідуально, щомісячна відвідуваність порталу повинна перевищувати 100 000 чоловік.

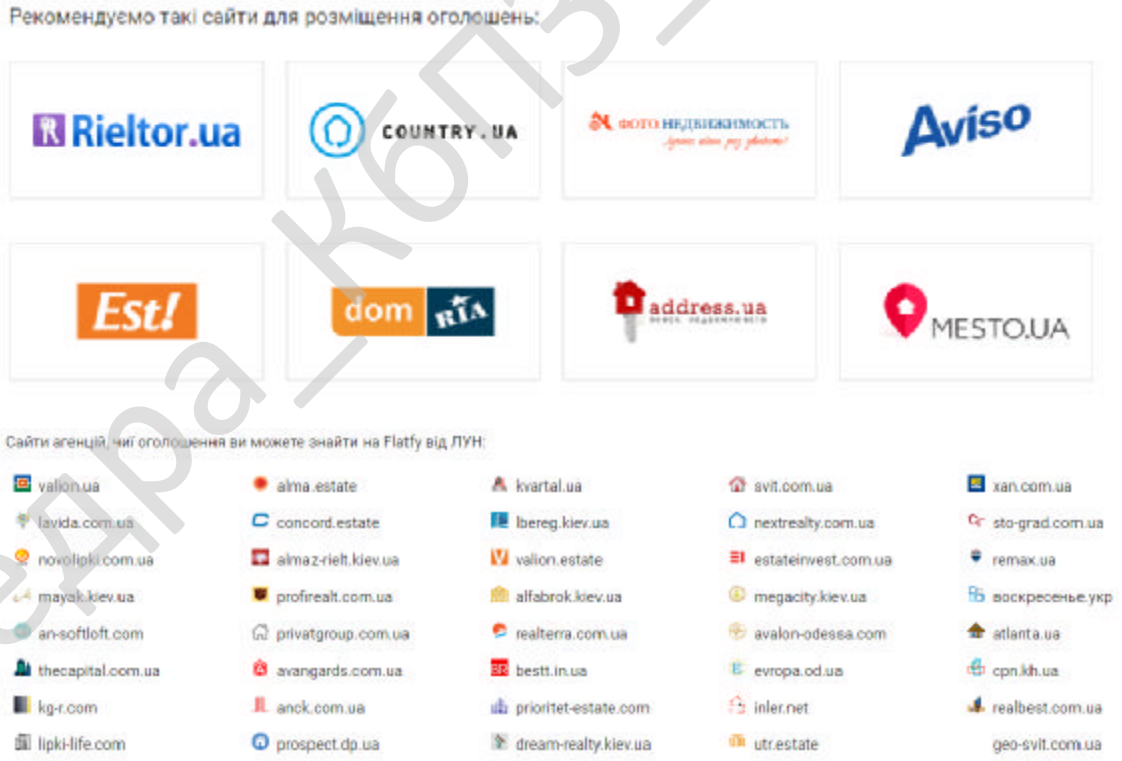


Рисунок 2.1 – Список сайтів-партнерів

Для невеликих агенцій ефективнішим буде розміщення оголошень на порталах з нерухомості, з якими портал співпрацює. Flatfy від ЛУН — пошукова система, як Google, що знаходить оголошення на всіх сайтах та порталах із нерухомості в Україні. Аби оголошення з'явилося на Flatfy від ЛУН, його потрібно додати на один чи кілька таких сайтів або порталів.

Перейдемо до основного компонента на порталі - карти. Для цього в навігації потрібно натиснути «На карті», потім ви потрапите на сторінку з картою.

Рисунок 2.2 – Карта з поясненням для позначень

Точками відмічені будинки, за кольором яких можна дізнатися потрібний статус. Коли натиснути на відмітку, можна побачити загальну інформацію про житловий комплекс та посилання на сайт, на якому було виставлено оголошення.

Наступний приклад – це мобільні карти BatchGeo для IPhone і IPad. Якщо є IPhone, IPad чи IPod Touch, то цей додаток завжди працює, незалежно від наявності підключення до інтернету.

Функції додатку:

1. Доступ до всіх карт BatchGeo.
2. Рідний додаток спеціально для IOS.

3. Більш швидка та плавна робота, ніж у мобільної веб-версії.
4. Додавання нових функцій із оновленнями програми.

Якщо пристрій оснащено модулем GPS, BatchGeo може перенести місцезнаходження на будь-яку з карт. Але навіть за відсутністю GPS-модуля сервіс робить все можливе для пошуку маркерів, що поряд. Також карти з додатковими полями, наприклад, категорії та обсяг продаж, можуть бути відфільтровані за одним чи більше значеннями. Функція працює на основі групування й забезпечує можливість зосередитися на пошуку потрібних місць. Також є швидкий повнотекстовий пошук за всіма даними на карті.

Додаток надає можливість знайти необхідну точку на карті, для цього потрібно натиснути іконку із зображенням автомобіля для переходу в режим створення автомобільного маршруту.

Функція спільного доступу дозволяє відправляти як посилання для мобільного додатку, так і для веб-версії електронною поштою.

BatchGeo полегшує спілкування з клієнтами, зберігаючи адреси офісів, надає можливість скласти план поїздки чи заїхати до того клієнта, який знаходиться найближче на даний момент часу. Додаток також пропонує перенести на карту список потенційних клієнтів, скласти план поїздки чи заїхати до того клієнта, який знаходиться найближче на даний момент часу.

Мобільна версія BatchGeo Mobile має доступ до карт, що прив'язані до адреси електронної пошти чи існуючого акаунту BatchGeo Pro. Щоб розпочати процес, потрібно виконати три простих кроки:

1. Встановити додаток.
2. Вказати свою адресу електронної пошти, щоб додаток знову знайшов карти.
3. Активувати додаток (Відправляється вам лист на електронну пошту, щоб переконатися, що це саме ви).

BatchGeo надає можливість створення карт за допомогою таких засобів: електронних таблиць, таблиць на веб-сайтах, баз даних — будь-яких даних, на основі яких можна конструювати таблицю з даними про місцеположення та використовувати для експорту в додаток. При створенні таблиці з нуля розробники радять використовувати їхній шаблон таблиці. Потім, підготувавши свої дані, просто скопіювати їх в BatchGeo, щоб створити карту з маркерами.

BatchGeo робить все можливе, щоб угадати потрібний стовпчик із геоінформацією, наприклад: адресами, назвами міста, району, поштовими індексами чи широтою і довготою. Якщо когось налаштування сервісу за замовчуванням не влаштовують, то він може задати власні налаштування.

Додаток також виконує геокодування поштових індексів / адрес і створює потрібну карту. У залежності від кількості адрес у таблиці процес может зайняти декілька хвилин, але якщо вказана широта і довгота, то карта буде створена практично миттєво. Закінчивши картографування, можна зберегти карту на веб-сторінці, натиснувши кнопку «Зберегти і продовжити».

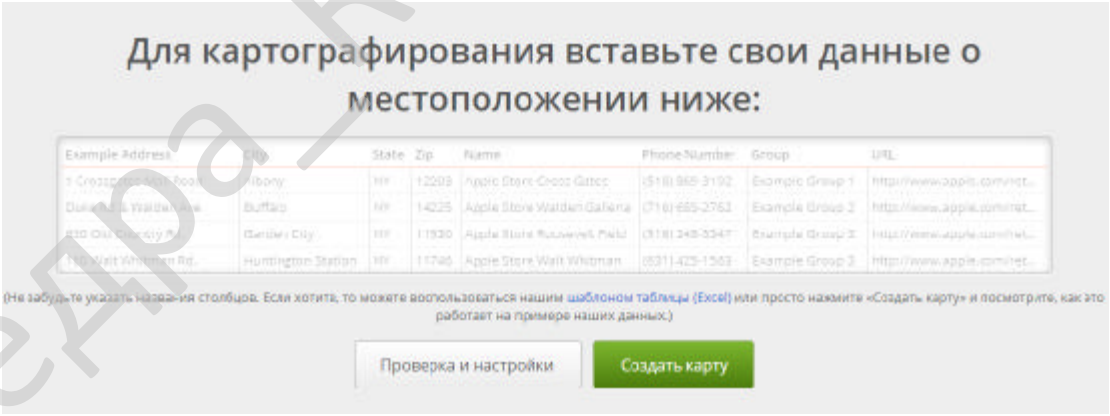
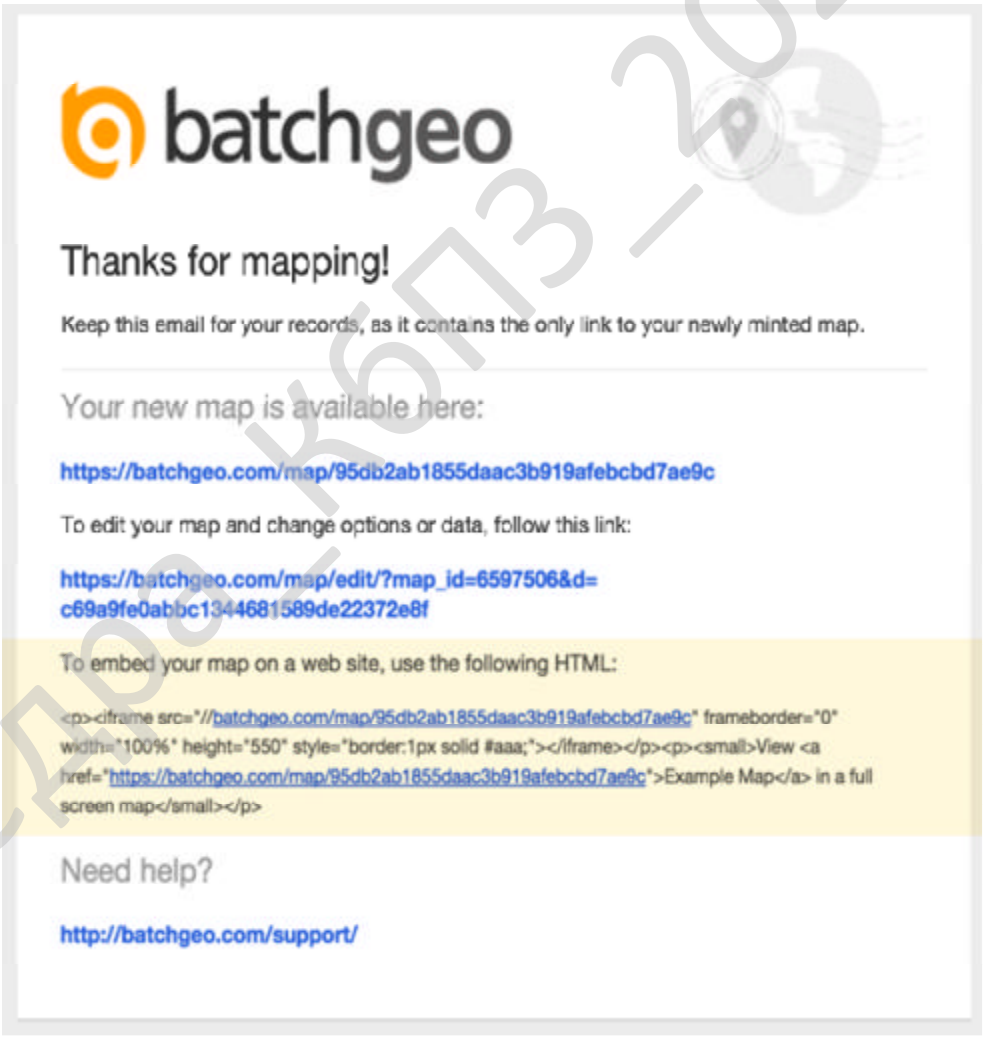


Рисунок 2.3 – Приклад заповнення даних для створення карти

Розробники наголошують, що карти BatchGeo можуть бути особистими, ними можна ділитися з вибраними людьми або зробити їх доступними для загалу. Однією з популярних функцій є можливість

вставляти карти на веб-сайти, для цього не потрібно додавати JavaScript. Карти розміщуються на серверах BatchGeo, але працюють у складі будь-якого сайту абсолютно органічно, як єдине ціле.

Після додавання даних для створення карти за допомогою нашого інтерактивного робочого процесу вам буде запропоновано зберегти карту. Не забудьте вказати адресу електронної пошти при збереженні карти, оскільки автоматично буде відправлено лист із кодом для вставки карти на ваш веб-сайт. Завдяки збереженню карти згодом зможете змінити вашу карту з використанням оновлених даних, створити інший стиль підкладки або скористатися іншими розширеними налаштуваннями картографування.


batchgeo

Thanks for mapping!

Keep this email for your records, as it contains the only link to your newly minted map.

Your new map is available here:

https://batchgeo.com/map/95db2ab1855daac3b919afebcbd7ae9c

To edit your map and change options or data, follow this link:

https://batchgeo.com/map/edit/?map_id=6597506&d=c69a9fe0abbc1344681589de22372e8f

To embed your map on a web site, use the following HTML:

<iframe src="//batchgeo.com/map/95db2ab1855daac3b919afebcbd7ae9c" frameborder="0" width="100%" height="550" style="border:1px solid #aaa;"></iframe> <small>View Example Map in a full screen map</small>

Need help?

http://batchgeo.com/support/

Рисунок 2.4 – Приклад листа з кодом для вставки на сайт, який прийде після збереження карти

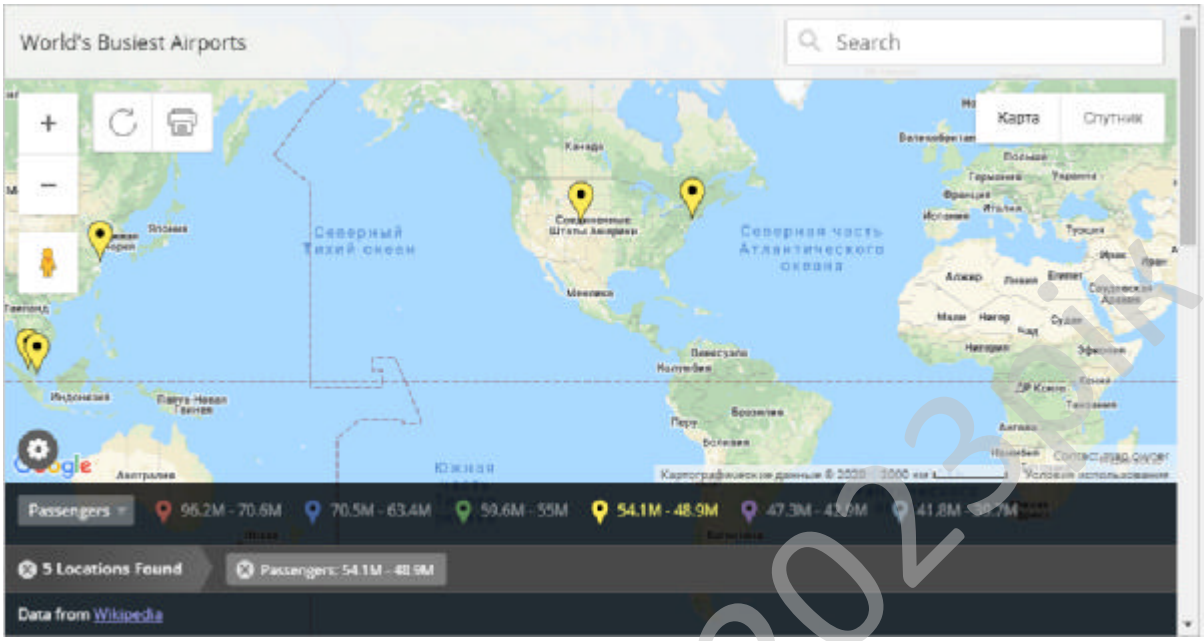


Рисунок 2.5 – Приклад карти

Вище можете побачити приклад карти, що була створена з використанням BatchGeo і потім вбудована в цю сторінку. Є різні способи, щоб поділитися карткою: пряме посилання на повноекранну версію, значок карти з моментальним знімком, а також найпопулярніший варіант - вставка на сайт, як показано вище.

ArcGIS Online – додаток, що є частиною Esri Geospatial Cloud, дозволяє об'єднувати користувачів, дані та місцеположення за допомогою інтерактивних карт. Розробники пропонують роботу з інтелектуальними стилями, керованими даними та інтуїтивно зрозумілими інструментами аналізу, які забезпечують аналітику місця розташування.

ArcGIS Online має такі функції:

1. Створення карт із вбудованою інтерактивністю.
2. Аналіз даних.
3. Зберігання, управління та спільне використання даних у вигляді файлів та веб-слоїв.

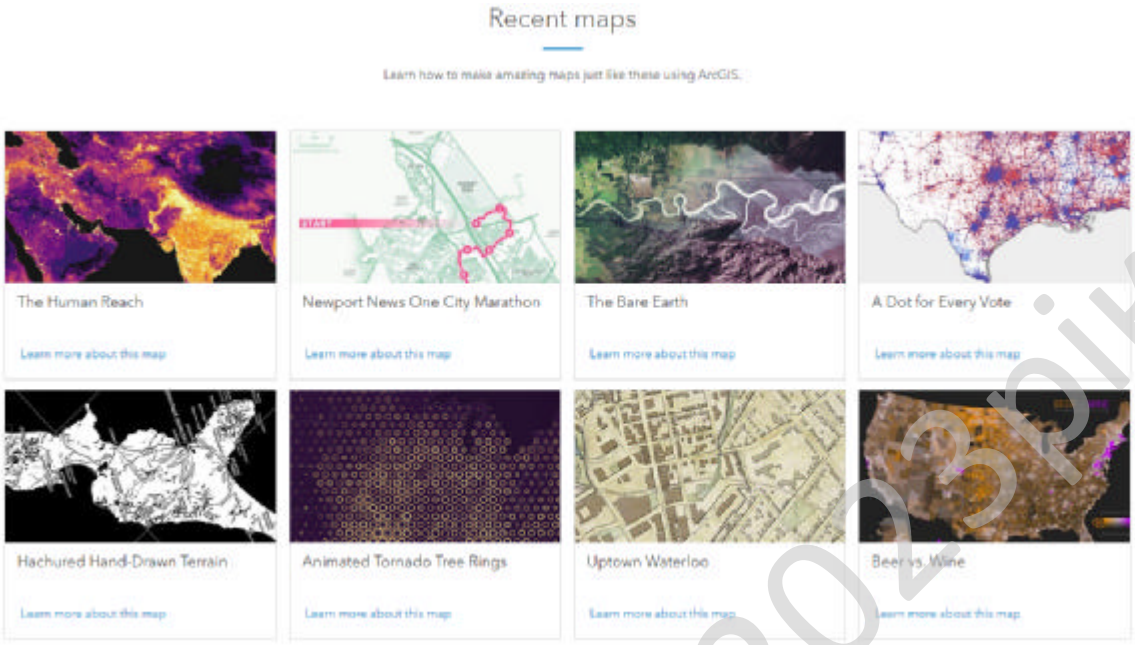


Рисунок 2.6 – Деякі з доступних стилів

Під час роботи з інтерактивними картами додаток надає можливість бачити подробиці при наближенні, клацанням по кнопці викликати інформацію у вигляді тексту, таблиць та зображень, пошуку відомих світових та власних місцеположень, бачити у різних перспективах через фільтрацію даних і зміну кольорів.

Проведення аналізу складається з аналізу шаблонів для прогнозування та визначення наступних кроків, з'ясування зв'язків та викидів у даних. Також можна приєднати дані, розташовані в кількох місцях, чи додати елемент місця розташування у свої.

Скажімо, є завдання оцінити можливі місця для будівництва нового складу. Ця оцінка повинна бути заснована на транспортній доступності, наявності особливих обмежень, наприклад, історичного оточення, доступ до ресторанів та інших закладів, що можуть знадобитися працівникам, доступність суспільного транспорту та тип землекористування, що може обмежувати чи, навпаки, сприяти новому будівництву. Щоб достовірно

оцінити ці місця в кількісному виразі, крім даних, знадобляться ще інструменти аналізу та виміру географічних відношень.

Дивлячись на карту, ви не можете візуально визначити багато закономірностей і відношень. Інструменти просторового аналізу дозволяють їх кількісно відображати як карти, таблиці та графіки.

Інструменти для сумування даних обчислюють загальну кількість об’єктів, довжини, площі та основну описову статистику об’єктів та їхніх атрибутів у межах указаних областей або поблизу інших об’єктів.

Інструменти:

1. Агрегирування точок.
2. Об’єкти, що приєднуються.
3. Сумувати прилеглі.
4. Сумувати у межах.
5. Сумований центр і дисперсія.

Інструменти знаходження місця розташування виконують пошук об’єктів, що задовольняють будь-яке число критеріїв, що задані,:

1. Знайти існуюче місцеположення.
2. Отримати нові місця розташування.
3. Вибір найкращих пунктів обслуговування.
4. Створення області видимості.
5. Створення вододілів.
6. Трансування вниз за течією.
7. Знайти центроїди.

Інструменти збагачення даних домагають виявити характеристики різноманітних областей. Для вибраних районів виходять докладні демографічні дані та статистика. Для великих територій також можна отримувати порівняльну інформацію.

Інструменти:

1. Збагатити шар.

2. Обчислення щільності.

3. Знайти гарячі точки.

4. Пошук викидів.

5. Пошук кластерів точок.

6. Інтерполювати точки.

Інструменти просторового аналізу:

1. Створити буфери.

2. Створити області часу і дорозі.

3. Знайти найближчі.

4. Планувати маршрути.

5. З’єднати початкові точки з пунктами призначення.

Інструменти щоденного управління географічними даними та комбінування останніх перед аналізом:

1. Витягти дані.

2. Злиття кордонів.

3. Створити замощення.

4. Злиття шарів.

5. Накладення шарів.

ArcGIS Online можна використовувати коли завгодно і будь-де, тому що воно є «програмним забезпеченням-сервісом» (SaaS). Esri турбується про оновлення та обслуговування програмного забезпечення. ArcGIS Online забезпечує ведення журналу та інші розширені звіти, дає можливість інтегрувати корпоративну систему аутентіфікації.

Всі можливості ArcGIS Online доступні через API і SDK. Розробники можуть почати свої проекти, використовуючи карти, аналізи і стилі, створені їх колегами по роботі з картами. Можна розширити і налаштовати елементи ArcGIS Online, або ж розробити власні програми за допомогою інструментів розробника.

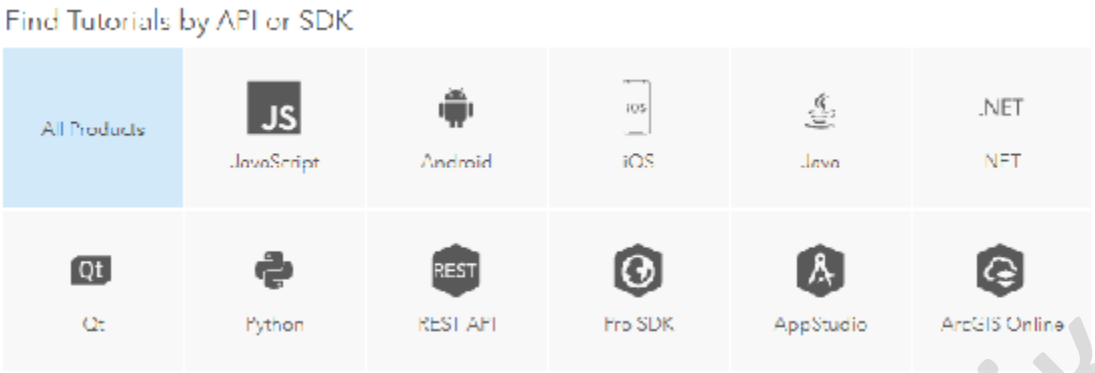


Рисунок 2.7 – Мови програмування та технології, у яких наявні практичні керівництва

Рисунок 2.8 – Практичні керівництва за розділами

Перший розглянутий веб-додаток ЛУН може використовуватися тільки для рієлторських контор. Щоб ваші оголошення з'явилися на порталі, потрібно виставляти їх тільки на сайтах-партнерах проекту. Якщо потрібно стати партнером самому, то для цього треба відповідати багатьом умовам, й це є на платній основі. Тому не всі можуть стати партнерами через те чи інше.

Для того, щоб точно додати свої позначки-оголошення на карту додатка, треба спочатку додати на сайт-партнер, а потім його повинна знайти пошукова система. Також потрібно, щоб ваш запис не був одним із останніх, бо чи дійде до нього користувач під час пошуку. Тому можливо деяким саме такі умови або ж щомісячна оплата не підійдуть.

Другий розглянутий додаток BatchGeo полегшує зв'язок із клієнтами, зберігаючи адреси офісів, надає можливість скласти план поїздки чи заїхати до найближчого на даний момент клієнта, переносить на карту список потенційних клієнтів. Він має, крім веб-версії, ще й мобільний додаток для кращої роботи з IPhone і IPad.

Основою задачею додатка є відмічання об'єктів на карті. Надається можливість створення карт за допомогою різних засобів - електронних таблиць, таблиць на веб-сайтах, баз даних — будь-яких даних, що будуть основою конструювання таблиці з даними про місцеположення, а також використовувати для експорту в додаток. Це допоможе прискорити заповнення карти потрібними об'єктами. Після заповнення карта може бути доступною тільки вам, або можна поділитися нею. Якщо потрібно вставити карту на свій сайт, то тільки скопіювати й вставити в HTML-файл. Для цього не треба використовувати ні бібліотеки, ні JavaScript коду.

Цей приклад можна використовувати для підприємств з великою кількістю складських приміщень. Також тим, хто хоче побудувати маршрут для візитів до клієнтів. Або ж для виведення на карті більшої кількості об'єктів, що можна додати навіть за допомогою бази даних. Так що мета використання додатка залежить від специфіки підприємства.

Оплата за сервіс є помісячною, та кількість доступних користувачів залежить від пакету послуг, які, в свою чергу, різняться цінами. Тому підприємці вже будуть розраховувати, чи доцільно використовувати саме такі послуги.

Останнім розглянутим додатком був ArcGIS Online. Це дуже потужний інструмент для інтерактивних дій із картами. Всі створенні карти мають вбудовану інтерактивність, тому у вас буде доступ до аналізу даних, зберігання, управління та спільного використання даних у вигляді файлів та веб-слоїв, інструменти для сумування даних, що обчислюють

загальну кількість об’єктів, довжини, площі та основну описову статистику об’єктів та їхніх атрибутів у межах указаних областей або поблизу інших об’єктів.

Можна вибирати один з інтелектуальних стилів для карт, а потім апелювати керованими даними та інтуїтивно зрозумілими інструментами аналізу, що забезпечують аналітику місця розташування.

Цей додаток краще підійде, коли тільки планується розміщення складів чи інших будівель підприємства. На основі даних можна побачити оцінку для обраного розташування, яка заснована на транспортній доступності, наявності особливих обмежень, наприклад, історичного оточення, доступу до ресторанів та інших закладів, що можуть знадобитися працівникам, доступність суспільного транспорту та тип землекористування, що може обмежувати чи, навпаки, сприяти новому будівництву.

Ще є можливість покращити вже існуючий або той, що планується, додаток для потреб підприємства за допомогою доступних API чи SDK. Використовуючи карти, аналізи і стилі, створені по роботі з картами, розробники можуть почати свої проекти. Додаток дозволяє розробити власні програми за допомогою інструментів або розширити і налаштувати елементи ArcGIS Online**.** Але тоді потрібно враховувати витрати на доступ до функцій, на оплату розробникам, щоб визначити, чи буде це доцільним і рентабельним.

Система, яка буде розроблятися, матиме в собі основний елемент – карту з маркерами, при натисканні на які буде видно інформацію про них. Також розробиться API для отримання всіх об’єктів, дій над ними. Це можна використовувати для вже існуючих додатків. Проект буде простішим у експлуатації, ніж розглянуті приклади, й не передбачає внесення щомісячної плати за використання.

Спочатку потрібно буде авторизуватися користувачу, бо доступ до карти мають тільки авторизовані. Потім буде визначена роль цього користувача. Вже в залежності від ролі він буде мати різні права – адміністратор може додавати, правити об’єкти на карті. А до їх перегляд й вивантаження файлу з інформацією мають доступ всі авторизовані користувачі.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Продукт, що розробляється, буде веб-додатком. Основною мовою програмування є PHP із використанням фреймворку Laravel. База даних - MySQL. Для карти застосовуватиметься Google Maps Platform, також для вставки й пошуку потрібен JavaScript.

Середовищем для розробки є JetBrains PhpStorm. Воно є зручним для використання й підходить для обраних мов програмування. Також в ньому можна тестувати API requests.

СЧ проекту буде побудована за методологією об'єктно-орієнтованого програмування. Скрипти з використанням JavaScript будуть функціональним програмуванням. Ще знадобиться jQuery — популярна JavaScript-бібліотека з відкритим кодом. Синтаксис jQuery розроблений, щоб зробити орієнтування у навігації зручнішим завдяки вибору елементів DOM, створенню анімації, обробки подій і розробки AJAX-застосунків. Також будуть використані стандартна мова розмітки для створення веб-сторінок і веб-додатків HTML й Sass, що являється скриптовою метамовою, яка інтерпретується в каскадні таблиці стилів (CSS), для стилю сторінок. Щоб полегшити формування сітки сторінок, буде задіяний Bootstrap – це безкоштовний набір інструментів із відкритим кодом, призначений для створення веб-сайтів та веб-додатків, який містить шаблони CSS та HTML для типографіки, форм, кнопок, навігації та інших компонентів інтерфейсу, а також додаткові розширення JavaScript. Він спрощує розробку динамічних веб-сайтів і веб-додатків.

Laravel має багато засобів для розробки, тому, якщо прочитати документацію по фреймворку та слідувати їй, то це пришвидшить процес написання проекту, й сам код буде структурованішим. Одним із засобів є Eloquent ORM, що являється технологію програмування, яка зв'язує бази

даних із концепціями об'єктно-орієнтованих мов програмування, створюючи «віртуальну об'єктну базу даних».

Google Maps Platform має TMJAPI, яке, в свою чергу, дозволяє налаштувати карти з власним вмістом та зображеннями для відображення на веб-сторінках та мобільних пристроях. API JavaScript Maps пропонує чотири основні типи карт (дорожня карта, супутник, гібрид та місцевість), які можна змінювати, використовуючи шари та стилі, елементи керування та події, а також різні сервіси та бібліотеки.

2.3 Розгорнута постановка завдання

Проектування бази даних:

- спроектувати базу даних для проекту;
- зробити міграції для кожної запланованої таблиці, у яких будуть описані всі поля та зв'язки за потребою.

Планування карти сайту:

- визначити, з яких сторінок буде складатися веб-додаток;
- створити головний шаблон для сторінок й інші, в залежності від кількості різного контенту.

Робота над серверною частиною:

- створення моделей для кожної з таблиць, прописування зв'язків за потребою;
- додавання авторизації для користувачів;
- реалізація CRUD - create read update delete, 4 базові функції управління даними «створення, зчитування, зміна і видалення» - для об'єктів;
- робота зі збереженням\видаленням файлів, що будуть додаватися до об'єктів.

Реалізація API:

- авторизація юзера для отримання унікального токена, який буде використовуватися для всіх подальших запитів;
- реалізація CRUD для об'єктів, але з іншим response – даними, що повернуться після виконання запиту;
- написання документації для запитів.

Робота з HTML та CSS:

- додати імпорт потрібних файлів у головний шаблон;
- розмітити сітку на головному шаблоні за допомогою стилів із Bootstrap;
- розмітка інших шаблонів;

- допис нових потрібних стилів.

Робота з TMJAPI:

- дослідження документації, визначення з неї потрібних функцій для додатку, що розробляється.

Робота з JavaScript:

- додавання до одного файлу, який буде мініфікуватися (для швидшого завантаження сторінок), всіх потрібних бібліотек;
- написання функцій, потрібних для роботи з картою – виділення точок, виведення вже створених об’єктів, пошук за назвою міста та інші.

3 ОПИС І ОБГРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Основні частини системи:

- база даних;
- СЧ, у яку входить віддача та збереження об’єктів, авторизація користувачів – однаковий набір дії для сайту й API;
- файлове сховище;
- веб-сайт;
- TMJAPI;
- будь-яка система, що має доступ до API.

У базі даних будуть зберігатися всі потрібні дані в таблицях. До неї звертатиметься СЧ для створення, модифікування, перегляду записів у різних таблицях. Тому ця частина системи має зв'язок тільки з серверною частиною.

СЧ буде посередником між веб-сайтом або системою, що зробила запит, й базою даних. Після отримування даних вона чи поверне потрібну відповідь, чи модифікує дані. Також буде виконувати роботу над файловим сховищем. Це спрощений опис усіх функцій. Частина має зв'язок із базою даних, файловим сховищем, веб-сайтом й будь-якою системою, що виконала запит до неї.

Файлове сховище, у якому будуть знаходитися файли, прив'язані до створених об'єктів, буде мати зв'язок із серверною частиною.

Веб-сайт буде авторизувати користувача, відображати об'єкти на карті та інші дані, відправляти форми на зберігання серверній частині. Тому матиме зв’язок із серверною частиною та TMJAPI.

TMJAPI – це ізольована частина, до якої будуть виконуватися запити з веб-сайту, і, як наслідок, вона буде мати зв’язок тільки з веб-сайтом.

Система, яка має доступ до API нашого веб-додатку через запити, буде авторизувати користувача, отримувати дані, відправляти дані на збереження або видалення. Потім апелюватиме отриманими даними так, як це потрібно їй для функціоналу; матиме зв'язок тільки з серверною частиною.

Таблиця 3.1 – Відмічено всі зв’язки частин системи

Назва	База даних	СЧ	Файлове сховище	Веб-сайт	TMJAPI	Система з доступом до API
База даних		+				
СЧ	+		+	+		
Файлове сховище		+				
Веб-сайт		+			+	
TMJAPI				+		
API		+				+

Для наглядного прикладу побудуємо таблицю, у якій відмічені зв'язки. З цією таблицею буде простіше розробити структурну схему, спираючись на зв’язки між функціональними частинами.

3.2 Розробка структурної схеми

Структурна схема призначена для відображення загальної структури системи, тобто її основних частин та головних зв'язків між ними. Тому для її побудови спочатку треба визначити основні функціональні частини додатку, потім їхні взаємозв’язки та призначення.

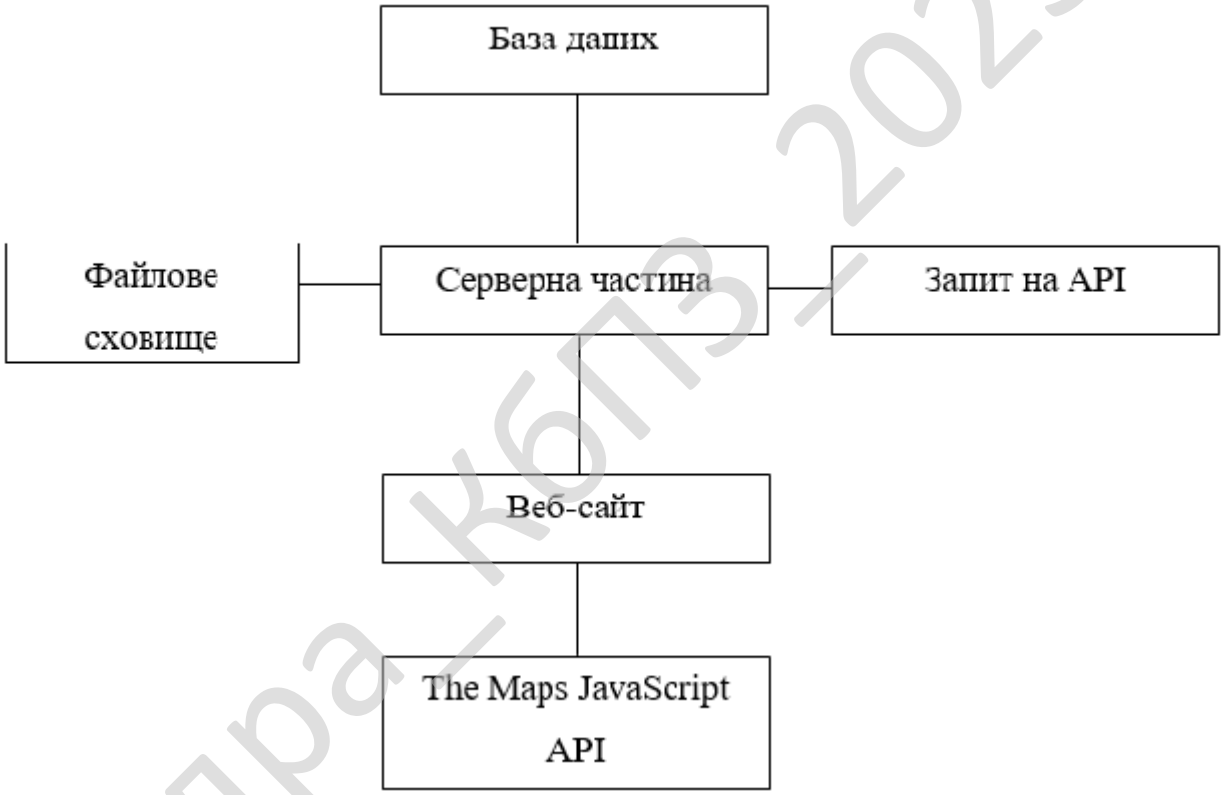


Рисунок 3.1 – Структурна схема системи

На рисунку 3.1 наведена структурна схема. Вона була побудована за допомогою таблиці 3.1, яка, в свою чергу, побудувалася після визначення зв’язків між частинами системи. За допомогою розбору функцій частин можна було окреслити зв’язки між ними.

3.3 Розробка функціональної схеми системи

Функціональна схема відображає взаємодії компонентів програмного забезпечення з описом інформаційних потоків, склад даних в потоках і вказівкою частин, що використовуються.

Автентифікація виконує функцію, спрямовану на забезпечення безпеки інформації. За допомогою неї ідентифікується користувач та визначаються його права доступу.

Автентифікації користувача включає в себе введення логіну та паролю; також забезпечує безпомилкове визначення користувача під час отримання доступу.

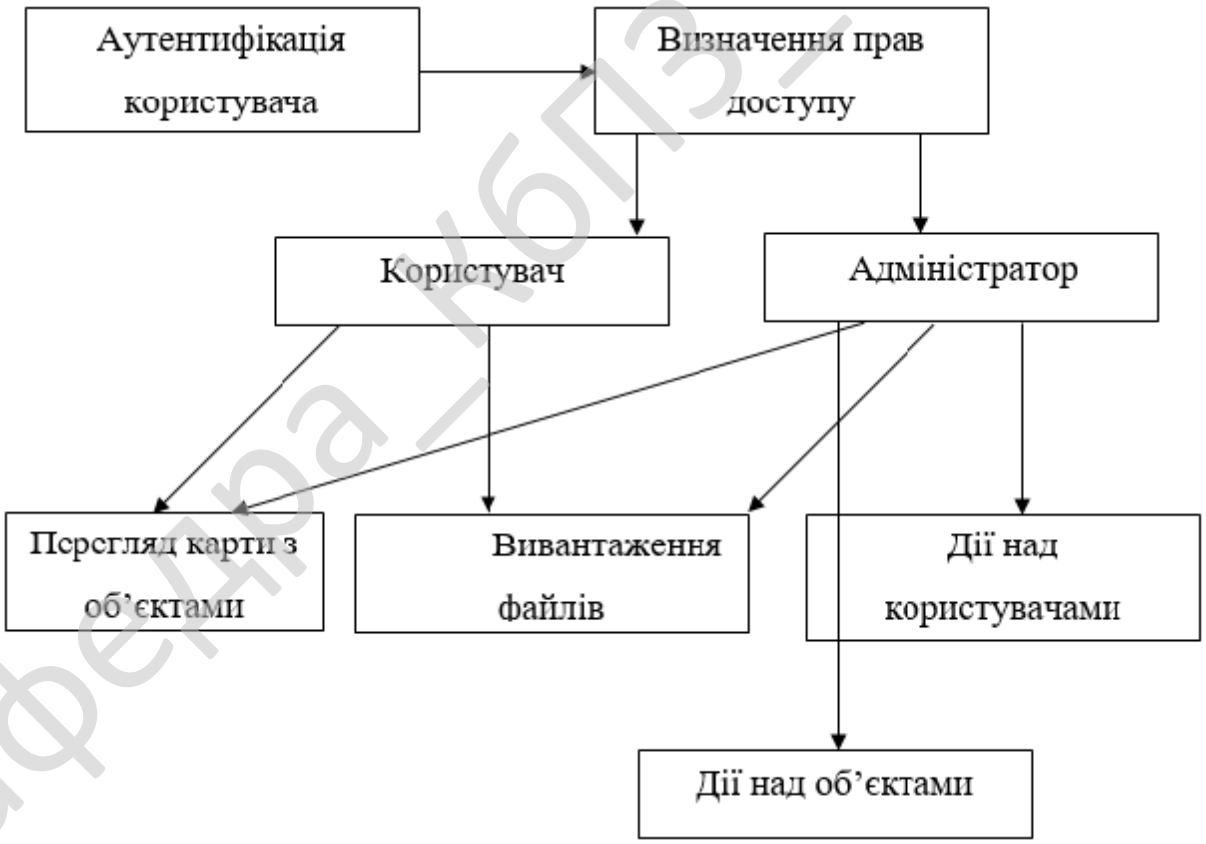


Рисунок 3.2 – Функціональна схема системи

Для визначення функціональних можливостей програмного продукту розробляється функціональна схема. Вона дозволяє відобразити принципи роботи та визначити зв’язки між компонентами програми. На рисунку 3.2 наведена функціональна схема додатку.

Функціональна схема містить інформацію про способи реалізації додатком заданих функцій. За такою схемою можна визначити, як здійснюються перетворення, і які для цього необхідні функціональні елементи. Кожен функціональний елемент містить лише ті входи і виходи, що необхідні для його коректної роботи.

3.4 Розробка діаграми процесів

Діаграма потоків даних (англ. Data Flow Diagram) — модель проектування, графічне представлення «потоків» даних в інформаційній системі. Діаграма потоків даних також може використовуватись для візуалізації процесів обробки даних (структурне проектування).

Для розробника вважається звичним спочатку креслити діаграму потоків даних рівня контексту, завдяки чому буде показано взаємодію системи із зовнішніми модулями. Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати розлого розроблювану систему.

Діаграми потоків даних містять чотири типи графічних елементів:

- процеси - являють собою трансформацію даних у рамках описуваної системи;
- сховища даних (репозиторії);
- зовнішні по відношенню до системи сутності;
- потоки даних між елементами трьох попередніх типів.

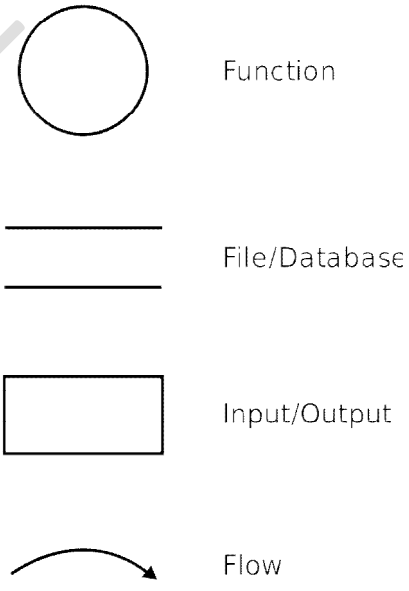


Рисунок 3.3 – Типи графічних елементів

На рисунку 3.3 наведені типи графічних елементів, які використовуватимуться для побудови діаграм.

Спочатку побудуємо діаграму процесу автентифікації користувача. До серверної частини приходить запит на перевірку введеного логіну та паролю користувачем. Запит може прийти з веб-сайту або через API. Сама СЧ перевірить, чи є такий користувач в системі та створює сесію, якщо логін та пароль коректні. Після цього повертає результат на запит. Потім вже, у залежності від прав користувача, він матиме доступ до різних дій на веб-сайті або запитів у випадку API.

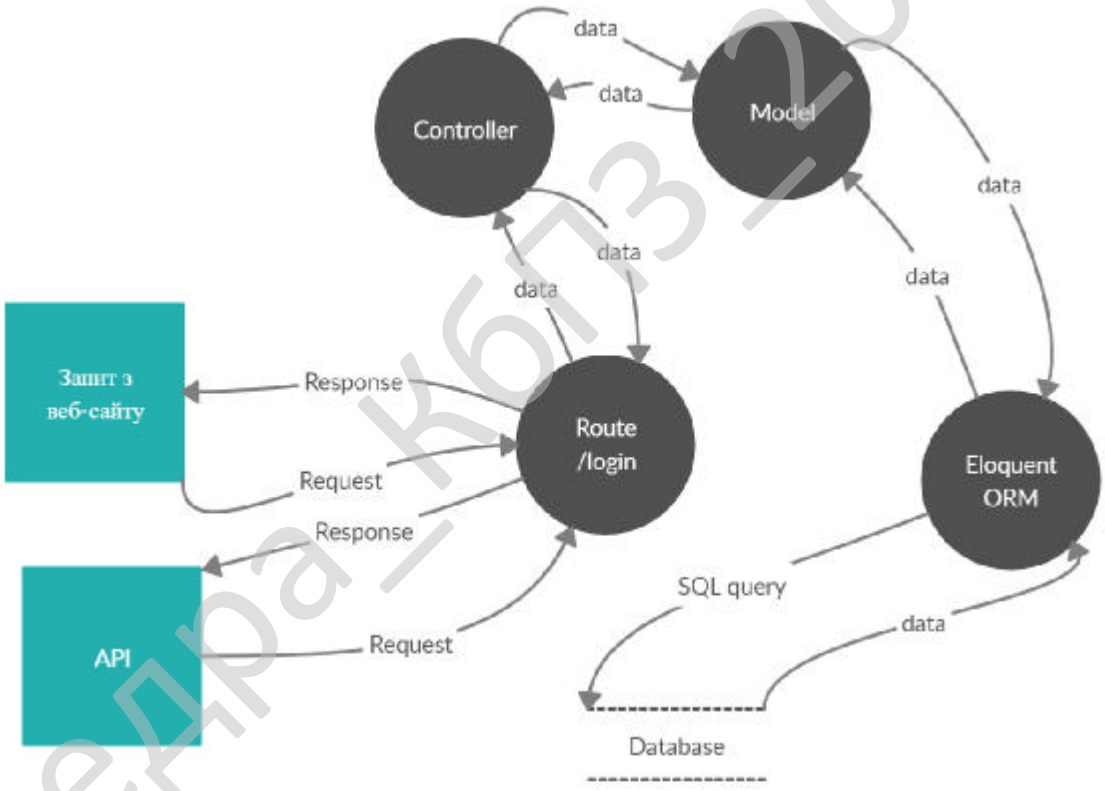


Рисунок 3.4 – Діаграма процесу авторизації користувача

Розглянемо побудовану діаграму процесу авторизації користувача з рисунку 3.4. Методом POST по http буде приходити запит з веб-сайту або з системи, що має доступ до API. Параметри, які будуть приходити, це – request. У них буде емейл та пароль, введені користувачем.

Запити будуть потрапляти на роут /login, вже далі фреймворк направить на потрібний контролер.

Контролер – це клас, із функціями та властивостями, що наслідується від основного класу фреймворку. Функції в ньому виконують дії потрібні саме для цього запиту.

Для роботи з базою даних використовуються моделі. З отриманих даних за допомогою моделі User можна визначити, чи є в базі користувач із таким емейлом, й чи правильний пароль.

На основі цих висновків вже буде відпрацьовувати далі логіка в контролері. Після відпрацювання формуватиметься відповідь(response), формат якої буде залежати від того, звідки прийшов запит. Це буде php-шаблон чи json.

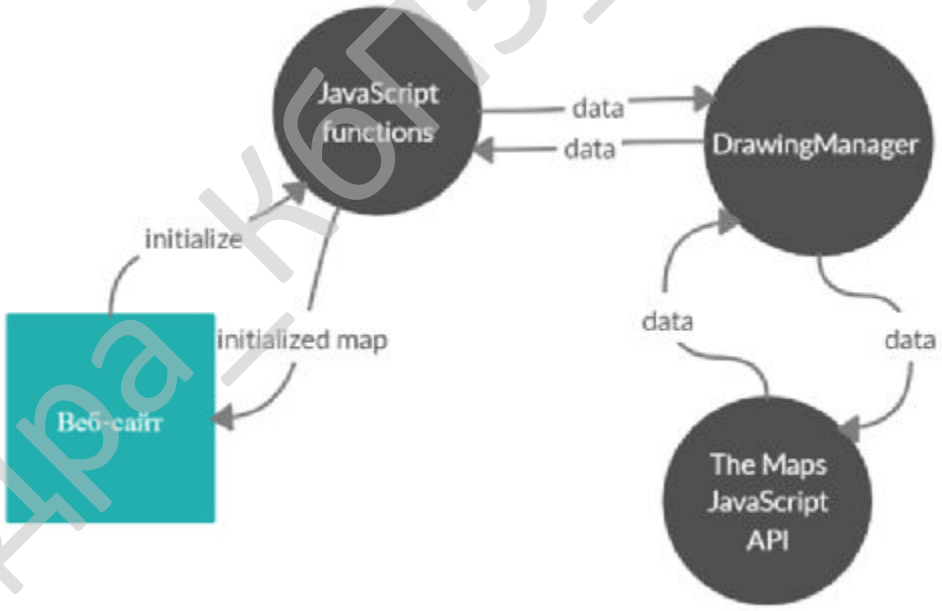


Рисунок 3.5 – Діаграма процесу ініціалізації карти на веб-сайті з можливістю креслення на ній

Наступною побудуємо діаграму процесу збереження об’єкту, накресленого на карті. Але для креслення на карті треба спочатку

ініціалізувати її на веб-сторінці за допомогою JavaScript, додати можливість креслення за допомогою DrawingManager з TMJSAPI. Тому спочатку побудуємо діаграму ініціалізації карти для креслення.

Розглянемо побудовану діаграма процесу ініціалізації карти на веб-сайті з можливістю креслення на ній з рисунку 3.5. Кожен раз, коли користувач заходить на веб-сторінку для додавання нового об’єкту на карту, буде ініціалізуватися карта з включеними опціями для креслення. Після цього користувач вже може відмічати точки для об’єкту. А потім зберігати їх.

Тепер вже можна побудувати процес збереження об’єктів з файлами, прикріпленими до них. Вже маємо веб-сторінку з ініціалізованою картою та опцією DrawingManager. Й після того, як користувач відмітить точки, він вже зможе зберегти об’єкт в базі, натиснувши на кнопку «Зберегти». При натисненні на неї буде відпрацьовувати функція Javascript, яка спочатку валідує кількість точок об’єкту, та якщо їх більше 20, то користувач побачить повідомлення, що перевищено ліміт по кількості. Якщо ж кількість задовольнить валідацію, то відправиться http-запит методом GET на роут. Параметрами будуть координати всіх точок, їх кількість, координати центру об’єкту, назва об’єкту. Потім за описаною вище логікою буде виконано направлення на потрібний контролер та функцію в ньому.

Вже в функції на серверній частині, будуть виконуватися дії для збереження об’єкту в базі. В ці дії входять:

- збереження в базу даних за допомогою моделі;
- створення файлу для цього об’єкту за замовчуванням, збереження його в файловому сховищі;
- перевірка на унікальність назви об’єкту, якщо такий вже є в базі, то назва поточного буде приведена до формату «Назва-час-створення»;

- додавання флеш-повідомлення для користувача, яке він побачить на веб-сайті, про те, що об’єкт був збережений;
- формування відповіді(response), формат якої буде json; при успішному виконанні збереження статус буде 200 й повернеться ідентифікатор об’єкту для подальших дії над ним.

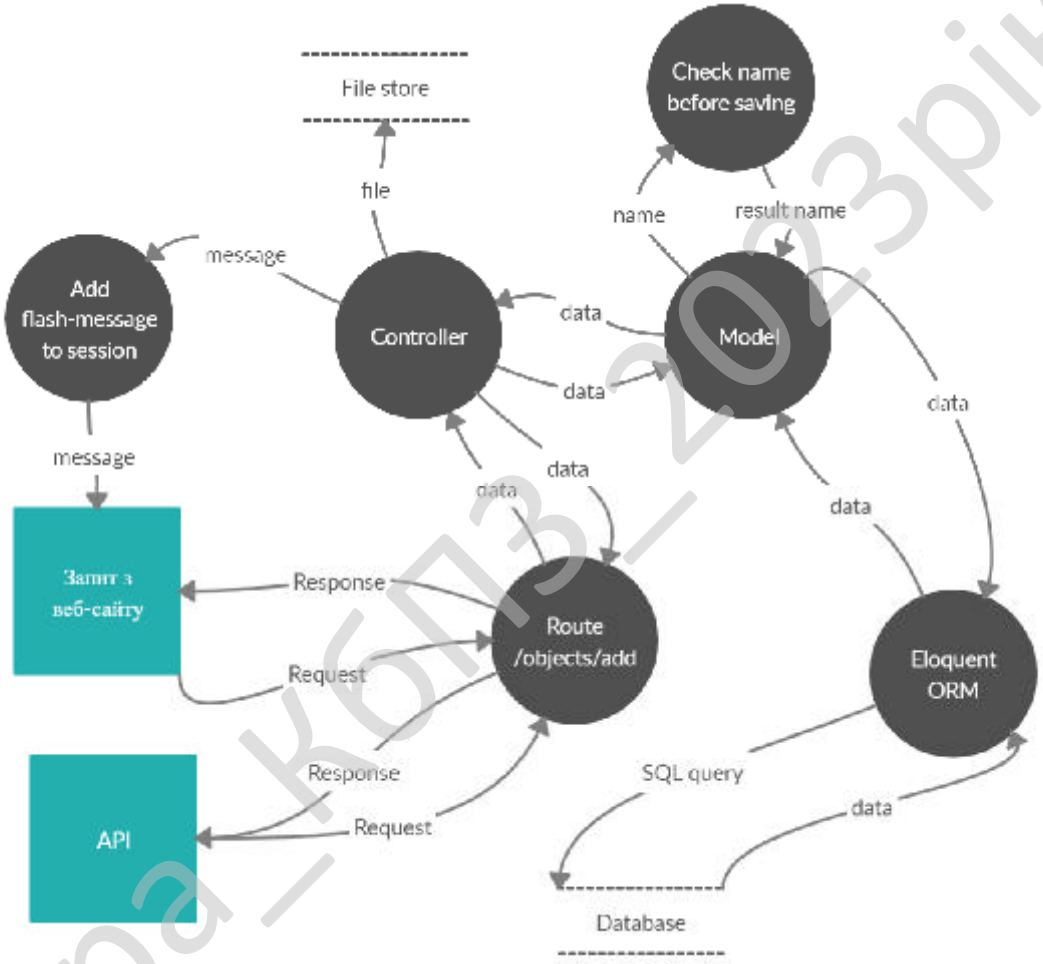


Рисунок 3.6 – Діаграма процесу збереження об’єктів з файлами, прикріпленими до них

Розглянемо побудовану діаграма процесу збереження об’єктів із файлами, прикріпленими до них з рисунку 3.6. На роут /objects/add будуть приходити дані, потрібні для створення об’єкту в базі. Після цього в контролері зберігаються дані в базу за допомогою моделі, перед зберіганням перевіряється назва на унікальність. А потім зберігається файл для цього об’єкту; для дії використовується файлове сховище. Додається

флеш-повідомлення для користувача про те, що об’єкт був створений. Після цього формується відповідь у форматі json.

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ

4.1 Розробка блок-схем та опис алгоритмів функціонування системи

Перед розробкою алгоритмів розглянемо всі можливості, які надає Laravel, – обраний PHP-фреймворк для розробки веб-додатку. Розберемо архітектурний шаблон, який лежить у його основі. Шаблон сприяє впорядкованості структури системи й задає каркас, за яким уже буде працювати вся логіка.

Laravel — безкоштовний, з відкритим кодом PHP-фреймворк, створений Taylor Otwell і призначений для розробки веб-додатків відповідно до шаблону model–view–controller (MVC). Деякі з особливостей Laravel є модульна система упакування з виділеним менеджером залежностей Composer, різні способи для доступу до реляційних баз даних, утиліти, які допомагають у розгортанні додатків і технічного обслуговування, а також його орієнтація на синтаксичний цукор.

Станом на березень 2015 року, Laravel вважається одним із найпопулярніших PHP фреймворком, разом із Symfony2, Nette, CodeIgniter, Yii2 й іншими фреймворками. Сирцевий код Laravel'a розміщується на GitHub і ліцензований відповідно до умов MIT License.

Синтаксичний цукор (англ. syntactic sugar) — термін, що позначає доповнення синтаксису мови програмування, які не додають нових можливостей, а роблять використання мови зручнішим для людини.

Моде́ль–вигляд–контро́лер (або Модель–представлення–контролер, англ. Model-view-controller, MVC) — архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення.

Цей шаблон передбачає поділ системи на три взаємопов'язані частини: модель даних, вигляд (інтерфейс користувача) та модуль керування. Застосовується для відокремлення даних (моделі) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача.

Мета шаблону — гнучкий дизайн програмного забезпечення, який повинен полегшувати подальші зміни чи розширення програм, а також надавати можливість повторного використання окремих компонентів програми. Крім того, використання цього шаблону у великих системах сприяє впорядкованості їхньої структури і робить їх більш зрозумілими за рахунок зменшення складності.

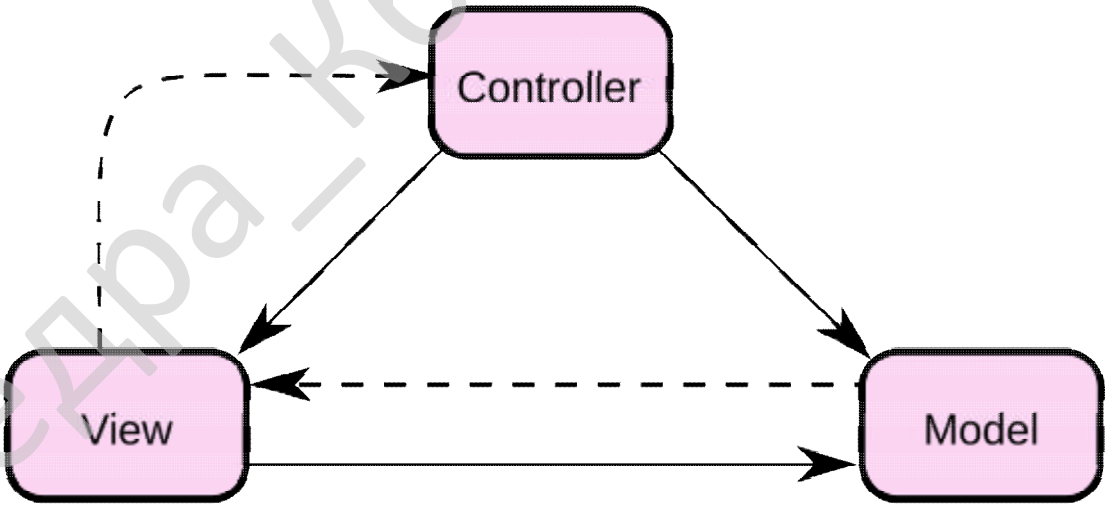


Рисунок 4.1 – Діаграма взаємодії між компонентами шаблону

У рамках архітектурного шаблону модель–вигляд–контролер (MVC) програма поділяється на три окремі, але взаємопов'язані частини з розподілом функцій між компонентами. Модель (Model) відповідає за

зберігання даних та їхню структуру. Вигляд (View) відповідальний за представлення цих даних користувачеві, тобто інтерфейс програми. Контролер (Controller) керує компонентами, отримує сигнали у вигляді реакції на дії користувача (зміна положення курсора миші, натискання кнопки, ввід даних в текстове поле) і передає дані у модель.

Модель є центральним компонентом шаблону MVC і відображає поведінку застосунку, незалежну від інтерфейсу користувача. Модель стосується прямого керування даними, логікою та правилами застосунку.

Вигляд може являти собою будь-яке представлення інформації, одержуване на виході, наприклад, графік чи діаграму. Одночасно можуть співіснувати кілька виглядів (представлень) однієї і тієї ж інформації, наприклад, гістограма для керівництва компанії й таблиці для бухгалтерії.

Контролер одержує вхідні дані й перетворює їх на команди для моделі чи вигляду.

Модель інкапсулює ядро даних і основний функціонал їхньої обробки і не залежить від процесу вводу чи виводу даних.

Вигляд може мати декілька взаємопов'язаних областей, наприклад, різні таблиці і поля форм, у яких відображаються дані.

У функції контролера входить відстеження визначених подій, що виникають в результаті дій користувача. Контролер дозволяє структурувати код шляхом групування пов'язаних дій в окремий клас. Наприклад, у типовому MVC-проекті може бути користувацький контролер, що містить групу методів, пов'язаних із управлінням обліковим записом користувача, таких як реєстрація, авторизація, редагування профілю та зміна пароля.

Зареєстровані події транслюються в різні запити, що спрямовуються компонентам моделі або об'єктам, відповідальним за відображення даних. Відокремлення моделі від вигляду даних дозволяє незалежно використовувати різні компоненти для відображення

інформації. Таким чином, якщо користувач через контролер внесе зміни до моделі даних, то інформація, подана одним або декількома візуальними компонентами, буде автоматично відкоригована відповідно до змін, що відбулися.

Ключові особливості, що лежать в основі архітектури Laravel:

- пакети (англ. Packages) - дозволяють створювати і підключати модулі в форматі Composer до додатка на Laravel; багато додаткових можливостей вже доступно у вигляді таких модулів;
- Eloquent ORM - реалізація шаблону проектування ActiveRecord на PHP; дозволяє строго визначити відносини між об'єктами бази даних; стандартний для Laravel будівник запитів Fluent підтримується ядром Eloquent;
- логіка програми - частина додатку, який розроблюється, оголошена або за допомогою контролерів, або маршрутів (функцій-замикань); синтаксис оголошень схожий на синтаксис, який використовується в каркасі Sinatra;
- зворотня маршрутизація пов'язує між собою посилання і маршрути, які генеруються додатком, дозволяючи змінювати останні з автоматичним оновленням пов'язаних посилань; при створенні посилань за допомогою іменованих маршрутів Laravel автоматично генерує кінцеві URL;
- REST-контролери - додатковий шар для поділу логіки обробки GET- і POST-запитів HTTP;
- автозавантаження класів - механізм автоматичного завантаження класів PHP без необхідності підключати файли їх визначень в include; завантаження на вимогу запобігає завантаження непотрібних компонентів; завантажуються тільки ті з них, які дійсно використовуються;

- укладачі уявлень (англ. View composers) - блоки коду, які виконуються при генерації уявлення (шаблону);
- інверсія управління (англ. Inversion of Control) - дозволяє отримувати екземпляри об'єктів за принципом зворотного управління. Також може використовуватися для створення і отримання об'єктів-одинаків (англ. Singleton);
- міграції - система управління версіями для баз даних; дозволяє зв'язувати зміни в коді програми зі змінами, які потрібно внести в структуру БД, що спрощує розгортання і оновлення програми;
- модульне тестування (юніт-тести) - грає дуже велику роль в Laravel, який сам по собі містить велику кількість тестів для запобігання регресій (помилок внаслідок поновлення коду або виправлення інших помилок);
- посторінкове виведення (англ. Pagination) - спрощує генерацію сторінок, замінюючи різні способи вирішення цього завдання єдиним механізмом, вбудованим в Laravel;
- підтримка noSQL СУБД Redis;
- безліч готових адмін-панелей, шаблонів і CRUD;
- шаблонізатор Blade (за замовчуванням); можна також використовувати інший аналог - Twig (як в Symfony 4);
- можливість підключати CSS шаблони.

Шаблон active record — це шаблон проєктування, що використовується при реалізації доступу до реляційних баз даних. Вперше згадується Мартіном Фаулером у книжці «Patterns of Enterprise Application Architecture». Інтерфейс такого об'єкта включає функції CRUD, а також поля, що більш чи менш прямо відповідають полям відповідної таблиці в базі даних.

Active Record реалізує популярний підхід об'єктно-орієнтованого проєкціювання (ORM). Кожен клас AR відображає таблицю (чи

представлення) бази даних, екземпляр AR — запис цієї таблиці, а загальні операції CRUD реалізовані як методи AR. У результаті можна працювати з більшою об'єктно-орієнтованістю.

Першою побудуємо блок-схему алгоритму відпрацювання запиту на збереження об’єкту. Основна логіка не залежить від того, звідки прийшов запит, бо відповідь завжди буде в json-форматі.

Спочатку запит пройде через маршрутизацію й потрапить в REST-контролер. І вже тоді дійде до функція, яка містить у собі потрібний алгоритм відпрацювання.

Тепер можна представити логіку для самого збереження. Спочатку треба присвоїти змінним, якими можна буде потім оперувати, відповідні значення з вхідних параметрів.

Потім для роботи з базою буде використовуватися модель – клас AR, який відображає таблицю (чи представлення) бази даних. Для цього потрібно описати спочатку модель й зробити міграцію для створення таблиці в базі. Це буде описано в наступному підрозділі. У модель можна додавати функції, що будуть викликатися при якійсь дії з CRUD методів. Таким чином, буде описана функція, що викликатиметься перед збереженням даних в таблицю, й перевірятиме назву на унікальність. Якщо така вже є, то назва поточного об’єкта зміниться. І після цього вже буде збережено в базу.

Тепер вже для створеного об’єкту буде збережено файл. Всі файли, які можна прикріпляти для нього, мають бути . xls або . xlsx формату. Тому для створення файлу за замовчуванням використаємо бібліотеку для роботи з Excel-файлами. З її допомогою створимо файл, у якому для початку будуть тільки координати об’єкта, а вже потім користувач може його перезалити. Й збережемо його в файловому сховищі.

Рисунок 4.2 – Блок-схема алгоритму створення об’єкту

Далі якщо запит був з веб-сайту, то в сесію користувача буде додано флеш-повідомлення про те, що об’єкт було створено. Потім сформується й повернеться відповідь на запит.

Наступною побудуємо блок-схему алгоритму відображення всіх об’єктів на карті. Цей алгоритм буде написаний мовою програмування Javascript з використанням бібліотеки JQuery та TMJAPI.

Після завантаження сторінки буде викликатися функція, яка буде ініціалізувати карту та отримувати всі об’єкти через http-запит до серверної частини.

Як об’єкти вже будуть отримані, то можна починати процес їх відображення. Це буде робитися за допомогою циклу for, таким чиною можна пройтися по всім елементам в масиві. Але в основний цикл по проходженню об’єктів буде вкладений ще один цикл для збирання координат всіх точок. Початкову точку треба додати ще раз в кінці, щоб об’єкт був цільним. Потім отриманий масив з координатами передамо в функцію з TMJAPI для відмічення об’єкта на карті за цими точками.

Після цього витягуємо координати центру кожного об’єкту для того, щоб поставити в середину маркер. Це теж буде робитися за допомогою функції з TMJAPI. До вже доданого маркеру додаємо функцію по натисненню на нього. По кліку на нього буде показуватися карточка, в якій буде назва об’єкта, кнопки для його видалення, вивантаження файлу, прикріплення нового файлу до об’єкта, відображення координат.

Рисунок 4.3 – Блок-схема алгоритму відображення об’єктів

Для початку роботи треба встановити базовий Laravel проект. Фреймворк Laravel має кілька системних вимог. Усі ці вимоги задовольняє віртуальна машина Laravel Homestead, тому рекомендується використовувати Homestead як місцеве середовище розробки Laravel.

Однак, якщо не використовуєте Homestead, потрібно буде переконатися, що ваш сервер відповідає таким вимогам:

- PHP >= 7.2.5;
- BCMath PHP Extension;
- Ctype PHP Extension;
- Fileinfo PHP extension;
- JSON PHP Extension;
- Mbstring PHP Extension;
- OpenSSL PHP Extension;
- PDO PHP Extension;
- Tokenizer PHP Extension;
- XML PHP Extension.

Надалі не будемо використовувати Homestead, тому потрібно переконатися, що сервер відповідає таким вимогам.

Спочатку треба завантажити та встановити Composer, а вже потім із його допомогою завантажити інсталятор Laravel. Консольна команда для завантаження - composer global require laravel/installer.

Після встановлення за допомогою команди laravel можна отримати базовий Laravel проект останньої версії у вказаному вами каталозі. Наприклад, laravel new project створить каталог із іменем project, що містить свіжу установку Laravel з усіма встановленими залежностями Laravel. Команда для цього - laravel new project.

Якщо PHP встановлено локально, можна використовувати вбудований сервер розробки PHP для обслуговування програми, для цього треба скористатися командою serve Artisan. Ця команда запустить сервер розробки за адресою http://localhost: 8000. Команда для цього - php artisan serve.

Тепер у нас є основа проекту та можна приступити до подальшої реалізації функціоналу. Тому перейдемо до проектування бази даних. Створення й опис таблиць буде реалізовано за допомогою міграції.

Міграції - це як контроль версій для бази даних, що дозволяє змінювати та ділитися схемою баз даних програми. Міграції, як правило, поєднуються з конструктором схем Laravel для створення схеми бази даних додатка.

Фасад схеми Laravel забезпечує агностичну підтримку бази даних для створення та маніпулювання таблицями в усіх підтримуваних системах баз даних Laravel.

Для створення міграції треба використати команду make:migration Artisan. Наприклад - php artisan make:migration create_objects_table.

Нова міграція буде розміщена у вашій database/migrations директорії. Кожне ім'я файлу міграції містить часову позначку, яка дозволяє Laravel визначати порядок міграцій.

Клас міграції містить два методи: up та down. Метод up використовується для додавання нових таблиць, стовпців або індексів у вашу базу даних, тоді як метод down повинен змінити операції, виконані методом up.

Щоб запустити всі міграції, треба виконати команду - php artisan migrate.

Міграція для створення таблиці користувачів із базовими полями для них вже є в фреймворку, тому треба тільки додати додаткові потрібні для проекту поля. А для об’єктів створимо й опишемо нову міграцію. Після цього потрібно буде їх запустити.

Тепер можна описувати моделі до відповідних таблиць. Модель для користувачів вже є, тому її потрібно тільки дописувати. А для об’єктів потрібно додати.

Eloquent ORM, що входить до Laravel, забезпечує прекрасну просту реалізацію ActiveRecord для роботи з вашою базою даних. Кожна таблиця бази даних має відповідну "Модель", яка використовується для взаємодії з цією таблицею. Моделі дозволяють запитувати дані у ваші таблиці, а також вставляти нові записи в таблицю.

Додатково до моделі об’єктів опишемо функцію для перевірки й, якщо потрібно, модифікації назви на унікальність перед збереженням у базу.

Вже маємо таблиці в базі даних й описані моделі, тому перейдемо до контролерів.

Замість того, щоб визначати всю логіку обробки запиту у файлах маршрутів, можна організувати цю поведінку за допомогою класів Controller. Контролери можуть групувати пов'язані логіки обробки запитів в один клас. Контролери зберігаються в каталозі app/Http/Controllers.

Контролер на авторизацію користувачів входить у базовий проект, тому його реалізовувати не треба.

Створимо контролер для об’єктів. У ньому потрібно реалізувати методи для збереження, віддачі, оновлення та видалення об’єктів.

Тепер потрібно описати шаблони для сторінок веб-сайту.

Blade - це простий, але потужний шаблоновий двигун, забезпечений Laravel. На відміну від інших популярних двигунів шаблонів PHP, Blade не обмежує вас використовувати звичайний PHP-код у своїх переглядах. Насправді всі перегляди Blade складаються у звичайний код PHP та кешуються до тих пір, поки вони не будуть змінені, тобто Blade додає фактично нульові накладні витрати до програми. Файли перегляду blade використовують розширення файлу .blade.php і зазвичай зберігаються в каталозі resources/views.

Після шаблонів вже можна починати працювати з JavaScript. Для збору всіх файлів для веб-сторінок буде використовуватися Laravel Mix. Він надає вільний API для визначення етапів складання Webpack для програми Laravel, використовуючи декілька загальних попередніх процесорів CSS та JavaScript. Завдяки простому методу ланцюга ви можете чітко визначити свій конвеєр активів. Для цього потрібно переконатися, що Node.js та NPM встановлені на машині.

Після реалізації всіх алгоритмів для JavaScript й написання стилів на CSS потрібно запустити команди - npm install, npm run dev. Після них уже будуть сформовані всі додаткові файли для сторінок.

Тепер, коли вже все було реалізовано, можна починати тестувати всі функції, що були закладені в систему з самого початку розробки.

4.2 Захист розробленого програмного забезпечення

Для того, щоб доступ до карти мали тільки авторизовані користувачі, буде використовуватися auth Middleware.

Middleware забезпечує зручний механізм фільтрації HTTP-запитів, що надходять у систему. Наприклад, Laravel включає auth Middleware, що підтверджує автентичність користувача системи. Якщо користувач не має автентифікації, тоді Middleware перенаправить його на екран входу. Однак, якщо користувач має автентифікацію, Middleware дозволить запиту продовжувати роботу в додатку.

Для запитів API буде теж Middleware, але там вже буде перевірятися токен із header запиту.

Таким чином, всі не авторизовані користувачі будуть бачити тільки сторінку авторизації. А для API у кожного користувача буде токен, з яким можна відправляти подальші запити.

Ще реалізовано role Middleware, яка вже йду по пріоритету після auth Middleware. Наприклад, якщо користувач авторизований й хоче додати об’єкт на карту, але він не адміністратор, то role Middleware перенаправить його на іншу сторінку або поверне негативну відповідь, у разі якщо це API запит.

5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Для розгортання системи потрібен сервер з описаними в попередньому розділі вимогами для фреймворку Laravel. Також потрібна база даних MySQL.

Тепер можемо перейти до встановлення проекту. Для початку скачаємо проект за допомогою команди git clone. Наступним кроком буде створення файлу з конфігураціями, у якому потрібно прописати доступи до бази. Після цього можемо встановити проект за допомогою Composer – composer install. Й встановити інші пакети, які потрібні для веб-сайту, – npm install. Далі запуск міграцій – php artisan migrate. Було створено Seeder для збереження користувача-адміністратора. Тому після запуску міграції треба виконати команду - php artisan db:seed.

І, у залежності від домену, можна заходить на веб-сайт, яким являється встановлена система. Заповнювати карту об’єктами, завантажувати файли до них.

6 ОСНОВНІ ВИСНОВКИ

Було реалізовано систему, яка відповідає технічному завданню, яке було описане на початку роботи.

За основу було взято PHP фреймворк Laravel. Він має багато зручних інструментів для розробки, як для PHP, так й для шаблонів веб-сторінок, й JavaScript та CSS файлів.

Веб-додаток - це самостійна система, до якої можуть авторизуватися користувачі. В залежності від ролі користувача він має різні права в системі. Адміністратор може зберігати, редагувати, дивитися об’єкти та користувачів. А сам користувач може тільки дивитися об’єкти на карті та вивантажувати інформацію про них. Це саме стосується й API, й залежить від того токен, якого за роллю користувача, був надісланий в запиті.

База даних, використана у додатку, - MySQL. Створення таблиць було реалізовано за допомогою міграцій з фреймворку, а запити до неї через Eloquent ORM.

Для ініціалізації карти на веб-сторінці було використано TMJAPI, зв’язок з яким реалізовано за допомогою JavaScript. Також для зручнішої роботи з елементами DOM на веб-сторінці було додано jQuery, популярну JavaScript-бібліотека з відкритим кодом.

Для верстки було використано Bootstrap, безкоштовний набір інструментів із відкритим кодом, призначений для створення веб-сайтів та веб-додатків

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Цифровий ресурс - https://developers.google.com/maps/documentation/javascript/tutorial
2. Цифровий ресурс - https://laravel.com/docs/7.x
3. Скляр Д. Изучаем PHP 7: руководство по созданию интерактивных веб-сайтов. – Диалектика, 2017. – 464с.
4. Зандстра М. PHP. Объекты, шаблоны и методики программирования. 5-е издание. – Диалектика, 2019. – 736с.
5. Цифровий ресурс - https://developer.mozilla.org/uk/docs/Web/JavaScript
6. Цифровий ресурс - https://api.jquery.com/
7. Цифровий ресурс - https://getbootstrap.com/docs/4.5/getting-started/introduction/
8. Стаффер М. Laravel. Полное руководство. 2-е издание. - Питер Пресс, 2020. – 512с.
9. Ніксон Р. Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5. 4-е издание. – Питер, 2016. – 768с.
10. Конноллі Т. Базы данных. Проектирование, реализация и сопровождение. Теория и практика. 3-е издание. – Вильямс, 2018. – 1440с.
11. https://habr.com - веб-сайт, який поєднує ознаки соціальної мережі і колективного блогу, створений для публікації новин, аналітичних статей, думок, пов'язаних із інформаційними технологіями, бізнесом та Інтернетом.
12. https://getbootstrap.com/ - клієнтський фреймворк.
13. https://idg.net.ua/blog/uchebnik-css - Підручник CSS.
14. http://www.elibukr.org/ - Електронна бібліотека України.

15. Закон України «Про захист інформації в інформаційно-телекомунікаційних системах» [Электронный ресурс]. – Режим доступа к ресурсу: http://zakon4.rada.gov.ua/laws/show/2594-15

16. Бен Фрейн «HTML5 и CSS3 разработка сайтов для любых браузеров и устройств», 2014 г.

17. Селезнев К. От SQL к NoSQL и обратно [Електронний ресурс] / Константин Селезнев. – Режим доступу: http://www.osp.ru/os/2012/02/13014127.

18. Алекс Феррара, Мэтью Мак-Дональд Программирование web-сервисов для .NET.

19. jQuery. Подробное руководство по продвинутому JavaScript, 2-е издание, из-во “Символ-Плюс”, 624 стр., 2011р.

20. PHP 5, из-во “BHV-СПб”, 1104 стр., 2008р.

21. Гмурман В.Е. Теория вероятностей и математическая статистика / В.Е. Гмурман. – М.: Высшая школа, 2003. – 479 с.

22. Хоган Б. Книга веб-программиста. Секреты профессиональной разработки веб-сайтов / Б. Хоган, К. Уоррен, М. Уэбер, К. Джонсон, А. Годин. – СПб.: Питер, 2013. – 288 с.

23. Касимов Р.Р. Вдосконалення алгоритмів QoS маршрутизації в мережах з технологією IP/MPLS на основі прогнозу трафіка: автореф. дис. на здобуття наук. ступеня кандидата техн. наук: спец. 05.12.02 – телекомунікаційні системи та мережі / Р.Р. Касимов. – Київ: Київський державний університет інформаційно-комунікаційних технологій, 2011. – 24 с.

24. А. Філд, П. Харрісон Функціональне програмування. / А. Філд, П. Харрісон. – К: Москва "Мир", 1993. – 55 с.

25. Б. Лоусон Изучаем HTML5. Библиотека специалиста / Б. Лоусон, Р. Шарп. – Питер : Питер, 2011. – 235 с.

26. Хоган Б. Книга веб-программиста. Секреты профессиональной разработки веб-сайтов / Б. Хоган, К. Уоррен, М. Уэбер, К. Джонсон, А. Годин. – СПб.: Питер, 2013. – 288 с.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

					ВКРБ-123.23.0003.00.00.ТЗ		
Вим.	Арк.	№ документа	Підпис	Дата			
Розробив		Бондаревський С.О.			Програмне забезпечення системи керування відносинами з клієнтами	Літ.	Аркуш 1 / Аркушів 6
Перевірив		Пархоменко Ю.М.				Б	
Н. Контр.		Гермак В.С.				ЦНТУ КМ-19	
Затв.		Смірнов О.А.					

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи керування відносинами з клієнтами.

2 Підстава для розробки

Підставою для розробки служить завдання на бакалаврську роботу, видане на кафедрі програмування комп'ютерних систем і мереж (нак. №10-02 від 5.01.2023 року).

3 Мета та призначення розробки

Метою бакалаврської роботи є дослідження та програмна реалізація розробка системи керування відносинами з клієнтами.

4 Джерела розробки

Джерелом цієї бакалаврської роботи є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка системи, а також розробка взаємодії враховуючи використання декількох каналів Інтернет;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– серверну частину, яка задовольняє визначеним можливостям;

– зберігання даних у базі даних;

- веб-сайт з авторизацією користувачів.

5.3 Вимоги до функціональних характеристик

Розроблений веб-сайт повинен коректно працювати у браузерах

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Система повинна відповідати усі критеріям та стандартам, які притаманні аналогічним системам та мати високу надійність при експлуатації.

5.6 Умови експлуатації

Місце дислокації повинно задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ архітектури IBM PC, працювати в ОС Ubuntu і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Ubuntu

5.8.1 Обладнання

Комп'ютер Intel® Core Duo/2048 Mb/80 Gb/SVGA 14" 1Mb або сумісні з ним.

5.8.2 Мова програмування

PHP, JavaScript.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Конфігураційні файли.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Перелік документів, що розробляються

– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуші.
– Пояснювальна записка – 53 аркушів.

8 Етапи розробки

8.1 Збір і обробка інформації по темі бакалаврської роботи. Постановка задачі на виконання бакалаврської роботи (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень бакалаврської роботи.

8.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

9.1 Подання бакалаврської роботи на попередній захист 16.05.2023 р.

9.2 Подання бакалаврської роботи на захист 21.05.2023 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник бакалаврської дипломної роботи

_____ Пархоменко Ю.М.

Програмне забезпечення системи керування відносинами з клієнтами

Лістинг програми

Код документу 12

Носій: CD/DVD-диск

Загальна кількість аркушів: 33

Літера: РП

Кропивницький 2023

Міграції

Файл 2023_02_12_120123_create_users_table.php – міграція для створення таблиці користувачів

```
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateUsersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('users', function (Blueprint $table) {
            $table->increments('id');
            $table->string('login')->unique();
            $table->string('password');
            $table->boolean('is_admin')->default(false);
            $table->string('api_token');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('users');
    }
}
```

Файл 2023_02_12_122311_create_objects_table.php - міграція для створення таблиці об’єктів

```
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateObjectsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('objects', function (Blueprint $table) {
            $table->increments('id');
            $table->string('name');
            $table->text('coordinates');
            $table->string('center');
            $table->integer('count');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('objects');
    }
}
```

Файл AdminSeed.php – Seeder для створення користувача-адміністратора

```php
<?php

use Illuminate\Database\Seeder;
use App\Models\User;

class AdminSeed extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        User::create([
            'login' => 'Адмін',
            'password' => bcrypt('123456'),
            'is_admin' => true
        ]);
    }
}
```

Файл User.php – модель-представлення таблиці users

```
<?php

namespace App\Models;

use Illuminate\Notifications\Notifiable;
use Illuminate\Foundation\Auth\User as Authenticatable;

class User extends Authenticatable
{
    use Notifiable;

    /**
     * The attributes that are mass assignable.
     *
     * @var array
     */
    protected $fillable = [
        'login', 'password', 'is_admin', 'api_token'
    ];

    /**
     * The attributes that should be hidden for arrays.
     *
     * @var array
     */
    protected $hidden = [
        'password'
    ];
}
```

Файл MapObject.php – модель-представлення таблиці objects

```
<?php

namespace App\Models;

use Carbon\Carbon;
use Illuminate\Database\Eloquent\Model;

class MapObject extends Model
{
    protected $table = 'objects';

    protected $fillable = [
        'name', 'coordinates', 'center', 'count'
    ];

    protected $appends = [
        'link'
        ];

    protected $casts = [
        'count' => 'integer'
    ];

    public function getLinkAttribute()
    {
        if (file_exists(public_path('/files/' . $this->name . '.xls')))
        {
            return '/files/' . $this->name . '.xls';
        }

        if (file_exists(public_path('/files/' . $this->name . '.xlsx')))
        {
            return '/files/' . $this->name . '.xlsx';
        }

    }

    public function getCoordinatesAttribute($value)
    {
        return unserialize($value);
    }

    public function setNameAttribute($value)
```

```
            {
                $count = MapObject::where('name', '=', $value )->where('id',
'!=', $this->id)->count();
                if ($count != 0) {
                    $value = str_replace(' ', '-', $value . '-' . Carbon::now());
                    $value = str_replace(':', '-', $value);
                    $this->attributes['name'] =  $value;
                } else {
                    $this->attributes['name'] = $value;
                }
            }
        }
```

Файл ObjectsController.php – контролер для роботи з об'єктами

```
<?php

namespace App\Http\Controllers;

use App\Http\Middleware\RedirectIfAuthenticated;
use Illuminate\Http\Request;
use App\Models\MapObject;
use Illuminate\Support\Facades\File;
use Illuminate\Support\Facades\Input;
use Illuminate\Support\Facades\Redirect;
use Maatwebsite\Excel\Facades\Excel;

class ObjectsController extends Controller
{
    public function create()
    {
        return view('objects.create');
    }

    public function store(Request $request)
    {
        $name = $request['name'];

        $coordinates = serialize($request['polygonArray']);

        $count = $request['count'];

        $center = $request['center'];


        $object = MapObject::create([
            'name' => $name,
            'coordinates' => $coordinates,
            'center' => $center,
            'count' => $count
        ]);

        $array = [];
        $array = $object->coordinates;
        Excel::create($object->name, function ($excel) use ($array) {
            $excel->sheet('Sheet 1', function ($sheet) use ($array) {
                $sheet->fromArray($array);
```

```
            });
        })->save('xls', public_path('/files'), true);

        session()->flash('message', "Додано новий об'єкт");

        return response()->json([
            'id' => $object->id
        ], 200);
    }

    public function show()
    {
        return view('objects.show');
    }

    public function file(MapObject $object)
    {
        return view('objects.file', compact('object'));
    }

    public function get()
    {
        $objects = MapObject::all();
        $count = $objects->count();

        return response()->json([
            'objects' => $objects,
            'count' => $count
        ], 200);
    }

    public function edit(MapObject $object)
    {
        return response()->json([
            'object' => $object
        ], 200);
    }

    public function saveCoordinates(MapObject $object, Request $request)
    {
        $oldname = $object->name;
        $count = $object->count;

        $coordinates = [];

        $a = 0;
```

```
                for ($i = 0; $i < $count; $i++) {
                    $coordinates[$i] = $request[$a] . ',' . $request[$a + 1];
                    $a += 2;
                }

                $object->coordinates = serialize($coordinates);
                $object->name = $request['name'];
                $object->save();

                if ($oldname != $request['name']) {
                    if (file_exists(public_path('/files/' . $oldname . '.xls')))
{
                        $extention = '.xls';
                    } elseif (file_exists(public_path('/files/' . $oldname .
'.xlsx'))) {
                        $extention = '.xlsx';
                    }

                    if ($extention != null) {
                        File::Copy(public_path('/files/' . $oldname .
$extention), public_path('/files/' . $object->name . $extention));
                        File::Delete(public_path('/files/' . $oldname .
$extention));
                    }
                }

                return redirect()->action('ObjectsController@show');
            }


            public function save(MapObject $object, Request $request)
            {
                $extension = Input::file('file')->getClientOriginalExtension();

                if ($extension !== 'xls' && $extension !== 'xlsx') {
                    return Redirect::back()->withErrors(['Цей формат файлу не
підтримується']);
                }

                File::Delete(public_path('/files/' . $object->name . '.xls'));
                File::Delete(public_path('/files/' . $object->name . '.xlsx'));

                if ($extension == 'xls') {
                    $file = $request->file('file');
                    $name = $object->name;
```

```
            $filename = $name . '.xls';
        }

        if ($extension == 'xlsx') {
            $file = $request->file('file');
            $name = $object->name;
            $filename = $name . '.xlsx';
        }

        $file->move(public_path() . '/files', $filename);

        return redirect()->action('ObjectsController@show');
    }

    public function delete(MapObject $object)
    {
        $object->delete();

        return redirect()->action('ObjectsController@show');
    }
}
```

Файл SessionsController.php – контролер для авторизації користувачів

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class SessionsController extends Controller
{
    public function create()
    {
        return view('sessions.create');
    }

    public function store()
    {
        $this->validate(request(), [
            'login' => 'required',
            'password' => 'required',
        ]);

        if (!auth()->attempt(request(['login', 'password']))) {
            return back()->withErrors([
                'message' => 'Please check your credentials and try
again.'
            ]);
        }

        return redirect()->home();
    }

    public function destroy()
    {
        auth()->logout();

        return redirect()->home();
    }
}
```

Файл TokenApiMiddleware.php – перевірка авторизації для API

```
<?php

namespace App\Http\Middleware;

use App\Models\User;
use Closure;

class TokenApiMiddleware
{
    /**
     * Handle an incoming request.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  \Closure  $next
     * @return mixed
     */
    public function handle($request, Closure $next)
    {
        if($user = User::whereToken($request->header('token'))->first())
{

            return $next($request);
        } else {

            return response()->json([
                'error' => 'wrong token'
            ], 401);
        }
    }
}
```

Файл AdminMiddleware.php – перевірка ролі користувача

```
<?php

namespace App\Http\Middleware;

use App\Models\User;
use Closure;
```

```
class AdminMiddleware
{
    /**
     * Handle an incoming request.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  \Closure  $next
     * @return mixed
     */
    public function handle($request, Closure $next)
    {
        if(auth()->user()->is_admin) {
            return $next($request);
        } else {
            return redirect('/home');
  }
    }
}
```

Файл web.php – файл з роутами

```
<?php

/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| contains the "web" middleware group. Now create something great!
|
*/


Route::get('/login', 'SessionsController@create')->name('login');
Route::post('/login', 'SessionsController@store');

Route::group(['middleware' => 'auth'], function () {

    Route::get('/', 'ObjectsController@index')->name('home');

    //Login/out
    Route::get('/logout', 'SessionsController@destroy')->name('logout');

    //Objects
    Route::get('/objects', 'ObjectsController@show');
    Route::get('/objects/get', 'ObjectsController@get');

    Route::group(['middleware' => 'admin'], function () {
        Route::get('/objects/create', 'ObjectsController@create');
        Route::get('/objects/add', 'ObjectsController@store');
        Route::get('/objects/{object}/delete',
'ObjectsController@delete');
        Route::get('/objects/{object}/edit', 'ObjectsController@edit');
        Route::post('/objects/{object}/file/save',
'ObjectsController@save');
        Route::post('/objects/{object}/save',
'ObjectsController@saveCoordinates');
        Route::get('/objects/{object}/file', 'ObjectsController@file');
```

```
    });

    Route::get('{object}', function (App\Models\MapObject $object) {
        return $object->link;
    });
});
```

Файл master.blade.php – головний шаблон для всіх веб-сторінок

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta name="csrf-token" value="{{ csrf_token() }}">
    <title>Maps</title>
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
    <link href="{{ asset('css/app.css') }}" rel="stylesheet">
</head>

<body>

<header>
    @include ('layouts.nav')
</header>

<div class="container">
    @if ($flash = session("message"))
        <div id="flash-message" class="alert alert-success" role="alert">
            {{ $flash }}
        </div>
    @endif
        <div class="row">
            @yield('content')
        </div>
</div>

@yield('scripts')
<script src="{{ asset('js/app.js') }}"></script>
</body>
</html>
```

Файл nav.blade.php – шаблон навігації

```
<div class="blog-masthead">
    <div class="container">
        <nav class="nav blog-nav">
```

```
                    @if (Auth::check())
                        <a class="nav-link active" href="{{
action('ObjectsController@create') }}">Додати новий об'єкт</a>
                        <a class="nav-link active" href="{{
action('ObjectsController@show') }}">Показати всі об'єкти</a>
                        <a class="nav-link ml-auto"
                           href="/">{{ Auth::user()->login }}</a>
                        <a class="nav-link ml-auto" href="{{
action('SessionsController@destroy') }}">Вийти</a>
                    @endif
                </nav>
            </div>
        </div>
```

Файл objects/create.blade.php – шаблон сторінки додавання нового об’єкту

```
        @extends ('layouts.master')

        @section ('content')
            <div class="col-sm-3 blog-sidebar">
                <div class="sidebar-module">
                    <div class="form-group">
                        <button id="add" type="button" class="btn btn-success
form-control" data-toggle="modal"
                                data-target="#exampleModal" disabled>Додати новий
об'єкт
                        </button>
                    </div>
                    <div class="form-group">
                        <button id="delete" type="button" class="btn btn-danger
form-control">Очистити</button>
                    </div>
                    <div class="form-group">
                        <input type="text" id="search-create" name="search-
create" class="form-control" placeholder="Пошук">
                    </div>
                    <div class="form-group">
                        <p id="search-info"></p>
```

```
                        <button id="search-btn-create" type="button" class="btn
btn-primary form-control">Пошук</button>
                    </div>
                </div>
            </div>
            <div class="col-md-8">
                <script
src="https://maps.googleapis.com/maps/api/js?key=AIzaSyCDUEenUN0OjhDKFFJbL_D4IS4
g7XjP2q0&libraries=geometry,drawing"></script>
                <div id="map" style="height: 800px;width: 800px;margin:
0px;padding: 0px;"></div>
            </div>

            <!-- Modal -->
            <div class="modal fade" id="exampleModal" tabindex="-1" role="dialog"
aria-labelledby="exampleModalLabel"
                 aria-hidden="true">
                <div class="modal-dialog" role="document">
                    <div class="modal-content">
                        <div class="modal-header">
                            <h5 class="modal-title"
id="exampleModalLabel">Створення нового об'єкта</h5>
                            <button type="button" class="close" data-
dismiss="modal" aria-label="Close">
                                <span aria-hidden="true">&times;</span>
                            </button>
                        </div>
                        <div class="modal-body">
                            <div class="form-group">
                                <label for="name">Назва об'єкта:</label>
                                <input type="text" class="form-control" id="name"
name="name" required>
                            </div>
                            <p id="info"></p>
                        </div>
                        <div class="modal-footer">
                            <button type="button" class="btn btn-secondary" data-
dismiss="modal">Закрити</button>
                            <button type="submit" class="add btn btn-
primary">Зберегти</button>
                        </div>
                    </div>
                </div>
            </div>
        @endsection
```

Файл show.blade.php – шаблон сторінки відображення всіх об’єктів

```
@extends ('layouts.master')

@section ('content')
    <div class="col-sm-4 blog-sidebar">
        <div class="sidebar-module">
            <div class="form-group">
                <input type="text" id="search" name="search" class="form-
control" placeholder="Пошук">
            </div>
            <div class="form-group">
                <p id="search-info"></p>
                <button id="search-btn" type="button" class="btn btn-
primary form-control">Пошук</button>
            </div>
            <div class="coordinates">
                <form method="POST" id="coordinates-form" action=""
novalidate>
                {{ csrf_field() }}
                </form>
            </div>
        </div>
    </div>
    <div class="col-md-8">
        <script
src="https://maps.googleapis.com/maps/api/js?key=AIzaSyCDUEenUN0OjhDKFFJbL_D4IS4
g7XjP2q0&libraries=geometry,drawing"></script>
        <div id="map_canvas"></div>
    </div>
@endsection
```

Файл errors.blade.php – шаблон відображення помилок

```
@if (count($errors))
    <div class="form-group">

        <div class="alert alert-danger">
            <ul>
                @foreach ($errors->all() as $error)
                    <li>{{ $error }}</li>
                @endforeach
```

```
                </ul>
            </div>

        </div>
    @endif
```

Файл file.blade.php – шаблон сторінки додавання файлу до об’єкту

```
    @extends ('layouts.master')

    @section ('content')

        <div class="col-md-4 center p-3 rounded border border-primary">
            <form method="POST" enctype="multipart/form-data" id="file-form"
action="/objects/{{ $object->id }}/file/save" >
                {{ csrf_field() }}
                <h4>Завантажити файл з інформацією про об'єкт</h4>
                <div class="form-group">
                    <label for="file">Файл:</label>
                    <input type="file" class="form-control" id="file"
name="file" required>
                </div>
                <div class="form-group">
                    <button type="submit" class="btn btn-primary form-
control" >Зберегти</button>
                </div>
            </form>
            @include ('layouts.errors')
        </div>

    @endsection
```

Файл sessions/create.blade.php – шаблон сторінки авторизації користувачів

```
    @extends ('layouts.master')

    @section ('content')
        <div class="col-md-4 center p-3 rounded border border-primary">
            <h1>Авторизуватися</h1>

            <form method="POST" action="/login">
                {{ csrf_field() }}
```

```
                    <div class="form-group">
                        <label for="login">Логін:</label>
                        <input type="text" class="form-control" id="login"
name="login" required>
                    </div>

                    <div class="form-group">
                        <label for="password">Пароль:</label>
                        <input type="password" class="form-control" id="password"
name="password" required>
                    </div>

                    <div class="form-group">
                        <button type="submit" class="btn btn-primary form-
control">Авторизуватися</button>
                    </div>

                    @include ('layouts.errors')

                </form>
            </div>
        @endsection
```

Файл addObject.js – файл з функціями для створення об’єкту з веб-сторінки

```
var geocoder;
var map;
var polygonArray = [];
var count;
var center;
var centerMap = {lat: 50.27, lng: 30.21};
var zoom = 7;

//map
var init = function () {
    map = new google.maps.Map(
        document.getElementById("map"), {
            center: centerMap,
            zoom: zoom,
            mapTypeId: google.maps.MapTypeId.ROADMAP
        });
    var drawingManager = new google.maps.drawing.DrawingManager({
        drawingMode: google.maps.drawing.OverlayType.POLYGON,
        drawingControl: true,
        drawingControlOptions: {
            position: google.maps.ControlPosition.TOP_CENTER,
            drawingModes: [
                google.maps.drawing.OverlayType.POLYGON
            ]
        },
        polygonOptions: {
            fillColor: '#BCDCF9',
            fillOpacity: 0.5,
            strokeWeight: 2,
            strokeColor: '#57ACF9',
            clickable: false,
            editable: false,
            zIndex: 1
        }
    });
    drawingManager.setMap(map);

    google.maps.event.addListener(drawingManager, 'polygoncomplete',
function(polygon) {
        $('#add').removeAttr('disabled');
        count = polygon.getPath().getLength();
        for (var i = 0; i < count; i++) {
```

```
                polygonArray.push(polygon.getPath().getAt(i).toUrlValue(6));
            }
            var result = polygonCenter(polygon);
            center = String(result[0].toFixed(6)) + ',' +
String(result[1].toFixed(6));
        });

    }

    window.init = init;
    google.maps.event.addDomListener(window, "load", init);


    //ajax
    $('body').on('click', '.add', function(){
        if(count <= 20){
            $name = $('#name').val();
            $.ajax({
                url: "/objects/add",
                method: "GET",
                data: {
                    'name': $name,
                    'polygonArray': polygonArray,
                    'center': center,
                    'count': count
                },
                dataType : "json",
                complete: function() {},
                statusCode: {
                200: function(answer) {
                    id = answer.id;
                    console.log(id);
                    var url = "/objects/" + id + "/file";
                    $(location).attr('href',url);

                }
                }
            });
        }
        else
        {
            $('#info').text("Об'єкт складається більше ніж з 20 точок, тому
його не можна зберегти.");
        }
    });
```

```
        function polygonCenter(poly) {
            var lowx, highx, lowy, highy, lats = [], lngs = [], center_x,
center_y, vertices = poly.getPath();
            for (var i = 0; i < vertices.length; i++) {
                lngs.push(vertices.getAt(i).lng());
                lats.push(vertices.getAt(i).lat());
            }

            lats.sort();
            lngs.sort();
            lowx = lats[0];
            highx = lats[vertices.length - 1];
            lowy = lngs[0];
            highy = lngs[vertices.length - 1];
            center_x = lowx + ((highx - lowx) / 2);
            center_y = lowy + ((highy - lowy) / 2);
            return [center_x, center_y];
        }

        $('#delete').on('click', function()
        {
            init();
            show();
        });

        var show = function () {
            $url = "/objects/get";
            $.ajax({
                url: $url,
                method: "GET",
                data: {},
                dataType: "json",
                complete: function () {
                },
                statusCode: {
                    200: function (answer) {

                        countObject = answer.count;

                        //drawing all objects
                        for ($a = 0; $a < countObject; $a++) {
                            coordinates = [];
                            $object = answer.objects[$a];
                            count = $object['count'];

                            //getting coordinates for drawing object
```

```
                            for ($i = 0; $i < count; $i++) {
                                $str = $object['coordinates'][$i];
                                arr = $str.split(',');
                                coordinates.push({lat: Number(arr[0]), lng:
Number(arr[1])});
                            }

                            flightPlanCoordinates = [];

                            for ($i = 0; $i < count; $i++) {
                                flightPlanCoordinates.push(coordinates[$i]);
                            }

                            flightPlanCoordinates.push(coordinates[0]);

                            flightPath = new google.maps.Polyline({
                                path: flightPlanCoordinates,
                                geodesic: true,
                                strokeColor: '#FF0000',
                                strokeOpacity: 1.0,
                                strokeWeight: 2
                            });

                            //drawing current object
                            flightPath.setMap(map);

                            //getting the marker's coordinates
                            $str = $object['center'];
                            arr = $str.split(',');
                            $center = {lat: Number(arr[0]), lng: Number(arr[1])};

                            var marker = new google.maps.Marker({
                                position: $center,
                                map: map,
                                title: $object['name']
                            });
                            (function (marker, $object) {
                                google.maps.event.addListener(marker, 'click',
function (event) {
                                    //adding the marker
                                    var contentString = '<div id="content"
class="information"><h4>' + $object['name'] + '</h4><br>' +
                                        '<a href="/objects/' + $object['id'] +
'/delete">Видалити</a>' +
                                        '<br>' +
```

```
                                        '<a href=' + $object['link'] + '
download>Завантажити інформацію</a>' +
                                        '<br>' +
                                        '<a href="/objects/' + $object['id'] +
'/file">Завантажити нову інформацію</a>' +
                                        '<button id=' + $object['id'] + '
class="btn btn-primary show-coordinates m-4">Показати координати</button>'
                                    '</div>';
                                    var infowindow = new google.maps.InfoWindow({
                                        content: contentString
                                    });

                                    infowindow.open(map, marker);
                                });
                            })(marker, $object);


                        }
                    }
                }
            });
        };

        window.init = show;
        google.maps.event.addDomListener(window, "load", show);

        $('body').on('click', '#search-btn-create', function () {
            var city = $('#search-create').val();
            $url = 'https://maps.googleapis.com/maps/api/geocode/json?address=' +
city + '&sensor=false&language=ru';
            $.ajax({
                url: $url,
                method: "GET",
                dataType: "json",
                complete: function () {
                },
                statusCode: {
                    200: function (answers) {
                        console.log(answers);
                        if (answers['results'] && answers['results'].length == 0)
{
                            $('#search-info').text('За запитом нічого не
знайдено');
                        }
                        else {
                            $('#search-info').text('');
```

```
                            centerMap =
answers['results'][0]['geometry']['location'];
                            zoom = 10;
                            init();
                            show();
                        }
                    }
                }
            });

        });
```

Файл showObject.js – файл з функціями для відображення всіх об’єктів на веб-сторінці

```
var coordinates = [];
var arr = [];
var count = 0;
var countObject = 0;
var flightPath;
var flightPlanCoordinates = [];
var id = 0;
var id_coordinates = 0;
var center = {lat: 50.27, lng: 30.21};
var zoom = 7;

var show = function () {
    $url = "/objects/get";
    $.ajax({
        url: $url,
        method: "GET",
        data: {},
        dataType: "json",
        complete: function () {
        },
        statusCode: {
            200: function (answer) {

                countObject = answer.count;

                //initMap
                let map = new
google.maps.Map(document.getElementById('map_canvas'), {
                    zoom: zoom,
                    center: center,
                    mapTypeId: google.maps.MapTypeId.ROADMAP
                });

                //drawing all objects
                for (let a = 0; a < countObject; a++) {
                    coordinates = [];
                    let object = answer.objects[a];
                    count = object['count'];

                    flightPlanCoordinates = [];

                    //getting coordinates for drawing object
```

```
                            for (let i = 0; i < count; i++) {
                                let str = object['coordinates'][i];
                                arr = str.split(',');
                                coordinates.push({lat: Number(arr[0]), lng:
Number(arr[1])});
                                flightPlanCoordinates.push({lat: Number(arr[0]),
lng: Number(arr[1])});
                            }

                            flightPlanCoordinates.push(coordinates[0]);

                            flightPath = new google.maps.Polyline({
                                path: flightPlanCoordinates,
                                geodesic: true,
                                strokeColor: '#FF0000',
                                strokeOpacity: 1.0,
                                strokeWeight: 2
                            });

                            //drawing current object
                            flightPath.setMap(map);

                            //getting the marker's coordinates
                            let str = object['center'];
                            arr = str.split(',');
                            center = {lat: Number(arr[0]), lng: Number(arr[1])};

                            let marker = new google.maps.Marker({
                                position: center,
                                map: map,
                                title: object['name']
                            });
                            (function (marker, object) {
                                google.maps.event.addListener(marker, 'click',
function (event) {
                                    //adding the marker
                                    let contentString = '<div id="content"
class="information"><h4>' + object['name'] + '</h4><br>' +
                                        '<a href="/objects/' + object['id'] +
'/delete">Видалити</a>' +
                                        '<br>' +
                                        '<a href=' + object['link'] + '
download>Завантажити інформацію</a>' +
                                        '<br>' +
                                        '<a href="/objects/' + object['id'] +
'/file">Завантажити нову інформацію</a>' +
```

```
                                    '<button id=' + object['id'] + '
class="btn btn-primary show-coordinates m-4">Показати координати</button>'
                                    '</div>';
                                    let infowindow = new google.maps.InfoWindow({
                                        content: contentString
                                    });

                                    infowindow.open(map, marker);
                                });
                            })(marker, object);


                        }
                    }
                }
            });
        };

        window.init = show;
        google.maps.event.addDomListener(window, "load", show);


        $('body').on('click', '.show-coordinates', function () {
            id = $(this).attr("id");
            let url = "/objects/" + id + '/edit';
            $.ajax({
                url: $url,
                method: "GET",
                data: {
                    'id': id
                },
                dataType: "json",
                complete: function () {
                },
                statusCode: {
                    200: function (answer) {
                        let object = answer.object;
                        count = object['count'];
                        let name = object['name'];
                        $('#coordinates-form').append(`<div class="form-group p-3
rounded border border-success"> <label for="name">Назва обьекта:</label><input
type="text" name="name" class="form-control" value="${name}"></div>`);
                        //getting coordinates for object
                        for (let i = 0; i < count; i++) {
                            let str = object['coordinates'][i];
                            arr = str.split(',');
```

```
                            $('#coordinates-form').attr('action',
`/objects/${id}/save`);
                            $('#coordinates-form').append(`<span>Точка ${i + 1}:
</span><div class="form-group p-3 rounded border border-success">
                                <div class="column">
                                <label>lat:</label>
                                <input type="number" name="${id_coordinates}"
class="form-control" value="${arr[0]}">
                                </div>
                                <div class="column">
                                <label>lng:</label>
                                <input type="number" name="${id_coordinates + 1}"
class="form-control" value="${arr[1]}">
                                </div>
                            </div>`);
                            id_coordinates = id_coordinates + 2;
                        }
                        $('#coordinates-form').append('<div class="form-
group"><button type="submit" class="btn btn-success save form-control">Зберегти
зміни</button></div>');
                    }
                }
            });
        });

        $('body').on('click', '#search-btn', function () {
            var city = $('#search').val();
            let url = 'http://maps.googleapis.com/maps/api/geocode/json?address='
+ city + '&sensor=false&language=ru';
            $.ajax({
                url: url,
                method: "GET",
                dataType: "json",
                complete: function () {
                },
                statusCode: {
                    200: function (answer) {
                        if (answer['results'] && answer['results'].length == 0) {
                            $('#search-info').text('За запитом нічого не
знайдено');
                        }
                        else {
                            $('#search-info').text('');
                            center =
answer['results'][0]['geometry']['location'];
                            zoom = 10;
```

```
                        show();
                    }
                }
            }
        });

    });
```