МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХЕРСОНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Факультет комп’ютерних наук, фізики та математики

Кафедра комп’ютерних наук та програмної інженерії

РОЗРОБКА ТА ВПРОВАДЖЕННЯ ВЕБ-СЕРВІСУ “ПАРК ХЕРСОНСЬКОГО ДЕРЖАВНОГО УНІВЕРСИТЕТУ”

(назва теми)

Кваліфікаційна робота (проєкт)

на здобуття ступеня вищої освіти «бакалавр»

Виконав: здобувач 4 курсу 431 групи

Спеціальності:122 Комп’ютерні науки

Освітньо-професійної програми:
Комп'ютерні науки

Микитас Данило Олексійович

Керівник: кандидат педагогічних наук
доцент Вінник М.О.

Рецензент кандидат педагогічних наук,
доцент Кушнір Н.О.

Херсон – Івано-Франківськ – 2024

ЗМІСТ

ВСТУП

Зростаюча свідомість суспільства щодо екологічних проблем і важливості природних резервів визначає необхідність розробки інформаційних інструментів для підтримки екологічної свідомості та взаємодії з природним середовищем. У цьому контексті, актуальність створення веб-сервісу "Парк Херсонського державного університету" набуває непередбаченої вагомості, оскільки дане дослідження висвітлює прагнення забезпечити взаємодію громадян з природними ресурсами в онлайн-режимі, а саме з ресурсами парку та ботанічного саду Херсонського державного університету.

Сучасне суспільство стає учасником глобальної дискусії про екологічні виклики, та тому розробка інтерактивного інформаційного ресурсу для Херсонського державного університету, що дозволяє віртуально пізнавати та досліджувати природне навколишнє середовище, визначається потребою у відкритому доступі до інформації про рослинний світ, архітектурні перлини та природні озера парку.

Серед найбільш актуальних проблем сучасності - зменшення доступності екологічної освіти та низький рівень екологічної грамотності. Веб-сервіс "Парк Херсонського державного університету" вирішує ці питання, створюючи можливість для віртуального вивчення та пізнання природних унікальностей парку, що, в свою чергу, сприяє підвищенню екологічної обізнаності та формуванню відповідального ставлення до природи.

Таким чином, веб-сервіс не лише відповідає на тенденції сучасності, але і визначається як стратегічний інструмент у розвитку здорової екологічної культури та підтримки екологічного туризму. Активна участь у вирішенні цих завдань є необхідною для створення стійкого та екологічно збалансованого майбутнього.

Метою цього наукового дослідження є створення та вдосконалення інформаційної системи, спрямованої на інтерактивне взаємодію громадськості з природними резерватами через віртуальний простір, зокрема, реалізація веб-сервісу "Парк Херсонського державного університету". Це дослідження має на меті вирішити ряд ключових завдань для створення інноваційного інструменту, який об'єднає науку, освіту та технології для підтримки взаємодії людини з природним середовищем.

Завдання дослідження містить в собі:

1)Аналіз існуючих веб-сервісів парків;

2)Розробка вимог та проектування сервісу на основі аналізу;

3)Безпосередня розробка веб-сервісу, який має бути реалізованим у вигляді сайту;

4)Тестування

5)Впровадження веб-сервісу.

Об'єктом дослідження є реалізація веб-сервісів для парків, а **предметом** дослідження є - Парк Херсонського державного університету.

Методи дослідження для досягнення мети були використані з загальнонауковою прив’язкою (аналіз, синтез, порівняння), методи емпіричного дослідження (спостереження, експеримент) та аналізу літературних джерел. Аналіз - метод дозволяє детально вивчити потреби користувачів, конкурентний ринок та технічні можливості для визначення оптимальних рішень для веб-сервісу. Аналіз також враховує найновіші технологічні та дизайнерські тенденції. Застосування методу синтезу дозволяє об'єднати різноманітні аспекти функціоналу, інтерфейсу та технічних вимог, щоб створити цілісний та ефективний веб-сервіс. Порівняння як метод використовується для визначення конкурентних переваг веб-сервісу, порівнюючи його з аналогічними проектами. Це допомагає виділити унікальні особливості та позиціонувати сервіс ефективно на ринку. Метод спостереження використовується для аналізу поведінки користувачів, їхніх потреб та реакцій на функціонал веб-сервісу, що є ключовим для вдосконалення його використання. Застосування експериментального методу дозволяє перевірити ефективність конкретних елементів або функціоналу веб-сервісу та внести необхідні корективи. Аналіз літературних джерел важливий для оцінки наявних технологій, кращих практик та актуальних тенденцій у розробці веб-сервісів.

Структура роботи складається з змісту, вступу, першого розділу, другого розділу, висновку та використаних джерел інформації.

Практичне значення результатів. Результати дослідження будуть мати безпосереднє практичне застосування в контексті розробки та впровадження веб-

сервісу "Парк Херсонського державного університету". Отримані в ході роботи дані та рекомендації відкривають широкі можливості для реалізації інформаційного інструменту, спрямованого на взаємодію громадськості з природним резерватом.

Усе це робить тему дослідження актуальною та перспективною для розвитку інформаційних технологій у сфері парків та екології.

РОЗДІЛ 1
АНАЛІЗ ІСНУЮЧИХ ПАРКІВ ТА ПРОЕКТУВАННЯ

1.1. Аналіз найкращих парків та ботанічних садів світу на основі їх веб-сервісів

У сучасному світі, завдяки стрімкому розвитку інформаційних технологій, веб-сервіси стали невідкладною складовою сфери розваг та культурно-освітніх заходів. Зокрема, парки та ботанічні сади, як символи природної краси та біорізноманіття, удосконалюють свою інфраструктуру, використовуючи новітні технології для створення веб-сервісів, що покликані полегшити взаємодію з відвідувачами.

Дане дослідження присвячена глибокому вивченню та порівнянню веб-сервісів, які впроваджуються у найпрестижніших природних об'єктах світу. Дослідження охоплює аспекти розробки, функціоналу та технологічної інноваційності, які роблять ці сервіси ефективними і привабливими для відвідувачів.

Проведений аналіз спрямований на виявлення та висвітлення кращих практик у галузі використання веб-технологій у парках та ботанічних садах, а також на визначення можливостей для подальшого вдосконалення і розвитку. Висновки дослідження мають на меті сприяти вдосконаленню веб-сервісів природних об'єктів, забезпечуючи їхню актуальність та конкурентоспроможність у сучасному інформаційному середовищі.

Провівши аналіз найпопулярніших парків різних категорій та ботанічних садів, можна виділити, що веб-сервіси парків та ботанічних садів стали ключовим елементом взаємодії між природними об'єктами та їхніми відвідувачами, забезпечуючи не лише інформаційний, але й віртуальний досвід. Ці сервіси прийшли на заміну традиційним методам інформаційної підтримки, надаючи відвідувачам зручний інструментарій для ознайомлення з природною спадщиною та активного взаємодії з нею.

Веб-сервіси парків і ботанічних садів можуть приймати різні форми, включаючи офіційні веб-сайти, мобільні додатки, інтерактивні карті, аудіо та відеогіди, веб-камери та інші інноваційні рішення. Ці сервіси часто впроваджують сучасні технології, такі як розширена реальність (AR), віртуальна реальність (VR), аудіогіди, QR-коди та інші елементи, щоб розширити можливості взаємодії відвідувачів з природним оточенням.

Інформаційні системи, які вбудовані у веб-сервіси парків та ботанічних садів, включають різноманітні дані про ландшафт, біологічні види, історію та культурну спадщину. Вони також можуть містити практичні ресурси, такі як карти маршрутів, розклади заходів, інформацію про послуги та інше. Ці системи допомагають створити зручне та інформативне середовище для відвідувачів, покращуючи їхній загальний досвід та сприяють утриманню природних об'єктів на високому рівні.

Рис 1. - Гранд-парк Чикаго.

Також аналіз цих сервісів дав можливість виокремити та виділити пункти які є модулями, що містять ці сервіси, і дозволяють максимально точно налагодити взаємодію з користувачем. Кожна інформаційна система несе собою мету дати користувача необхідну кількість інформації та даних, що максимально заохочує їх до дослідження даного природного об’єкту. Це може дати як певні емоції, так і нагадування тобто спогади, якщо користувач відвідував даний парк. З етичної точки зору це може дати користувачу незабутні враження. Модулі сервісів:

1. Головна сторінка: Вітаючий текст: Ласкаво запрошуємо в парк або ботанічний сад. Візуальний матеріал: Зображення основних ландшафтів та рослинного світу.

Навігаційне меню: Зручний доступ до ключових розділів.

2. Про нас: Історія: Огляд історії та розвитку парку або ботанічного саду. Місія та цінності: Визначення основних принципів та цілей управління.

3. Експозиції та Зони: Віртуальні Тури: Відображення основних експозицій через віртуальні тури або фотогалереї. Карта Об'єкту: Інтерактивна карта з відміченням ключових точок і об'єктів.

4. Розклади та Події: Події та Виставки: Актуальна інформація про плановані події та виставки. Розклад Роботи: Графік роботи, включаючи вихідні та святкові дні.

5. Відвідувачам: Практична Інформація: Поради для відвідувачів, включаючи правила поведінки та безпеки. Квитки та Абонементи: Інформація про ціни та умови відвідування.

6. Освітні та Навчальні Ресурси: Освітні Програми: Пропозиції для шкіл, студентів та групових екскурсій. Вчительські Ресурси: Матеріали для вчителів та педагогічних заходів.

7. Контакти та Зворотній Зв'язок: Адреса та Телефони: Офіційні контактні дані для зв'язку. Форма Зворотного Зв'язку: Зручний спосіб висловити питання чи залишити відгук.

8. Спільнота та Соціальні Мережі: Посилання на Соціальні Мережі: Активність у Facebook, Instagram, Twitter тощо. Блог або Новини: Огляди подій та цікаві факти.

9. Технологічні Інновації: Використані Технології: Опис інновацій, таких як AR, VR, мобільні додатки тощо. Веб-камери та reallife-трансляції: Якщо доступно, зображення в реальному часі.

10. Партнерство та Співпраця: Партнери та Спонсори: Інформація про компанії чи організації, що підтримують парк або сад. Можливості Співпраці: Інформація для бізнесу чи потенційних партнерів.

Ці елементи дозволяють веб-сайту парку або ботанічного саду надавати збалансовану та повну інформацію для відвідувачів, забезпечуючи їм зручний та цікавий досвід.

Рис 2. - Схема дендропарку ‘Софіївка’.

При порівнянні технологічних аспектів веб-сайтів парків та ботанічних садів, можна звернутися до кількох ключових категорій. Головні елементи для аналізу включають візуальний дизайн та інтерфейс, інтерактивність та віртуальні можливості, освітні ресурси та мультимедіа, навігацію та карти, використання мобільних додатків, актуальність та обрані технології, забезпечення безпеки та приватності, співробітництво та соціальні засоби, аналітику та відстеження відвідувань, інтеракцію з відвідувачами, технічну підтримку, швидкість та витрати ресурсів, підтримку різних мов тощо. Усі ці аспекти можна порівнювати в двох теоретичних половинах: наявність у сервісі та спосіб реалізації з технічної точки зору, наприклад яким способом реалізовано карту парку.

Порівняння цих категорій дозволяє визначити різниці в технологічних рішеннях та функціональності між веб-сайтами різних природних об'єктів. Основна мета - визначити, як веб-сервіси використовують сучасні технології для поліпшення взаємодії з відвідувачами та забезпечення їм якісного та інноваційного досвіду взаємодії з природним навколишнім середовищем. Для порівняння буде використано таблиці та

додаток What Runs, який дозволяє переглядати технічну інформацію конкретного сервісу.

Назва	Analytics	Web Framework or Language	Web Server	Javascript Frameworks	CDN	Dev Tools	Карти
Central Park, Нью-Йорк, США	Google Analytics UA	Bootstrap	Resin 3.1.8	jQuery 1.10.2 Respond JS	CloudFlare CDN JS	HTML5 Shiv	Присутня інтерактивна карта парку з цікавими місцями на основі Google Maps
Yellowstone National Park, США	Google Analytics UA	Bootstrap	Apache 2.4.6	Modernizr 2.6.2 jQuery 1.12 Respond JS	Amazon Cloudfront	HTML5 Shiv	Інтерактивна карта заснована на технологіях Esri та Mapbox
Grand Canyon National Park, США	Google Analytics UA	Bootstrap	Apache 2.4.6	Modernizr 2.6.2 jQuery 1.12 Respond JS	Amazon Cloudfront	HTML5 Shiv	Інтерактивна карта заснована на технологіях Esri та Mapbox
Hakone Gora Park, Японія	Google Analytics UA	PHP	Apache 2.4.33	jQuery 1.11.0	-	HTML5 Shiv	Карта не інтерактивна, зроблена у вигляді скріншотів
Everglades National Park, США	Google Analytics UA	Bootstrap	Apache 2.4.6	Modernizr 2.6.2 jQuery 1.12 Respond JS	Amazon Cloudfront	HTML5 Shiv	Інтерактивна карта заснована на технологіях Esri та Mapbox
Yosemite National Park, США	Google Analytics UA	Bootstrap	Apache 2.4.6	Modernizr 2.6.2 jQuery 1.12 Respond JS	Amazon Cloudfront	HTML5 Shiv	Інтерактивна карта заснована на технологіях Esri та Mapbox
Kruger National Park, ПАР	Google Analytics UA Infer	Bootstrap	Apache 2.4.6	jQuery 1.12.4 jQuery UI	-	HTML5 Shiv	Присутня інтерактивна карта парку з цікавими місцями на основі карт Google і даних від компанії AfriGIS
Royal National Park, Австралія	Google Analytics UA Quantcast Infer	PHP	IIS 10.0	Modernizr 2.7.1 jQuery 1.11.0 jQuery UI 1.11.0 Angular JS Respond JS JSON 3	-	HTML5 Shiv	Власна інтерактивна карта заснована на технологіях карт Google

Kew Gardens, Велика Британія	Google Analytics UA Hotjar NewRelic	Drupal 8	Nginx 1.11.8	Hammer.js jQuery 3.3.1 Async	-	HTML5 Shiv	Інтерактив і карта в часності відсутні
Singapore Botanic Gardens, Сінгапур	Google Analytics UA SiteCatalyst	PHP	Apache 2.4.6	Head JS Modernizr 2.6.2 jQuery 1.9.0 jQuery UI	Amazon Cloudfront	HTML5 Shiv	Інтерактив відсутній. Карти у вигляді завантажуваного файлу
New York Botanical Garden, США	Google Analytics	Bootstrap	Apache 2.4.39	jQuery 1.11.1 embed JS jQuery Instagram	-	HTML5 Shiv	Власна розроблена інтерактивна карта без застосувань сторонніх сервісів
Royal Botanic Garden Edinburgh, Велика Британія	Google Analytics	PHP	Apache 2.4.6	Lo-dash Modernizr Moment JS TweenMax jQuery 1.11.2 Vue JS 2.5.16 Scrollmagic GSAP	CloudFlare	Polyfill	Відсутня інтерактивна карта, посилання на Google Maps
Jardin des Plantes, Франція	Google Analytics	PHP	Nginx	Modernizr jQuery 1.7	-	-	Інтерактив та карта відсутні
Missouri Botanical Garden, США	Google Analytics UA	PHP	Apache 2.4.6	jQuery 1.3.2 HoverIntent JS Minified	-	HTML5 Shiv	Інтерактивна карта відсутня використовується Google Maps
Melbourne Royal Botanic Gardens, Австралія	Google Analytics UA	Typekit	Apache	Modernizr 2.6.2 Moment JS jQuery 1.9.1 Respond JS	-	HTML5 Shiv	Присутня власна інтерактивна карта заснована на технологіях Google Maps
Botanic Gardens Conservation International (BGCI)	Google Analytics UA	PHP	Nginx 1.10.3	Hammer.js Modernizr TweenMax jQuery	-	HTML5 Shiv	Інтерактив у вигляді карти відсутній

На основі даних порівнянь можна створити діаграму, яка покаже перевагу певної технології.

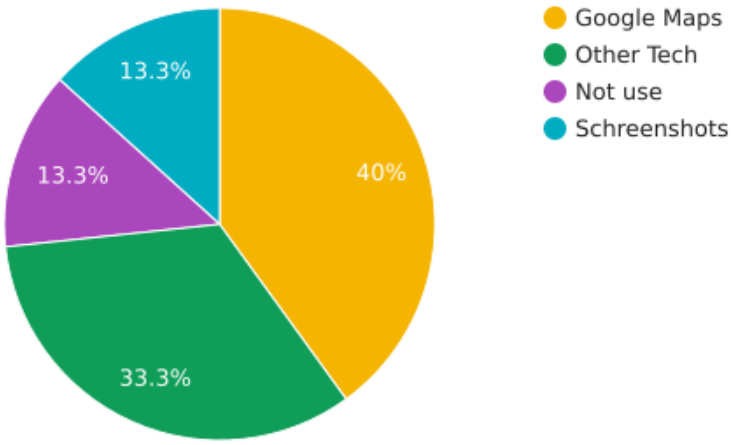

Рис 3. - Діаграма технологій інтерактивних карт.

Як можна побачити, найбільшу перевагу мають карти Google Maps, і це не дивно, адже Google в цьому та інших планах є світовим гігантом технологій, що мають зручну інтеграцію та зрозумілий інтерфейс для взаємодії, тому карти від Google серед порівняних парків та ботанічних садів займають перше місце серед кількості карт заснованих на них.

1.2. Технологічні аспекти розробки веб-сервісів для парків

Мобільна Адаптація. Створення веб-сайту для парку повинно включати мобільну адаптацію, щоб забезпечити зручний та ефективний доступ до інформації на різних пристроях. Респонсивний дизайн вирішує цю проблему, дозволяючи веб-сайту автоматично адаптуватися до розмірів екранів, щоб надавати оптимальний вигляд та функціональність.

Технологія CSS Media Queries. Респонсивний дизайн реалізується за допомогою CSS Media Queries, які дозволяють визначати стилі для різних розмірів екрану. Це забезпечує оптимальне відображення контенту для різних пристроїв, включаючи комп'ютери, планшети та смартфони.

Mobile-First Розробка. Методологія Mobile-First передбачає спроектування та розробку веб-сервісу спочатку для мобільних пристроїв, а потім розширення функціоналу для більших екранів. Цей підхід гарантує, що веб-сайт оптимізований саме для користувачів з мобільними пристроями.

Інтерактивність та Візуалізація. Для залучення відвідувачів до вивчення парку важливо використовувати інтерактивні елементи та візуалізацію. Інтерактивні карти, що дозволяють відвідувачам легко знаходити об'єкти та місця, а також розгалужені галереї та відеоматеріали, створюють захоплюючий досвід та стимулюють інтерес.

JavaScript та API для Карт. Використання JavaScript-бібліотек, таких як Google Maps API або Leaflet, дозволяє створювати інтерактивні карти парку. Вони надають можливість взаємодії, маршрутизації та отримання інформації про конкретні об'єкти на карті.

WebGL та Three.js для створення складних тривимірних візуалізацій можна використовувати технології WebGL разом із бібліотекою Three.js. Це дозволяє створювати вражаючі інтерактивні візуальні ефекти та об'єкти.

AJAX та Інтерактивні Елементи. Використання технології AJAX для динамічного завантаження контенту та використання інтерактивних елементів, таких як вкладки, слайдери або плавні анімації, покращує користувацький досвід та забезпечує активність взаємодії.

Онлайн-Бронювання та Квитки. Розробка системи онлайн-бронювання дозволяє відвідувачам парку планувати свій візит, заздалегідь резервуючи екскурсії чи участь у подіях. Це полегшує організацію та підвищує зручність відвідування, а також сприяє ефективному використанню ресурсів парку.

Backend Системи Бронювання: Використання мов програмування, таких як Python, Node.js чи PHP, для розробки backend-частини системи бронювання. Backend відповідає за обробку та зберігання даних бронювань, а також взаємодію з базою даних.

Frontend Компоненти: Використання фреймворків, таких як React або Angular, для розробки frontend-частини системи. Це забезпечує швидке та ефективне відображення інтерфейсу для користувачів, які бронюють квитки чи послуги онлайн.

Безпека та Захист Даних. Забезпечення безпеки та захисту особистих даних відвідувачів – це важлива складова розробки веб-сайту для парку. Використання SSL-шифрування гарантує, що дані передаються у безпечному форматі, а заходи проти вторгнень допомагають убезпечити веб-сайт від потенційних загроз.

SSL/TLS Протоколи: Використання останніх версій протоколів SSL/TLS (наприклад, TLS 1.2 або TLS 1.3) для забезпечення ефективного та безпечного шифрування даних між веб-сайтом та користувачами.

Сертифікати Безпеки. Отримання та використання SSL-сертифікатів від відомих центрів сертифікації для підтвердження автентичності веб-сайту та забезпечення безпеки обміну даними.

SEO-Оптимізація. SEO-оптимізація грає ключову роль у просуванні веб-сайту в пошукових системах. Використання правильних ключових слів та оптимізованих метатегів, а також підтримка високої швидкості завантаження, сприяє покращенню рейтингу та видимості веб-сайту для потенційних відвідувачів.

SEO-Інструменти. Використання SEO-інструментів, таких як Google Keyword Planner або SEMrush, для вибору ефективних ключових слів та оптимізації метатегів.

Структуровані Дані. Додавання структурованих даних на веб-сайт для полегшення індексації пошуковими системами та відображення додаткової інформації у виведенні пошуку.

Соціальна Інтеграція. Спільне використання контенту в соцмережах є ефективним засобом розповсюдження інформації та привертання нових відвідувачів. Включення кнопок та можливостей поділу на популярних платформах створює можливість для відвідувачів долучати друзів та ділитися своїми враженнями.

Соціальні Плагіни: Використання соціальних плагінів або API для інтеграції з популярними соцмережами. Це дозволяє користувачам легко ділитися контентом на своїх сторінках.

Спільні Медіа-Елементи: Додавання кнопок "Поділитися" та віджетів соцмереж до мультимедійних елементів, таких як фотографії чи відео, для стимулювання обміну враженнями серед відвідувачів.

Аналітика та Відстеження. Використання аналітичних інструментів, таких як Google Analytics, дозволяє веб-сайту для парку відстежувати та аналізувати поведінку відвідувачів. Це надає важливі дані щодо ефективності контенту, джерел трафіку та інших метрик, які можуть використовуватися для подальшої оптимізації та удосконалення веб-сервісу.

Google Analytics. Користування сервісом Google Analytics для збору та аналізу різних метрик, таких як кількість відвідувачів, джерела трафіку та поведінка користувачів.

Налаштування цілей. Визначення цілей в Google Analytics для відстеження конкретних дій відвідувачів, таких як завантаження квитків або реєстрація на подію.

Інтеграція з Системами Бронювання та Касовими Системами. Розробка API для інтеграції з системами бронювання та касовими системами робить процес бронювання та оплати ще більш зручним для відвідувачів. Ця інтеграція сприяє автоматизації та оптимізації обліку та обробки платежів, покращуючи загальний досвід користувачів. Розробка API. Створення API для взаємодії з системами бронювання та касовими системами. Це дозволяє автоматизувати процеси обробки замовлень та обліку платежів. Забезпечення безпеки та конфіденційності даних в процесі інтеграції за допомогою шифрування та заходів безпеки на рівні API.

Ці технологічні аспекти розробки веб-сайту для парку покликані забезпечити високий рівень зручності, безпеки та ефективності користування. Їх інтеграція сприяє створенню вражаючого та функціонального веб-середовища для відвідувачів парку, забезпечуючи незабутній досвід та задоволення від взаємодії. Об'єднуючи їх, розробка веб-сервісу парку стає не лише інформативною платформою, але і інтерактивним досвідом для відвідувачів. Врахування зручностей, безпеки та інтерактивності дозволяє створити сервіс, який відповідає сучасним стандартам та вимогам.

1.3. Розробка вимог

Функціональні та нефункціональні вимоги є важливою частиною процесу розробки програмного забезпечення.

Функціональні вимоги включають опис функціональності, введення та виведення даних, а також поведінку системи.

Нефункціональні вимоги включають продуктивність (вимоги до швидкодії та відповіді системи), надійність (вимоги до стійкості до помилок), безпеку (вимоги до захисту інформації), сумісність (можливість взаємодії з іншими системами) та масштабованість (здатність адаптуватися до збільшення обсягу даних чи користувачів).

Ці вимоги розроблюються для забезпечення високої якості та придатності системи до використання, а також для відповідності розробленого ПЗ очікуванням користувачів та бізнес-вимогам. Розробка програмного забезпечення включає в себе увесь цей комплекс вимог для створення ефективної та надійної системи.

Функціональні Вимоги.

Керування рослинами: Додавання рослин експертами з вказанням основних характеристик, таких як назва, клас, родина, опис тощо. Можливість завантаження та прикріплення фотографій для ідентифікації рослин.

Карта Парку: Інтерактивна карта парку та ботанічного саду

Пошук та Фільтрація: Можливість швидкого пошуку рослин за їхніми характеристиками або назвою. Фільтрація та категоризація рослин за класифікаційними ознаками.

Інформаційні сторінки: Індивідуальні інформаційні сторінки для кожної рослини з детальною інформацією та фотографіями.

Нефункціональні Вимоги.

Швидкодія: Висока швидкість завантаження сторінок та ефективність обробки запитів.

Сумісність: Підтримка різних веб-браузерів для забезпечення доступу користувачам з різних платформ.

Надійність: Забезпечення стабільної роботи системи та уникнення втрати даних при можливих збіях.

1.4. Прототипування сервісу

Прототипування веб-сайту є невід'ємною частиною сучасної веб-розробки. Це процес створення спрощених моделей майбутнього сайту, які допомагають візуалізувати його структуру, функціональність та дизайн до початку розробки коду. Прототипи можуть бути статичними або інтерактивними, варіюватися від простих ескізів до складних симуляцій користувацького інтерфейсу.

Переваги прототипування це виявлення та усунення проблем на ранніх етапах, адже воно дозволяє виявити та виправити проблеми з використанням, дизайном та логікою сайту до того, як буде витрачено значні ресурси на розробку. Прототипи слугують візуальною мовою, яка полегшує спілкування між замовниками, дизайнерами та розробниками, забезпечуючи чітке розуміння бачення кінцевого продукту. Раннє виявлення та виправлення проблем може значно скоротити витрати на розробку, адже зміни коду на пізніх етапах проекту є значно дорожчими. Залучення клієнтів до процесу прототипування дає їм можливість висловити свої думки та побажання, що веде до кращого розуміння їхніх потреб та, як наслідок, до вищого рівня задоволеності кінцевим продуктом.

Види прототипів

Ескізи: Прості, намальовані від руки креслення, які візуалізують основну структуру та компонування сайту.

Макет: Цифрові зображення, які деталізують візуальний дизайн сайту, включаючи розташування елементів, кольори та шрифти.

Інтерактивний прототип: Симуляція реального сайту, яка дозволяє користувачам взаємодіяти з елементами інтерфейсу та бачити, як сайт буде функціонувати.

Інструменти для прототипування

Існує безліч інструментів для прототипування, які варіюються за складністю та функціональністю. Деякі з популярних варіантів включають:

Adobe XD: Комплексний інструмент для створення прототипів та дизайну інтерфейсів.

Axure RP: Чудовий та безкоштовний інструмент для прототипування дизайну і сайту в цілому.

Figma: Безкоштовний онлайн-інструмент для прототипування та дизайну інтерфейсів.

Sketch: Популярний інструмент для дизайну інтерфейсів з можливостями прототипування.

InVision: Інструмент для створення інтерактивних прототипів з широкими можливостями співпраці.

Proto.io: Хмарний інструмент для створення прототипів з акцентом на простоті використання.

Процес прототипування

Процес прототипування складається з наступних етапів.

Визначення цілей. Треба визначити цілі прототипування та те, що ви бажано досягти.

Створення користувацьких сценаріїв. Опис, як користувачі будуть взаємодіяти з сайтом.

Розробка ескізів. Створення простих ескізів, щоб візуалізувати структуру та компонування сайту.

Створення макета. Перетворення ескізи на цифрові зображення з деталізацією візуального дизайну.

Для прототипування веб-сервісу Парк Херсонського державного університету було обрано такий інструмент як Axure RP. Axure RP (Axure Rapid Prototyping) є одним з найбільш популярних інструментів для створення прототипів та специфікацій веб-сайтів і додатків. Він дозволяє дизайнерам та розробникам проєктувати високоякісні, інтерактивні прототипи, які допомагають візуалізувати та оцінити функціональність, юзабіліті та дизайн майбутнього продукту.

Axure RP пропонує зручний для користувача інтерфейс drag-and-drop, що дозволяє легко створювати прототипи без необхідності написання коду. Програма включає бібліотеку готових компонентів інтерфейсу, таких як кнопки, форми, таблиці, тощо. Це дозволяє швидко створювати прототипи без необхідності малювати кожен елемент з нуля. Axure RP дозволяє додавати інтерактивність до прототипів, щоб користувачі могли взаємодіяти з елементами та бачити, як сайт чи додаток буде функціонувати. Програма дозволяє легко створювати зв'язки між різними сторінками прототипу, що забезпечує реалістичний досвід користувача. Axure RP може збирати дані користувачів під час тестування прототипу, наприклад, кліки мишею чи введені значення форм. Це допомагає оцінити юзабіліті прототипу та зробити його більш зручним для користувачів. Також дозволяє генерувати документацію з прототипу, що включає описи сторінок, компонентів та взаємодій. Це корисно для розробників, оскільки допомагає їм зрозуміти вимоги до проєкту.

Після вибору інструменту в дослідженні було почато процес прототипування, який як зазначено вище починається з визначення цілей. **Основна ціль** цього прототипування це облегшити розробку веб-сервісу, створенням інтуїтивного макету.

Це надасть зрозумілості та визначеності розробці, та в десятки разів пришвидшить написання коду.

Наступним етаном стане створення **користувацьких сценаріїв**, що дозволить передбачити сценарії взаємодії користувачів з сайтом, і на основі цих сценаріїв розробити сервіс таким чином, щоб ці сценарії відбулися.

Особливість: Інтерактивна карта та інформація про рослини ботанічного саду університету

Опис: Функціонал для навігації та ознайомлення з рослинами у ботанічному саду університету через веб-сайт.

Основний користувач: Відвідувачі ботанічного саду та студенти університету.

Мета: Ознайомлення з рослинами, що ростуть у ботанічному саду, та навігація по його території.

Use Case 1: Пошук рослини за назвою

Користувач відвідує головну сторінку сайту.

Він обирає пункт меню "Усі рослини" або використовує пошукову функцію.

Користувач вводить назву рослини у поле пошуку та натискає кнопку "Пошук".

Система відображає результати пошуку з рослинами, які відповідають введеному запиту.

Користувач обирає потрібну рослину та переходить на її окрему сторінку для подальшого вивчення.

Use Case 2: Перегляд карти ботанічного саду

Користувач переходить на сторінку з картами ботанічного саду.

Він вивчає карту, розглядаючи розташування різних рослинних колекцій та атракцій у ботанічному саду.

Користувач обирає цікаву локацію та переходить до неї, щоб дізнатися більше про рослини, які там ростуть.

Use Case 3: Перегляд детальної інформації про рослину

Користувач використовує фільтри на сторінці "Усі рослини" для пошуку певного виду рослини або категорії.

Після знаходження цікавої рослини, він переходить на її окрему сторінку.

На сторінці рослини користувач переглядає фотографії, опис та характеристики цієї рослини для отримання детальної інформації.

Наступним і останнім етапом прототипування виключно є розробка ескізів та самого макету сайту. Для розробки ескізів буде використано звичайний Paint, а макету вже підібраний Axure RP. Буде продемонстровано ескіз та макет сторінки з рослинами.

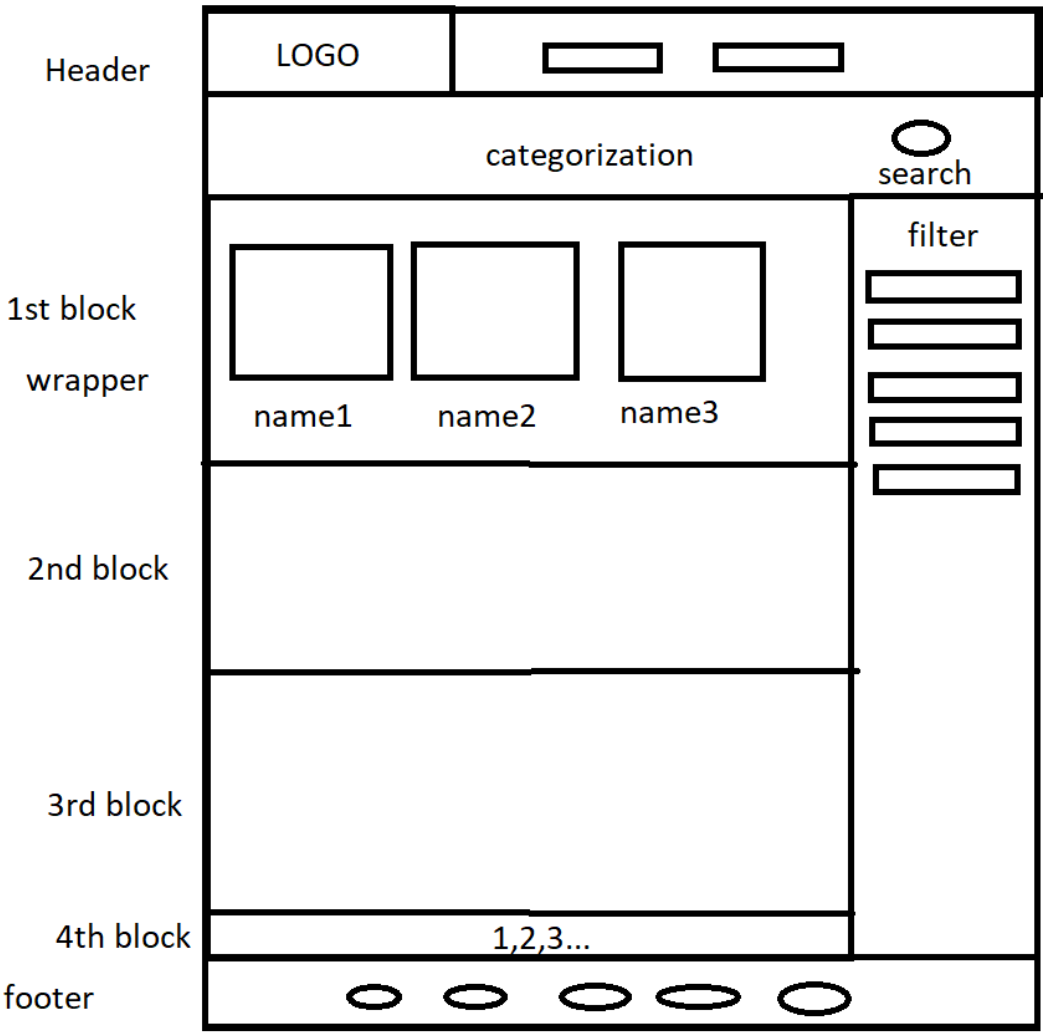

Рис 4.1 – Ескіз сторінки з рослинами у Paint.

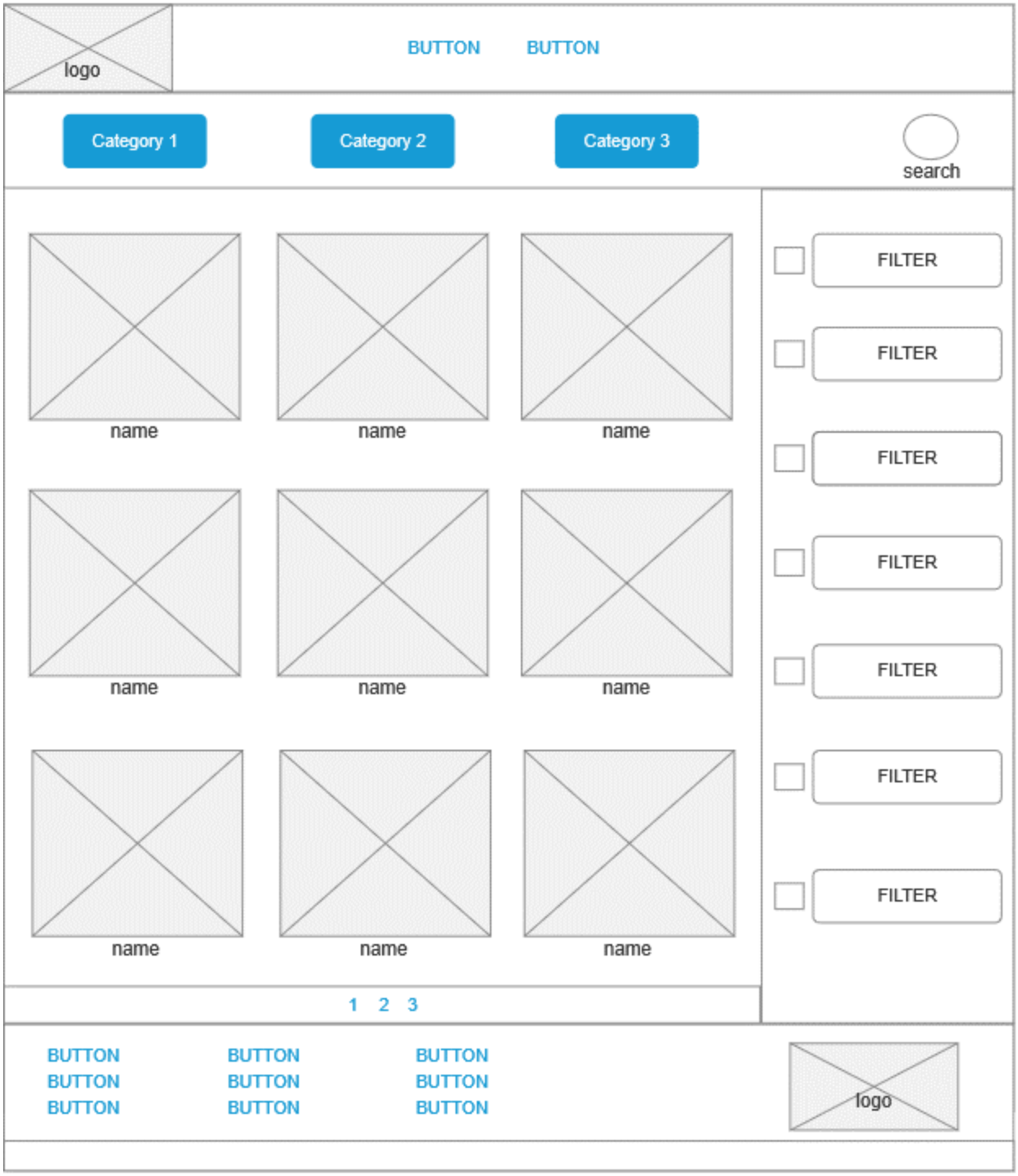

Рис 4.2 – Макет сторінки з рослинами у Axure RP.

1.5. Вибір технологій для розробки

Вибір технологій для розробки це ключовий момент, адже від цього залежить як швидкість, так і якість готового продукту. Тому в даному дослідженні, при певному аналізі існуючих технологій веб-розробки було надано перевагу наступним.

Для клієнтської частини веб-сервісу, тобто дизайну і інтерактиву були використані HTML, CSS, JavaScript. Це базові технології, які притаманні будь-якій розробці. HTML відповідає за розмітку елементів на сторінці, CSS це мова стилів, відповідає за надання певного стилю HTML елементам. Мова програмування JavaScript в даному контексті використовується для надання “життя” сайту. Додавання анімацій, плавні переходи, інтерактив, це все Java.

Для серверної частини, тобто бази даних та функціоналу сайту на серверному рівні були використані наступні можливості. База даних MySQL. MySQL - це безкоштовна система керування реляційними базами даних (СУБД), яка широко використовується для організації доступу до інформації та її управління. Вона пропонує широкий спектр функцій, що робить її популярним вибором як для приватних осіб, так і для підприємств. Переваги використання саме цієї бази даних:

Відкритий код. Програмне забезпечення з відкритим кодом, що дає вам свободу використовувати, модифікувати та розповсюджувати його безкоштовно. Це робить її доступним рішенням для користувачів з будь-яким бюджетом.

Висока продуктивність. Ця база даних відома своєю швидкістю та здатністю обробляти великі обсяги даних. Це робить її ідеальним вибором для веб-сайтів та програм, які потребують швидкого доступу до інформації.

Простота використання. Порівняно з деякими іншими СУБД, MySQL відносно проста у вивченні та використанні. Це робить її доступною як для досвідчених розробників, так і для початківців.

За обробку запитів що будуть передаватися на сервер, та за весь функціонал буде відповідати мова програмування PHP. PHP - це універсальна мова програмування, спеціально створена для веб-розробки. Вона дозволяє створювати інтерактивні та динамічні веб-сторінки, які реагують на дії користувачів і можуть обробляти дані. Синтаксис PHP схожий на інші популярні мови програмування, такі як C, Java та JavaScript, що робить його легким для вивчення, особливо для розробників з досвідом.

PHP пропонує широкий набір функцій для роботи з базами даних, формами, сесіями, файлами, зображеннями та іншим, що дозволяє створювати складні веб-застосунки. Також було обрано фреймворк для роботи, а саме Laravel.

Laravel - це відкритий фреймворк для розробки веб-додатків на мові програмування PHP. Він був створений Тейлором Отвеллом і випущений в 2011 році. Laravel досить популярний серед розробників завдяки своїй простоті, зручності та багатофункціональності. Він пропонує чистий і елегантний синтаксис, має потужну систему маршрутизації, вбудований ORM під назвою Eloquent, шаблонний двигун Blade для створення виглядів, систему міграцій та сідів для керування базою даних, вбудовану систему автентифікації та авторизації, широкий вибір офіційних та спільнотних пакетів для інтеграції з різними сервісами, засоби для тестування, Artisan для автоматизації рутинних завдань, активну спільноту розробників і велику базу документації.

РОЗДІЛ 2
РОЗРОБКА ТА ВПРОВАДЖЕННЯ ВЕБ-СЕРВІСУ

2.1.Розбиття розробки на етапи

Після розробки вимог до майбутнього сервісу, його прототипування та визначення технологій, кроком до реалізації веб-сервісу стає безпосередньо початок розробки. Розбиття на етапи дає певну концепцію, дотримуючись якої розробка стає структурованою, зрозумілою, і не має виникати складнощів. Кожен етап був описаний детально.

1.Першим етапом є підготовка бази даних для подальшої роботи. Він включає в себе ознайомлення з базовим функціоналом обраної бази даних, у випадку даного дослідження це MySQL. Після чого починається процес створення самої бази даних та її таблиць. Для більш зрозумілішого концептуального розуміння бази даних необхідно розробити її проєкт. Даний проєкт має містити в собі опис усіх сутностей та моделей, що містить БД. Також має бути візуальне представлення з відображенням усіх залежностей та загальної моделі бази даних.

2. Другим етапом є розробка клієнтської частини або front-end. Цей етап включає в себе створення скелету сайту, його розмітки за допомогою HTML. Розмітка являє собою набір елементів які розкладаються по блокам, адже майже усі елементи є блочного типу. Можна провести асоціацію з будівництвом, де цегла почергово складається одна на одну. Дана аналогія чудово демонструє процес розмітки. Після того як скелет сайту створений, його потрібно стилізувати, задати палітри кольорів, змінити форми об’єктів і ще багато чого іншого. За даний функціонал відповідає CSS, який інтегрується у HTML файл і здійснює перетворення елементів та об’єктів. Після того як скелет сайту сформовано та задано основні стилі елементам починається робота з функціоналом, який відповідає за взаємодію користувача, та елементів цього сайту. JavaScript дозволяє додати певні властивості елементам сайту, що створює нові сценарії їх поведінки. Наприклад анімацію появи елементів при заході на певну сторінку, плавну модель поведінки. Також дозволяє інтегрувати динамічну поведінку об’єктів у front-end середовищі, перемикання сторінок і багато чого іншого.

3. Третій етап являє собою роботу з серверною частиною веб-сервісу, а саме розробкою цієї частини. На даному етапі клієнтський функціонал підв’язується до серверного, що дає певну реакцію на взаємодію. Тобто при взаємодії користувача з сайтом, починають виконуватися певні серверні запити, що дають можливість виводити інформацію з баз даних. Ця інформація може бути інтегрована будь-яким чином. Даний функціонал реалізується на серверних мовах програмування у вигляді певних скриптів або сценаріїв.

4. Після розробки клієнтської та серверної частини сервісу починається тестування. Цей процес є одним з найважливіших у розробці, адже дає можливість виявити помилки та недоліки продукту перед його випуском. Тестування може охоплювати як глобально увесь веб-сервіс, так і його конкретні функціональності.

Основні етапи тестування:

1.Тестування функціоналу – перевірка усіх функціональних можливостей що надає сайт. Перевірка взаємодії з усіма об’єктами що присутні на сервісі, як вони реагують на ту чи іншу дію.

2.Тестування сумісності – відповідає за перевірку роботи сайту на різних браузерах.

3.Тестування сценаріїв – перевірка конкретних сценаріїв та Use Case використання.

5.Останнім етапом розробки після тестування є безпосереднє впровадження сервісу. Сайт переводиться з dev. у prod. режим і розгортається на сервері, отримує власний домен та адресу, що відкриває можливість іншим користувачам взаємодіяти з ним.

2.2.Проєктування бази даних

В даному досліджені, проєктування бази даних було почато з опису усіх моделей. В даному випадку база даних має дві моделі, рослини і їх характеристики. Модель Рослини відповідає за наповнення короткими відомостями про рослини що буде додана на сайт, такими як ім’я, опис, тип, фото і т.д., а модель Характеристики відповідає за вміст певних характеристик кожної рослини, яких дуже велика кількість.

Модель Рослини

Назва поля	Тип поля	Опис	Обов’язково	Приклад
name	varchar	Назва рослини	Так	Дуб звичайний
description	text	Інформація про рослину	Так	Велике дерево і гарне
type	varchar	Вид рослин	Так	Дерево
photo	varchar		Так	/tree/image.jpg
data	date	Дата додавання в базу даних	Так	18.12.2023
link	varchar	Посилання на Wiki для детального ознайомлення	Ні	link

Таблиця 1.1 – Модель Рослини

Модель Характеристики

Назва поля	Тип поля	Опис	Обов'язково	Приклад
plant_id	int	FOREIGNKEY, має зв'язок з id рослини який ставить реакцію на публікацію	Так	223
edgetype	varchar	форма краю листка (наприклад, цілий, зубчастий, хвилястий).	Так	Цілий
veining	varchar	розташування та розгалуження жилок на листку (сітчасте, паралельне, пальчасте).	Так	Сітчасте
phyllotaxis	varchar	розташування листків на стеблі (протилегле, чергове, спіральне).	Так	Протилегле
stipules	varchar	маленькі листочки біля основи черешка листка (їх наявність або відсутність, форма, розмір)	Так	Наявні
pubescence	varchar	тип покриття листка волосками (голе, опушене, повстисте).	Так	Опушене
sex	varchar	роздільностатеві або двостатеві квітки (чоловічі, жіночі)	Так	Чоловічі
symmetry	varchar	форма квітки (актиноморфна – симетрична навколо осі, зигоморфна – симетрична лише в одній площині)	Так	Актиноморфна
pollination	varchar	спосіб перенесення пилку з тичинок на маточку (вітром, комахами, самозапилення).	Ні	Вітром
shape	varchar	тип плоду (ягода, кістянка, яблуко,	Так	Яблуко

		горіх, стручок тощо)		
size	varchar	розмір плоду	Ні	Великий
color	varchar	Колір плоду	Ні	Червоний

Таблиця 1.2 – Модель Характеристик

Для характеристик можна додати наступні описи.

тип краю: форма краю листка (наприклад, цілий, зубчастий, хвилястий).

жилування: розташування та розгалуження жилок на листку (наприклад, сітчасте, паралельне, пальчасте).

розташування: розташування листків на стеблі (наприклад, протилегле, чергове, спіральне).

прилистники: маленькі листочки біля основи черешка листка (їх наявність або відсутність, форма, розмір).

опушення: тип покриття листка волосками (наприклад, голе, опушене, повстисте).

2. Деталі квітки:

стать: роздільностатеві або двостатеві квітки (чоловічі, жіночі).

симетрія: форма квітки (актиноморфна - симетрична навколо осі, зигоморфна - симетрична лише в одній площині).

кількість пелюсток: кількість пелюсток у квітці.

кількість тичинок: кількість тичинок (чоловічих репродуктивних органів) у квітці.

кількість маточок: кількість маточок (жіночих репродуктивних органів) у квітці.

запилення: спосіб перенесення пилку з тичинок на маточку (вітром, комахами, самозапилення).

3. Деталі плоду:

форма: тип плоду (ягода, кістянка, яблуко, горіх, стручок тощо).

розмір: розмір плоду.

колір: колір плоду.

наявність насіння: чи є в плоді насіння.

спосіб поширення: як плоди поширюються в природі (вітром, тваринами, водою).

4. Сезонні характеристики:

період вегетації: час року, коли рослина активно росте.

період спокою: час року, коли рослина перебуває в стані спокою.

осінні характеристики: зміни, що відбуваються з рослиною восени (зміна кольору листя, опадання листя).

5. Екологічні характеристики:

тип ґрунту: тип ґрунту, в якому росте рослина (кислий, лужний, піщаний, глинистий).

водний режим: потреба рослини у воді (вологолюбна, посухостійка, потребує доброго дренажу).

освітлення: потреба рослини у світлі (сонячне, півтінь, тінь).

стійкість до морозів: здатність рослини витримувати морози (зона зимостійкості).

6. Хімічний склад:

алкалоїди: хімічні сполуки, які можуть мати психоактивні та інші фармакологічні властивості.

ефірні олії: леткі сполуки, які надають рослині запах і можуть мати лікувальні властивості.

вітаміни: органічні сполуки, необхідні для здоров'я людини.

мінерали: неорганічні сполуки, необхідні для росту та розвитку рослин.

7. Господарське значення:

декоративне використання: вирощування рослини як декоративного елемента в садах, кімнатних умовах.

харчове використання: використання їстівних частин рослини в їжу.

лікарське використання: використання рослини для лікування захворювань.

інші використання: використання рослини в промисловості, ремеслах, наукових дослідженнях.

8. Походження та історія:

дикоростучий вид: рослина, яка росте в природі без втручання людини.
культурний вид: рослина, яка вирощується людиною (може мати різні сорти).
гібриди: рослини, отримані в результаті схрещування різних видів.

Далі при проєктуванні було описано зв’язки

№	Модель 1(M1)	Модель 2(M2)	Назва зв’язку для M1	Назва зв’язку для M2	Роль M1	Роль M2	Тип зв’язку
1	Рослини	Характеристики	Характеристики рослини	Надання характеристик рослині	Parent	Child	1 до багатьох

Таблиця 1.3 – Зв’язки

І останнім етапом було створено візуальну модель бази даних

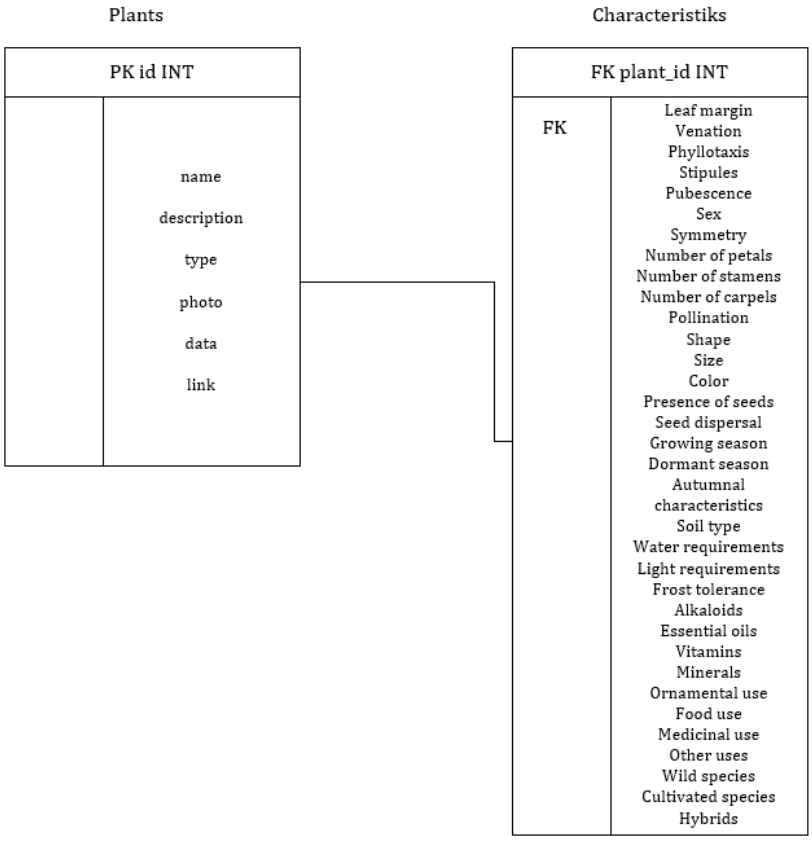

Рис 5. – Візуальна схеми бази даних сервісу

2.3. Розробка front-end частини

1. Дизайн сайту:

HTML: Почали зі створення загальної структури сайту, визначивши різні розділи як <header>, <nav>, <main>, <footer> та інші.

CSS: Використовуючи CSS, створили стиль, що відображається на всіх елементах сторінки, включаючи колір, розмір шрифту, розташування тощо.

Медіа-запити CSS: Для забезпечення адаптивності сайту використовували медіа-запити, які змінювали стиль в залежності від розміру екрану, забезпечуючи оптимальне відображення на різних пристроях.

2. Головна сторінка:

HTML: Розробили структуру головної сторінки, включаючи вступний матеріал та розділи для кнопок "Карти" та "Рослини".

CSS: Створили стилі для елементів головної сторінки, зокрема для хедера, кнопок та текстових блоків.

JavaScript: Додали JavaScript для обробки подій натискання на кнопки "Карти" та "Рослини", забезпечивши перехід на відповідні сторінки.

3. Сторінка "Карти":

HTML: Створили HTML-структуру для відображення карт ботанічного саду та парку Херсонського державного університету.

CSS: Використовуючи CSS, надали картам відповідний стиль, включаючи розмір, рамки тощо.

JavaScript: Додали інтерактивність за допомогою JavaScript, щоб забезпечити можливість переключення між картами та взаємодії з їхніми зонами при натисканні на них.

4. Сторінка "Рослини":

HTML: Розробили HTML-структуру для відображення категорій рослин, фільтрації та результатів пошуку.

CSS: Стилізували блоки з рослинами та пагінацію, використовуючи CSS для кращого відображення та зручності користувача.
JavaScript: Додали JavaScript для реалізації функціоналу фільтрації рослин за характеристиками, пошуку та виведення результатів на сторінці.
5. Детальна інформація про рослини:
HTML: Розробили HTML-структуру для відображення короткого опису, фото та характеристик рослин на окремій сторінці.
CSS: Використовуючи CSS, стилізували ці елементи для привабливого відображення та кращого розуміння контенту.
JavaScript: Додали JavaScript для підвантаження додаткової інформації про рослину при натисканні на неї, що забезпечило зручність для користувача та покращило швидкість завантаження сторінки.
В результаті кожен етап розробки був детально пророблений і оптимізований для забезпечення високої якості і зручності використання вашого сайту.

2.4.Розробка back-end частини

Розробка back-end частини сайту з використанням PHP та фреймворка Laravel, а також бази даних MySQL, включатиме кілька етапів, таких як підключення до бази даних, створення моделей для роботи з даними, розробка контролерів для обробки запитів, а також використання маршрутів та шаблонів для відображення вмісту на сторінках.

1. Підключення до бази даних:

Для початку потрібно налаштувати з'єднання з базою даних MySQL у конфігураційних файлах Laravel. Для цього було використано файл .env, де визначено параметри підключення до бази даних, такі як ім'я бази даних, користувач, пароль тощо.

2. Моделі для роботи з даними:

Створено моделі для кожної таблиці в базі даних (Plants та Characteristics). Використовуючи міграції, визначено структуру кожної таблиці. Моделі Laravel дозволяють взаємодіяти з даними у базі даних шляхом створення, оновлення, видалення та отримання записів.

3. Контролери для обробки запитів:

Створено контролери, які будуть обробляти запити з клієнтської частини сайту. Наприклад, контролер для сторінки з рослинами може отримувати дані з бази даних та передавати їх у відповідний шаблон для відображення.

4. Маршрути для направлення запитів:

Визначено маршрути в файлі web.php, які вказують, які URL повинні бути оброблені кожним контролером. Наприклад, /plants може вказувати на метод index контролера, що відповідає за виведення списку рослин.

5. Шаблони для відображення вмісту:

Розроблено шаблони для кожної сторінки сайту. Вони написані з використанням Blade - шаблонної системи Laravel. Шаблони дозволяють відображати дані, отримані з контролерів, та робити їх динамічними.

На початку розробки з'єднання з базою даних MySQL у конфігураційних файлах Laravel було визначено в файлі .env. Використали параметри підключення до бази даних, такі як ім'я бази даних, користувач, пароль тощо.

```
// .env
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=ksugarden
DB_USERNAME=user
DB_PASSWORD=`` ``
```

Рис 6.1. - Підключення.

Використовуючи Laravel Eloquent ORM, ми створили моделі для кожної таблиці в базі даних. Це дозволило нам взаємодіяти з даними у базі даних шляхом створення, оновлення, видалення та отримання записів.

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Plant extends Model
{
    protected $table = 'plants';
    protected $fillable = ['name', 'description', 'type', 'photo', 'data', 'link'];

    public function characteristics()
    {
        return $this->hasOne(Characteristics::class);
    }
}
```

Рис 6.2. – ORM Plants.

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Characteristics extends Model
{
    protected $table = 'characteristics';
    protected $fillable = ['leaf_margin',
     'venation', 'phyllotaxis', 'stipules', 'pubescence', 'sex',
     'symmetry', 'number_of_petals', 'number_of_stamens', 'number_of_carpels',
     'pollination', 'shape', 'size', 'color', 'presence_of_seeds', 'seed_dispersal', 'growing_season',
     'dormant_season', 'autumnal_characteristics', 'soil_type', 'water_requirements', 'light_requirements',
      'frost_tolerance', 'alkaloids', 'essential_oils', 'vitamins', 'minerals', 'ornamental_use', 'food_use',
      'medicinal_use', 'other_uses', 'wild_species', 'cultivated_species', 'hybrids'];

    public function plant()
    {
        return $this->belongsTo(Plant::class);
    }
}
```

Рис 6.2. – ORM Characteristics.

Створили моделі для кожної таблиці в базі даних за допомогою міграцій Laravel. Визначили структуру кожної таблиці у міграціях.

```php
// database/migrations/create_plants_table.php
Schema::create('plants', function (Blueprint $table) {
    $table->id();
    $table->string('name');
    $table->text('description');
    $table->string('type');
    $table->string('photo');
    $table->int('data');
    $table->string('link');
    $table->timestamps();
});
```

Рис 6.4. - Migrations.

Моделі Eloquent дозволяють зручно взаємодіяти з базою даних, надаючи доступ до методів для вибору, оновлення, видалення та вставки даних.

```php
// app/Plant.php
namespace App;

use Illuminate\Database\Eloquent\Model;

class Plant extends Model
{
    protected $table = 'plants';

    // Додаткові методи для роботи з даними
}
```

Рис 6.5. - Eloquent.

Створили контролери, які обробляють запити з клієнтської частини сайту. Наприклад, контролер для сторінки з рослинами отримує дані з бази даних та передає їх у відповідний шаблон для відображення. Контролери Laravel використовуються для обробки HTTP-запитів. Вони приймають вхідні дані, виконують потрібні дії та повертають відповідь, часто у вигляді відображення відповідного шаблону.

```php
// app/Http/Controllers/PlantController.php
namespace App\Http\Controllers;

use App\Plant;
use Illuminate\Http\Request;

class PlantController extends Controller
{
    public function index()
    {
        $plants = Plant::paginate(9);
        return view('plants.index', compact('plants'));
    }

    // Інші методи контролера
}
```

Рис 6.6 - Контролери.

Визначили маршрути в файлі web.php, які вказують, які URL повинні бути оброблені кожним контролером.

Маршрутизація у Laravel відповідає за направлення вхідних HTTP-запитів до відповідних контролерів. Вони дозволяють визначити, який код буде виконуватися при отриманні певного URL.

```php
Route::get('/plants', 'PlantController@index');
```

Рис 6.7. - Маршрутизація.

Створили шаблони для кожної сторінки сайту за допомогою Blade - шаблонної системи Laravel.

Це дозволяє нам створити динамічні сторінки з використанням даних, які ми отримали з контролерів та моделей. Цей підхід дозволяє нам створювати складні та динамічні веб-сайти з високою продуктивністю та зручністю використання.

```
@foreach ($plants as $plant)
    <div>{{ $plant->name }}</div>
@endforeach

{{ $plants->links() }}
```

Рис 6.8. - Шаблонізація.

2.5.Тестування

1.Після завершення тестування фронтенду на сторінках з рослинами, картами, категоріями, фільтрами та іншими функціями ми отримали позитивні результати без виявлення будь-яких проблем. Ось деякі деталі тестування:

Рендеринг сторінок: Переконалися, що сторінки з рослинами, картами, категоріями і фільтрами коректно відображаються у браузері без помилок та збоїв.

Навігація: Перевірили роботу навігаційних посилань та кнопок, щоб забезпечити коректне перехід між сторінками та їх функціоналом.

Фільтрація та пошук: Перевірили роботу фільтрів та пошуку, переконавшись, що вони коректно фільтрують та відображають рослини відповідно до введених критеріїв.

Категорії та картки рослин: Впевнилися, що рослини правильно класифікуються за категоріями та їх картки відображають відповідну інформацію.

Взаємодія з картами: Перевірили функціонал взаємодії з картами, такий як відображення маркерів, детальна інформація про рослини при кліку на маркери тощо.

Адаптивність і кроссбраузерність: Перевірили, як сторінки відображаються на різних пристроях та браузерах, щоб забезпечити коректну роботу на різних платформах.

Загальний висновок після тестування фронтенду є позитивним, з відсутністю виявлених проблем на сторінках з рослинами, картами, категоріями, фільтрами та іншими функціями.

2. В результаті успішного тестування функціоналу взаємодії з базою даних для рослинного додатку на Laravel були отримані наступні позитивні результати:

Модельні тести: Упевнились, що моделі Plants і Characteristics працюють належним чином, зберігаючи, оновлюючи та видаляючи дані безперешкодно.

Тести контролерів: Перевірено всі методи контролерів для обробки запитів, такі як створення, оновлення, видалення та отримання рослин та їх характеристик. Всі вони працюють правильно та повертають очікувані результати.

Зв'язок між моделями: Перевірено, що зв'язок "один до одного" між моделями Plants і Characteristics працює коректно, дозволяючи звертатися до характеристик кожної рослини через модель Plants та навпаки.

Функціональні тести: Перевірено, що введені дані коректно зберігаються у базі даних та можуть бути успішно витягнуті з неї. Також впевнилися, що зображення рослин коректно зберігаються та відображаються у відповідях сервера.

Тестування API: Переконалися, що всі ендпоінти API працюють належним чином і повертають очікувані дані у форматі JSON.

Загальною висновок після тестування є те, що функціонал взаємодії з базою даних для рослинного додатку на Laravel працює стабільно та надійно.

2.6.Впровадження

Компонування файлів з режиму розробки в режим продукції:
Використано інструменти збірки, такі як Laravel Mix та Webpack, для компіляції та оптимізації ресурсів, таких як JavaScript, CSS, із режиму розробки в режим продукції. Налаштування видалення зайвих коментарів, мініфікації коду та інших оптимізацій.

Компонування в Docker:
Було створено Dockerfile для back-end і front-end додатків, в якому вказується потрібне середовище виконання та налаштування залежностей. Додатково використано Docker Compose для керування кількома контейнерами. У Docker Compose також було налаштовано мережі та об'єднати back-end і front-end в одному комплекті.

Розгортання на сервері:

Було розгорнуто Docker контейнер на обраному сервері. Це включало завантаження Docker образів на сервер та їх запуск. Також важливо було налаштувати веб-сервер для проксі-перенаправлення запитів на відповідні Docker контейнери.

Реєстрування домена:

Було зареєстровано домен сайту через доменний реєстратор. Після реєстрації домену налаштовано DNS записи, з вказівкою на IP-адресу сервера, на якому розгорнутий додаток.

Цей процес впровадження дозволив ефективно розгорнути back-end та front-end додаток на живий сервер для широкого кола користувачів.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи на тему ‘Розробка та впровадження Веб-сервісу ‘Парк Херсонського державного університету’’, було виконане певне дослідження з наступними завданнями.

Проаналізовано існуючі веб-сервісі для парків. Цей аналіз виявився глибоким, і дозволив виокремити тенденції та технології, що дали розуміння в якому векторі рухатися та розвивати проєкт. Дали можливість оцінити з різних сторін можливість такої розробки та цільові аудиторні вподобання.

Розроблено вимоги до сервісу та проведено його проєктування. Ці дві взаємопов’язані функціональні можливості дали структурування та певний план дій, у подальшій роботі. Були розроблені як функціональні, так і нефункціональні вимоги, які дали розуміння та бачення подальшого розвитку дослідження. Під час проєктування вже було складено візуальне представлення майбутнього веб-сервісу, з усіма його елементами.

Безпосередня розробка самого веб-сервісу, з розділенням роботи на етапи, що стало заключним етапом у перед підготовці до розробки. Після цього було створено проєкт глобальної бази даних та її моделей, що у подальшому були інтегровані у систему. Розроблена front-end частина, з простим та інтуїтивно зрозумілим дизайном, інтерактивом, що покращує досвід взаємодії користувача з сайтом. Розроблена back-end частина, яка відповідала за відображення усієї інформації з баз даних на сервісі. Ця частина одна з найважливіших, адже від неї потім залежить прямий процес інтеграції та впровадження.

Проведено тестування, де було підтверджено успішну функціональність та виправлено деякі недоліки, що у результаті дали можливість висунути веб-сервіс у подальший розвиток.

Впроваджено веб-сервіс ‘Парк Херсонського державного університету’ на серверному рівні, що дало можливість взаємодіяти з ним звичайним користувачам.

У висновку, усі поставлені завдання дослідження були реалізовані. Було розроблено та впроваджено веб-сервіс ‘Парк Херсонського державного університету’, який підвищить зацікавленість як людей що знайомі з даним університетом, так і звичайних жителів міста Херсон. Також присутня можливість розвивати проєкт, підіймаючи його рівень.

ВИКОРИСТАНІ ДЖЕРЕЛА

1. KSPU. URL**:** https://www.kspu.edu/
2. Посібник з PHP. URL: https://php.org.ua/manual/uk/manual.md/
3. Книга: Matt Stauffer. Laravel: Up and Running: A Framework for Building Modern PHP Apps 1st Edition. O'Reilly. 2016. 454с.
4. HTML Підручник. URL: https://w3schoolsua.github.io/html/index.html#gsc.tab=0
5. CSS Підручник. URL: https://w3schoolsua.github.io/css/index.html#gsc.tab=0
6. JavaScript Підручник. URL: https://w3schoolsua.github.io/js/index.html#gsc.tab=0
7. Книга: Пасічник В. В., Пасічник О. В., Угрин Д. І. Веб-технології та веб-дизайн. Книга 1. Веб-технології. Магнолія. 2021. 336с.
8. Книга: Бородкіна І.Л., Бородкін Г.О. WEB-технології та WEB-дизайн: застосування мови HTML для створення електронних ресурсів. Ліра-К. 2020. 212с.
9. Книга: Мельник Р.А. Програмування веб-застосувань (фронт-енд та бек-енд). Львівська політехніка. 2018. 248с.
10. Стаття: `Як вчити Laravel початківцям`. URL: https://foxminded.ua/laravel-dlia-pochatkivtsiv/
11. Стаття: `7 причин використовувати laravel`. URL: https://brainlab.com.ua/uk/blog-uk/7-prychyn-vykorystovuvaty-laravel
12. Книга: Douglas Crockford. How JavaScript Works. Virgule-Solidus. 2018. 280с.
13. Книга: Joe Attardi. Web API Cookbook: Level Up Your JavaScript Applications 1st Edition. O'Reilly Media. 2024. 278с.
14. Книга: Martine Dowden, Michael Gearon. Tiny CSS Projects. Manning. 2023. 392с.
15. Книга: Бен Фаррелл. Веб-компоненти в дії. Print2print. 2020. 462с.
16. Стаття: ‘Веб-розробка: вчора, сьогодні, завтра’. https://dou.ua/lenta/articles/web-development-status-2020/

17. Стаття: ‘Laravel Multi Service: як масштабувати Laravel-застосунок’. https://dou.ua/forums/topic/43911/
18. Central Park Conversancy. URL: https://www.centralparknyc.org/
19. Grand Canyon National Park. URL: https://www.nps.gov/grca/index.htm
20. Стаття: ‘CSS для дизайнерів і не тільки. Основи, які має знати кожен. Частина 1’. URL: https://dou.ua/forums/topic/38091/