МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
_____ Литвиненко О.Є.

“_____”_____2020 р.

ДИПЛОМНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ
“МАГІСТР”

Тема: «Інформаційний геолокаційний додаток для автоматизації задач»

Виконавець: студентка, СП-235М, Примушко Аліна Віталіївна

Керівник: к. ф.-м. н. Кучерява Ольга Миколаївна

Нормоконтролер: Тупота Є.В.

Київ 2020

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії

Кафедра комп'ютеризованих систем управління

Освітнього ступеня магістр

Напрям (спеціальність) 123 «Комп'ютерна інженерія»
(шифр, найменування)

ЗАТВЕРДЖУЮ

Завідувач кафедри
Литвиненко О.Є.

"_____"_____2020 р.

ЗАВДАННЯ
на виконання дипломної роботи (проекту)
Примушко Аліни Віталіївни
(прізвище, ім'я, по батькові)

1.Тема роботи: «Інформаційний геолокаційний додаток для автоматизації задач» затверджена наказом ректора від «27» серпня 2020 року № 1203 /ст.
2. Термін виконання роботи: з 05 жовтня 2020 р. по 13 грудня 2020р.
3. Вихідні дані до роботи: мова програмування *JavaScript*, мова програмування *PHP*, мова розмітки даних *HTML*, мова стилю вебсторінок *CSS*.
4.Зміст пояснювальної записки (перелік питань, що підлягають розробці):
1. Огляд різновидів інформаційних та мобільних додатків.
2. Теоретичні основи розробки додатків.
3. Технології створення геолокаційного інформаційного додатку.
4. Розробка інформаційного геолокаційного додатку для автоматизації задач користувача.

5.Перелік обов'язкового графічного матеріалу:
1. Робота геолокаційного мобільного додатку (схема алгоритму).
2. Життєвий цикл розробки мобільного додатку.
3. Діаграма класів геолокаційного мобільного додатку.
4. Діаграма прецедентів геолокаційного мобільного додатку.
5. Сторінка інформаційного геолокаційного додатку.
6. Сторінка мобільного додатку.

6. Календарний план

№ п/п	Етапи виконання дипломного проекту	Термін виконання етапів	Примітка
1	Проведення огляду різновидів інформаційних додатків	05.10.2020-12.10.2020	
2	Написання першого розділу	12.10.2020-15.10.2020	
3	Опрацювання теретичних основ розробки мобільних додатків	15.10.2020-20.10.2020	
4	Написання другого розділу	20.10.2020-16.10.2020	
5	Аналіз літературних джерел щодо технологій створення інформаційного геолокаційного додатку.	17.10.2020-27.10.2020	
6	Написання третього розділу	27.10.2020-02.11.2020	
7	Розробка інформаційного геолокаційного додатку для автоматизації задач користувача.	02.11.2020-20.11.2020	
8	Оформлення пояснювальної записки	20.11.2020-07.12.2020	
9	Підготовка графічних та ілюстративних матеріалів	07.12.2020-13.12.2020	

7. Дата видачі завдання « 05 » жовтня 2020 р.

Керівник дипломного проекту_____ Кучерява О.М.
(підпис)

Завдання прийняв до виконання_____ Примушко А.В.

РЕФЕРАТ

Пояснювальна записка до дипломної роботи « Інформаційний геолокаційний додаток для автоматизації задач»: 92 сторінки, рисунків, 30 літературних джерел, 1 додаток.

Ключові слова: МОБІЛЬНИЙ ДОДАТОК, ІНФОРМАЦІЙНИЙ, ДОДАТОК, АВТОМАТИЗАЦІЯ ЗАДАЧ, ЗБІР ІНФОРМАЦІЇ, ГЕОЛОКАЦІЯ.

Об’єкт дослідження – процес розробки геолокаційних мобільних додатків.

Предмет дослідження – автоматизація процесу збору інформації.

Мета дипломної роботи – розробка геолокаційного програмного засобу для автоматизації процесу збору інформації.

Методи дослідження – набір інструкцій та команд мов програмування *JavaScript* та *PHP* , мова розмітки даних *HTML*, мова стилю вебсторінок *CSS*.

Результатом дипломної роботи є мобільний додаток призначений для звичайних користувачів та груп дослідників. Додаток надає користувачам доступ до інформації в режимі реального часу під час вивчення і дослідження різних геологічних чи історичних ландшафтів. З його використанням користувачі при відвідуванні і дослідженні нової для себе місцевості, можуть відповідати на різні запитання, задані групами дослідників.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

HTML (HyperText Markup Language) – стандартизована гіпертекстова мова розмітки.

CSS (Cascading Style Sheets) – мова опису зовнішнього вигляду вебсторінок. Мовою описують стилістичні інструкції щодо відображення контенту.

JavaScript – динамічна мова програмування, що будовується в *HTML*-код і забезпечує динамічну інтерактивність.

PHP (Hypertext Preprocessor) – скриптова мова програмування. Вона використовується для розробки динамічних інтерактивних веб-сторінок і в цілому орієнтується на серверні сценарії.

СУБД – система управління базами даних.

MySQL – вільна СУБД. Дозволяє створювати, редагувати та отримувати доступ до декількох баз даних на сервері.

JSON (*JavaScript Object Notation*) – текстовий формат обміну даними між комп'ютерами.

API (Application Programming Interface) – прикладний програмний інтерфейс, надає розробнику засоби швидкої розробки програмних засобів.

GPSS (General Purpose Simulation System) – система моделювання загального призначення).

ВСТУП

Актуальність теми. Сучасне життя складно уявити без використання обчислювальної техніки тією чи іншою мірою. Комп'ютери тією чи іншою мірою проникли в усі сфери людської діяльності, маючи на меті зробити наше життя більш комфортним.

Кожен день перед людиною стоїть питання пошуку найкращого рішення. Дослідження показують, що значну частину свого робочого дня працівник витрачає на рутинні задачі, наприклад, на збір інформації.

Автоматизація непродуктивних складових робочого процесу є хорошим способом для покращення власного виробничого потенціалу. Метою її впровадження є перш за все заміна рутинної роботи на автоматизовані дії. Разом зі зменшенням впливу людського фактору, зменшується можливість загубити дані чи допустити помилку внаслідок втрати концентрації. Кількахвилинні рутинні завдання, які виконує людина, впливають на її концентрацію. Вона втрачає пильність і не може розглянути проблему з іншого боку чи висувати нові ідеї. Чим більше рішень потрібно прийняти, тим важчим кожне з них стає для людського мозку. Автоматизація допоможе звільнити час та розумову енергію з метою зосередитись на більш серйозних та важливих питань. Наприклад, замість того, що витрачати час на рутину, працівник його може використати на пошук нових способів для покращення якості і вдосконалення продукту, на впровадження цих ідей чи взаємодію з клієнтами.

Автоматизація дозволяє працювати швидше. Кількахвилинні рутинні завдання, які виконує людина, впливають на її концентрацію. Вона втрачає пильність і не може розглянути проблему з іншого боку чи висувати нові ідеї. Чим більше рішень потрібно прийняти, тим важчим кожне з них стає для мозку працівника. Автоматизація може звільнити робочий час та розумову енергію з метою зосередитись на більш серйозних та важливих питань. Наприклад, замість того, що витрачати час на рутину, працівник його може використати на пошук нових способів для покращення якості і вдосконалення продукту, на впровадження

цих ідей чи взаємодію з клієнтами. Впровадження програмного забезпечення для автоматизації вирішить проблему з нестачею робочої сили, коли немає можливості розширити штат. Врешті автоматизація необхідна, коли обсяги інформації, яку необхідно опрацювати і обробити, надто великі, для обробки людиною вручну.

Програмний засіб бере на себе виконання багатоповторних задач. Орієнтовані на споживача інструменти значно спрощують розробку складної серії задач. Поруч із цим постає питання розробки програмного забезпечення для автоматизації конкретних задач.

Створення додатків набирає популярності, тому їх розробка стає все більш затребуваною. Користувачі все частіше між десктопним і мобільним додатком роблять вибір на користь другого варіанту.

Сучасне життя обертається навколо інформації. Це спричиняє попит на створення інформаційних додатків. Слід дослідити, що собою являє поняття інформаційного додатку, виділити найбільш популярні області їх застосування.

Існує великий клас інформаційно-довідкових систем, що базуються на гіпертексті і мультимедіа. Гіпертекст являє собою набір логічно зв'язаних текстових, графічних, аудіо- та відеоматеріалів. Велика кількість програмних продуктів поставляються або з довідковими системами, або з електронною документацією, організованої подібним чином. Для розробки навчальних додатків також часто використовується принцип гіпертексту. Подальший розвиток такі інформаційні додатки отримали у Всесвітній мережі Інтернет. Тут поняття гіпертекстового посилання розширюється на глобальну мережу. Вона включає імена вузла та файлу і позицію всередині файлу. Спеціальні *WWW*-сервери є сховищем для такого гіпертексту, клієнти мають програми для їх перегляду – браузери. Останнім часом пасивний характер таких систем змінився, в них з'явилися засоби діалогу для формування замовлень на продукти і для здійснення оплати товару. Перспективи розвитку при цьому пов'язують з мовою *Java*, що орієнтована на розробку інформаційних додатків.

Самостійний підклас інформаційно-довідкових систем складають географічні інформаційні системи. В них інформація прив'язується до точок на

карті або плану місцевості. Для однієї географічної області може бути представлено кілька картографічних шарів з різними об'єктами і відповідно різною інформацією щодо них. Наприклад, міські комунікації, транспортні зв'язки, лісові масиви, водойми. Шари можна накладати, тим самим утворюючи карту, що орієнтована на вирішення конкретних задач.

Всі галузі нашого життя впевнено прямують в сторону онлайн засобів і сфера досліджень не є виключенням. В процесі дослідження збір інформації часто проводиться у формі живого опитування. Набагато зручніше та швидше було б здійснювати це за допомогою мобільного додатку. Для автоматизації процесу збору інформації було вирішено розробити власний геолокаційний інформаційний додаток. Дослідник створити власний проект в додатку та розмістити ряд запитань, а користувач додатку може, відвідуючи певну місцевість в режимі реального часу допомагати дослідникам, відповідаючи на їх запитання в мобільному додатку.

Мета і завдання виконання дипломної роботи. Метою дипломної роботи є розробка геолокаційного програмного засобу для автоматизації процесу збору інформації. Для того, щоб досягти поставленої мети необхідно дослідити, чому загалом потрібне вровадження автоматизації задач і чим воно може бути корисне, визначити, чи можна повністю автоматизувати робочі процеси. Проаналізувати поняття інформаційного додатку, виділити осовні аспекти розробки. Визначити, що собою являє геолокаційне програмне забезпечення, з якою метою його розробляють, виділити основні напрями розробки та визначити які складнощі можуть виникати в процесі їх реалізації.

Ретельно дослідити поняття мобільного додатку. В цілому їх можна розділити на три основні групи: нативні додатки, гібридні додатки та вебдодатки. Кожен із перерахованих підходів має як ряд перваг, так і недоліків. В процесі пошуку правильного підходу до реалізації бажаного проекту, розробники, опираються на користувацький досвід, обчислювальні потужності та власні навики, необхідні для розробки мобільного додатку, бюджет і часові обмеження на релізацію та ресурси, доступні для обслуговування програмного продукту. Слід виділити основні етапи процесу розробки мобільних додатків.

Об’єкт і предмет дослідження. Об’єктом дослідження даної роботи є процес розробки геолоакаційних мобільних додатків. Предметом даної роботи є автоматизація процесу збору інформації.

Методи дослідження. Для досягнення поставленої мети в процесі розробки додатку було застосовано наступні технології: для реалізації користувацького інтерфейсу застосовано стандартизовану гіпертекстову мову розмітки *HTML*, мову стилів *CSS*. Мову *HTML* інтерпретує браузер, а отриманий внаслідок інтерпретації контент відображається на екрані пристрою користувача. Структура, утворена *HTML*-кодом є певного роду фундаментом. Каскадні таблиці стилів *CSS* визначають, у якому вигляді уже існуючий *HTML*-документ побачить користувач. *CSS*-файли містять стилістичні інструкції щодо відображення документа. За динамічну обробку даних відповідають мова *JavaScript* та бібліотека *jQuery*. Так як інформаційні додатки працюють в режмі діалогу з користувачем, слід забезпечити динамічну інтерактивність. *JavaScript* являє собою динамічну мову програмування, яку застосовують як мову сценаріїв. *JavaScript* вбудовується в *HTML*-документ і забезпечує динамічну інтерактивність. Для розробки внутрішньої серверної системи мобільного додатку застосовано мову програмування *PHP* Сфера застосування *PHP* розширилась до створення мобільних додатків. Переваги *PHP* зроблять розробку більш ефективною та простою з точки зору обслуговування. Завдяки широкому спектру функціоналу, набору фреймворків, що також розширяють його можливості, для *PHP* доступним є майже все: збір даних, програмування серверної частини, динамічна генерація контенту. Використання *PHP* підтримується майже на всіх операційних системах, також має підтримку для більшості серверів і баз даних.

При розробці геолокаційного додатку для побудови карт для компонентів досліження було використано прикладний програмний інтерфейс *Google Maps*. Для реалізації бази даних, щоб зберігати всі дані розроблюваної системи застосовано *MySQL* та *GeoJSON*. В якості середовища розробки обрано *Microsoft Visual Studio 2015 Professional*.

Наукова новизна отриманих результатів. Результатом виконання дипломної роботи є геолокаційний додаток, що дозволяє автоматизувати процес збору інформації.

Розроблюваний мобільний додаток створено дослідження в природних умовах, наприклад, національних, державних та місцевих парках. Програма використовує геодані для інтеграції інформації про розташування в мобільний додаток, а також надаватиме інформацію про місцезнаходження та навчальну інформацію для науковців.

Практичне значення отриманих результатів. Отриманий в процесі виконання дипломної роботи мобільний додаток призначений для звичайних користувачів та груп дослідників. Додаток надає користувачам доступ до інформації в режимі реального часу під час вивчення і дослідження різних геологічних чи історичних ландшафтів. З його використанням користувачі при відвідуванні і дослідженні нової для себе місцевості, можуть відповідати на різні запитання, задані групами дослідників.

РОЗДІЛ 1
ОГЛЯД РІЗНОВИДІВ ІНФОРМАЦІЙНИХ ТА МОБІЛЬНИХ ДОДАТКІВ

1.1. Поняття інформаційного додатку

Сучасне життя обертається навколо інформації. Це спричиняє попит на створення інформаційних додатків і їх робробка стає все більш широко поширеним завданням, яке вирішують різні комерційні організації. Слід розібратися, що собою являють інформаційні додатки, виділити найбільш їх популярні області застосування та співвіднести їх з типами інформаційних систем.

Інформаційний додаток – прикладна програмна підсистема, що орієнтується на збір, зберігання, пошук та обробку інформації. У більшості випадків інформаційні додатки працюють в режимі діалогу з користувачем. Типовий інформаційний додаток включає в себе:

1. Діалогове введення і виведення інформації.
2. Логіка діалогу.
3. Прикладна логіка обробки даних.
4. Логіка керування даними.
5. Операції, виконувані над файлами чи базами даних.

Переважна частина функціоналу додатка закладена в системному програмному забезпеченні, в бібліотеках і конструкціях інструментальних засобів розробки. Однак залишається частина програми, що змінюється в залежності від конкретної предметної області. Основними об'єктами розробки стають компоненти майбутньої програми, що будуть визначати логіку діалогу та логіку обробки та керування даними. Часто переважаюче значення у розробці має діалог, що проходить через весь розроблюваний додаток, тому більшість інструментів орієнтуються конкретно на спрощення та пришвидшення реалізації діалогу в ньому. Попри маніпуляційний характер процесу розробки, проміжне представлення програми оформляється у вигляді мовного опису, що надалі

дозволить швидко розроблений шаблон надалі наповнювати змістовною обробкою даних, але вже з використанням мови програмування.

Окрім програмного компоненту інформаційного додатку важливу роль грає його інформаційна складова. Вона задасть структуру, атрибутику та типізацію даних, а також обмеження цілісності інформації для баз даних. Інформаційна складова та логіка керування даними тісно пов'язані між собою. Через це засоби автоматизації проектування додатків віддають пріоритет інформаційної моделі, з якої виводиться все інше, включаючи діалог.

За масштабом інформаційні системи можна поділити на одиночні, групові та корпоративні.

1. Одиночні інформаційні системи можна розробити на автономному, як правило, персональному комп'ютері. Така система може містити кілька простих додатків, пов'язаних загальним інформаційним фондом, і розрахована на роботу одного користувача або групи користувачів, які діляють одне робоче місце. Такого виду додатки розробляють з використанням так званих десктопних систем керування базами даних, зокрема *Clarion, Clipper, FoxPro, Paradox, dBase, MS Access* або з використанням файлової системи та діалогової оболонки для введення, редагування і обробки даних.

2. Групові інформаційні системи орієнтуються на колективне застосування інформації об’єктами робочої групи, наприклад, в межах одного робочого підрозділу. Найчастіше групові інформаційні системи будуються у виді локальної обчислювальної мережі персональних комп’ютерів або іноді у виді багатотермінальної централізованої обчислювальної системи. Однотипні чи спеціалізовані робочі місця можуть забезпечити виклик одного або кількох конкретних програм. Загальний інформаційний фонд представляє базу даних або множину різних файлів. Спільний доступ до інформації організовується за допомогою блокувань записів і файлів. В ході розробки групових інформаційних програм використовуються десктопні системи управління базами даних, які розраховані на більшу кількість, сервери баз даних для робочих груп, зокрема *Btrieve, NetWare SQL, Gupta SQLBase, Sybase Anywhere SQL, MS SQL Server,*

Progress, Informix-SE, Workgroup Oracle та відповідні інструменти для розробки чи системи керування документами, їх інструментальні засоби. Взаємодія між користувачами відбувається через централізовану БД, шляхом електронного листування або використанням мережевої файлової системи.

3. Корпоративні інформаційні системи є вдосконаленим варіантом систем для робочих груп. Вони орієнтуються на розміри компанії, можуть підтримувати територіально віддалені вузли чи мережі. Корпоративні системи можуть мати ієрархічну структуру, що складаються з кількох рівнів. Головною особливістю такої системи є забезпечення доступу з підрозділу до центральної або розподіленої БД організації, окрім доступу до інформаційного фонду робочої групи. Таким систем характерна клієнт-серверна архітектура із спеціалізацією серверів. Корпоративні ІС будують на корпоративних *SQL*-серверах баз даних, зокрема *Oracle7, Informix-OnLine, Informix-DSA, Sybase, CA-Ingress* та відповідних інструментальних засобах. Крім власних ресурсів розробки часто застосовують незалежні багатоплатформні інструментальні засоби, що доповнюються інтерфейсами, драйверами і шлюзами для зв'язку з різними системами управління базами даних.

Для таких інформаційних систем вимоги до надійності функціонування і збереження даних більш суворі. Остання властивість забезпечується підтримкою цілісності даних, посилань і транзакцій в серверах баз даних. Транзакція – неподільний набір операцій з базою даних, вона завершується успішно, коли всі її операції виконано, інакше відбувається відкочування до попереднього стану.

Розглянемо основні типи інформаційних систем і пов'язані з ними додатки. За оперативністю обробки даних розрізняють пакетні інформаційні системи (реального часу) та оперативні інформаційні системи. В чистому вигляді системи з пакетною обробкою можна спостерігати на великих централізованих обчислювальних машинах. В системах організаційного управління переважає режим оперативної обробки транзакцій *OnLine Transaction Processing (OLTP)* для відображення поточного стану предметної області в режимі реального часу, а пакетна обробка займає досить обмежену нішу. Для *OLTP* систем характерний

регулярний інтенсивний потік простих транзакцій, які відіграють роль замовлень, платежів, запитів тощо. Висока продуктивність обробки транзакцій і гарантована передача інформації при віддаленому доступі до базі даних є основними вимогами.

Іншим типом інформаційних систем є системи підтримки прийняття рішень *Decision Support System (DSS)*. На основі складних запитів в них проводиться відбір та аналіз даних в різних розрізах: часових, географічних і за різними показниками. Крім традиційних засобів доступу до БД, прогресивні системи підтримки прийняття рішень включають наступні засоби:

- вилучення даних з джерел різних видів, включно з неструктурованою інформацією;
- багатовимірного аналізу даних;
- обробки статистики;
- моделювання правил та стратегії ділової діяльності;
- ділової графіки для представлення результатів аналізу;
- аналізу *What-If*;
- штучного інтелекту (ШІ).

Засоби ШІ складають експертну підсистему, що базується на правилах або прецедентах із бази знань та відповідних механізмів виведення. Загалом вимога оперативності необов'язкова для таких систем з урахуванням комплексності транзакцій та аналітичної обробки.

У класі систем підтримки прийняття рішень виділяється окремий клас систем оперативної аналітичної обробки *OnLine Analysis Processing (OLAP)*. Оперативності обробки можна досягти шляхом застосування потужної мультипроцесорної обчислювальної техніки, спеціальних *OLAP*-серверів, методів багатовимірного аналізу та спеціальних сховищ даних *Data Warehouse*. Такі сховища зберігають інформацію із різних джерел протягом великого проміжку часу та забезпечують до них оперативний доступ. Крім готових *OLAP*-систем для спеціалізованих областей, наприклад, фінансової, існують інструментальні набори для реалізації подібних програм на базі систем керування базами даних або *OLAP*-серверів та сховищ даних.

Існує великий клас інформаційно-довідкових систем, що базуються на гіпертексті і мультимедіа. Гіпертекст являє собою набір логічно зв'язаних текстових, графічних, аудіо- та відеоматеріалів. Велика кількість програмних продуктів поставляються або з довідковими системами, або з електронною документацією, організованої подібним чином.

Для розробки навчальних додатків також часто використовується принцип гіпертексту. Подальший розвиток такі інформаційні додатки отримали у Всесвітній мережі Інтернет. Тут поняття гіпертекстового посилання розширюється на глобальну мережу. Вона включає імена вузла та файлу і позицію всередині файлу. Спеціальні *WWW*-сервери є сховищем для такого гіпертексту, клієнти мають програми для їх перегляду - браузери. Останнім часом пасивний характер таких систем змінився, в них з'явилися засоби діалогу для формування замовлень на продукти і для здійснення оплати товару. Перспективи розвитку при цьому пов'язують з мовою *Java*, що орієнтована на розробку інформаційних додатків.

Самостійний підклас інформаційно-довідкових систем складають географічні інформаційні системи. В них інформація прив'язується до точок на карті або плану місцевості. Для однієї географічної області може бути представлено кілька картографічних шарів з різними об'єктами і відповідно різною інформацією щодо них. Наприклад, міські комунікації, транспортні зв'язки, лісові масиви, водойми. Шари можна накладати, тим самим утворюючи карту, що орієнтована на вирішення конкретних задач.

Значна частина інформації, що поширюється в установах, являє собою неструктуровані дані з паперових документів. Сучасні системи управління електронними документами *EDMS*, що мають на меті перетворення паперових документів в їх електронну версію, забезпечені засобами індексування та пошуку, деякі з них мають властивості гіпертексту. Розвинені офісні системи включають засоби колективної роботи *Groupware*, що забезпечують автоматизацію робочих процесів з використанням електронної пошти, засобів створення звітів, заповнення документів, електронних таблиць і текстових редакторів.

Методи і засоби *Workflow* використовуються для автоматизації документообігу та контролю виконавської дисципліни. У систему закладено графи взаємозв’язків працівників організації, задачі, прив'язано документи та базу даних, маршрути руху документів. Предметом розробки для цих систем є діалогові форми у вузлах графових моделей. Спеціальні технології допомагають у генерації прототипу форми, а розробник лише розширює її для обробки нетипових ситуацій. Складність цієї розробки полягає в побудові цілісної системи графів, що описують ділову діяльність установи.

В інформаційному додатку, що взаємодіє з базою даних, як правило, прийнято виокремлювати наступні функціональні компоненти: засоби представлення, логіка представлення, компонент прикладної логіки, компонент логіки керування даними, операції, що здійснюються над базою даних, компонент файлових операцій.

1. *Presentation Services (PS).*

Цей компонент являє собою засоби представлення. Вони забезпечуються пристроями, які приймають введену користувачем інформацію та відображають те, що вказує пристрою компонент логіки представлення *PL*, з відповідною програмною підтримкою. Це можуть бути різноманітні текстові термінали або персональні комп’ютери в режимі програмної емуляції терміналу.

2. *Presentation Logic (PL).*

Логіка представлення здійснює управління взаємодією між користувачем і електронною обчислювальною машиною. Вона обробляє дії користувача за вибором пунктів меню, натискання кнопок або при виборі елемента зі списку.

3. *Business Logic (BL).*

Прикладна логіка представляє собою визначений перелік правил, що потрібний для прийняття рішень, обчислень та операціями, котрі має виконувати додаток.

4. *Data Logic (DL).*

До логіки керування даними відносять операції, виконувані з базою даних. *SQL*-оператори вибору (*SELECT*), видалення (*DELETE*), оновлення (*UPDATE*) та

вставки (*INSERT*) виконуються для реалізації прикладної логіки управління даними.

5. *Data Services (DS).*

Компонент, що відповідає за операції, виконувані над базою даних. Дії системи управління базою даних, що викликаються для виконання логіки керування даними, такі як маніпулювання даними, визначення даних, фіксація або відкочування транзакції.

6. *File Services (FS).*

Компонент файлових операцій виконує дискові операції читання і запису даних для системи управління базою даних, зазвичай є функціями операційної системи.

Залежно від розташування типових компонентів додатку у вузлах мережі, можна навести кілька схем побудови інформаційних систем (табл. 1.1).

Таблиця 1.1.

Централізовані багатотермінальні системи

Опис схеми	Клієнт	Сервер 1	Сервер 2	Приклад реалізації
Централізована багатотермінальна система	*PS*	*PL, BL, DL, DS, FS*		Сервер *Sun* з *X*-терміналами в операційній системі *Solaris*
Локальна комп'ютерна мережа з файл-серверними додатками	*PS, PL, BL, DL*	*FS*		Локальна мережа ПК в середовищі *NetWare*, програми на *FoxPro, Clipper*
Віддалений доступ до даних на сервері БД	*PS, PL, BL, DL*	*DS, FS*		Клієнт-серверна система з доступом ПК до серверу БД *Informix (NetWare)*

Продовження таблиці 1.1

Віддалений доступ до бази даних із використанням збережених процедур	*PS, PL, DL*	*BL, DS, FS*		Клієнт-серверна система, доступ ПК до серверу *ORACLE* в середовищі *SCO Unix*
Віддалений доступ до бази даних із розділенням логіки додатку	*PS, PL, BL, DL*	*BL, DL, DS, FS*		Клієнт-серверна система, доступ ПК до серверу *ORACLE* на *Sun (Solaris)*
Віддалене представлення даних з доступом до *Unix*-системи	*PS, PL*	*BL, DL, DS, FS*		Мережа ПК станцій, додатки на моніторі транзакцій і СУБД в *Unix*
Віддалене керування файл-серверним додатком у мережі	*PS*	*PL, BL, DL, DS*	*FS*	Зв’язок віддалених ПК із сервером доступу *WinView* в мережі для роботи з СУБД *FoxPro*
Багатотермінальний сервер додатків для доступу до СУБД	*PS*	*PL, BL, DL*	*DS, FS*	Мережа ПК, сервер додатків на *SCO Unix*, доступ терміналів до *ORACLE* на *HP*

Закінчення таблиці 1.1

3-хланкова система на Unix з монітором транзакцій	*PS, PL*	*BL, DL*	*DS, FS*	Мережа ПК, сервер додатків на *SCO Unix*, доступ терміналів до *ORACLE* на *HP*
3-хланкова система на Unix з монітором транзакцій і розділенням логіки	*PS. PL, BL*	*BL. DL*	*DS, FS*	Аналогічно до попереднього, з виконанням контролю даних на клієнтських вузлах

1.2. Автоматизація задач

Сучасне життя – це постійне порівняння альтернатив і пошук найкращого рішення. Це однаково відноситься як до побуту, так і професіональної діяльності людини. Проте відмінність полягає в тому, що для простих задач не виникає необхідність створення цілих систем для прийняття рішень і розроблення для цього математичного апарату.

Згідно з дослідженням Мак-Кінсі, 28 відсотків середнього робочого тижня працівника займають на відповіді на електронні листи. Подібним чином, ще 19 відсотків витрачається на збір інформації та даних, а 14 відсотків на спілкування та співпрацю, залишаючи мізерні 39 відсотків, фактично витрачених на рольову діяльність людини.

Автоматизація – це перш за все заміна рутинної роботи на автоматизовані дії. Програмне забезпечення візьме на себе трудомісткі, багатоповторні задачі, наприклад, внесення даних, формування звітів, розсилка листів, виставлення рахунків, для яких працівник має регулярно виконувати типові дії. Завдяки зменшенню впливу людського фактору, не виникатимуть проблеми з обліком замовлень і заявок, так як програма навряд здатна забути чи загубити дані. Впровадження на підприємстві програмного забезпечення для автоматизації

вирішить проблему з нестачею робочої сили, коли немає можливості розширити штат. Врешті автоматизація необхідна, коли обсяги інформації, яку необхідно опрацювати і обробити, надто великі, для обробки людиною вручну.

Автоматизація непродуктивних частин роботи є хорошим способом для покращення власного виробничого потенціалу. Моніторинг бюджетів, відстеження робочого часу, планування зустрічей, створення звітів – всі ці малоцінні та трудомісткі завдання робочого процесу можна піддати автоматизації, використовуючи відповідні інструменти.

Автоматизація дозволяє працювати швидше. Кількахвилинні рутинні завдання, які виконує людина, впливають на її концентрацію. Вона втрачає пильність і не може розглянути проблему з іншого боку чи висувати нові ідеї. Чим більше рішень потрібно прийняти, тим важчим кожне з них стає для мозку працівника. Автоматизація може звільнити робочий час та розумову енергію з метою зосередитись на більш серйозних та важливих питань. Наприклад, замість того, що витрачати час на рутину, працівник його може використати на пошук нових способів для покращення якості і вдосконалення продукту, на впровадження цих ідей чи взаємодію з клієнтами.

Автоматизація задач забезпечує зменшення ручних повторюваних операцій. Передбачаючи використання програмного забезпечення для зменшення ручної обробки простих завдань чи ряду більш складних, вона допомагає підвищити прискорити процес роботи і покращує її продуктивність.

Програмне забезпечення для автоматизації задач стає все більш і більш популярним серед користувачів. Орієнтовані на споживача інструменти значно спрощують розробку складної серії задач.

У більшості випадків задачі впорядковуються за допомогою інтерфейсу перетягування, що дозволяє впорядковувати та підключати різноманітні типи завдань, щоб імітувати реальний робочий процес задач. Це також іноді називають автоматизацією робочого процесу. Завдання в процесі можуть включати форми, які потрібно заповнити, підтвердження, які необхідно надати, розподіл робіт, спрацьовування іншої системи, пошук даних, надсилання електронних листів та

сповіщень тощо. Більшість програм для автоматизації задач включають в себе множину подібних попередньо побудованих, що можуть бути організованими послідовно або паралельно, виходячи з порядку типових задач.

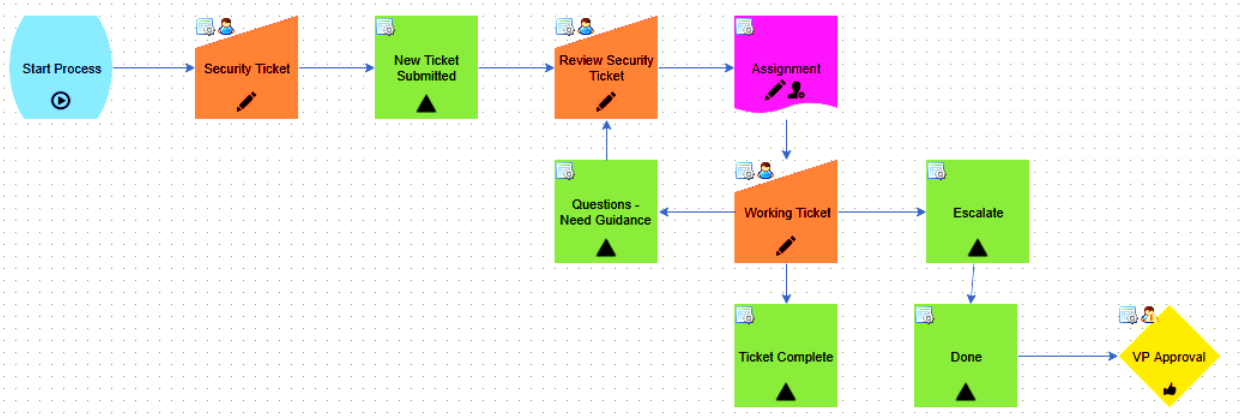

Рис. 1.1. Серія автоматизованих завдань для обробки випадків безпеки

Перехід до автоматизації завдань ґрунтується не просто на бажанні скоротити повторювані завдання та звільнити час, а насправді для підвищення надійності та вилучення помилок, пов’язаних з людським фактором. Від простого до вдосконаленого, виявляється, є цілий ряд, який завдань, які можна автоматизувати, щоб підвищити продуктивність робочого дня.

Проте навіть максимально автоматизовані задачі вимагають втручання людини в певний момент процесу.

Хоча розумні правила, боти та подібні інтелектуальні інструменти можуть допомогти зробити процеси ефективнішими, процедури, які повністю автоматизовані процеси, без залучення людини можуть з часом призвести до несподіваних результатів.

Коли перед людиною поставлене завдання, їй також потрібні допоміжні інструменти, що дозволять співпрацювати та відповідати на запитання. Наприклад, хтось може надати недостатньо інформації, щоб вимагати витрат, і дивуватися, чому їх запит було повернуто. Включення інструментів для обговорення як частини процесу дозволяє зацікавленим сторонам спілкуватися безпосередньо, незалежно від того, до, під час або в кінці процесу.

В результаті впровадження автоматизації можна досягти наступного:

- Економія робочого часу, який працівник би витратив на оформлення звітів, складання графіків, формування статистики, заповнення бази даних чи розміщення оголошень.
- Спрощення роботи з клієнтами. Автоматичне оформлення облікового запису клієнта, запис наступних подій та нагадування про них звільнить від цього менеджерів. В результаті буде дотримані всі терміни на обслуговування і оброблено більше заявок.
- Зменшення похибки, зумовленої людським фактором. Що примітивніше та нудніше завдання, тим вища ймовірність припуститися помилки в процесі вирішення такої задачі. Проте такого не станеться, коли процес вирішення задачі буде автоматизовано.
- Збільшення швидкості та якості роботи по обслуговуванню клієнтів.
- Збільшення швидкості отримання звітів та статистики по компанії. Без автоматизації працівник може витрачати до кількох днів на розробку звіту, щоб вирахувати всі компоненти, підсумувати та здійснити перевірку на помилки повторно. Натомість програмне забезпечення виведе звіт менше, ніж за хвилину.
- Пришвидшення процесу підготовки документів, рахунків, актів.
- Завдяки налаштуванням прав доступу до інформації, цінні конфіденційні дані будуть ліпше захищені.

Для постійного розвитку і зростання, наприклад, компанії, всі її процеси мають бути впорядковані в єдину логічну систему. Це фундаментальна основа для швидкості і точності прийняття рішень. Точність та достовірність даних спрощує аналіз різних ситуацій, прогнозування та покращує показники.

У групі засобів автоматизації бізнес-процесів виробництва та документообігу можна виокремити наступні підгрупи:

- засоби автоматизації організацій діяльності *Office Automation*;
- системи керування електронними документами *EDMS*;
- засоби забезпечення колективної роботи *Groupware*;
- засоби автоматизації документобігу *Workflow*.

Засоби автоматизації установчої діяльності включають до свого складу: текстові редактори для підготовки і коригування документів; процесори електронних таблиць для розрахунків, аналізу та графічного представлення даних; програми генерації запитів за зразком з різних баз даних; мережеві планувальники для призначення робочих зустрічей та нарад; засоби розробки і демонстрації ілюстративних матеріалів для презентацій; словники та системи порядкового перекладу; програми надсилання і прийому факсів; локальну електронну пошту для обміну повідомленнями та пересилки файлів.

Ці засоби можуть являти собою окремі пакети (WinWord, WordPerfect, Excel), інтегровані пакети програм (MS Works) або узгоджені набори пакетів (Microsoft Office або Corel Perfect Office). Для багатьох пакетів програм характерне використання так званих "майстрів" (Wizard), які в режимі діалогу із користувачем допомагають йому розібратись зі складним функціоналом, виконати складну процедуру обробки.

Засоби забезпечення колективної роботи Groupware загалом орієнтуються на автоматизацію роботи невеликого колективу і підтримують коректне спільне використання інформації малою групою користувачів. Такий поділ інформаційних ресурсів можливо організувати в локальній комп’ютерній мережі за допомогою пакетів засобів офісної автоматизації (Microsoft Office) і мережевої файлової системи або електронної пошти та форматного введення документів (Lotus Notes). При цьому відсутня структуризація в проведенні робіт: немає конкретних правил і вказівок правил виконання робіт в системі. Ця підгрупа практично збігається із засобами автоматизації установчої діяльності, за виключенням інтеграції засобів, колективного використання ресурсів і застосування електронної пошти.

Засоби автоматизації документообігу Workflow націлені на автоматизацію діяльності корпорації і підтримують розподіл робіт: декілька користувачів здійснюють обробку декількох процесів одночасно в синхронному або в

асинхронному режимі. Для автоматизації документообігу та контролю виконавської дисципліни застосовуються методи теорії Workflow. У систему закладаються опису графів взаємин працівників, виконання робіт чітко структуризовано: розписано по ролям, документам, часу обробки документів і задані маршрути руху документів. Ці графові моделі і їх параметри зберігаються в базі даних. Основні труднощі при цьому полягає в побудові системи графів, що описують ділову діяльність всієї установи. У найпростішому випадку для підтримки документообігу буває необхідно забезпечити маршрутизацію документів по електронній пошті. Система може бути інтегрована як з системою управління документами, так і з СУБД. Вона реалізується в середовищі клієнт-сервер і орієнтована на корпоративну мережу.

1.3. Застосування Інтернет-ресурсів в дослідницькій сфері

Процес пошуку інформації швидкий і доступний кожному: за умови наявності з'єднання з мережею та пристрою для пошуку за лічені секунди користувач отримує доступ до незлічненного обсягу знань. Пошукові системи містять відповіді на безліч запитань в будь-який момент часу.

Завдяки поліпшенню комунікації та вдосконаленню існучих технологій, Всесвітня мережа набирає значущості і на даний момент вже встигла стати потужним інструментом в світі, який суспільство користуться відповідно до своїх потреб та інтересів.

Навчальна функція мережі Інтернет полягає не лише в простоті пошуку знань та вивченні деяких речей, але й в унікальній можливості ділитись власними знаннями і результатами досліджень з усім світом.

Для освітньої сфери можна виділити такі позитивні аспекти:

1. Економічна доступність знань. Фундаментальною основою неперервного розвитку суспільства є якість освіти. Просте досить часто перешкодою в навчальному процесі стає висока ціна. Натомість застосування

Інтернет-технологій поліпшує якість навчального процесу і разом з тим, підтримує можливість отримання знань шляхом навчання за допомогою відео, онлайн-посібників, що доступні для кожного, а також допомагає учням навчатись з будь-якої точки планети в будь-який момент часу.

2. Взамодія між учнем і вчителем. Всесвітня мережа дає можливість учням постійно підтримувати зв’язок із своїми колегами чи викладачами шляхом електронного листування чи з використанням додатків для обміну повідомлень. Тим самим вивчаються можна збагатитись новими ідеями та знаннями.

3. Інструмент ефективного навчання. Вчителі можуть використовувати Інтернет-ресурси в якості навчальних посібників, ділитись в мережі власними напрацюваннями. Завдяки застосуванню різноманітних навчальних засобів, наприклад, презентацій, фото-, відео- та аудіоматеріалів, анімацій, процес отримання знань стає цікавішим і більш захопливим, бо ефективно притягує увагу.

4. Легкість доступу до знань.Для всіх учасників начального процесу спрощено доступ до знань: учень має доступ до якісний та безкоштовних начальних ресурсів на різноманітних платформах або ж може за допомогою мережі купити ще якісніші матеріали; викладач, використовуючи Всесвітню мережу, для надання своїм учням додаткві матеріали, дистанційно провести освітню віторину, інтерактивний урок, а також записати свою лекцію і надавати її учням для повноцінного повторення вивченого матеріалу, замість перечитування своїх заміток.

1.4. Огляд існуючих категорій геолокаційних додатків

Впродовж історії розорбки додатків для мобільних пристроїв геолокаційні засоби не втрачають своєї популярності. За особливостями застосування в них геолокаційні сервісів *LBS* (*Location-based services*), додатки цього виду можна розподілити на наступні категорії:

1. Геолокаційні додатки для надання інформації про найближчі об'єкти і відстань до них.

Це найпростіший для розробки вид геолокаційних додатків. Головне у процесі реалізації такого виду додатків – спроектувати алгоритм і розрахувати за формулою відстань між двома координатами: поточним місцезнаходженням користувача і локацією об'єкта з бази даних. Приклади таких мобільних додатків є *2GIS* або *Yillio*.

2. Геолокаційні додатки, що відображають розташування інших осіб.

Особливістю даного виду є проектування алгоритмів оновлення розташування всіх користувачів системи на сервері. Основна складність полягає у балансі між частотою передачі координат на сервер і енергоспоживанням мобільного пристрою: зі зростанням швидкості оновлення даних різко підвищуться навантаження на сервер і значно зменшується час автономної роботи пристрою користувача. Рішенням проблеми може стати застосування алгоритмів фільтрації даних, що отримуються з *GPS* датчиків. Приклади: *Swarm* або *Patrolife*.

3. Екскурсійні програми та додатки-путівники.

Головна особливість полягає у можливості роботи мобільного додатку без підключення до мережі. Типовий користувач цього виду додатку – турист. Найчайстіше дані мобільного додатку використовуються в роумінгу під час подорожі або без засобів зв'язку. У разі, якщо потрібно працювати в *offline* режимі, слід розумно обирати формат зберігання карт в додатку. Вибір полягає між растровим і векторним типом карт. Найчастіше векторний формат доцільніше застосовувати через його компактність. Растрові карти краще рішення для невеликих територій, наприклад, парки, музеї, завдяки можливості намалювати на таких картах реалістичні об'єкти. Прикладом такого додатку є *Azbo*.

4. Геолокація з доповненою реальністю.

Найскладніший для розробки вид геолокаційний додаток. Крім роботи з *GPS* пристроями, необхідними для отримання інформації про місцезнаходження, додавати розпізнавання графічних об'єктів і обробляти інформацію з гіроскопа. Цей вид додатків набирає популярності серед освітніх програм. Користувачі рухаються між експонатами виставки чи музею, наводять камери смартфонів на об'єкти, що їх цікавлять і бачать подробиці виділеного об'єкта. Для реалізації таких

функцій розробникам потрібно навчити додаток розпізнати зображення, отриманого з камери мобільного пристрою користувача. Для цього геолокація може використовуватися як додатковий механізм, що підвищує точність процесу розпізнавання. Також для спрощення можня впроваджувати встановлення спеціальних маркерів – *QR* кодів. Прикладом додатку з доповненою реальністює відомий *Pokemon GO*.

5. Навігатори.

Для даного виду додатків, крім відображення карт, додаться функція побудови маршруту. Поруч із переліченими вище додатками, які ведуть користувача вулицями міста, набирають популярність додатки для побудови маршрутів в торгових центрах. Користувачі буду маршрут до деякої секції в торговому центрі. Основна розробки полягає у визначенні локації користувача в закритому приміщенні. Впроваджується систему «маяків», за якими додаток визначає, де в даним момент знаходиться користувач, і чим більше їх використано в даному приміщенні, тим точніше визначиться місце розташування користувача. Прикладом є *Google Maps*.

Незалежно від виду геолокаційного додатка, є ряд питань, які потребують уваги:

1. Автономність роботи програми.

Часто можна спостерігати додатки, що в процесі своєї роботи сильно нагрівають пристрій користувача і спричиняє дискомфорт. Причина цього в неактуальних алгоритмах обробки зібраних з датчиків даних, відсутності методів кешування і фільтрації інформації.

2. Складність проектування інтерфейсу користувача.

На карті потріюно відобразити велику кількість інформації на різних рівнях масштабування. Проектувальниками і дизайнерами проводяться декілька ітерацій *Usabilty* тестування, щоб створити дійсно зручний у користуванні додаток.

3. Забезпечення якості.

Якщо обмежитись лише емуляцією розташування, не завжди можна знайти слабкі місця і можливі дефекти. Тому фахівцям з якості потрібно відтворювати багато тестових сценаріїв в реальних умови експлуатації.

Процес проектування геолокаційного додатку вимагає більш детального опрацювання логіки й інтерфейсу, а також наявності перевірених алгоритмів рішення. Але попри складність процесу створення геолокаційних додатків, вони приносять велику користь.

1.5 Висновки до розділу

У даному розділі було розглянуто поняття інформаційного додатку. Він являє собою прикладну програмну підсистему, що орієнтується на збір, зберігання, пошук та обробку інформації. Типовий інформаційний додаток включає в себе:

1. Діалогове введення і виведення інформації.
2. Логіка діалогу.
3. Прикладна логіка обробки даних.
4. Логіка керування даними.
5. Операції, виконувані над файлами чи базами даних.

В інформаційному додатку, що взаємодіє з базою даних, прийнято виокремлювати наступні функціональні компоненти: засоби представлення, логіка представлення, компонент прикладної логіки, компонент логіки керування даними, операції, що здійснюються над базою даних, компонент файлових операцій.

Інструментальні програмні засоби розробки інформаційних додатків повинні забезпечувати наступні важливі властивості: підтримку багатоплатформності; незалежність від виробника; уніфікацію засобів розробки; створення надійного і якісного програмного забезпечення; підтримку розробленого програмного засобу протягом усього часу життя; проектування з використанням різних сучасних методик; ведення версій; підтримку вебтехнології.

Також проведено ознайомлення з поняттям автоматизації задач. Автоматизація забезпечує зменшення ручних повторюваних операцій. Врешті

автоматизація необхідна, коли обсяги інформації, яку необхідно опрацювати і обробити, надто великі, для обробки людиною вручну.

В результаті впровадження автоматизації можна досягти: економії робочого часу працівника; спрощення роботи з клієнтами; зменшення похибки, зумовленої людським фактором; збільшення швидкості та якості роботи по обслуговуванню клієнтів; збільшення швидкості отримання звітів та статистики; пришвидшення процесу підготовки документів, рахунків, актів; захист цінних конфіденційних даних.

Проведено огляд існуючих категорій геолокаційних додатків, а саме: додатки для надання інформації про найближчі об'єкти і відстань до них; геолокаційні додатки, що відображають розташування інших осіб; екскурсійні програми і додатки-путівники; геолокація з доповненою реальністю; навігатори. Виявлено ряд питань, які потребують уваги: автономність роботи програми; складність проектування інтерфейсу користувача, забезпечення якості.

РОЗДІЛ 2
ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ ДОДАТКІВ

2.1. Поняття мобільного додататку

Поява поняття мобільного додатку є результатом технологічних інновацій. Впродовж кількох останніх десятиліть застосування інформаційних технологій зазнало чіткого переходу зі стаціонарних комп'ютерів до портативних мобільних пристроїв. Цьому розвитку посприяв зріст продажу смартфонів, планшетних комп'ютерів, який, до того ж, уже багато років перевищує продажі страндартних настільних персональних комп'ютерів (ПК).

Поки молоде покоління набуває цифрової грамотності, хмарні обчислення продовжують витісняти стаціонарні. Обчислювальні роботи покинули межі офісів та домівок, таку діяльність можна здійснювати будь-де і будь-коли. Мобільні програми з'явилися завдяки тісному переплетенню мережі Інтернету з засобами масової інформації та інформаційних технологій.

На сьогоднішній день проектування та розробка мобільних додатків займає провідні місця у сфері надання інформаційних послуг. Мобільні додатки користуються попитом серед сотень тисяч користувачів.

Мобільний додаток – це вид додатку, що створений ддя запуску на мобільному пристрої, наприклад, смартфон або планшет. Типовий мобільний додаток використовує мережеве підключення для роботи з віддаленими обчислювальними ресурсами Незалежно від того, що програмний продукт такого виду, як правило, є невеликим програмним блоком з конкретним обмеженим функціоналом, мобільному додатку вдається надати користувачу якісну послугу. Мобільні додатки часто слугують для надання послуг, подібних тим, що доступні на персональному комп'ютері.

На сучасному ринку мобільних додатків є дві домінуючі платформи.

1. Платформа *iOS* від *Apple Inc*: *iOS* платформа – це операційна система, яка працює на продуктах від *Apple.*

2. Платформа *Android* – це операційна система, яка працює від *Google*. Операційною системою *Android* користуються не лише пристрої *Google*, а й багато інших виробників устаткування, що розробляють смартфонів та інші смарт-пристрої.

Хоча у цих двох платформ можна прослідкувати подібні риси в процесі створення програмних продуктів, розробка для *iOS* проти розробки для *Android* передбачає використання різних *SDKs* (*software development kits* – набори для розробки програмного заюезпечення) та різного ланцюжка інструментів розробки. Принциповою відмінністю є використання компанією *Apple* операційної системи *iOS* виключно для власних пристроїв, компанія *Google* робить *Android* доступною платформою для інших компаній за умови, що вони задовольняють деяким вимогам, таким як додавання певних програмних продуктів *Google* на пристрої, які вони розробляють. Розробники мають можливість розробляти мобільні продукти для пристроїв, орієнтуючись на обидві платформи.

Мобільні додатки забезпечують швидший доступ до вмісту та плавну взаємодію. Щодо конкретних потреб бізнесу, існують загальні сценарії, коли створення додатка є найкращим рішенням: якщо замовник планує налаштувати функції, пов’язані з функціоналом мобвльного пристрою, наприклад, камера, *GPS*, дзвінок за допомогою доступу до контактів. В такому разі більш доцільною буде саме розробка додатку, а не вебсайту.

2.2. Види мобільних додатків

Загалом мобільні додатки можна розділити на три основні групи: нативні додатки, гібридні програми та вебдодатки. Кожен із перерахованих підходів має як ряд перваг, так і недоліків. В процесі пошуку правильного підходу до реалізації бажаного проекту, розробники, опираються на користувацький досвід, обчислювальні потужності та власні навики, необхідні для розробки мобільного додатку, бюджет і часові обмеження на релізацію та ресурси, доступні для обслуговування програмного продукту.

1. Нативні додатки.

Нативні додатки – мобільні програми, спеціально створені для певного типу операційної системи. Їх називають нативними, бо вони належать певному пристрою або платформі. Для створення мобільного додатку такого виду розробник використовує конкретну мову програмування, залежно від пристрою, під який створюється програмний продукт. Наприклад, *Objective-C* для пристроїв *Apple* або *Java* для пристроїв *Android*. Програма, написана на *Objective-C* не запуститься на пристрої *Android*, а програма мовою *Java* не працюватиме на пристрої *Apple*. Однак суть такого методу полягає у створенні найшвидшого додатку для конкретного пристрою. Розробник працюватиме з різними наборами для розробки програмного забезпечення для кожного середовища, що забезпечує повний доступ до всіх елементів керування пристроєм, зокрема, контакти, камера, датчики тощо (*Xcode* для *Apple* та *Eclipse* для *Android*), щоб створити кінцевий продукт, наприклад, файл *apk* для пристроїв *Android*.

Проблемою розробки нативних додатків є необхідність у вузьконаправлених висококвалафікованих спеціалістах. І хоча, є спеціалісти з розробки мовами *C* або *Java* , які зокрема і застосовуються для програмування нативних додатків, але існує не так багато розробників, які знають спеціальні версії цих мов та орієнтуються в їх середовищах розробки. Кваліфікованих розробників важко знайти і утримувати в штаті через високий попит на них.

До додатків можна отримати доступ через відповідні магазини додатків, наприклад, додатки *Android* користувач може завантажити на свій пристрій через магазин *Google Play*, додатки для *iOS* – в *AppStore*.

Мобільний додаток для платформи *Android* можна створити з використанням будь-якого комп’ютера, включаючи *Mac*, на відміну від розробки мобільних додатків для *Apple*: кінцевий продукт, який завантажується в *AppStore*, має обов’язково компілюватись на комп’ютері *Mac*.

Компанія *Apple* впровадила суворі правила, яких необхідно дотиримуватись, щоб потрапити до *AppStore*. *Google Play* застосовує лише кілька правил щодо форми та вмісту.

До переваг нативних додатків можна віднести:

- швидкі для конкретних пристроїв;
- легко розповсюджуються в *AppStore* або *Google Play;*
- програмний продукт інтерактивний та інтуїтивно зрозумілий;
- легко комунікуються з функціями пристрою.

До недоліків нативних додатків можна віднести:

- побудовані лише для конкретної платформи;
- для розробки такого виду мобільних додатків використовуються складні для вивчення мови програмування: *Swift* та *Java*;
- громіздкий і тривалий процес розробки;
- складність в управлінні.

2. Гібридні додатки.

Гібридні програми використовують для розробки продукту комбінацію технологій *HTML, CSS* та *Javascript*, а потім вони інкапсулюються в контейнер типу *PhoneGap* або *Cordova*, який пов'язує код із пристроєм і дозволяє йому працювати на пристрої в якості нативного додатку. Далі готовий додаток компілюється у вигляді інсталяційних пакетів для кожної мобільної операційної системи. За допомогою *PhoneGap* або будь-якого подібного програмного забезпечення можна використовувати плагіни для доступу до різних функцій телефону, таких як камера або супутникова система навігації (*GPS*). Гібридні програми, як правило, дещо повільніші та не настільки плавні, як їх нативний аналог. Але вони є гарним рішенням для створення програмних продуктів, які потребують повного доступу до пристрою, але не вимагають високої продуктивності.

Гібридна програма – просто вебдодаток, що працює у вбудованому середовищі браузера, значну частину її програмного коду можна використати для реалізації мобільного додатку.

Як і нативний додаток, гібридний може використовувати функціонал пристрою, на якому працює. Додаток можна завантажити на пристрій через магазин *Google Play* або *AppStore*. І хоча код такого мобільного додатку можна запрограмувати на будь-якому комп'ютері, для остаточної компіляції продукту

перед завантаженням в *AppStore* розробнику все одно знадобиться *Mac*, і все одно знадобиться *SDK* для кожного пристрою, для якого розробляється програмний продукт.

Оскільки ефективність рендерингу та продуктивність мобільних браузерів невпинно зростає, розробка гібридних додатків є гідною альтернативою для розробників, які хочуть швидко проектувати мобільні додатки.

Існують веб-сайти, присвячені створенню додатків за допомогою форм, що дозволяють просто заповнивши деякі поля або перетягнувши програмне забезпечення, яке дозволяє створювати безліч різних додатків. Але вкрай малою є ймовірність того, що їх приймуть до *AppStore*.

До переваг гібридних додатків можна віднести:

- легкість у побудові;
- значно дешевші за нативні програми;
- додаток незалежний від платформи і пристрою;
- не потребують браузера;
- зазвичай отримують доступ до утиліт пристрою за допомогою прикладного програмного інтерфейсу;
- розробка значно швидша за нативні додатки.

До переваг гібридних додатків можна віднести:

- розробка гібридного додатку вимагає більше матеріальних ресурсів за розробку вебдодатків;
- швидкість такого виду додатків буде значно нижчою за швидкість нативного додатку;
- менш інтерактивні, ніж нативні додатки.

3. Вебдодатки.

Вебдодатки являють собою вебсторінки, що розміщені на звичайних вебсерверах, але їх компактний розмір дозволяє зручно вмістити функціонал у смартфоні. Такий пограмний продукт розробляться з використанням будь-якого числа технологій, включаючи *PHP*/*MySQL* та інших технологій баз даних, оскільки їх основою є вебсайт. Вебдодатки не використовують *SDK* для пристроїв, оскільки,

як і будь-який звичайний вміст вебсайту, вони завантажуються на вебсервера. Проте без використання *SDK*, вони не мають доступу до функціональних можливостей пристрою, таких як, наприклад, камера. Вебдодаток має коректно відображатись на будь-якому пристрої, на якому його запущено, оскільки він переглядається аналогічно будь-якій іншій вебсторінці: у браузері чи на пристрої. Сучасні вебдизайнери для створення макетів сторінок вебдодатку користуються методами адаптивного вебдизайну, що використовують медіа-запити та інші методи, що дозволяють контенту підлаштуватись під різну ширину екранів пристроїв.

До переваг вебдодатків можна віднести:

- зниження витрат бізнесу;
- немає необхідності в завантаженні та встановленні додатку на пристрій;
- немає необхідності в створювати проекти для певних операційних систем;
- доступність, оскільки для користування такого виду додатком достатньо лише з’днання з мережею;
- розробка вимагає менше матеріальних ресурсів.

До недоліків вебдодатків можна віднести:

- прослідковується пряма залежність від швидкості з’єднання з мережею Інтернет;
- примітивний інтерфейс;
- розробка займає більше часу;
- ризик небезпеки.

2.3. Переваги та недоліки мобільних додатків

Використання компанією мобільних додатків принесе їй ряд переваг, як в загальному, так і локально, наприклад, підприємствам, що надають послуги культури, освіти, туризму, охорони здоров’я.

Грамотне користування перевагами впровадження мобільних додатків для компанії може поліпшити багато чинників: збільшення продажів, оптимізація маркетингових кампаній тощо.

Нижче наведено основні переваги:

1. Створення іміджу бренду або затвердження вже існуючого імені.

Впровадження компанією мобільного додатка для просування своїх товарів і послуг допоможе виділити її поміж конкурентів, затвердити ім’я вже існуючого бренду, що, в свою чергу, може значно підняти довіру клієнта шляхом природної взаємодії.

2. Потужний інструмент маркетингу.

Додаток дозволяє рекламувати товари і послуги фірми, надсилати *push*-повідомлення і тримати клієнта в курсі всіх справ: повна інформація про підприємство, його продукти, доступ до оновлень в будь-який момент і можливість ділитись інформацією на своїх сторінках у соціальних мережах, дає клієнтові повну свободу. Якщо споживачеві подобається функціонал мобільного додатка компанії, він просуватиме її діяльність, поділившись із родиною чи друзями.

3. Лояльність клієнтів.

Після моменту завантаження корситувачем мобільного додатку на свій пристрій між клієнтом і компанією, що надає йому послуги, починається взаємодія. Поки додаток завжди доступний на пристрої клієнта, разом з тим зростатиме ймовірність того, що він буде ним частіше користуватись, створюючи прямий зв’язок.

4. Більш висока швидкість, порівняно з сайтом.

Коли мобільний додаток встановлено на смартфоні чи планшеті користувача, він має швидший доступ до продукції компанії, оскільки програма працює швидше за вебсайт

5. Збільшена видимість.

Після виходу розробленого додатку на найвідоміші цифрові вітрини, зокрема *AppStore* та *Google Play*, послуги компанії стають доступними мільйонам потенційних споживачів.

6. Створення засобу продажу.

Зазвичай користувач завантажує та встановлює на свій пристрій мобільний додаток компанії, коли товари чи послуги цієї фірми його зацікавили і він виявив бажання придбати у неї певний продукт. В цей самий момент він стає потенційним постійним клієнтом даної компанії, створюючи тим самим канал продажу.

7. Оновлення та звіти живого ринку.

Набір функціональностей мобільного додатку дозволяє в режимі реального часі вивчати звіт про ринок компанії, тим самим з'являється можливість економити час на на прийняття важливих рішень, оскільки дізнатись показники можна у будь-який момент часу.

8. Технологічність.

Ще однією з особливих опцій, що передбачає наявність мобільного додатку, є можливість доступу до периферійних пристроїв смартфонів і планшетів користувача. Наприклад, доступ до супутникової системи навігації або камери тощо. З їх допомогою можна аналізувати інформацію про споживачів, а також підібрати індивідуально пропозиції та акції під смак кожного клієнта, в чому полягає особлива цінність своєму продукту. Компанія матиме лояльну базу споживачів, якщо її мобільний додаток спрощує їм процес пошуку знижок чи доволяє відслідковувати замовлення в режимі реального часу. Такі відносини зі споживачем підвищують ймовірність того, що співпраця буде довгостроковою.

9. Надання способу комунікації.

Мобільний додаток може працювати в якості каналу зв'язку між компанією і споживачем її послуг. Клієнт можна задавати питання, здійснювати запит на обслуговування.

10. Унікальний функціонал.

У мобільному додатку можна поєднати бізнес із розвагами або несподіваними для користувача функціями як, наприклад, доповнена реальність.

Глобалізація та технології змінили і продовжують змінюювати світ, і слід повноцінно скористатися цими змінами. Мобільні додатки є важливою частиною нового ринку.

Мобільні додатки вже встигли стати частиною сучасного життя і культури. Але дефекти в них можуть виникати будь-де і носити абсолютно різний характер. Не можна виділити багато недоліків мобільних додатків, але вони все ж є. Далі наведено деякі з них:

1. Проблема сумісності.

Задля забезпечення коректної роботи мобільного додатку, він має відповідати вимогам конкретної платформи. Тобто операційна система, наприклад, *iOS, Android* чи *Windows* буде вимагати окремої версії програмного засобу.

2. Недосконалий захист транспортного рівня передачі.

Транспортний рівень забезпечує додаткам або верхнім рівням стека – прикладному й сеансовому – передачу даних з тим ступенем надійності, що їм необхідний. Для захисту інформації у протоколах *TLS/SSL* знадобляться надійні алгоритми криптографічного захисту. В програмний код слід додавати попередження для користувача, щоб міг взаємодіяв із конфігурацією зашифрованого з'єднання.

3. Підтримка та обслуговування.

Мобільний додаток, що створений для кількох платформ, буде більше ресурсозатратним в плані його обслуговування. Підтримка вже розробленої програми вимагає зусиль, часу та матеріальних затрат. Процеси забезпечення оновлень та виправлення помилок, проблем сумісності з деякими пристроями є постійними. Крім цього, слід інформувати користувача про оновлення і спонукати його завантажувати оновлену версію програмного продукту.

2.4. Етапи розробки мобільних додатків

Розробка мобільного додатку – це процес створення програмних модулів, що працюють на мобільному пристрої. Додаток такого виду, як правило, використовує підключення до мережі для роботи з віддаленими обчислювальними ресурсами. Отже, процес розробки програмного продукту для мобільного пристрою включає в себе створення програмних пакетів, що встановлюються, реалізацію серверної

частини, такої як доступ до даних за допомогою *API,* та перевірка програми на помилки на мобільних пристроях.

Над створенням мобільних додатків, як правило, працює ціла команда експертів з маркетингу, інформаційних технологій, мобільного контенту та дшзайнерів. Розробка навчального мобільного додатку починається зі збирання та аналізу інформації. Інтернет маркетолог аналізує ринок подібних мобільних додатків, збирає відгуки на існуючі додатки, вивчає потреби користувачів, цільову аудиторію. В результаті він надає детально оформлений звіт, до якого включено всі дані: про ринок, про конкуректів, створено портрет користувача і розроблено позиціонування майбутнього програмного продукту. Важлива абсолютно кожна дрібниця: від дизайну користувацького інтерфейсу до функціональних вимог.

Такий підхід дозволяє створити мобільний додаток, що користується попитом.

Можна виділити основні етапи розробки:

1. Обговорення ідеї майбутнього продукту.

Замовник, як правило приходить в компанію розробників, щоб реалізувати власну ідею програмного продукту. На цьому етапі важливим є діалог замовника з бізнес-аналітиком, щоб допрацювати ідейну складову проету, враховуючи побажання замовника.

2. Оцінка проекту.

Після оцінки проекту, здійсненої комадою розробників, замовник отримає попередню комерційну пропозицію на розробку додатку.

3. Створення прототипу.

Знаючи, чого чекають користувачі від мобільного додатку, розробники починають процес створення прототипу – своєрідний скелет дизайну, де зображено кожну деталь майбутнього програмного продукту (рис. 2.1). Коректно побудований інтерфейс є запорукою того, що користувач зможе швидко зорієнтуватись яким саме чином програмний продукт зможе вирішити його проблему. На цьому етапі замовник має можливість ознайомитись із функціоналом майбутнього мобільного додатку без його програмної частини.

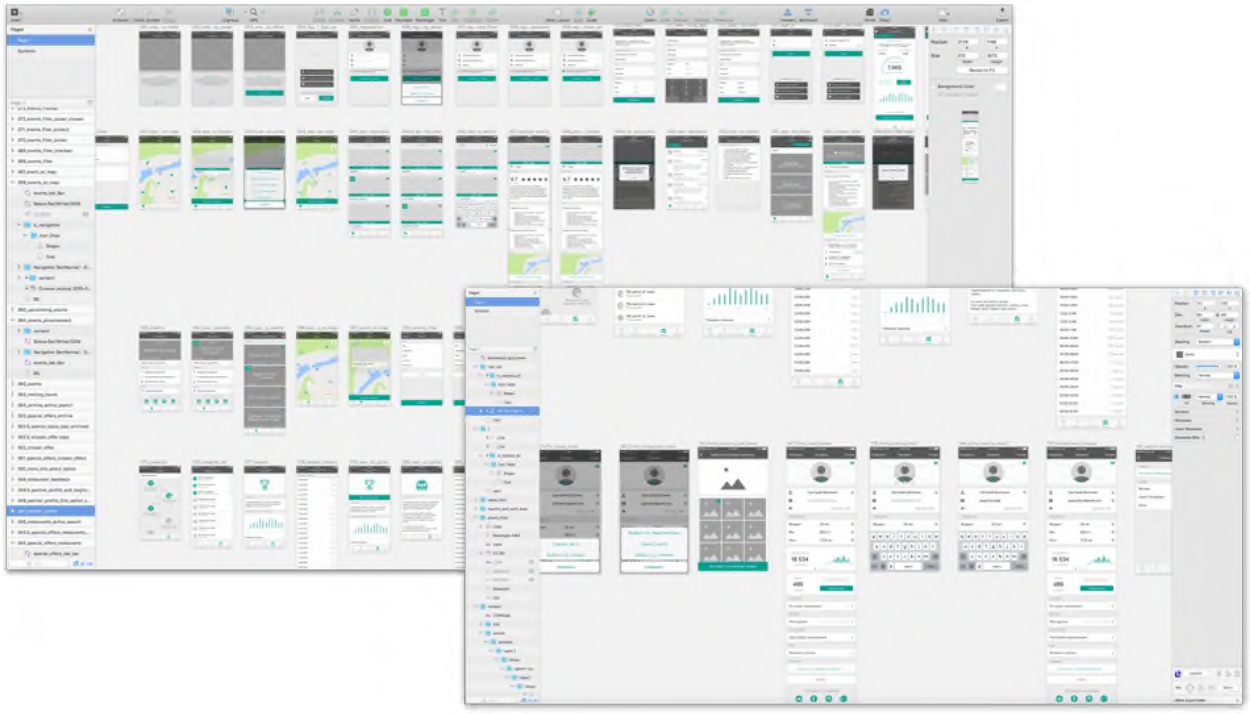

Рис. 2.1. Прототип мобільного додатку

4. Розробка концепції дизайну продукту.

Важливо створити саме той дизайн програмного продукту (рис. 2.2), що буде вирізняти його серед конкурентів. Графічні рішення мають бути, в першу чергу, зручними і зрозумілими для користувача.

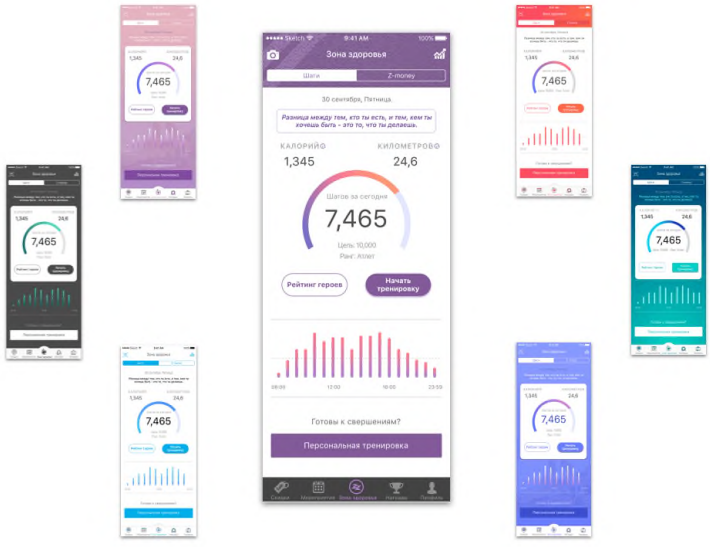

Рис. 2.2. Приклади макетів дизайну мобільного додатку

5. Розробка технічного завдання і клієнт-серверної архітектури.

Для розробки і тестування продукту важливо на базі затвердженого інтерактивного прототипу і дизайну продукту створити документацію, необхідну розробникам і тестувальникам. В технічому завданні необхідно передбачити всі сценарії, переходи між екранами та стани екранів. На цьому ж етапі спеціалісти мають пропрацювати високорівнему архітектуру проекту та модель зберігання даних (рис. 2.3). Від цього етапу буде залежати те, як побудована програмна частина, швидкість роботи мобільного додатку, конфігурація запитів. В технічому завданні також описано всі системи, з якими буде необхідно синхронізуватись.

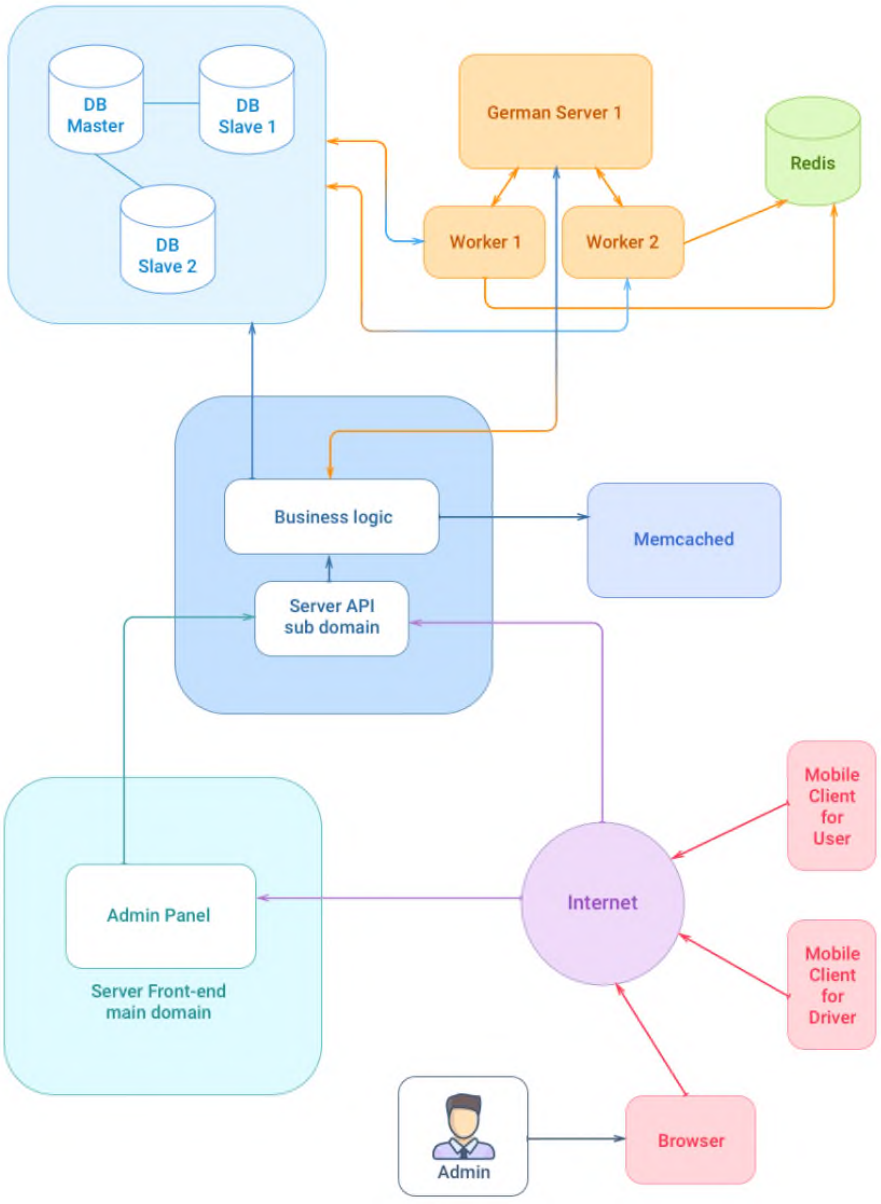

Рис. 2.3. Розробка архітектури додатку

6. Верстка візуальної частини.

Затверджений дизайн передаться фронтенд розробникам, що складає окремі частини в цілісну картину. В результаті створюється код, який доступний для перегляду у веб-браузері.

7. Програмування та тестування.

Процеси програмування та тестування тісно зв'язані між собою, тому доречно буде їх об'єднати в один комплексний процес.

Проект поетапно програмується. Після завершення реалізації певної програмної частини, мобільний додаток відправляється на тестування. Процес тестування може включати перевірки на помилки (рис. 2.4) в різних умовах, за наявності яких слід повертати додаток програмістам, поки всі несправності не буде усунено.

Tester's Name:					
	XP	Vista	7	8	Comments
Installing the Application					
Install the application	Pass				
Running the Application					
Run the application from the installer	Fail				Bug #s
Run the application from the shortcut	Warning				
Restart and verify application launches at login with "Launch on Startup" enabled	Blocked				
Restart and verify application does not launch at login with "Launch on Startup" disabled	Skipped				

Рис. 2.4. Приклад тестової документації для програмного продукту

8. Публікація мобільного додатку.

Після завершення всіх етапів розробки мобільного додатку, його готовий програмний продукт потрібно підготувати до публікації. Для цього потрібно оформити опис цього мобільного додатку, графіку відповідного до вимог магазинів *Google Play* і *AppStore*. В інакшому разі мобільний додаток не пройде етап модерації.

9. Супровід готового програмного продукту.

Після введення в експлуатацію продукт має перейти на стадію технічої підтримки.

2.5. Життєвий цикл розробки мобільного додатку

Існує два основні компоненти мобільного додатка, що тісно пов’язані між собою:

- *Front-end* мобільного додатку, що знаходиться на мобільному пристрої;
- *Back-end*, що підтримує мобільний інтерфейс.

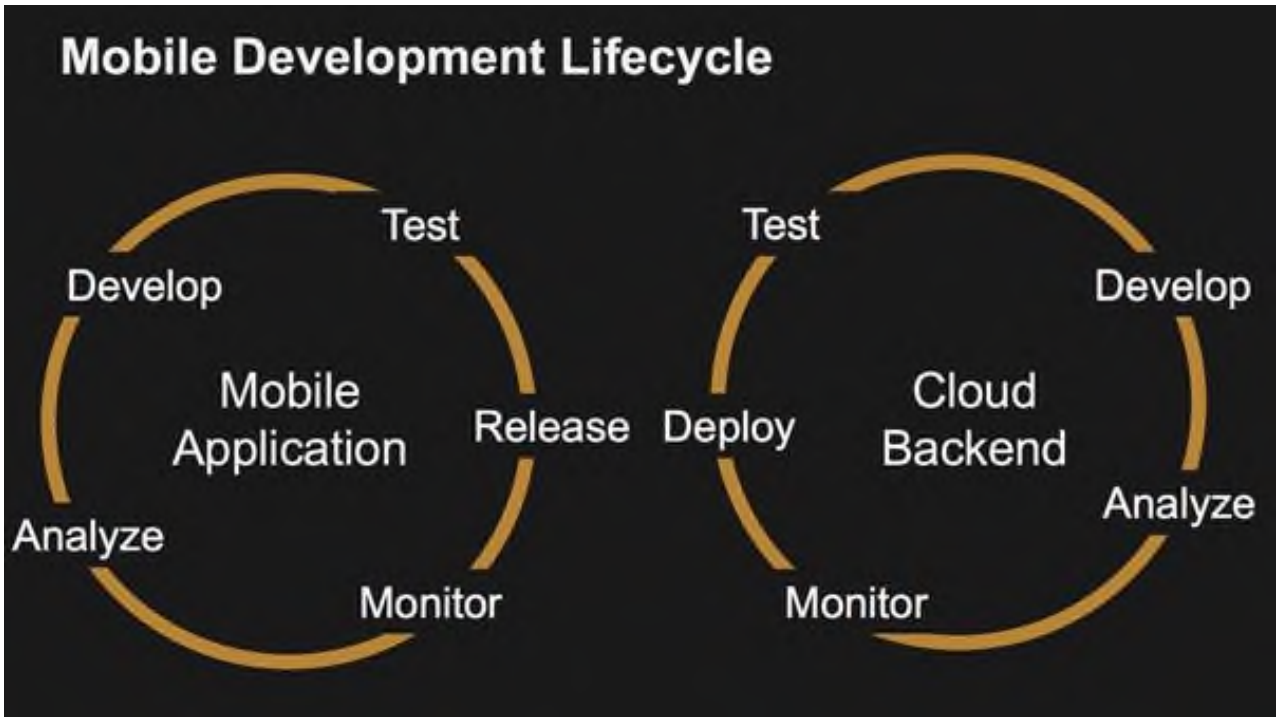

Рис. 2. 5. Життєвий цикл мобільної розробки

У перші дні епохи сучасних додатків для смартфонів мобільні додатки пройшли подібний шлях, як і перші веб-сайти. Спочатку додатки та сайти діяли лише в якості статичної реклама бренду, компанії, товару чи послуги.

У ході покращення можливостей зв’язку та мережі додатки дедалі більше почали підключатися до джерел даних та інформації поза їх межами, а додатки ставали дедалі динамічнішими, оскільки вони могли оновлювати свій візуальний

інтерфейс та контент даними, отриманими через мережу. шляхом запитів до баз даних.

В результаті, мобільні *front-end* програми все частіше покладаються та інтегруються із *back-end* (серверними) сервісами, які забезпечують споживання даних через мобільний інтерфейс. Такого виду дані можуть включати, наприклад, інформацію про товари для програм електронної комерції або інформацію про рейс для додатків для подорожей та бронювання. Для мобільної гри дані можуть включати нові рівні або зв'язок з іншими реальними гравцями в режимі реального часу.

Мобільний *front-end* отримує дані з *back-end* за допомогою різноманітних сервісних викликів, таких як *API*. У деяких випадках ці *API* можуть належати і бути застосованими тим самим суб'єктом, який розробляє мобільний додаток. В інших випадках *API* може контролюватися третьою стороною, а доступ до мобільного додатку надається за комерційною угодою.

Наприклад, розробник може отримати соціальні засоби комунікації або рекламний контент, зв'язавшись із засобами масової інформації або службами реклами. У такому випадку розробнику, можливо, доведеться підписати контракт, щоб отримати облікові дані та ключ, що надає права доступу до *API* та визначає, як цей розробник може ним користуватися, скільки це буде коштувати або як часто його можна буде викликати, або скільки даних можна запитати протягом якого періоду часу.

Для більшості програм розробники мобільних додатків відповідають за створення та управління *back-end* сервісами для розроблюваних додатків.

Front-end мобільного додатка – це візуальна та інтерактивна частина програми, призначеня для користувача. Зазвичай візуальна частина відображаться на пристрої, або є принаймні піктограма, що представляє програму, яка знаходиться на головному екрані або закріплена в каталозі програм пристрою. Додаток можна завантажити безпосередньо на пристрій з магазину додатків платформи або отримати доступ через браузер пристрою, як у випадку з вебдодатком.

Залежно від розміру команди, яка розробляє додаток, у розробці та розробці інтерфейсного мобільного додатка може бути задіяно багато різних працівників. Розмір команди може варіюватися від одного розробника, який робить все, що пов'язано зі створенням програми, до більшої кількості вже вузьконаправлених спеціалістів.

Наприклад, можуть бути окремі графічні дизайнери, які відповідальні за створення візуальних елементів програм, таких як піктограми, фони, кольори та інші частини програми. Команда може також мати *UI/UX* архітекторів, які працюють над компонуванням компонентів, як вони взаємодіють між собою та користувачем. У випадку певних типів ігор команда може включати розробників графіки руху і навіть інженерів, які розробляють двигуни, які регулюють фізику руху компонентів у додатку, як автомобіль у гоночній грі.

Незалежно від розміру команди розробників, важливим елементом реалізації є побудова логіки програми, яка відповідає за здійснення мережевих викликів до серверних служб, отримання даних та оновлення даних у *back-end* системах новою інформацією, що генерується з додатку.

Доступ до *back-end* служб, як правило, здійснюється через різноманітні прикладні програмні інтерфейси *(API)*. Існують різні типи *API*, такі як *REST* та *GraphQL*, а також існує багато варіантів засобів та стилів доступу до них. Хоча деякі *API* службових інтерфейсів доступні безпосередньо для програми за допомогою дзвінків на самій платформі, багато спеціалізованих служб повинні бути інтегровані в програму за допомогою набору для розробки програмного забезпечення *(SDK)*. Як тільки *SDK* буде додано до програми через середовище розробки, програма зможе використовувати *API,* що визначені в *SDK*.

Прикладом *back-end* служби для мобільного інтерфейсу може бути база даних, що містить інформацію, яка використовується в розроблюваному додатку. Для прямого доступу до бази даних розробник мобільних програмних засобів повинен знати мережеве розташування бази даних, протокол доступу до бази даних, облікові дані для аутентифікації та авторизації для доступу до даних та

конкретні команди бази даних, необхідні для виконання запитів на отримання необхідних даних.

Крім того, розробник може використовувати спеціалізований *API* під час взаємодії з базою даних. Розробник може знати лише параметри, необхідні для виклику методу для отримання або оновлення необхідної інформації. У деяких випадках розробник мобільних програмних засобів може розробляти ці *API* самостійно або використовувати визначення *API*, надане їм власником оператором *back-end* ресурсу.

Зазвичай *REST API* використовується для взаємодії з джерелами даних у хмарі, наприклад, хмарною базою даних. *API GraphQL* також є ще одним варіантом для розробників, оскільки спрощує роботу з серверними даними додатку. *GraphQL* забезпечує підтримку запитів через одну кінцеву точку *API* та схему даних, яка може бути використана для побудови та легкого розширення моделей даних, використовуваних у додатку.

Незалежно від того, яка *front-end* платформа або методологія розробки використовується, надання високоякісних мобільних додатків вимагає надійних сервісних послуг.

З огляду на критичну важливість *back-end* сервісів для успіху розроблюваного мобільного додатку, розробники мають кілька важливих архітектурних рішень, які потрібно врахувати. Ці рішення включають, які служби вони повинні будувати самостійно і які сторонні послуги вони повинні використовувати.

З метою підвищення власної продуктивності та ефективності розробники мобільних програмних засобів повинні створювати власні сервіси лише у тому випадку, якщо вони є дуже специфічними для домену програми та втілюють унікальну інтелектуальну власність. Крім того, навіть для служб, які вони створюють самі, вони майже завжди повинні використовувати хмарні сервіси для побудови та підтримки своєї внутрішньої інфраструктури.

Існують сотні хмарних та сторонніх служб, які розробники мобільних додатків можуть використати для пришвидшення своєї розробки. Однак навряд чи розробник зможе стати експертом у кожній із цих окремих послуг.

Натомість розробники мобільних пристроїв повинні шукати середовище для розробки, що полегшує їм швидку та легку інтеграцію, використання та використання найпоширеніших можливостей у своєму додатку, зберігаючи свободу користування багатьма доступними окремими послугами.

2.6. Висновки до розділу

В даному розділі було досліжено, що мобільний додаток являє собою вид додатку, що створений для запуску на мобільному пристрої, наприклад, смартфон або планшет та використовує мережеве підключення для роботи з віддаленими обчислювальними ресурсами Мобільні додатки часто слугують для надання послуг, подібних тим, що доступні на персональному комп’ютері.

Проведено ознайомлення з осоновними видами мобільних додатків, загалом їх можна розділити на три основні групи: нативні додатки, гібридні програми та вебдодатки. Кожен із перерахованих підходів має як ряд перваг, так і недоліків.

Було виділено основні етапи розробки мобільних додатків, а саме:

1. Обговорення ідеї майбутнього продукту.
2. Оцінка проекту.
3. Створення прототипу.
4. Розробка концепції дизайну продукту.
5. Розробка технічного завдання і клієнт-серверної архітектури.
6. Верстка візуальної частини.
7. Програмування та тестування.
8. Публікація мобільного додатку.
9. Супровід готового програмного продукту.

Також було досліджено життєвий цикл розробки мобільних додатків.

РОЗДІЛ 3
ТЕХНОЛОГІЇ СТВОРЕННЯ ГЕОЛОКАЦІЙНОГО ІНФОРМАЦІЙНОГО ДОДАТКУ

3.1. Вимоги до розроблюваного додатку

Розроблюваний інформаційний геолокаційний додаток створено для викладання та навчання в природних умовах, наприклад, національних, державних та місцевих парках. Додаток надає користувачам доступ до інформації в режимі реального часу під час вивчення і дослідження різних геологічних чи історичних ландшафтів. Програма використовує геодані для інтеграції інформації про розташування в мобільний додаток, а також надаватиме інформацію про місцезнаходження та інформацію для дослідн.

Дана розробка зосереджена на представленні трьох функціональних моделей.

1. Модель карти обробляє географічні дані для демонстрації тисяч точок в геоінформаційній системі. Важливо зауважити, що в багатьох випадках оновлення виконуються дистанційно підключеними користувачами.

2. Модель компоненту, що дозволяє користувачам досліджувати місцевість у межах маршрутів. Мобільний додаток надсилатиме повідомлення користувачеві, коли він зайде у геозону певного пункту. Після цього у мобільному додатку буде перелічено ряд запитаннь, на які користувач повинен відповісти. Мобільний додаток може редагувати інформацію про маршрут та оновлювати дані. Він також підтримує обробку запитань та відповідей користувачів за допомогою методу *Ajax*.

3. Компонент аналізу даних користувачами відповідей і допомагає дослідницькій групі отримати потрібну їм інформацію.

3.2. Засоби для реалізації мобільного додатку

У ході даної роботи для реалізації різних моделей було використано різні технології. Для створення базового статичного інтерфейсу користувача було

використано мову розмітки гіпертексту *HTML* і мову стилів *CSS*; для динамічної обробки даних – мову *JavaScript* та бібліотеку *jQuery*; для керування логічними функціями моделей – мову *PHP* та застосовано систему *MySQL* для управління базою даних.

1. Мова програмування *PHP* є надійним та ефективним інструментом для створення систем. Сфера застосування *PHP* розширилась до створення мобільних додатків. Чіткість дизайну, добре організовані модулі, філігранне обслуговування різноманітних технологій роблять її найбільш популярною мовою в Інтернет-галузі на даний час. Вона використовується для розробки динамічних інтерактивних веб-сторінок. Розроблюваний додаток вимагає високих інтерактивних здібностей для мобільних клієнтів. Переваги *PHP* зроблять розробку більш ефективною та простою з точки зору обслуговування. Завдяки широкому спектру функціоналу, набору фреймворків, що також розширяють його можливості, для *PHP* доступним є майже все: збір даних, програмування серверної частини, динамічна генерація контенту.

PHP доступна для використання майже на всіх операційних систем, також має підтримку для більшості серверів і баз даних.

Мова *PHP* в цілому орієнтується на серверні сценарії, тому може робити все, що може будь-яка інша програма *CGI*, наприклад збирання даних з форм, генерувати динамічний контент сторінки або надсилати та отримувати файли *cookie*. Але *PHP* може зробити набагато більше.

Є три основні напрями застосування *PHP*.

- Серверні сценарії є найбільш традиційним і основним цільовим полем для *PHP*. Для цього потрібні три конпоненти: парсер *PHP* (*CGI* або серверний модуль), вебсервер і веббраузер. Вебсервер потрібно запустити із підключеною установкою *PHP*. Доступ до вихідних даних програми *PHP* можна отримати за допомогою веббраузера, переглядаючи сторінку *PHP* через сервер.
- Сценарії командного рядка. *PHP*-скрипт можна створити для запуску без будь-якого сервера або браузера. Необхідним для цього є лише парсер *PHP*, щоб використовувати його таким чином. Цей тип використання ідеально підходить

для сценаріїв, які регулярно виконуються за допомогою *cron* в системах класу *UNIX* або планувальника завдань на *Windows*. Ці сценарії також можна використовувати для простих завдань по обробкці тексту.

- Написання десктопних додатків. Хоча *PHP* і не найкраща мова для створення десктопного додатка з графічним інтерфейсом користувача, але якщо розробник добре знає *PHP* і хоче використовувати розширені функції *PHP* у своїх проектах на стороні клієнта, також можна використовувати *PHP-GTK* для написання таких програми. Таким чином також можна розробляти росплатформні програми. *PHP-GTK* – це розширення *PHP*, і воно недоступне в основному дистрибутиві.

PHP може використовуватися у всіх основних операційних системах, включаючи *Linux*, у багатьох варіантах *Unix* (включаючи *HP-UX*, *Solaris* та *OpenBSD*), *Microsoft Windows*, macOS, *RISC OS* та інших.

На сьогоднішній день *PHP* підтримує більшість веб-серверів сьогодні. Сюди входять *Apache*, *IIS* та багато інших. До цього числа також включається будь-який веб-сервер, який може використовувати бінарний файл *FastCGI PHP*, наприклад, *lighttpd* та *nginx*. *PHP* працює або як модуль, або як процесор *CGI*.

Отже, з *PHP* розробник має свободу вибору операційної системи та веб-сервера. Крім того, є можливість вибрати процедурне програмування або об'єктно-орієнтоване програмування (ООП) чи поєднувати їх.

З *PHP* не обмежує вихідним *HTML*. Можливості мови включають виведення зображень, *PDF*-файлів і навіть флеш-фільмів (за допомогою *libswf* та *Ming*), створених на льоту. Ви також можете легко вивести будь-який текст, такий як XHTML та будь-який інший XML-файл. PHP може автоматично генерувати ці файли та зберігати їх у файловій системі, замість того, щоб роздруковувати їх, формуючи кеш на стороні сервера для динамічного вмісту.

Однією з найсильніших і найважливіших функцій *PHP* є підтримка широкого спектру баз даних. Написати веб-сторінку з підтримкою бази даних легко, використовуючи одне з розширень бази даних (наприклад, для *MySQL*), або використовуючи рівень абстракції, такий як *PDO*, або підключившись до будь-якої

бази даних, що підтримує стандарт *Open Database Connection* через розширення *ODBC*. Інші бази даних можуть використовувати *cURL* або сокети, наприклад *CouchDB*.

PHP також підтримує контакт з іншими службами за допомогою натупних протоколів, як *LDAP, IMAP, SNMP, NNTP, POP3, HTTP, COM* (у *Windows*) та інших. Можна також відкривати необроблені мережеві сокети та взаємодіяти, використовуючи будь-який інший протокол. *PHP* підтримує складний обмін даними *WDDX* практично між усіма мовами вебпрограмування. Говорячи про взаємозв'язок, *PHP* підтримує інстанціювання об'єктів *Java* та прозоре використання їх як об'єктів *PHP*.

PHP має корисні функції обробки тексту, що включає регулярні вирази, сумісні з *Perl (PCRE)*, а також безліч розширень та інструментів для синтаксичного аналізу та доступу до *XML*-документів. *PHP* стандартизує всі розширення *XML* на надійній основі *libxml2* та розширює набір функцій, додаючи підтримку *SimpleXML, XMLReader* та *XMLWriter*.

І існує багато інших цікавих розширень, які класифікуються як за алфавітом, так і за категоріями. Є також додаткові розширення *PECL*, які можуть бути зафіксовані або не зафіксовані в самому посібнику *PHP*, наприклад *XDebug*.

Світ технологій надає розробникам велику кількість мов сценаріїв для створення додатків та веб-сайтів, таких як *ASP, CGI, Perl* та *PHP*. Серед усіх них *PHP* є популярною і є широко використовуваною мовою сценаріїв.

Можна виділити переваги вибору *PHP* серед інших мов.

- Мова проста у вивченні. *PHP*-сценарії легко вивчати. Розробники легко засвоюють знання *PHP*. Це може бути пов’язано з тим, що *PHP* певною мірою схожа на *C* та *Java*. Зазвичай розробники вивчають *PHP* як першу мову сценаріїв. У зв’язку з цим, набагато простіше знайти квалафікованих спеціалістів, бо простота *PHP* робить її популярною серед розробників.
- Підтримка якості. Мову *PHP* легко вивчити. Так як вона є популярною, розробникам легко зв’язатись з експертами та отримати допомогу в Інтернеті, оскільки є багато блогів, форумів та *PDF*-файлів, що присвячені розробці мовою

PHP, де обговорюються проблеми, пов'язані з програмуванням *PHP*. Розробники можуть отримати допомогу з цих джерел.

- Гнучкість і свобода. Відкритий код дозволяє використовувати будь-який текстовий редактор для написання коду програм, включаючи *Emacs*, *jEdit*, *Notepad++* тощо. Немає обмежень для розробки унікальної програми. Мова сценаріїв високого класу сумісна з багатьма платформами; не обмежується операційною системою. Програму можна запустити на *Linux*, *Mac OS*, *Windows* тощо.

- Інтеграція. *PHP* використовується з багатьма додатками. Він добре інтегрується з *Memcache*, *MongoDB* та *Pusher*. Додатки розроблені для компаній, що займаються інформаційною, банківською діяльністю та охороною здоров'я, певним чином використовують *PHP* для реалізації.

- Масштабованість. У сфері інформаційних технологій поняття масштабованості є багатогранним. *PHP* дозволяє додавати або збільшувати розмір кластера, додаючи сервери. Це дає платформу для більшого зростання.

- Індивідуальна розробка. Веб-сайт, розроблений на *PHP*, можна налаштувати індивідуально. за допомогою PHP можна одержати вид і представлення бажаного продукту. Це дає достатню гнучкість для розробки та налаштування саме того продукту, що задовольним потреби замовника.

- Гнучкість бази даних. *PHP* є гнучкою для підключення до бази даних. Він може підключатися до різних баз даних, найбільш популярною і часто вживаною є *MySQL*. *MySQL* можна використовувати безкоштовно.

PHP є однією з найбільш широко використовуваних мов програмування в Інтернеті, і вона дозволяє робити набагато більше, ніж простий *HTML*. *MySQL* дозволяє легко створювати та змінювати бази даних на сервері. Поєднуючись разом, ці інструменти можуть створювати складні та потужні власні додатки та бази даних.

PHP – це мова сценаріїв, яку можна використовувати для створення інтерактивних сценаріїв. Ці сценарії виконуються на вебсервері, а потім результати повертаються до засобу перегляду за допомогою *HTML*. *PHP* дозволяє створювати

дуже інтерактивні веб-сайти, орієнтовані на користувачів. *MySQL* - це мова баз даних з відкритим кодом, що дозволяє створювати, редагувати та отримувати доступ до декількох баз даних на сервері. Поєднання цих двох важливих технологій дозволяє розробляти надійні програмні засоби.

PHP може збирати інформацію про форму від користувача, створювати та редагувати файли на сервері, надсилати та отримувати файли *cookie*, обмежувати доступ, шифрувати дані та багато іншого.

Для використання *PHP* та *MySQL* розробникові знадобиться доступ до веб-сервера. Якщо немає доступу до веб-сервера, доведеться встановити його на комп'ютері.

Щоб розпочати створення *PHP*-скриптів та баз даних *MySQL*, буде потрібно буде завантажити кілька основних інструментів. Хоча *PHP* і можна редагувати в будь-якому текстовому редакторі використання спеціального редактора кодування значно спростить життя розробнику.

Для використання *MySQL* розробнику потрібна буде її інсталяція на своєму вебсервері.

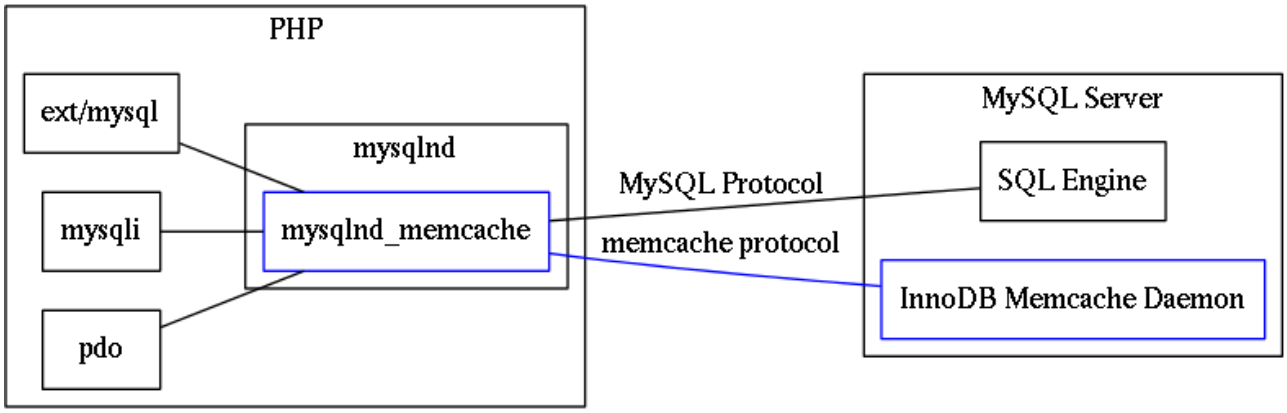

Рис. 3.1. Взаємодія *PHP* і сервера *MySQL*

2. *MySQL* – вільна система управління реляційними базами даних (СУБД). Вона працює в якості сервера, підтримуючи швидке і надійне виконання *SQL* запитів, багатопоточність та багатокористувацький доступ до ряду баз даних. Система *MySQL* відома у всьому світі як найбільш безпечна та надійна система управління базами даних, що використовується в таких популярних системах, як

WordPress, Drupal, Facebook та *Twitter*. У цій системі нам потрібно зберігати в базі даних тисячі точок та зображень. Характеристики *MySQL* здійснюють ефективну обробку даних.

MySQL - це безкоштовна база даних з відкритим кодом, яка спрощує ефективне управління базами даних, підключаючи їх до програмного забезпечення. Це стабільне, надійне та потужне рішення з розширеними функціями та перевагами, які є наступними:

- Захист даних. *MySQL* є безпечною та надійною системою управління базами даних, що відома у всьому світі та використовується в таких популярних засобах, як *WordPress, Drupal, Joomla, Facebook* та *Twitter*. Захист даних та підтримка обробки транзакцій, які супроводжують найновішу версію *MySQL*, можуть стати велику у нагоді для будь-якого бізнесу, особливо якщо це електронна комерція, яка передбачає постйніі грошові перекази.
- Масштабованість. *MySQL* пропонує філігранну масштабованість, для полегшення управління глибоко вбудованими програмами, використовуючи розміри, які є меншими навіть у масивних сховищах, де розміщується терабайт даних. Основною особливістю *MySQL* є гнучкість за запитом. Це рішення з відкритим кодом дозволяє повністю налаштувати бізнес електронної комерції з урахуванням унікальних вимог до сервера баз даних.
- Висока продуктивність. Особлива структура механізму зберігання даних *MySQL* дозволяє системним адміністраторам налаштовувати сервер баз даних *MySQL* для бездоганної продуктивності. Незалежно від того, чи є це веб-сайт електронної комерції, який щодня отримує мільйон запитів, або високошвидкісна система обробки транзакцій, *MySQL* розроблена задовольнити навіть найвибагливіші програми, забезпечуючи при цьому оптимальну швидкість, повнотекстовий індекс та унікальний кеш пам’яті для покращених характеристик.
- Безперебійна робота. *MySQL* дає повну гарантію безперебійної роботи цілодобово і пропонує широкий спектр високодоступних рішень, таких як спеціалізовані сервери кластерів та конфігурації реплікації *master/slave*.

- Комплексна підтримка транзакцій. На сьогоднішній день *MySQL* є лідером у списках надійних систем баз даних транзакцій, що доступні на ринку. Завдяки таким функціям, як повна автоматична, послідовна, ізольована та довговічна підтримка транзакцій, підтримка транзакцій з декількома версіями та необмежене блокування на рівні рядків, це рішення для повної цілісності даних. Це гарантує миттєву ідентифікацію тупикової ситуації через примусову посилальну цілісність сервера.

- Повний контроль робочого потоку. Оскільки середній час завантаження та встановлення становить менше 30 хвилин, *MySQL* означає зручність користування з першого дня. Незалежно від платформи, чи це *Linux, Microsoft, Macintosh* або *UNIX, MySQL* – це комплексне рішення з функціями самоуправління, яке автоматизує все, починаючи від розширення простору і конфігурації для проектування даних до адміністрування даних.

- Знижена загальна вартість власності. Переміщуючи поточні програми баз даних на *MySQL*, підприємства значно заощаджують матеріальні засоби коштів для нових проектів. Надійність та простота управління, що супроводжують *MySQL*, економить час на усунення несправностей, який інакше витрачається на вирішення проблем простою та продуктивності.

- Гнучкість відкритого коду. Всі страхи та занепокоєння, що виникають у рішеннях з відкритомим кодом, *MySQL* можна довести до кінця за допомогою цілодобової підтримки та ідентифікації підприємств. Безпечна обробка та надійне програмне забезпечення *MySQL* поєднує в собі ефективні транзакції для проектів. великих обсягів. Це робить обслуговування, налагодження та оновлення швидким та простим, одночасно покращуючи досвід роботи кінцевого користувача.

3. *GeoJSON* – це формат обміну даними, що призначений для зберігання геоданих, заснований на *JSON*. Спершу здійснюється *SQL* запит до бази даних на отримання інформації із сутностей, а потім певним чином потрібно передати одержану інформацію клієнту, щоб показати її користувачеві.

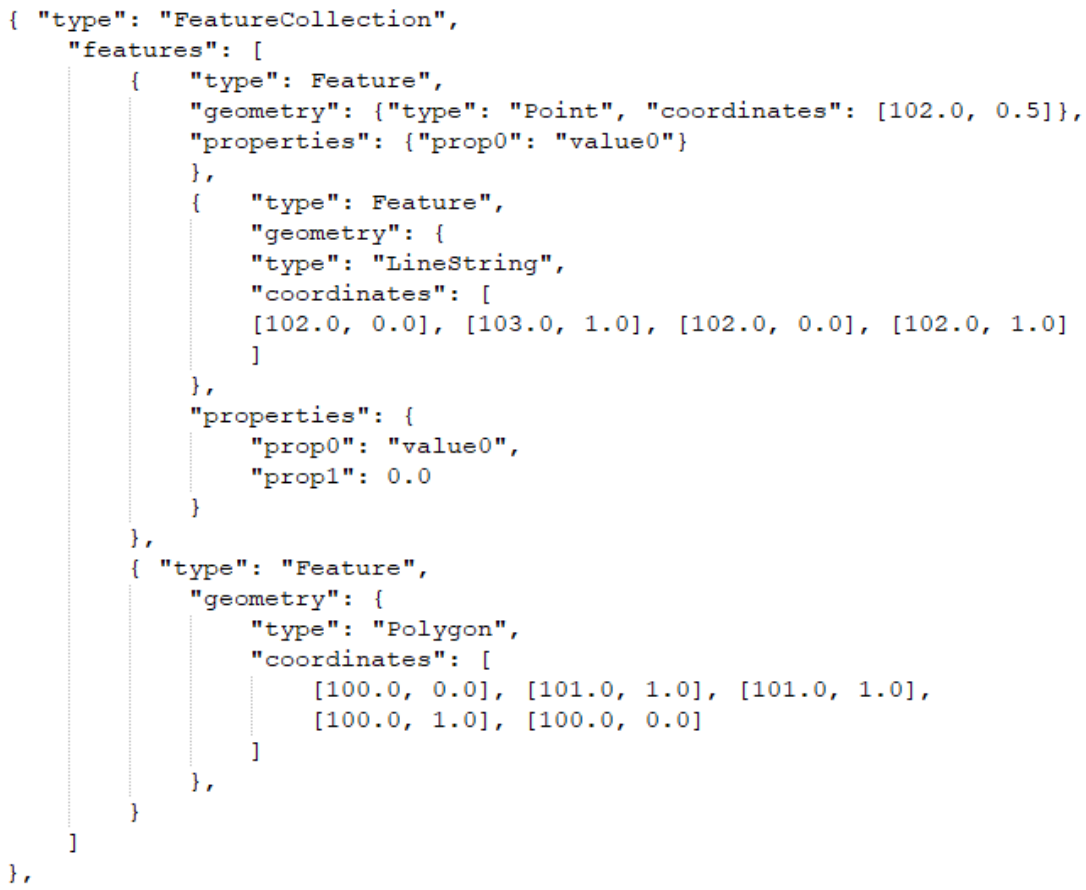

Рис. 3.2. Фрагмент структури *GeoJSON*

Основні переваги *GeoJSON*:

- Один сервісний сервер може обслуговувати однаковим чином декілька клієнтів, або навпаки, клієнт може відображати карти незалежно від способу реалізації сервісного сервера, якщо він використовує *GeoJSON*. Це дозволяє клієнту бути незалежним від сервера карт, він стає просто користувачем *GeoJSON*, незалежно від того, як він був розроблений.

- Доступність *JavaScript* без подальшого аналізу. Тобто дані легко переглядати, оскільки це звичайний об’єкт *JavaScript*, що спрощує подальшу обробку.

В даній роботі увагу зосереджено на обробці та відображенні геоданих в мобільному додатку в режимі реального часу. Порівняно з новою технологією, за допомогою якої дані оброляють з використанням *JavaScript Object Notation (JSON)*, процес *RDBMS (Relational Database Management System)* робить систему дуже інертною.

З метою покращення продуктивності обробки даних необхідний професійний метод зберігання, індексації та оновлення геоданих. Формат *GeoJSON* є відкритим стандартним форматом для кодування різноманітних географічних структур даних, заснованих на *JSON*. Його структура наведена на рис. 3.2.

Простота та швидкість завантаження *GeoJSON* роблять роблять систему обробки величезних географічних даних дуже швидкою. І хоча для нього потрібно трохи більше сховищ, загалом це значно сучасніше за використання традиційної бази даних *SQL* для обробки даних різних умов. Причини в основному наступні:

- Використання мови *SQL* для запиту понад тисячі пунктів у базі даних та завантаження їх на карту затратить час більше чотирьох секунд. Це спричинить збій карти, оскільки надалі буде зберігатися все більше і більше геоданих.
- У цій системі ми маємо три типи карт. Перша – це карта, яка використовується для відображення загальнодоступних пунктів. Загальнодоступна інформаційна карта використовує статичні геодані, вона менше контактуватиме з веб-сервером. Тобто використовується завантаження статичного файлу *GeoJSON* для обробки даних, а не лише для зменшення тиску на сервер при обробці даних, це також забезпечує хорошу взаємодію з користувачем при використанні карти.
- Іншим двом типам карт потрібно часто підтримувати зв’язок зі стороною сервера. Для кожного користувача дані зберігаються в одному файлі *GeoJSON*, середній розмір яких становить близько 50 КБ. Виходячи з цього, для зберігання двадцяти тисяч геоданих користувача потрібно близько 1 Гб пам’яті. Порівняно із величезним сховищем сервера, використовувати менший об’єм пам’яті для покращення швидкої реакції карти, дуже важливо для розроблюваної системи.

3.3. Тестування

Наявність етапу тестування у життєвому циклі розроблюваного програмного забезпечення допомагає виявляти та виправляти несправності до того, як програмне забезпечення почне працювати, тим самим значно зменшується ризик

виникнення помилок під час його експлуатації. Багато модулів у системі мають інтегруватися та функціонувати з іншими існуючими модулями. Таким чином, підтримка коректної роботи кожного окремого модуля дуже важлива для розроблюваної системи. У процесі виконання роботи використовується кілька методів тестування для перевірки системи.

У ході розробки системи буе використано декілька способів тестування. Перший з них – аналіз граничних значень. Даний метод передбачає перевірку поведінки продукту на граничних значеннях вхідних даних, оскільки на крайніх значеннях найчастіше виникають збої роботи програмного продукту. Граничний випадок трапляться поза нормальними робочими параметрами, зокрема, коли безліч змінних одночасно знаходяться на екстремальних рівнях Граничне тестування також може включати тести, що перевіряють стійкість системи для вхідних даних, що виходять за допустимий діапазон значень. При цьому система має певним способом обробляти такі ситуації, наприклад, надсилати повідомлення про некоректні дані чи помилку.

На кожній межі діапазону допустимих значень слід перевірити три набори значень:

1. Граничне значення.
2. Значення перед межею діапазону.
3. Значення після межі діапазону.

Метою такої перевірки є знайти помилки, пов’язані з граничним значенням. Аналіз граничних значень дозволяє переконатися, що система не просто коректно працює, але також може добре працювати в різних суворих умовах.

Також у ході розробки буде проводитись тестування методом *black-box*, щоб дозволити багатьом користувачам, які не знають внутрішньої структури розроблюваної системи, забезпечити вхідні дані (клацання, натискання клавіш) і перевірити результати на основі очікуваних результатів. Тобто метод передбачає тестування програмного продукту, дизайн, внутрішня структура та реалізація якого невідомі тестувальнику. *Black-box* тестування – процедура отримання і вибору

тестових випадків на основі аналізу функціональної чи нефункціональної специфікації, компонентів чи системи без посилання на їх внутрішню структуру.

В ході розробки додатку було виявлено проблему: рівень заряду акумулятора пристрою, на якому запущено мобільний додаток, відносно швидко знижується. Коли мобільні пристрої реєструють місцезнаходження користувачів за допомогою *GPS*, час автономної роботи може зменшитись приблизно на 12% за півгодини. Основною причиною цієї проблеми є те, що ми використовуємо мобільний додаток, щоб розрахувати, чи знаходиться користувач в районі геозони пункту. Незважаючи на те, що з розвитком мобільного техніки, продуктивність мобільного ЦП (центрального процесора) набагато стає набагато кращою, ніж раніше, проте вона все ще залишається значно нижчою, ніж у сервера. Таким чином, при використанні розробленого мобільного додатка для обчислення відстані між користувачем та певним пунктом збільшується тиск на центральний процесор мобільного пристрою. Крім того, система *GPS* буде використовувати певну частину заряду акумулятора мобільного пристрою.

Тим самим, це не лише допомагає у процесі пошуку несправностей, але і допомагає дослідити як впливає використання мобільного додатку швидкість зменшення рівня заряду акумулятора пристрою, на якому його запущено.

3.4. Висновки до розділу

Для розробки інформаційного геолокаційного додатку було обрано наступні технології:

- для реалізації користувацького інтерфейсу застосовано: стандартизовану гіпертекстову мову розмітки *HTML*, мову стилів *CSS* та фреймфорк *Bootstrap*;
- мову *JavaScript* та бібліотеку *jQuery* – для динамічної обробки даних;

мову програмування *PHP* – для розробки внутрішньої серверної системи мобільного додатку;

- *JavaScript* і *MapBox* використовуються для побудови загальнодоступних та приватних інформаційних пунктів,
- *API Google Maps* використано для побудови карт для компонентів досліження;
- для реалізації бази даних, щоб зберігати всі дані розроблюваної системи застосовано *MySQL* та *GeoJSON*.

У ході розробки системи буе використано декілька способів тестування. Перший з них – аналіз граничних значень. Даний метод передбачає перевірку поведінки продукту на граничних значеннях вхідних даних, оскільки на крайніх значеннях найчастіше виникають збої роботи програмного продукту. Також у ході розробки буде проводитись тестування методом *black-box*, передбачає перевірку програмного продукту, дизайн, внутрішня структура та реалізація якого невідомі тестувальнику.

РОЗДІЛ 4

РОЗРОБКА ІНФОРМАЦІЙНОГО ГЕОЛОКАЦІЙНОГО ДОДАТКУ ДЛЯ АВТОМАТИЗАЦІЇ ЗАДАЧ КОРИСТУВАЧА

У розділі буде представлено процес розробки системи. Перш за все слід визначити ролі користувачів та функціональні моделі відповідно до вимог. Слід використати уніфіковану мову моделювання *UML*, щоб візуалізувати процес проектування системи. *UML* дозволяє вказувати, візуалізувати, конструювати та документувати артефакти розроблюваної програмної системи. На основі моделей та їх функцій для проектування бази даних системи буде використано діаграму *ER (entity relationship diagram)*, щоб проілюструвати логічну структуру бази даних.

Для розробки системи на основі геолокації, потрібно реалізувати кілька важливих кроків для задоволення її вимог. Весь системний каркас – це перший крок до початку проекту. Він включає в себе розроблення основних моделей функцій, проектування баз даних, розроблення логічних процесів.

Ця частина потрібна для викладення короткого опису системи, складання чіткоїструктури моделі та визначення взаємозв'язку кожної моделі.

4.1 Система контролю доступу

Система контролю доступу – це компонент, котрий відповідає за підтримання захисту даних у системі управління даними. Загалом доступ до даних контролюється певним набором правил авторизації, в якому зазначено, хто може отримати доступ до якого ресурсу. У цій системі є три ролі користувача:

1. Адміністратор. Користувачі, авторизовані як адміністратори, мають найбільше прав по управлінню системою. Вони можуть редагувати публічну інформаційну карту, що включає в себе створення нових пунктів та оновлення інформації про них, керувати інформацією про *AdventureTrack* та коригувати питання до них, збирати дані про місцезнаходження користувачів та дані про

відповіді користувачів на питання, а також вони здійснюють управління правами користувачів.

2. Звичайний користувач. Усі звичайні користувачі, які зареєструвались чи авторизувались у системі, можуть використовувати функції приватної карти, вони можуть створювати та оновлювати власні пункти та бачити їх у мобільному додатку.

3. Користувач дослідник. Дослідники в процесі користування розроблюваним додатком відіграють особливу роль у цій системі, оскільки вони можуть створити свій приватний проект та самостійно додати в нього інформацію, тим самим ділитися власними знаннями.

На рис. 4.1 зображено діаграму прецедентів.

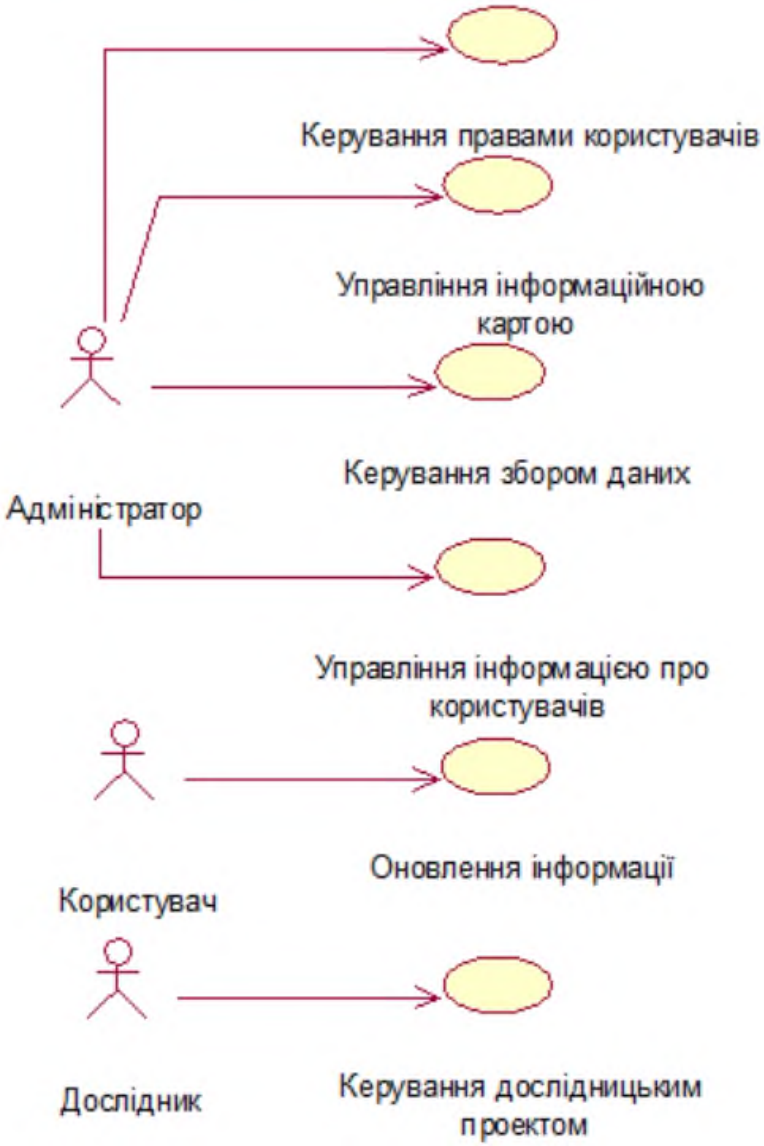

Рис. 4.1. Діаграма прецедентів

Для даної системи передачено наступні компоненти: карта, компонент збору даних, компоненти для дослідження місцевості та для наукових досліджень. Геодані можна завантажувати з двох сторін на карті. Маючи доступ до системи, адміністратори можуть використовувати карту для оновлення даних, які зберігаються в базі даних. Публічні пункти на публічній інформаційній карті показують, що кожен користувач може бачити інформацію про них. Звичайні користувачі використовують приватну карту для оновлення власних пунктів за допомогою файлів *GeoJSON*. Потім, відвідуючи певне місце кожного разу, вони не тільки можуть бачити загальнодоступні точки на карті, але й свої приватні маркери на карті. Компонент для дослідження місцевості використовується, щоб допомогти користувачам дізнатися більше про певну територію, коли вони використовують мобільний додаток при її відвідуванні. Компонент для наукових досліджень – це підпроект, який дослідники можуть використовувати для своїх досліджень. Структура компонентів для дослідження місцевості та для наукових досліджень подібна. Процес збору даних розроблений з метою допомогти дослідницькій групі проаналізувати відповіді анонімних користувачів на запитання та знайти рівень відвідуваності паркової зони в різні сезони.

Оскільки адміністратор має найбільше функціональних можливостей у цій системі, він може створювати, оновлювати та видаляти всі дані та визначати, який з пунктів не є публічним, а який може відображатися на загальнодоступній карті.

Модель компоненту для дослідження місцевості включає багато пунктів, якими має керувати адміністратор для надання їм спеціального ключа, щоб позначити, до якого саме маршруту вони відносяться. Адміністратор також може створювати та оновлювати інформацію про цей компонент, що включає наступні пункти: назву, вступ, опис, рівень складності, довжину маршруту, зображення маршруту, його позначку тощо. Нарешті, адміністратор має права для визначення, контенту, який можна затвердити для публікації в мобільному додатку.

Карта допомагає у відображенні пунктів на поточному маршруті, а вже в них або на основі списку можна створювати та оновлювати різні типи запитань.

4.2. *Ajax* метод для обробки даних

Ajax (Asynchronous JavaScript And XML) – це акронім від асинхронного *JavaScript* і *XML*, насправді це не єдина самостійна технологія, а поєднання декількох суміжних. Цей термін було запропоновано Джессі Джеймсом Гарреттом у 2005 році. Програми *Ajax* використовують об'єкт *XMLHttpRequest* у веб-браузері для асинхронного отримання даних із сервера або служби. Це означає, що дані можна завантажувати у вебпрограму у фоновому режимі, інформацію можна частково оновлювати на екрані без необхідності перезавантажувати всю вебсторінку. На рис. 4.2 показано різницю між класичним методом та методом *Ajax* для передачі даних.

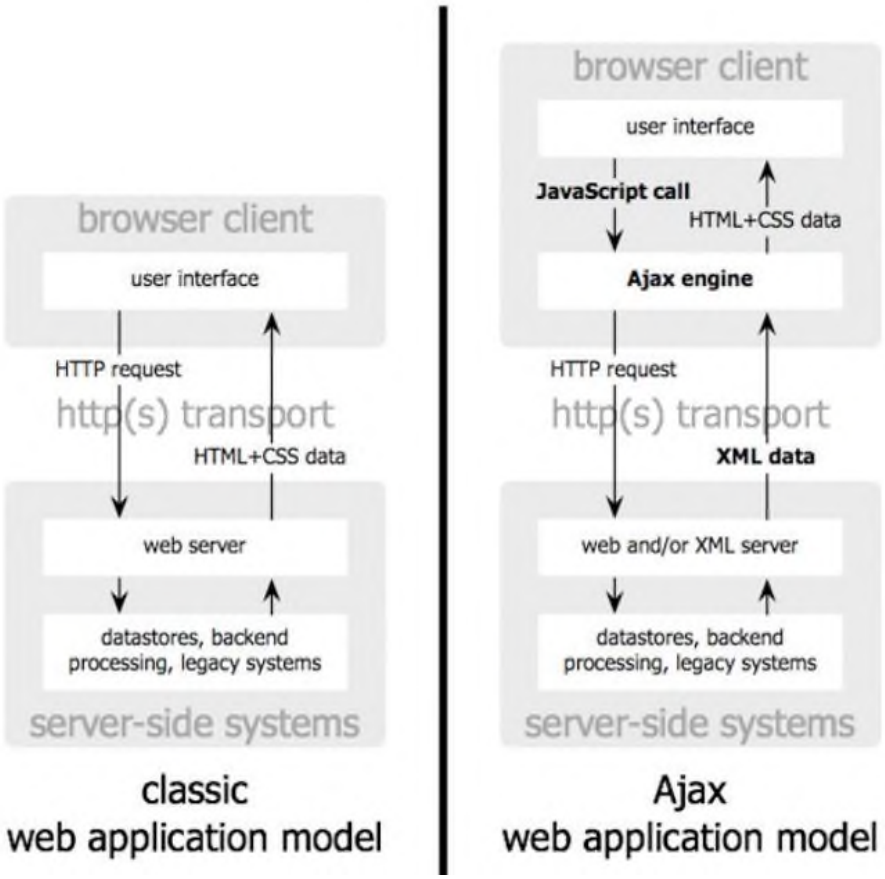

Рис. 4.2. Порівняння моделей класичного додатку з використанням *Ajax*

Оскільки розроблюваний мобільний додаток є геолокаційним, використовується *API Google Maps*, щоб встановити структуру карт. Поки питання потрібно оновлювати, бібліотека *JavaScript – jQuery* використовуватиме функцію *Ajax* для надсилання запиту на веб-сервер.

Завдяки використанню технології *Ajax* оновлення даних із сервера здійснюється швидше, а деякі пукнти на карті отримують відповідь без повного оновлення всієї карти. Дещо інакша справа з картою користувачів: потрібно оновлювати всю інформацію на карті, завантажувати статичні точки завантаження та витрачати більше часу задля оновлення даних одного пункту.

4.3. Компонент для дослідження місцевості

На карті використано три різні кольори для визначення трьох різних видів пунктів на ній. Червоним кольором позначено поточні пункти відстеження; жовтим – звичайні інформаційні пункти, які не належать до поточного маршруту, а синім кольором виділено ті пункти, що вивчаються на даний момент. Адміністратор може редагувати питання по пунктах після того, як вони будуть збережені користувачем в мобільному додатку. Вся інформація запитується з бази даних та віборажається на сторінці. Якщо користувач натискає на червону позначку, він може відкрити вікно зі списком запитань, яке відображає стан поточного пункту по кожному виду запитань. Адміністратор додає та редагує запитання за посиланням. Використання карти на основі відображення пунктів та запитань по них допомагає адміністраторові легко та вільно керувати питаннями.

Запитання будуть зберігатись в різних таблицях типів запитань у базі даних. Кожна таблиця запитань повинна мати свою особливу структуру. Наприклад, деякі текстові чи графічні запитання підтримують завантаження фотографій в якості відповіді на запитання, для цього типу запитань потрібен атрибут зображення. Для запитання, в якому передбачено більше однієї правильної відповіді, користувач має обрати декілька варіантів. В цьому разі для таблиці, що зберігає відповіді користувача, слід передбачити, що для запиту даних з неї потрібні два спеціальні атрибути, щоб записати кілька варіантів.

Запитання відображатимуться у мобільному додатку, коли користувачі обирають пункт або переглядають список запитань.

Користувач також може отримувати повідомлення, коли він перебуває у геозони компонента для дослідження місцевості, причому радіус геозони може задати адміністратор. Далі мобільний додаток надсилає запит, вебсервер отримує запит і використовує *API* для надсилання багаторівневих структурних даних до програми. Всі дані знаходяться в режимі реального часу, це гарантує, що користувачі можуть отримувати найбільш актуальну інформацію з боку сервера.

У всій розроблюваній системі, у зв'язку з деякими особливостями моделей, дані мають зберігатись по-різному. Традиційну реляційну базу даних *MySQL* доцільно використовувати для обробки невеликих об'ємів даних. Для зберігання більшості даних розроблюваної системи використовується база даних. Файл *GeoJSON* використовується для збереження багаторівневих структурних даних та спеціальних громіздких геоданих, які будуть використовуватися для відображення інформації на карті та збору даних користувачами відповідей. Наприклад, для зберігання даних про пункти на карті та інформацію користувачів використовується база даних. Обидві сутності мають зв'язки між собою, і використовуючи мову *SQL* інформацію з них можна легко запистувати із бази даних *MySQL* та оновлювати їх.

Таблиця користувачів складається з п'ятнадцяти атрибутів, дана сутність зберігає всю інформацію про всіх користувачів системи. Перший атрибут – це первинний ключ – *user_id*. Атрибути імені користувача та пароля створені для ідентифікації користувачів і допомагають користувачам отримати доступ до системи та керувати своїм обліковим записом. Атрибут *user_role* – це рівень прав користувачів, він включає три різні ступеня з різним набором функціональностей користувачів для використання системи. В розроблюваній сшстемі адміністратори мають найвищий користувацький рівень і найширший набір функціональностей: вони контролюють і можуть редагувати всю інформацію у системі та керувати набором прав інших користувачів, тобто змінювати дозвіл користувача на використання системи. Вони також можуть редагувати інформацію в компонентах для дослідження місцевості та запитання в них, переглядати всі набори даних. Звичайні користувачі можуть створювати та оновлювати свої пункти, бачити

основну інформацію. Гість у системі просто може бачити карту загальнодоступної інформації в Інтернеті, він не має доступу до системи і не може зберігати пункти на карті у свій обліковий запис. У таблиці з розташуванням пунктів зберігається вся інформація про них, в ній вказано координати, додані користувачами.

Сукупність точок на карті та дані відповідей користувачів зберігаються у файлі *JSON*. Для відображення на карті їм потрібно зберігати географічну інформацію з тисяч точок, а дані відповідей користувачів мають складну структуру. Щоб зберегти їх у файлі *JSON* слід переконатись, що вони можуть завантажувати дані за один раз після зчитання файлу, а не кожен раз для пошуку конкретної інформації здійснювати запит за допомогою мови *SQL*.

4.2. Підготовка до розробки

У ході розробки ситеми використовується ряд програмних засобів для реалізації мобільного додатку:

- мова програмування *PHP* необхідна для розробки внутрішньої серверної системи мобільного додатку;
- *JavaScript* і *MapBox* використовуються для побудови загальнодоступних та приватних інформаційних пунктів,
- *API Google Maps* використано для побудови карт для компонентів досліження;
- для реалізації користувацького інтерфейсу застосовано: стандартизовану гіпертекстову мову розмітки *HTML*, мову стилів *CSS* та фреймфорк *Bootstrap*;
- для реалізації бази даних, щоб зберігати всі дані розроблюваної системи застосовано *MySQL* та *GeoJSON*.

Важливою характеристикою *PHP* є гнучкість процесу розробки. та висока продуктивність у комлекті серверного програмного забезпечення – *LAMP*.

LAMP – це архетипічна модель стеків веб-служб, названа як абревіатура назв своїх оригінальних чотирьох компонентів з відкритим кодом:

- *Linux* – операційна система;
- *Apache* – *HTTP*-сервер;
- *MySQL* – реляційна система управління базами даних;
- *PHP* – мова програмування.

Більшість вебсерверів в якості операційно

У той час, як налаштовано середовище стеку служб, *PHP* та *MySQL* можуть ефективно працювати один з одним. У *PHP* існує безліч функцій, якими можна керувати *MySQL*, наприклад, функція *query()* використовується для виконання запитів мовою *SQL* у базі даних *MySQL*; функція *fetch_assoc()* використовується для отримання результату після виконання *SQL* запиту та форматування їх в якості структури даних масиву рядків; функція *num_rows()* використовується для повернення номера з результуючій вибірці в результаті виконання запиту мовою *SQL* тощо.

4.3. Проектування бази даних

Після налаштування сервісного середовища і завершення робіт по проектуванню слід розпочати розробку серверної частини. Перш за все, потрібно створити базу даних системи. Для чіткості розуміння ускладнених зв’язків, Тоді нам потрібно розпочати *back-end* розробку, перше, що потрібно створити цілу базу даних системи.

Побудована модель включає в себе декілька атрибутів, які ми будемо використовувати в цьому проекті:

- Сутності мають форму прямокутника, це об’єкт у системі.
- Зв’язки відображаються у вигляді ромбів, вони показують, як дві сутності діляться інформацією в базі даних.
- Атрибути позначаються в вигляді овалів. Ключовим атрибутом є унікальна характеристина сутності.

Більшість баз даних не обмежуються лише однією таблицею, тому деякі таблиці мають зовнішні зв’язки між собою. Всього існує три типи відношень:

- один-до-одного;
- один-до-багатьох;
- багато-до-багатьох.

Наприклад, для деякої таблиці *User_ID* встановлено як зовнішній ключ, який є первинним ключем сутності *User*. Таким чином, ці дві сутності пов’язані між собою і можна посилатись на індентифікатор користувача для пошуку необхідної інформації.

У розроблюваній базі даних створено багато зовнішніх ключів у сутностей з метою зв’язати таблиці між собою.

Для демонстрації різних карт у розроблюваному мобільному додатку в основному буде використано формат *GeoJSON* для завантаження геоданих.


```
"type": FeaturedCollection;
"features": [
    {
        "type: "Feature",
        "geometry": {
            "type": "Point",
            "coordinates": [
                51.526559,
                31.295917
            ]
        },
        "properties": {
            "decription": "",
            "marker-size": "medium",
            "marker-symbol": "star",
            "marker-color": "#ff8888",
            "category": 1348,
            "p_id": 1,
            "img_id": 239,
            "images": "./uploads/f803f1533dbc7df93c0ca825bfe217le.jpg",
            "img_num": 1,
            "attribute": "",
            "icon": {
                iconUrl": "../Categories/pictures/structures.jpg",
                "iconSize": [
                    30,
                    41
                ]
            }
        }
    },
```


Рис. 4.3. Фрагмент файлу *GeoJSON*

Атрибут координати визначатиме місце на карті; атрибут *p_id* – це збережений у базі даних ідентифікатор точки, що допомагає зв’язати її із запитанням по ній.

На карті точки можна відобразити, використовуючи атрибут іконки. У цей атрибут включено: *iconUrl* – який є шляхом до зображення, *iconSize* – який показує розмір зображення на карті.

Сутність для компонену дослідження місцевості використовується для зберігання всіх маршрутів та інформації користувача. Вона зв’язана із шістьма видами таблиць із запитаннями, а саме: «Питання по тексту/зображенню», «Заповнити порожнє поле», «Запитання з однією вірною відповіддю», «Запитання з кількома правильними варіантами», «Запитання з вибором правильної послідовності», «Встановити відповідність».

Для всіх типів запитань підтримується відображення кількох зображень та тексту на мобільній стороні. Щоразу, коли мобільний додаток використовує *API* для запиту даних запитань, сторона сервера використовує оператор внутрішнього з’днання *INNER JOIN*, щоб вибрати кілька таблиць для виконання операцій запиту. Використання оператора *INNER JOIN* може зробити запит ефективнішим, ніж аби кожного разу виконувався окремий запит у цих таблицях. Дані відповідей користувачів та дані місць на інформаційній карті зберігатимуться у файлах *GeoJSON*. Це зручний метод обробки такого роду даних. Всі вони мають складнішу структуру, ніж інші дані. Більшість з них для підключення потребують більше трьох таблиць.

Задля зменшення навантаження на базу даних та значно швидшого виконання запитів даних, дані будуть зберігаютися у файлі *JSON*. При їх створенні слід переконатится, що всі зв’язки у ньому зберігаються у форматі *JSON*. Таким чином, для читання файлу *JSON PHP* використовує функцію *file_get_content()*, а за допомогою функції *json_decode()* передає формат *JSON* у структуру масиву в *PHP*. *JavaScript* може читати формат *JSON* дуже швидко і без необхідності зміни формату, може завантажувати файл *JSON* безпосередньо за допомогою *loadURL().*

4.4 Компоненти та їх ролі

Системні моделі – це концептуальна модель, що є результатом моделювання системи, яка описує та представляє систему. Вони стануть у нагоді аналітику, щоб зрозуміти функціональність системи, а моделі використовуються для комунакації з користувачами. Крім того, вони демонструють різні перспективи, описують загальну поведінку системи. У розроблюванїй системі існує сім компонентів.

1. Карта. Функцією цього компоненту є побудова всіх карт та відображення інформації на різних типах карт. За допомогою нього можна легко встановити архітертуку карт. Для зміни певних параметрів у функціях карти, хоча всі карти мають побідну структуру, розробка цього методу дозволить зменшити час. Наприклад, маємо три карти для відображення різних видів пунктів: публічна карта, публічна інформаційна карта та приватна інформаційна карта. Вони використовують однакову структурну карту на базі *API Mapbox*, їх відмінністю є вміст. Для розгортання цих карт потрібно лише змінити дані на них. Таким чином, потрібно змінити інтерфейс завантаження даних; дати публічній карті дозвіл на читаня статичний файлу *GeoJSON*, який містить основну інформацію, дати публічній карті дозвіл надіслати запит до бази даних та отримати з неї більше інформації, а потім надіслати всю інформацію про публічні пункти на публічну інформаційну карту, дати дозвіл приватній інформаційній карті на читання файл *GeoJSON*, який отримує особисті дані користувачів з бази даних. У цьому методі системі потрібно створити дві різні структурні карти та змінити дані для завантаження, якщо вони використовують одну і ту ж структуру карти.

2. Зберігання даних. Компонент відповідає за збирання різного виду даних у розроблюваній системі. З дослідницькою метою всі дані з відповідями користувачів у компонентах для дослідження місцевості та для наукових досліджень, а також зберігаються дані про кожен пункт на карті. Крім цього, , збирається інформація про локацію користувача, коли він анонімно використовує мобільний додаток на певній території.

Стандартна координата користувачів або пункту включає в себе значення географічної широти та довготи. Дані про геолоакацію будуть збережені до бази даних у форматі рядка та відображені на карті, що показує частоту відвідування зон. Для зображення області, яку відвідали більше двохсот разів, використовується червоний колір, оранжевий колір показує територію, яку користувачі відвідували від двадцяти до двохсот разів, а зелений колір вказує на область, які відвідували менше двадцяти разів. Це зроблено з метою дізнатись, коли люди люблять відвідувати парк, і які місця є найбільш популярними для відвідування людьми.

Дані відповідей інших анонімних користувачів будуть збережені у файлах *JSON*. Передачено шість типів запитань, і кожен тип має різну структуру запитань. Для збереження даних відповідей користувачів потребується використання мови запитів *SQL* багато разів. Такий тип наборів даних охоплює інформацію з багатьох сутностей бази даних. Коли система використовує такий тип мультитаблиць для запитів з високою частотою, дані будуть збиратись дуже повільно. Так як надалі кількість даних буде лише зростати, система зазнає збою. Для вирішення проблеми обробки багатоструктурних дани застосовано формат *JSON* та впроваджено його після ініціалізації даних.

Більшість наборів даних буде автоматично збережено у файлах *Excel*.

У ході роботи використовуться вбудована в *PHP* функція *readdir()*, яка для пошуку всіх імен файлів у каталозі повертає ім'я файлів кожного разу, а далі застосовану вбудовану функцію *PHP* f*ile_get_contents()* для отримання вмісті файлів. Для того, щоб значно швидше прочитати більше тисячі файлів, їх слід перейменовуємо за їх ідентифікатором та створити дату за допомогою спеціального формату на основі ідентифікатора користувача, з метою отримання інформації, зокрема назва запитання, локація, тип запитання, відповідь і т. д.

3. Компонент для дослідження місцевості є головним у розроблюваній системі, адже ця частина допомагає у досягнення освітньої мети в ній. Користувачі з правами адміністратора використовують цей компонент для створення інформації, що відображатиметься у мобільному додатку. Вони також можуть керувати періодом часу для показу в мобільному додатку за допомогою мітки часу.

Наприклад, адміністратор може встановити показ визначеного місця на карті в додатку терміном в три місяці, бо саме в цей період відвідувачі можуть побачити якийсь особливий пейзаж. Візуалізована модель даних може показувати кількість типів запитань за допомогою кругової діаграми. Для їх використання в ході роботи використовується бібліотека *JavaScript canvasjs*. *Canvasjs* може використовувати функції *JavaScript* для динамічного відображення. Більшість карт потребують оновлення даних, коли користувач використовує мобільний додаток. Всю інформацію про місцезнаходження публічних точок та інформацію про всі типи запитань потрібно завантажувати на карту.

Для карти розроблено алгоритм сортування пунктів на ній, щоб допомогти адміністратору контролювати порядок пунктів для показу на мобільній стороні. За замовчуванням їх порядок сортується за часом створення. Використовучи алгоритм, адміністратор може самостійно присвоїти кожному пункту номер відображення. Ці дані будуть збережені у тимчасовому масиві, який буде впорядковано, використовуючи метод швидкого сортування, щоб надіслати його на мобільну сторону. Коли користувач обирає точку на карті пригод, у мобільному додатку буде відображена інформацію про запитання по ній. Інформацію про запитання адміністратор може редагувати та видаляти. Адміністратор може редагувати назву запитання та його опис, який буде відображатись на мобільних пристроях користувачів, коли відвідувачі зайдуть в зону спостереження.

Запитання для користувачів можуть включати демонстрацію кількох зображень, щоб допомогти користувачеві відповісти. Для зберігання конкретного типу запитань використовується окрема таблиця у базі даних, бо кожен тип запитання вимагає різної структури, різні таблиці можуть зберігати особливі характеристики кожного типу та значно спрощувати запити даних. Усі інтерактивні дані використовують метод *Ajax* для зв'язку із сценарієм на серверній стороні, він здійснює обробку інформації без перезавантаження браузера.

Алгоритм впорядкування пунктів на карті розроблено та впроваджено задля допомоги у їх сортуванні. Адміністратор може визначити їх порядок, що може допомогти відвідувачам вивчити місцевасть у правильній послідовності.

4. Компонент для наукових досліджень по своїй суті подібний до того, що використовується для дослідження місцевості, має подібні аспекти. Базова модель для розробки обох компонентів однакова. Функціонал дослідника в даному компоненті відрізняється від ролі адміністратора та звичайного користувача: йому надано спеціальний дозвіл на створення свого власного наукового проекту. Але ціль цих компонентів відрізняються. Метою створення компоненту для наукових досліджень є надання відвідувачам парку можливості допомогти дослідникам отримати нову інформацію. Для свого дослідження вони можуть створювати приватні місця на карті та перелічувати в них власні запитання. У цьому проекті відвідувачі використовують мобільний додаток, щоб знайти опублікований проект дослідника та відповісти на запитання, тим самим допомогти здобути корисну інформацію та обробити її за допомогою методів збору та аналізу даних. Усі створені дослідниками проекти не відображатимуться звичайним користувачам до того часу, поки адміністратор їх затвердить. Якщо адміністратор підтвердить, що проект легальний, його буде опубліковано і він буде доступний для всіх користувачів.

5. Для стимулювання користувачів додано механізм заохочення. Він стимулює користувачів досліджувати місцевість, щоб знайти в ній більше локацій та відповісти на більше запитань. Цей компонент створює та перевіряє адміністратор. У системі передбачено різні види цифрових значків: значкок для місцевості, значок відвідування парку, значок подоланої відстані. Коли користувач вперше використовує мобільний додаток для входу в систему, програма надішле повідомлення, що він отримав свій перший цифровий значок.

6. Розділ новин створено, щоб допомогти адміністратору поширювати новини, які можуть переглядати всі відвідувачі, що користуються мобільним додатком. Тільки адміністратор має право дозволу на редагування та оновлення новини. Порядок новин впорядковується за датою створення новини.

7. Функція завантаження зображень потрібна у системі майже всім компонентам. Публічна інформаційна карта потребує завантаження фотографій для кожного пункту на ній; компоненту для дослідження місцевості потрібно

передбачити можливість завантаження зображення із запитаннями, а для користувача можливість завантажити зображення, щоб відповісти на певні запитання; для розділу з новинами необхідна функція завантаження фотоновин. Для того, щоб зробити розробку більш ефективною, додано функцію завантаження зображень, щоб допомогти користувачам та адміністратору завантажувати фотографії різного призначення.

Для розробки слід використати технології *HTML* та *PHP*.

Форма - це основний тег *HTML*, який використовується для збору вхідних даних від користувача, він має атрибут *enctype*, який вказує, як слід кодувати дані форми під час надсилання їх на сервер. Коли файл буде завантажено на сторону сервера, він буде збережений тимчасово в кеш-пам'яті в сервері. *PHP* використовує вбудовану функцію *$_FILES["file"]["tmp_name"]*, щоб отримати тимчасову назву завантаженого зображення, а далі за допомогою функції *move_uploaded_file()*, щоб перемістити зображення у вказане місце. Зрештою шлях до зображення буде збережено у таблиці зображень у базі даних.

4.5. Роль технології *Mapbox* у розроблюваному додатку

Mapbox – це платформа для розробників, яка використовується в різних галузях для створення власних додатків. *Mapbox* являє собою інтерфейс створення додатків, заснований на *JavaScript*, що дозволяє створити власну електронну карту та опублікувати її зручним способом. Технологія *Mapbox* базутєься на векторних картах, вдосконаленому підході до картографування, за якого дані доставляються на пристрій і точно відображаються в режимі реального часу. В розроблюваному мобільному додатку необхідно встановити координату запуску, яка є першою точкою на карті, і використати *addLayer (L.mapbox.tileLayer ('mapbox.streets'))* для того, щоб побудувати базову статичну карту. Після чого необхідно завантажити деякі дані точок на статичну карту. *Mapbox* має вбудовану функцію *L.mapbox.featureLayer().loadURL()* для того, щоб отримати файл *GeoJSON*, використовуючи шлях.

В процесі розробки було використано мову *SQL* для отримання інформації про точки, а мову *PHP* – для керування базою даних *MySQL*, щоб розміщувати дані точок на карті по черзі. Для отримання даних та надсилання даних на карту використання циклу не є дуже ефективним способом, оскільки за такого методу карті потрібно більше, ніж 5 секунд, щоб відобразити всі точки, і кожен раз витрачатиметься ще декілька сенкуд на її перезавантаження для оновлення даних точок. Це поганий досвід взаємодії користувача з картографічною системою.

Для відображення динамічної карти використано інший, більш ефективний метод побудови. Використовується формат *JSON* для ініціації даних точок та дозволу *JavaScript* зчитувати файл безпосередньо. *JSON* – це формат для зберігання та передачі даних. Найкраще використовувати дані у веб-додатках з веб-серверів через *JavaScript*, який може зчитувати *JSON* швидше, ніж інші методи передачі.

Замість запитів до бази даних окремих пунктів і надсилання їх у *JavaScript* по одному, цей метод використовує *PHP* функцію *file_put_contents()* для запису даних у *JSON* файл у синтаксисі *JSON*. Синтаксис *JSON* бере початок із синтаксису нотації об’єктів *JavaScript*, він має компактну архітектуру і простий для читання. Таким чином, *API Mapbox* для читання файлів *JSON* та даних точок завантаження на карті буде використовувати менше однієї секунди.

4.6. Збирання багатоструктурних даних

У розроблюваній системі більшість даних складної структури у файлі *JSON*, що дозволяє уникати одержання одиниці даних шляхом багаторазового запиту декількох сутностей до бази даних. Коли дані ініціюються, для їх запису у *JSON* файл використовується *PHP* функція, з метою зменшити взаємодію з базою даних. Переконайтесь, що *JSON* файл може отримати всю корисну інформацію, а PHP її легко читає, тобто означає, що ми можемо збирати більше інформації одночасно. Слід переконатись, що *JSON* файл може отримати всю інформацію, а *PHP* – легко її зчитати, що означає можливість збирати більше інформації одночасно.

Дещо інакше з використанням *SQL* запиту для отримання даних.

Слід чітко встановити структуру даних, а для зміни всіх даних на *PHP* масив, застосувати *PHP* функцію *json_decode()*.

```
"title": "Розмір листка",
"question": {
    "q_answer": [
        "1",
        "2",
        "3",
    ],
    "q_options": [
        "Розмір менший за долоню",
        "Розміром з долоню",
        "Розмір більший за долоню"
    ],
    "q_attributes": "",
    "q_link": "",
    "u_response": [
        "Розмір менший за долоню; Розміром з долоню"
    ],
    "q_type": multi,
    "q_id": "100";
    "q_description": "Спробуйте знайти листя різного розміру",
    "q_response": "Добре! Листя може виростати до різних розмірів.
    Деякі дерева мають маленькі листки.
    Інші мають великі листки."
    "q_right": "0"
},
"last": false,
"desription": "",
```

Рис. 4.4. Структура даних відповідей

На рис. 4.4 зображено структуру даних відповідей, де:

- атрибут *q_id* – ідентифікатор точки, який може отримати назву досліджуваної місцевості, ідентифікатор питання та його заголовок;
- атрибут *q_type* вказує на тип запитання, який можна отримати, використовуючи *$ val ['questions'] [0] ['q_type']*;
- атрибут *u_response* – це дані відповіді користувача, які будуть збережені в останньому стовпці в файлі *Excel*.

Структура даних відповідей користувачів базується на типі запитання, Для запитань з кількома правильними варіантами записується масив з кількох варіантів в одному стовпці. Також у користувача є можливість відповідати на запитання

неодноразово, у кожному стовпці зберігаються певні дані відповідей, атрибут з даними користувачем відповідями буде створений на той час автоматично.

Далі описано реалізацію окремих компонентів мобільного додатку.

1. Реалізація компоненту для дослідження місцевості.

Впроваджено різні права доступу для контролю використання різними користувачами різних функцій. Звичайний користувач і дослідник бачать лише основну інформацію опублікованого матеріалу, а адміністратор має повний набір прав для керування.

Коли адміністратор використовує функцію створення нового компоненту для дослідження місцевості, там буде міститись лише основна інформація. Дані карт і дані запитань будуть порожні, їх ма додати адміністратор. Встановлено певні обмеження для створення нової місцевості*:* поля з назвою, описом, рівнем складності та відстанню не можуть залишатись порожніми. Завдяки чому, для всюди обов'язково буде додано основну інформацію, яка відображається у мобільному додатку.

Для питань компоненту дослідження місцевості створено алгоритм порядку проходження пунктів, щоб допомогти адміністратору задати їм правильний логічний порядок. Перш за все, дані пунктів отримуються за допомогою запиту з бази даних за допомогою параметрів. Далі адміністратор задає для пунктів правильний порядок подає їх на внутрішній сервер. Зрештою, ідентифікатори пунктів в правильному порядку зберігається у базі даних та сортуються в додатку.

Питання зберігаються в базі даних в різних таблицях і мають різну структуру. Тобто не можна завантажувати інформацію про них на карту за допомогою *PHP* та *MySQL*. *PHP* використовує функцію для запиту даних на карту, що кожного разу перезавантажує сторінку. Для вирішеня проблеми дуже частого оновлення сторінки, використовуєтьс метод *Ajax* для передачі даних між стороною сервера, базою даних та інтерфейсом користувача. При кожній обробці даних, наприклад, щоб показати різну кількість запитань на карті, з використанням *JavaScript* функцій встановлено так званий *click listener*: *onclick()* (коли подія запускається користувачем) і *post()* (відправляється запит до сервера без перезавантаження

сторінки). *PHP* використовується для запиту конкретних даних з бази даних. Коли *PHP* отримує результат даних з бази даних, сервер надсилає *JSON* дані на карту. У мобільний додаток дані, що передаються переважно таким методом.

2. Реалізація системи позначок

Щодо розробки моделі позначок є єдина вимога – в системі має бути позначка для місцевостей. Розроблювана система позначок не передбачає встановлення критеріїв для них, але потребує більше їх видів. Система позначок допомагає адміністратору у процесі управління, і може додавати нові позначки та оновлювати позначки за допомогою цієї системи. Для цього адміністратору спершу слід завантажити піктограму позначки, а далі ввести її назву, опис значка та критерії. Кожна позначка повинна мати принаймні один критерій. Позначка також може мати декілька критеріїв. *PHP* встановлює перевірку кожного з критеріїв на серверній стороні, коли вони спрацьовують, інформація про користувача та інформація про позначку буде зберігатися у відповідній таблиці в базі даних. У мобільному додатку користувач може бачити власні позначки в окремому розділі.

3. Реалізація системи новин.

Система новин є базовою системою управління. Вона заснована на контролі бази даних для вставки, видалення, оновлення та запиту новин. Адміністратору потрібно ввести заголовок, автора, посилання та контент для однієї новини. *PHP* використовує вбудовану функцію дати *('Y-m-d H: i: s', time ())*, щоб отримати поточні час і дату для встановлення її датою публікації новини. Формат контенту новин контролює *nicedit.js*, що допомагає адміністратору легко редагувати контент та передавати веб-вміст у *HTML*-код. *NicEdit* – простий кросплатформенний вбудований контекстний редактор, що дозволяє легко редагувати вебсторінки прямо з браузера. *NicEdit* швидко інтегрується через *JavaScript* в будь-який сайт. Коли адміністратор встановить прапорець «Опубліковано», ці новини будуть відображені на сторінці новин.

4. Завантаження зображень в додаток.

Завантаження зображень є однією з головних складових у розроблюваній системі: на сторінці профілю користувача слід завантажити зображення у

спеціально відведене місце, на карти – від одного до кількох зображень для деяких пунктів, для компоненту дослідження місцевості також слід передбачити завантаження зображень для відповідей на запитання.

Для мобільного додатку було розроблено дві функції, які допомагають завантажувати зображення. Однію із них є клас функції завантаження для сторони сервера. Переважна більшість зображень завантажується за допомогою *HTML*-форми: коли користувач надсилає форму, картинка з *<input type = file>* надсилається на сторону сервера і буде тимчасовим файлом на стороні сервера. У розроблюваній системі дозволяється завантажувати зображення *JPEG, JPG, GIF* та *PNG* формату, максимальний розмір завантажуваного файлу - 5 МБ. Користувачеві слід надіслати на сервер зображення коректного формату і не більше максимально допустимого розуміру. Коли користувач надсилає файл некоректного формату чи розміру, використовується *PHP* функція для перевірки формату та розміру файлу. *PHP* функція *$ _FILES ["file"] ["type"]* може витягувати розширення файлу, і якщо отримано значення *"image / jpeg"*, *"image / gif"* або *"image / png"*, то формат завантажуваного зображення коректний. Аналогічно для перевірки розміру завантаженого користувачем файлу. *PHP* функція *$ _FILES ["file"] ["type"]* для управління максимальним розміром файлу.

Всі кроки аналогічні для функції завантаження зображень. Лише слід встановити шлях, де зображення тимчасово зберігаються на сервері. *PHP* функція *move_uploaded_file(string $filename , string $destination)* може перемістити тимчасовий файл у певне місце. Якщо файл дійсно було завантажено на сервер, його буде переміщено за шляхом, вказаним в аргументі *destination*, і в разі успішного виконання функції, вона поверне значення *true*.

4.7. Вдосконалення розробленого мобільного додатку

З метою покращення розробленої системи в цілому та збільшеня конкретно часу автономної роботи, було розроблено і впроваджено алгоритм економії заряду в мобільному додатку. Алгоритм енергозбереження здійснює розрахунок відстані

на стороні сервера, використовуючи високу продуктивність його процесора, щоб розрахувати відстань на основі координат.

Для розрахунку використовується *PHP* функція *distance($lat1,$lng1,$lat2,$lng2)*, що обчислює відстань між двома точками за координатами: враховуючи їх широту та довготу, повертає масив значень з більшості одиниць загальної дистанції. Параметр *($ lat1, $ lng1)* – це *GPS* координата користувача, *($ lat2, $ lng2)* – *GPS* координата певного пункту на карті. Мобільний додаток розміщує ідентифікатор користувача та координати користувача на сервері. Сервер робить запит до бази даних, щоб отримати таблицю розміщення, знаходить пункти, що належать окремій місцевості, і зберігає їх у масив. Після чого виконується *MySQL* функція запиту масиву для пошуку елементів у цьому масиві по черзі. Функція *array_push()* використовується для встановлення ідентифікатора та координати об'єкту та збереження їх як елемента масиву (рис. 4.5).

```
$sql_gps = "SELECT * FROM Location WHERE ID IN('".implode("','",$point_arr)." ')";
$result_gps = $conn->query($sql_gps);
$gps_data = array();
while($r = $result_gps->fetch_assoc())
{
$datas = array(
'id' => $r['ID'],
'lat' => $r['Latitude'],
'lon' => $r['Longitude']
);
array_push($gps_data, $datas);
}
```

Рис. 4.5. Фрагмент коду алгоритму енергозбереження

Коли *PHP* переміщує дані до масиву, функція *distance()* обчислює відстань між користувачем і певним пунктом та переміщує обчислений результат в інший масив (рис. 4.6). Нарешті, функція *usort()* сортує конкретний атрибут у масиві. Вище згадана функція сортує елементи масиву, використовуючи для порівняння

деяку попередньо вказану функцію зворотного виклику. Її доречно застосовувати, якщо потрібно відсортувати масив за певною нетиповою ознакою. Якщо перший параметр менший, ніж другий параметр, функція поверне негативне значення, інакше – поверне позитивне значення. Впорядкований масив буде зростаючим. Коли відстань буде меншою за радіус певного пункту, *PHP* відправить результат в мобільний додаток у форматі *JSON*, що користувач знаходиться в геозоні цього пункту.

```
for($i=0;$i<count($gps_data);$i++)
{
$pid = $gps_data[$i]['id'];
$pdis = distance($now_lat, $now_lon, $gps_data[$i]['lat'], $gps_data[$ i]['lon']);
$re = array(
'pid' => $pid,
'distance' => $pdis
);
array_push($res, $re);
}
usort($res, function($item1,$item2){
if($item1['distance'] == $item2['distance']) return 0;
return $item1['distance'] < $item2['distance'] ? -1:1;
});
```

Рис. 4.6. Фрагмент коду алгоритму енергозбереження

4.8. Висновки до розділу

В даному розділі було проведено процес розробки та здійснено декілька способів тестування розробленого додатку та усунено поблему, що виникла в процесі розробки.

Для перевірки коректності роботи створеного мобільного додатку було обрано два способи: аналіз граничних значень та тестування методом *black-box*. Перший з них передбачає перевірку поведінки продукту на граничних значеннях

вхідних даних, оскільки на крайніх значеннях найчастіше виникають збої роботи програмного продукту. Метод *black-box* було використано з метою тестування програмного продукту, дизайн, внутрішня структура та реалізація якого невідомі тестувальнику і перевірити результати на основі очікуваних результатів.

В процесі перевірки коректності роботи розробленого мобільного додатку було виявлено недолік: рівень заряду акумулятора пристрою, на якому його запущено, відносно швидко знижується.

З метою покращення роботи розробленої системи в цілому та збільшеня конкретно часу автономної роботи, було розроблено і впроваджено алгоритм економії заряду в мобільному додатку.

ВИСНОВКИ

В результаті виконання дипломної роботи було розроблено інформаційний геолокаційний додаток для автоматизації задач користувача.

В процесі вивчення поняття інформаційного додатку було виявлено, що являє собою прикладну програмну підсистему, що орієнтується на збір, зберігання, пошук та обробку інформації. Типовий інформаційний додаток включає в себе наступні компоненти:

1. Діалогове введення і виведення інформації.
2. Логіка діалогу.
3. Прикладна логіка обробки даних.
4. Логіка керування даними.
5. Операції, виконувані над файлами чи базами даних.

В інформаційному додатку, що взаємодіє з базою даних, прийнято виокремлювати наступні функціональні компоненти: засоби представлення, логіка представлення, компонент прикладної логіки, компонент логіки керування даними, операції, що здійснюються над базою даних, компонент файлових операцій.

Інструментальні програмні засоби розробки інформаційних додатків повинні забезпечувати наступні важливі властивості: підтримку багатоплатформності; незалежність від виробника; уніфікацію засобів розробки; створення надійного і якісного програмного забезпечення; підтримку розробленого програмного засобу протягом усього часу життя; проектування з використанням різних сучасних методик; ведення версій; підтримку вебтехнології.

Також проведено ознайомлення з поняттям автоматизації задач. Автоматизація забезпечує зменшення ручних повторюваних операцій. Врешті автоматизація необхідна, коли обсяги інформації, яку необхідно опрацювати і обробити, надто великі, для обробки людиною вручну.

В результаті впровадження автоматизації можна досягти: економії робочого часу працівника; спрощення роботи з клієнтами; зменшення похибки, зумовленої людським фактором; збільшення швидкості та якості роботи по обслуговуванню

клієнтів; збільшення швидкості отримання звітів та статистики; пришвидшення процесу підготовки документів, рахунків, актів; захист цінних конфіденційних даних.

Проведено огляд існуючих категорій геолокаційних додатків, а саме: додатки для надання інформації про найближчі об'єкти і відстань до них; геолокаційні додатки, що відображають розташування інших осіб; екскурсійні програми і додатки-путівники; геолокація з доповненою реальністю; навігатори. Виявлено ряд питань, які потребують уваги: автономність роботи програми; складність проектування інтерфейсу користувача, забезпечення якості.

Досліжено, що мобільний додаток являє собою вид додатку, що створений для запуску на мобільному пристрої, наприклад, смартфон або планшет та використовує мережеве підключення для роботи з віддаленими обчислювальними ресурсами Мобільні додатки часто слугують для надання послуг, подібних тим, що доступні на персональному комп’ютері.

Проведено ознайомлення з осоновними видами мобільних додатків, загалом їх можна розділити на три основні групи: нативні додатки, гібридні програми та вебдодатки. Кожен із перерахованих підходів має як ряд перваг, так і недоліків.

Було виділено основні етапи розробки мобільних додатків, а саме:

1. Обговорення ідеї майбутнього продукту.
2. Оцінка проекту.
3. Створення прототипу.
4. Розробка концепції дизайну продукту.
5. Розробка технічного завдання і клієнт-серверної архітектури.
6. Верстка візуальної частини.
7. Програмування та тестування.
8. Публікація мобільного додатку.
9. Супровід готового програмного продукту.

Також було досліджено життєвий цикл розробки мобільних додатків.

Для розробки інформаційного геолокаційного додатку було обрано наступні технології:

- для реалізації користувацького інтерфейсу застосовано: стандартизовану гіпертекстову мову розмітки *HTML*, мову стилів *CSS* та фреймфорк *Bootstrap*;
- мову *JavaScript* та бібліотеку *jQuery* – для динамічної обробки даних;

мову програмування *PHP* – для розробки внутрішньої серверної системи мобільного додатку;

- *JavaScript* і *MapBox* використовуються для побудови загальнодоступних та приватних інформаційних пунктів,
- *API Google Maps* використано для побудови карт для компонентів досліження;
- для реалізації бази даних, щоб зберігати всі дані розроблюваної системи застосовано *MySQL* та *GeoJSON*.

Для перевірки коректності роботи створеного мобільного додатку було обрано два способи: аналіз граничних значень та тестування методом *black-box*. Перший з них передбачає перевірку поведінки продукту на граничних значеннях вхідних даних, оскільки на крайніх значеннях найчастіше виникають збої роботи програмного продукту. Метод *black-box* було використано з метою тестування програмного продукту, дизайн, внутрішня структура та реалізація якого невідомі тестувальнику і перевірити результати на основі очікуваних результатів.

З метою покращення роботи розробленої системи в цілому та збільшеня конкретно часу автономної роботи, було розроблено і впроваджено алгоритм економії заряду в мобільному додатку.

Отже, основні результати дипломної роботи:

- досліджено, чому загалом потрібне вровадження автоматизації задач і чим воно може бути корисне, визначити, чи можна повністю автоматизувати робочі процеси;
- проаналізовано поняття інформаційного додатку, виділено основні аспекти розробки;

- визначено, що собою являє геолокаційне програмне забезпечення, з якою метою його розробляють, виділено основні напрями їх розробки та визначено які складнощі можуть виникати в процесі реалізації;
- досліджено поняття мобільного додатку, виділено основні етапи процесу розробки мобільних додатків;
- на основі отриманих знань розроблено власний програмний засіб.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бойченко С.В., Іванченко О.В. Положення про дипломні роботи (проекти) випускників Національного авіаційного університету. – К.: НАУ, 2017. – 63 с.
2. ДСТУ ГОСТ 3008-95 «Документація. Звіти у сфері науки і техніки. Структура і правила оформлення.»
3. *github.com* [Електронний ресурс]. – Режим доступу: *https://github.com*
4. *developer.mozilla.org* [Електронний ресурс]. – Режим доступу: *https://developer.mozilla.org/en-US/docs/Web/HTML*
5. *htmlbook.ru* [Електронний ресурс]. – Режим доступу: *http://htmlbook.ru/html5*
6. *css-tricks.com* [Електронний ресурс]. – Режим доступу: *https://css-tricks.com/almanac/*
7. *idg.net.ua* [Електронний ресурс]. – Режим доступу: *https://idg.net.ua/blog/uchebnik-css*
8. *JSON* syntax [Електронний ресурс] – Режим доступу до ресурсу: *https://www.w3schools.com/js/js_json_syntax.asp.*
9. *habr.com* [Електронний ресурс]. – *Режим доступу: https://habr.com/ru/*
10. *Eric Freeman, Elisabeth Robson. Head First HTML and CSS. – O'Reilly Media*, 2005. – 694 с.
11. *Eric Freeman. Head First JavaScript Programming: A Brain-Friendly Guide. – O'Reilly Media*, 2014. – 704 с.
12. *Lynn Beighley. Head First SQL: Your Brain on SQL. – O'Reilly Media*, 2007. – 608 с.
13. *Lynn Beighley. Head First PHP & MySQL: A Brain-Friendly Guide. – O'Reilly Media*, 2009. – 814 с.

13. *Rebecca M. Riordan. Head First Ajax: A Brain-Friendly Guide. – O'Reilly Media*, 2008. – 528 с.

14. *Jon Duckett. Web Design with HTML, CSS, JavaScript and jQuery Set. – John Wiley & Sons*, 2014. – 1152 с.

15. *Jon Duckett. HTML and CSS: Design and Build Websites. – John Wiley & Sons*, 2011. – 490 с.

16. *Robin Nixon. Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5 (Learning PHP, MYSQL, Javascript, CSS & HTML5). – O 'Reilly Media*, 2018. – 832 с.

17. *Kevin Tatroe, Peter MacIntyre. Programming PHP: Creating Dynamic Web Pages 4th Edition. – O'Reilly Media*, 2020. – 544 с.

18. *Douglas Crockfor. JavaScript: The Good Parts: The Good Parts. – O'Reilly Media*, 2008. – 176 с.

19. *David Herman. Effective JavaScript: 68 Specific Ways to Harness the Power of JavaScript (Effective Software Development Series). – Addison-Wesley Professional, 2012. – 240 с.*

20. *Young Rewired State. Get Coding!: Learn HTML, CSS & JavaScript & Build a Website, App & Game. – Candlewick*, 2017. – 208 с.

21. *George Berkowski. How to Build a Billion Dollar App: Discover the secrets of the most successful entrepreneurs of our time. – Piatkus*, 2015. – 408 с.

22. *Ben Forta. SQL in 10 Minutes a Day, Sams Teach Yourself. – Sams Publishing*, 2019. – 256 с.

23. *Saied Tahaghoghi, Hugh E. Williams. Learning MySQL: Get a Handle on Your Data. – O'Reilly Media*, 2006. – 618 с.

24. *Walter Shields. SQL QuickStart Guide: The Simplified Beginner's Guide to Managing, Analyzing, and Manipulating Data With SQL. – ClydeBank Media LLC*, 2019. – 251 с.

25. *Baron Schwartz, Peter Zaitsev, Vadim Tkachenko. High Performance MySQL: Optimization, Backups, and Replication. – O'Reilly Media,* 2012. – 826 с.

26. *Michael B. White. Mastering JavaScript: A Complete Programming Guide Including jQuery, AJAX, Web Design, Scripting and Mobile Application Development. – Independently published,* 2019. – 546 с.

27. *Hazem Saleh. JavaScript Mobile Application Development. – Packt Publishing,* 2014. – 332 с.

28. *Semmy Purewal. Learning Web App Development: Build Quickly with Proven JavaScript Techniques. – O'Reilly Media,* 2014. – 306 с.

29. *MG Martin. PHP: The Complete Guide for Beginners, Intermediate and Advanced Detailed Approach To Master PHP Programming. – Independently published,* 2019. – 238 с.

30. *Laura Thomson, Luke Welling. PHP and MySQL Web Development (Developer's Library). – Addison-Wesley Professional,* 2016. – 688 с.

Додаток А