МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК ТА ТЕХНОЛОГІЙ

Кафедра Комп'ютерних інформаційних технологій

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач випускової кафедри
Аліна САВЧЕНКО
«_____» _____ 2023р.

КВАЛІФІКАЦІЙНА РОБОТА

(ДИПЛОМНИЙ ПРОЄКТ, ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ **"БАКАЛАВР"**
ЗА ОСВІТНЬО-ПРОФЕСІЙНОЮ ПРОГРАМОЮ
"ІНФОРМАЦІЙНІ УПРАВЛЯЮЧІ СИСТЕМИ ТА ТЕХНОЛОГІЇ"

Тема: «Допоміжний додаток для рибальства**»**

Виконавець: студентка групи УС-412 Адіанова Тетяна Сергіївна

Керівник: к.т.н., доцент Холявкіна Тетяна Володимирівна

Нормоконтролер: ст. викл. Олександр ШЕВЧЕНКО

Київ - 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних наук та технологій*

Кафедра *Комп'ютерних інформаційних технологій*

Галузь знань, спеціальність, освітньо-професійна програма: 12 "Інформаційні технології", 122 "Комп'ютерні науки", "Інформаційні управляючі системи та технології"

ЗАТВЕРДЖУЮ
Завідувач випускової кафедри
Аліна САВЧЕНКО
«____»_____2023р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи студента

Адіанової Тетяни Сергіївни
(прізвище, ім'я, по батькові)

1. Тема роботи: «Допоміжний додаток для рибальства» затверджена наказом ректора від 01.05.2023 р. за № 623/ст.

2. Термін виконання роботи: з 15.05.2023 р. по 25.06.2023 р.

3. Вихідні данні до роботи: програмний продукт для поширення та зберігання даних про місця риболовлі.

4. Зміст пояснювальної записки: аналіз наявних програмних продуктів, проведення перевірки наявних мов програмування, огляд інтегрованого середовища розробки програмного забезпечення, огляд фреймворків для реалізації інтерфейсу користувача.

5. Перелік обов'язкового графічного матеріалу: діаграма варіантів використання, діаграма класів системи, головна сторінка сайту.

6. Календарний план-графік

№ п/п	Завдання	Термін виконання	Підпис керівника
1.	Проаналізувати літературу та джерела за темою дипломного проекту.	15.05.2023 – 17.05.2023	
2.	Розроблення та затвердження плану дипломного проекту.	18.05.2023 – 20.05.2023	
3.	Провести консультації з науковим керівником щодо створення першого розділу.	21.05.2023– 23.05.2023	
4.	Розробка розділу 1	24.05.2023– 02.06.2023	
5.	Розробка розділу 2	03.06.2023– 09.06.2023	
6.	Розробка розділу 3	10.06.2023 – 13.06.2023	
7.	Висновки та оформлення пояснювальної записки дипломного проекту.	14.06.2023 – 16.05.2023	
8.	Підписання необхідних документів у встановленому порядку.	17.05.2023 – 19.05.2023	

7. Дата видачі завдання: «15» травня 2023 р.

Керівник дипломної роботи _____ Тетяна ХОЛЯВКІНА
(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання _____ Тетяна АДІАНОВА
(підпис випускника) (П.І.Б.)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи «Допоміжний додаток для рибальства» складається зі вступу, трьох розділів, висновку, списку бібліографічних посилань і містить 66 сторінки, 1 таблицю та 14 рисунків. Список бібліографічних посилань складається з 12 найменувань.

Об'єктом дослідження є процес створення допоміжного веб-додатку для риболовлі.

Предметом дослідження є засоби створення допоміжного веб-додатку для риболовлі.

Метою дипломної роботи є дослідження теоретичних аспектів розробки програмного забезпечення, інформаційної системи, веб-додатку; огляд основних принципів веб-розробки; аналіз та вибір архітектури програмного забезпечення; спроектувати майбутній функціонал системи; спроектувати внутрішню будову системи; вибір інструментальних засобів розробки веб-додатку; вибір середовища розробки; розробка графічного інтерфейсу користувача; тестування системи.

Методи дослідження включають у себе методи тестування програмного забезпечення, методи створення тест-кейсів.

Отримані результати включають у себе аналіз предметної області, обрані засоби реалізації, аналіз варіантів використання, проект внутрішньої будови, спроектований графічний інтерфейс користувача, тестування системи.

ВЕБ-ДОДАТОК, РИБОЛОВЛЯ, МОВИ ПРОГРАМУВАННЯ, JAVA, IntelliJ IDEA, SPRING FRAMEWORK

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

СУБД	Система управління базами даних
ІС	Інформаційна система
ІКТ	Інформаційно-комунікаційні технології
ОС	Операційна система
SQL	Structured query language
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
JSP	Java Server Pages
CMS	Content Management System
MVC	Model-View-Controller
GUI	Graphical user interface
UML	Unified Modeling Language
БД	База даних
JVM	Java Virtual Machine
LTS	Long-Term Support
PHP	Hypertext Preprocessor
HTTP	HyperText Transfer Protocol
ASP	Active Server Pages
IDE	Integrated development environment

ВСТУП

В сучасному світі тенденція створення програмного забезпечення для полегшення роботи і життя стає все більш явною.

Спрощення різноманітних процесів проникає в усі сфери людського життя, починаючи роботою і побутом, і закінчуючи освітою.

Сферу дозвілля дана тенденція також не оминула, кожного дня спрощуючи порядок отримання і надання різноманітних послуг.

Риболовля - популярне хобі, воно дозволяє людям відчути захоплення від процесу лову риби та релаксацію завдяки тиші та природі. Кожен рибак має унікальний досвід та емоції, які викликані рибальством. Саме розповіді батька надихнули мене на створення сучасного допоміжного веб-додатку для рибальства, адже за його досвідом сфера риболовлі мало диджиталізована.

За останні роки інтерес до риболовлі значно зріс, і все більше людей вступають в цю діяльність. Розробка сучасного веб-додатку для рибалки **актуальна саме тому**, бо може задовольнити попит на нові технологічні рішення, що полегшують організацію та покращують досвід рибалок. Наприклад, надання детальної інформації про придатність рибальського місця, ведення статистики уловів, що дозволяє розуміти, яку рибу, де краще ловити та в який час. Також додаток для рибалки може бути корисним для покращення комунікації та обміну інформацією між рибалками.

Новизна полягає в тому, що розроблено інтерактивну карту з різними місцями для риболовлі, такими як озера, річки, ставки та інші водойми. Користувачі зможуть переглядати місця для риболовлі, а також позначати місця, де вони спіймали рибу. Ця функція дозволить рибалкам обмінюватися цінною інформацією та знаходити нові місця для успішної риболовлі. Також розроблено персоналізовані рекомендації, а саме веб-додаток для риболовлі може робити аналіз даних про попередні улови користувача, місцезнаходження, та інші фактори. На основі цих даних додаток може

надавати персоналізовані рекомендації щодо найкращих місць, часу та способів риболовлі для кожного користувача.

Саме з цією тематикою і пов’язана тема даної роботи.

Мета роботи: розробка системи зберігання і шарінгу даних про місця риболовлі.

Для досягнення обраної мети слід виконати такі завдання:

- проаналізувати предметну область;
- обрати засоби реалізації;
- проаналізувати варіанти використання;
- спроектувати внутрішню будову;
- спроектувати графічний інтерфейс користувача;
- провести тестування системи.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Прикладне програмне забезпечення

Прикладне програмне забезпечення - це один з видів програмного забезпечення, який забезпечує виконання конкретних функцій для кінцевих користувачів через безпосередню взаємодію з ними. Головна мета прикладного програмного забезпечення полягає в тому, щоб надавати допомогу користувачеві у виконанні певних завдань.

В якості прикладів прикладного програмного забезпечення можна зазначити різноманітні веб-браузери, такі як Safari або Google Chrome, а також популярні програми, включаючи Microsoft Word або Excel, які широко використовуються на персональних комп'ютерах і ноутбуках. Також до цієї категорії належать різноманітні мобільні додатки, наприклад Viber або Telegram. Крім того, існують додатки, які представляють версії популярних сервісів, що користуються популярністю серед людей щодня, наприклад, додатки з інформацією про погоду або громадський транспорт. Зокрема, існують також додатки, що сприяють забезпеченню взаємозв'язку між кінцевими користувачами та їхніми бізнесами.

Прикладне програмне забезпечення є необхідною складовою сучасного цифрового середовища. Воно розробляється з метою надання користувачеві певних функцій та послуг, що спрощують його роботу, розваги, комунікацію або забезпечують вирішення конкретних завдань. Прикладне програмне забезпечення може бути встановлене на комп'ютер, ноутбук, смартфон, планшет або інші пристрої.

В сучасному світі існує широкий спектр прикладного програмного забезпечення для різних сфер життя. Наприклад, в бізнесі використовуються програми для управління проектами, фінансового обліку, клієнтського сервісу, маркетингу тощо. В освіті та навчанні широке застосування мають навчальні платформи, електронні підручники, мультимедійні інструменти. У сфері медицини розробляються медичні додатки для діагностики, моніторингу здоров'я, обміну медичними даними.

Функції прикладного програмного забезпечення є різноманітними і спрямовані на вирішення різного роду завдань. Деякі з них включають:

- правління інформацією та даними в організації, що допомагає впоратися з обробкою та організацією великого обсягу інформації;
- ведення та перевірка документів, що спрощує процеси документообігу та забезпечує точність та доступність даних;
- розробка програмного забезпечення для освітніх цілей, наприклад, системи управління навчанням і системи електронного навчання, що дозволяють покращити процеси вивчення;
- створення візуальних матеріалів та відео для презентаційних цілей, що дозволяє ефективно комунікувати та передавати інформацію широкій аудиторії;
- забезпечення зручних способів комунікації, включаючи електронну пошту, текстові повідомлення, аудіо- та відеоконференції, що допомагає підтримувати зв'язок та здійснювати ефективну комунікацію;
- управління бухгалтерським обліком, фінансами та нарахуванням заробітної плати, що полегшує процеси фінансового обліку та допомагає відділу кадрів та підприємствам з управління фінансовими ресурсами;
- планування ресурсів підприємства (ERP), що дозволяє ефективно керувати ресурсами компанії, включаючи виробничі потужності, запаси, фінанси та зв'язки з постачальниками;
- управління взаємовідносинами з клієнтами (CRM), що допомагає підприємствам підтримувати та розвивати взаємовідносини зі своїми

клієнтами, включаючи керування контактами, зберігання інформації про замовлення та зв'язок з клієнтами;

- управління будь-якими малими чи великими проектами, що надає інструменти для планування, керування ресурсами, відстеження прогресу та спілкування в команді для успішного виконання проектів;
- управління плавними та ефективними бізнес-процесами, що дозволяє автоматизувати та оптимізувати різні бізнес-процеси, включаючи замовлення, логістику, виробництво, облік та інші аспекти діяльності підприємства.

1.1.1. Типи прикладного програмного забезпечення

Прикладне програмне забезпечення можна класифікувати залежно від різноманітних факторів, таких як вартість і доступність. Давайте розглянемо декілька типів програмного забезпечення та їх особливості.

Програмне забезпечення для обробки тексту

Програмне забезпечення для обробки текстів, наприклад Microsoft Word (рис. 1.1.), є прикладним програмним забезпеченням, яке дозволяє створювати, редагувати, зберігати та друкувати документи. Вони широко використовуються професіоналами та іншими користувачами для обробки текстів.

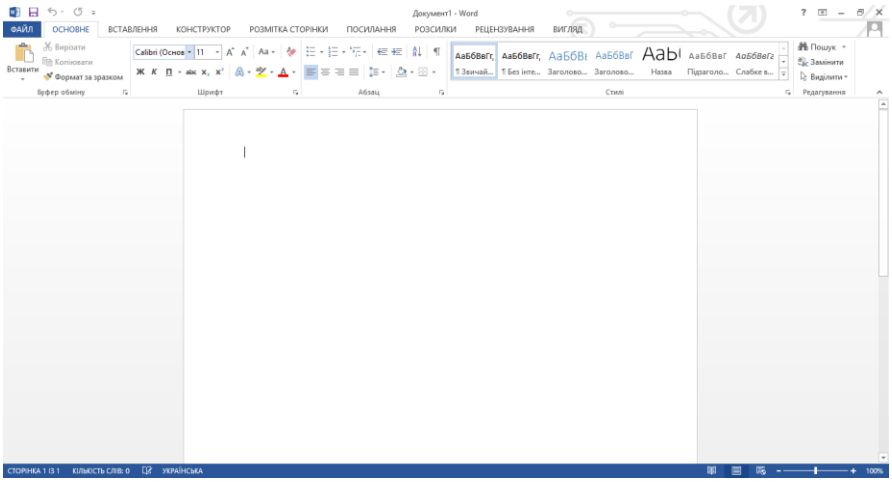

Рис 1.1. Microsoft Word 2010

Програмне забезпечення для роботи з електронними таблицями

Програмне забезпечення для роботи з електронними таблицями, як Microsoft Excel (рис 1.2.), надає можливість виконувати числові функції та проводити аналіз числових даних. Воно є потужним інструментом для бухгалтерів, аналітиків та інших професійних користувачів.

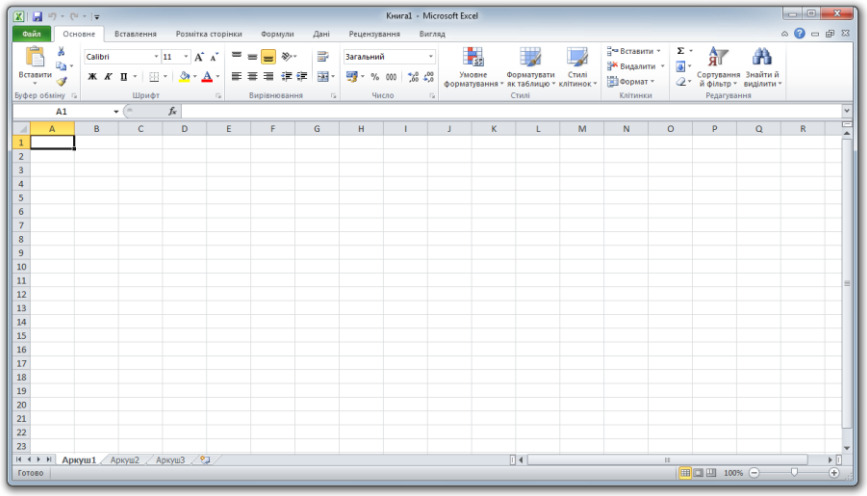

Рис. 1.2. Microsoft Excel 2010

Програмне забезпечення для презентацій

Програмне забезпечення для презентацій, такі як PowerPoint (рис. 1.3.), дозволяють створювати послідовності слайдів з текстом і зображеннями для ефективного виступу або публічної презентації.

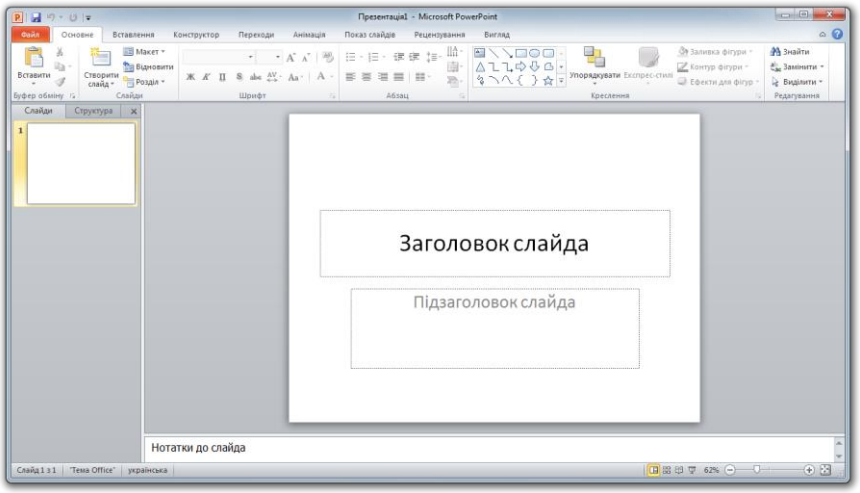

Рис. 1.3. Microsoft PowerPoint 2010

Мультимедійне програмне забезпечення

Мультимедійне програмне забезпечення дозволяє створювати, редагувати і відтворювати текст, зображення, аудіо та відео. Воно застосовується як у професійній, так і у особистій сферах, дозволяючи створювати різноманітний інтерактивний вміст. Відомими прикладами мультимедійного програмного забезпечення є програвач MX Player і Windows Media Player (рис. 1.4.) .

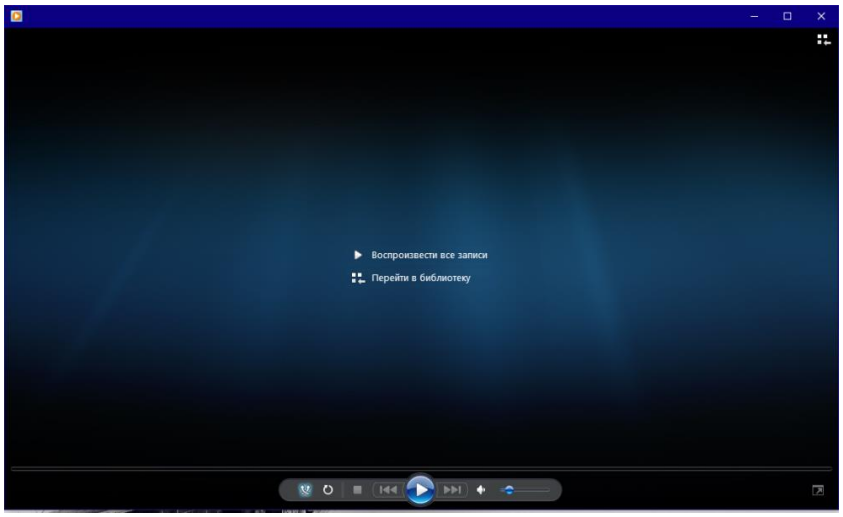

Рис. 1.4. Windows Media Player 12 у Windows 10

Веб-браузери

Веб-браузери, такі як Chrome, Firefox або Safari, використовуються для перегляду веб-сторінок і надають доступ до різноманітних веб-ресурсів.

Навчальне програмне забезпечення

Навчальне програмне забезпечення спеціально розроблено для освітніх цілей, включаючи програми для вивчення мови, керування класом та довідкове програмне забезпечення для студентів. Прикладами освітнього програмного забезпечення є EDX, MindPlay і Kid Pix.

Графічні програми

Графічне програмне забезпечення, таке як Photoshop або Illustrator, використовується для створення та редагування растрової і векторної графіки, включаючи дизайн етикеток та іншого візуального контенту.

Безкоштовне програмне забезпечення

Існує широке розмаїття програмного забезпечення, яке доступне безкоштовно для користувачів. Хоча зазвичай ці продукти створюються з комерційною метою, заробляючи на рекламі, підписках або додаткових функціях, вони також можуть використовуватися з комерційними цілями з метою просування власних платних аналогів або розширення частки ринку. Приклади таких безкоштовних програм з закритим кодом включають такі популярні продукти, як Adobe Reader, Free Studio та Skype.

Умовно-безкоштовна програма

Умовно-безкоштовне програмне забезпечення надається на пробній основі, де користувач може використовувати програму протягом обмеженого часу перед прийняттям рішення про придбання. Прикладом умовно-безкоштовного програмного забезпечення є WinZip (рис. 1.5.) .

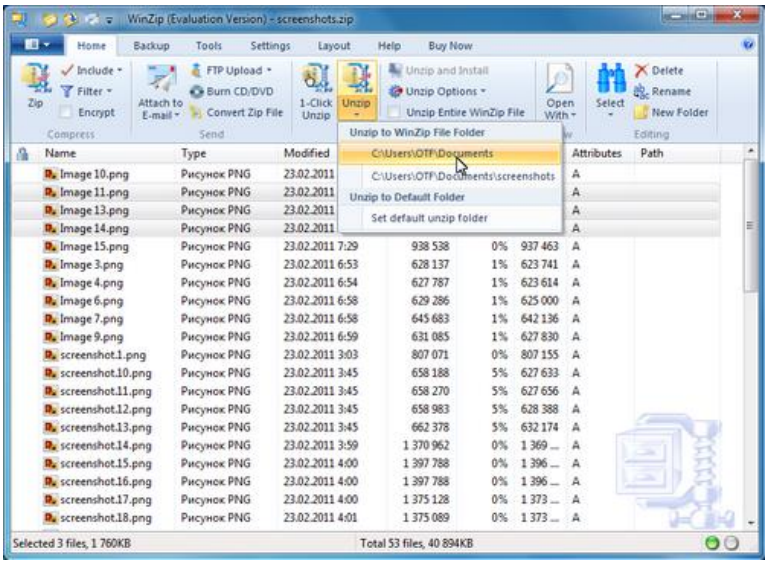

Рис. 1.5. WinZip 15 в Windows 7

Програмне забезпечення для моделювання

Програмне забезпечення для моделювання дозволяє інженерам оцінювати та оптимізувати дизайн продукту шляхом моделювання реальних подій у віртуальному середовищі. MATLAB (рис. 1.6.) є найкращим прикладом програмного забезпечення для моделювання.

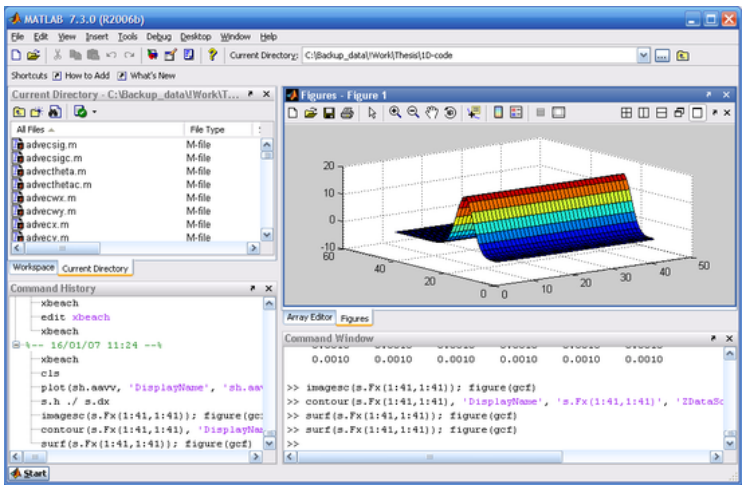

Рис. 1.6. MATLAB версії 7.3.0 (R2006b)

Відкрите джерело

Програмне забезпечення з відкритим кодом є варіантом програмного забезпечення, яке надається разом з вихідним кодом, що робить його доступним для використання, змін та поширення з усіма відповідними правами власності. Цей підхід сприяє відкритості та співпраці у розробці програмного забезпечення, дозволяючи розробникам адаптувати його до своїх потреб та спільно покращувати функціональність і безпеку продукту.

Закритий код

На відміну від програмного забезпечення з відкритим кодом, програмне забезпечення з закритим кодом не дозволяє користувачам копіювати, вилучати або змінювати частини вихідного коду. Це може мати наслідком різні наслідки, які можуть бути незначними або серйозними, включаючи анулювання гарантії, потенційні судові позови та інші правові наслідки.

Програмне забезпечення для бізнесу

Бізнес-програмне забезпечення є невід'ємною частиною бізнес-сфери і використовується для реалізації різних завдань та цілей. Термін "бізнес-програмне забезпечення" відноситься до програмного забезпечення, спеціально розробленого для підтримки бізнес-процесів та виконання конкретних завдань у контексті бізнесу. Воно може включати прикладні принципи та функціональні можливості, спрямовані на оптимізацію операцій, автоматизацію процесів, аналіз даних та прийняття

стратегічних рішень. Бізнес-програмне забезпечення відіграє важливу роль у підвищенні ефективності та конкурентоспроможності підприємств, надаючи їм потужні інструменти для керування та розвитку.

Управління взаємовідносинами з клієнтами (CRM)

CRM програмне забезпечення надає повну картину всіх взаємодій з клієнтами, відстежує продажі, організовує та встановлює пріоритети для можливостей та сприяє співпраці між різними командами. CRM-програмне забезпечення є необхідним інструментом для будь-якого бізнесу, що прагне забезпечити високу якість обслуговування клієнтів та зростання на ринку.

Планування ресурсів підприємства (ERP)

Планування ресурсів підприємства (ERP) є типом програмного забезпечення, яке використовується організаціями для керування різноманітними аспектами повсякденної бізнес-діяльності. Це включає такі функції, як бухгалтерський облік, управління закупівлями, проектами, ризиками та відповідність нормативним вимогам, а також операції в ланцюжку поставок. Крім того, повний пакет ERP може включати також інструменти для управління ефективністю підприємства, які допомагають в плануванні, бюджетуванні, прогнозуванні та звітуванні про фінансові результати організації.

Програмне забезпечення для керування проектами

Програмне забезпечення для керування проектами є невід'ємною частиною багатьох галузей і використовується для ефективного планування проектів, розподілу ресурсів і управління їх виконанням. Воно дозволяє керівникам проектів та командам повністю контролювати бюджет, якість та документацію, що обмінюється протягом проекту. Завдяки програмному забезпеченню для керування проектами, організації мають змогу підвищити продуктивність, зменшити ризики та досягти успішного завершення проектів.

База даних

Програмне забезпечення для баз даних відіграє важливу роль у створенні та управлінні базами даних, а також у зберіганні, пошуку та отриманні різноманітної інформації з них. Це спеціалізоване програмне забезпечення забезпечує надійність

та ефективність в роботі з даними шляхом організації і структурування інформації, забезпечення безпеки та доступу до даних, а також можливостей для пошуку, фільтрації та аналізу інформації за різними параметрами.

Управління бізнес-процесами

Управління бізнес-процесами включає в себе набір інструментів та рішень, які допомагають організаціям досягати гнучкості шляхом автоматизації, управління та оптимізації процесів.

Програмне забезпечення для керування ресурсами

Програмне забезпечення для управління ресурсами є невід'ємним цифровим рішенням, яке допомагає користувачам визначати та оптимізувати ресурси, необхідні для успішного виконання завдань та проектів. Цей інструмент відіграє важливу роль у плануванні та контролі ресурсів, допомагаючи визначити, коли саме ці ресурси будуть потрібні.

Навчальне програмне забезпечення

Навчальне програмне забезпечення відноситься до комп'ютерних програм та інших цифрових технологій, розроблених спеціально для підтримки процесу навчання та викладання. Навчальне програмне забезпечення має широкий спектр застосувань, включаючи інтерактивні навчальні матеріали, електронні підручники, навчальні платформи, ігрові та симуляційні програми, а також інструменти для оцінювання та відстеження успішності студентів.

1.2. Поняття і сутність інформаційної системи

Інформаційна система (ІС) - це комплекс взаємопов'язаних компонентів, що мають на меті збір, обробку, зберігання та поширення інформації для досягнення поставлених цілей. Завдяки механізму зворотного зв'язку, ІС сприяє досягненню організаційних цілей, підвищенню прибутковості, поліпшенню обслуговування клієнтів та сприяє прийняттю рішень та контролю в організаціях [4].

Компанії активно використовують інформаційні системи для збільшення прибутку та ефективного управління витратами.

В організаціях інформаційні системи базуються на чотирьох основних елементах, як запропонував Гарольд Лівітт у 1960-х роках (рис. 1.7.). Цей шаблон відомий як "діамант Лівітта".

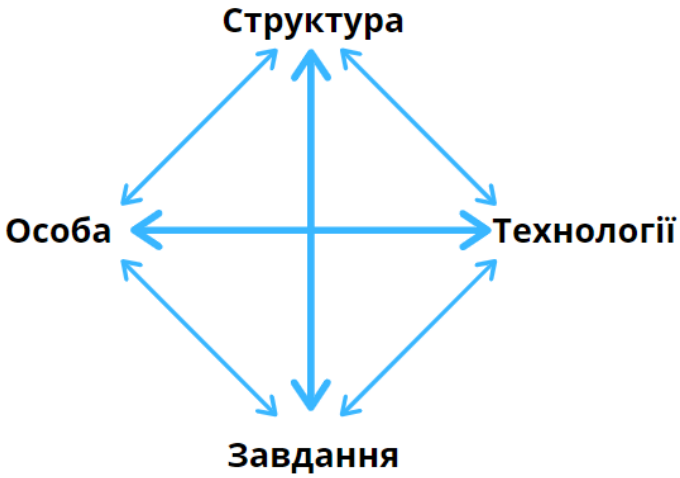

Рис. 1.7. Діамант Левітта: Соціально-технічний погляд на ІС

Технології: інформаційні технології ІС - це апаратне забезпечення, програмне забезпечення та телекомунікаційне обладнання, що використовується для захоплення, обробки, зберігання та поширення інформації. Більшість сучасних ІС ґрунтуються на інформаційних технологіях, оскільки вони забезпечують ефективне виконання операцій та управління будь-якого розміру.

Завдання: Цей компонент ІС включає в себе діяльність, необхідну для виробництва товарів або надання послуг. Ця діяльність підтримується потоком матеріалів, інформації та знань між різними учасниками.

Особа: Компонент, пов'язаний з людьми в інформаційній системі, охоплює всіх осіб, які безпосередньо беруть участь у функціонуванні системи. Серед них знаходяться менеджери, відповідальні за визначення цілей системи, користувачі, які використовують систему у своїй роботі, а також розробники, які відповідають за створення та підтримку інформаційної системи. Кожна з цих груп осіб має свої функції та відповідальності в рамках інформаційної системи, сприяючи її ефективному функціонуванню та досягненню організаційних цілей.

Структура: Даний елемент належить до структури організації та взаємозв'язку між окремими складовими частинами людського фактора в інформаційній системі. Він включає в себе ієрархічні структури, взаємовідносини між працівниками та системи оцінки їхньої роботи. Цей компонент допомагає встановити ефективну організаційну структуру, забезпечує зв'язок та координацію між працівниками, а також використовує системи оцінки для оцінки та розподілу ресурсів відповідно до результатів роботи кожного працівника.

Кожна інформаційна система має своєю основною метою забезпечення ефективності операцій, управління та процесів прийняття рішень. Інформаційна система включає в себе широкий спектр інформаційно-комунікаційних технологій (ІКТ), які застосовуються організацією, а також визначає способи взаємодії між людьми та цими технологіями з метою підтримки бізнес-процесів. Кожна інформаційна система розробляється з урахуванням конкретних потреб та вимог організації, сприяючи покращенню продуктивності, ефективності та конкурентоспроможності.

Деякі дослідники чітко відмежовують інформаційні системи, комп'ютерні системи та бізнес-процеси як різні концепції. Інформаційні системи, зазвичай, включають компоненти ІКТ, але не обмежуються лише ними, і акцентують увагу на використанні технологій для досягнення кінцевих цілей. Інформаційні системи відрізняються від бізнес-процесів тим, що вони сприяють контролю та управлінню ефективністю бізнес-процесів.

Альтер докладно освітлив переваги розуміння інформаційних систем як окремого типу робочих систем. Робоча система представляє собою комплекс, в якому люди або машини використовують різноманітні ресурси для здійснення процесів та операцій з метою створення конкретних продуктів або надання послуг клієнтам. З іншого боку, інформаційна система є робочою системою, спрямованою на збір, передачу, зберігання, пошук, обробку та відображення інформації, яка має велике значення для функціонування організації.

Тому інформаційні системи мають глибокі зв'язки з системами обробки даних з одного боку і бізнес-системами з іншого. Інформаційна система може бути

розглянута як форма комунікаційної системи, де дані надходять і обробляються як соціальна пам'ять. Крім того, інформаційні системи можна розглядати як напівформальну мову, що сприяє прийняттю людських рішень і дій, надаючи певну структуру та підтримку для ефективної взаємодії.

Розмаїтість типів інформаційних систем охоплює системи обробки транзакцій, системи підтримки прийняття рішень, системи управління знаннями, системи управління навчанням, системи управління базами даних та офісні інформаційні системи. Основною складовою більшості інформаційних систем є інформаційні технології, які призначені для виконання завдань, що перевищують можливості людського інтелекту, такі як обробка великих обсягів інформації, складні обчислення та керування багатьма паралельними процесами.

1.3. Поняття веб-додатку

Веб-додаток - це прикладна програма, яка використовується для виконання конкретних завдань і зберігається на віддаленому сервері. Користувачі мають можливість отримувати доступ до цього додатку шляхом використання спеціального програмного забезпечення - веб-браузера. Такий підхід дозволяє користувачам взаємодіяти з додатком через Інтернет без необхідності встановлювати його локально на своєму комп'ютері.

Веб-додатки використовують різноманітні сценарії як на стороні сервера (PHP і ASP), так і на стороні клієнта (JavaScript і HTML) для ефективного зберігання, отримання та відображення інформації для користувачів. Це дає можливість здійснювати взаємодію з компанією шляхом використання онлайн-форм, систем управління контентом, кошиків для покупок та інших функцій. Крім того, такі додатки дозволяють співробітникам створювати документи, обмінюватися інформацією, співпрацювати над проектами та працювати зі спільними документами незалежно від їх розташування або пристрою.

Веб-додатки часто розробляються мовами, які підтримуються браузерами, наприклад, JavaScript і HTML. Ці мови дозволяють браузеру відтворювати та

виконувати програмний код веб-додатків. Деякі програми мають динамічний характер і вимагають обробки на стороні сервера, тоді як інші є повністю статичними і не потребують обробки на сервері.

Для функціонування веб-додатку необхідний веб-сервер для обробки запитів від клієнта, сервер додатків для виконання задач і, в деяких випадках, база даних для зберігання інформації. Серверні технології можуть варіюватися від ASP.NET, ASP і ColdFusion до PHP і JSP. Кожна з цих технологій надає різноманітні можливості для розробки та виконання веб-додатків залежно від потреб і вимог проекту.

Що стосується інтерфейсу, використання JavaScript, CSS, Java, Flash, Silverlight дозволяє реалізувати різноманітні спеціальні функції, такі як малювання на екрані, відтворення звуку і взаємодія з клавіатурою та мишею. Протягом часу було розроблено багато служб, які поєднували ці функції в знайомому інтерфейсі, що нагадує зовнішній вигляд операційної системи. Ці технології також підтримують загальні методи, наприклад, перетягування елементів на екрані. Завдяки цим можливостям користувачі можуть насолоджуватися багатогранним та інтерактивним досвідом взаємодії з веб-додатками. Ajax, яка використовує комбінацію різних технологій, є прикладом технології, яка забезпечує більш інтерактивний досвід.

Наведемо приклад як виглядає типовий потік веб-програми:

1. Користувач ініціює запит до веб-сервера через Інтернет через веб-браузер або інтерфейс користувача програми.
2. Веб-сервер пересилає цей запит на відповідний сервер веб-додатків.
3. Сервер веб-додатків виконує потрібне завдання, наприклад надсилає запит до бази даних або обробляє дані, а потім генерує результати запитуваних даних.
4. Сервер веб-додатків надсилає результати на веб-сервер із запитуваною інформацією або обробленими даними.
5. Веб-сервер відповідає клієнту запитуваною інформацією, яка потім з’являється на дисплеї користувача.

1.4. Основні принципи веб-розробки

Веб-сайт є набором веб-сторінок, пов'язаних між собою гіперпосиланнями та організованих у логічну структуру. Веб-сайти можуть мати різні цілі та функціональні можливості. Серед найпоширеніших різновидів веб-сайтів можна виділити особисті веб-сторінки, інформаційні портали, електронні магазини, блоги та соціальні мережі.

Загальнодоступні веб-сайти є невід'ємною частиною Всесвітньої павутини, проте варто зазначити, що також існують приватні веб-сайти, які доступні лише в рамках приватних мереж, наприклад, внутрішніх корпоративних порталів для співробітників.

Кожен веб-сайт, зазвичай, має конкретну тематику або мету, такі як надання новин, освітні ресурси, комерційні послуги, розважальні контент або соціальні мережі. Навігація по веб-сайту здійснюється за допомогою гіперпосилань, які перенаправляють користувача між різними веб-сторінками, зазвичай починаючи з домашньої сторінки сайту.

Користувачі можуть отримати доступ до веб-сайтів з різноманітних пристроїв, включаючи настільні комп'ютери, ноутбуки, планшети та смартфони. Вони використовують веб-браузери для зручного перегляду веб-сторінок.

Веб-сайти мають різноманітне призначення: особисті блоги, корпоративні портали компаній, офіційні веб-сайти урядових органів, сторінки організацій та інші. Вони можуть належати приватним особам, підприємствам або організаціям, і зазвичай присвячені конкретній тематиці або меті. Важливо зазначити, що кожен веб-сайт може містити гіперпосилання на інші сайти, тому чітка межа між окремими сайтами з погляду користувача може бути розмитою.

Статичний веб-сайт представляє собою сайт, який складається зі сторінок, що містять незмінний вміст або простий код HTML, JavaScript або CSS. Цей тип веб-сайту не залежить від користувача або його місцезнаходження і має обмежену інтерактивність. Оскільки вміст статичного сайту залишається постійним, його можна передбачити та попередньо відобразити, що забезпечує високу

продуктивність та мінімальні витрати в простих сценаріях використання. Прості статичні сайти є ідеальним вибором для проектів, де вміст рідко змінюється або для веб-сайтів з простою структурою.

Динамічні веб-сайти представляють собою сайти, які надають вміст, залежно від користувача. При розробці таких сайтів враховуються такі фактори, як мова, місцезнаходження та цільова аудиторія, і вміст подається на основі цих вхідних даних користувача.

Для забезпечення динамічного вмісту потрібно, щоб кожний запит користувача оброблявся, шляхом отримання відповідної інформації з бази даних, заповнення сторінки вмістом і його подання користувачеві. Хоча цей підхід дозволяє надавати користувачеві дуже релевантну інформацію, він часто включає компроміси, такі як продуктивність та безпека.

Розробка динамічних веб-сайтів вимагає використання програмних мов та технологій, які дозволяють взаємодіяти з базою даних, обробляти інформацію та генерувати вміст на льоту. При цьому слід забезпечувати ефективну роботу сайту навіть при великому навантаженні та забезпечувати безпеку обробки даних користувачів. Для цього використовуються різні програмні системи та фреймворки, такі як CGI, Java-сервлети, JSP, Active Server Pages і ColdFusion (CFML), а також мови програмування, наприклад, Perl, PHP, Python та Ruby.

Динамічні веб-сайти надають можливість відображати актуальний стан діалогу з користувачами, враховувати змінні умови та надавати персоналізовану інформацію, відповідно до потреб кожного окремого користувача. Наприклад, головна сторінка новинного сайту може поєднувати збережені фрагменти HTML з новинами, які отримуються з бази даних або з інших джерел за допомогою RSS, для створення сторінки з найсвіжішою інформацією. Динамічні сайти часто мають інтерактивні елементи, використовують HTML-форми, зберігають і зчитують файли cookie браузера або створюють сторінки, які відображають попередню історію кліків користувача. JavaScript використовується для динамічної зміни вмісту сторінок у веб-браузері. Також існують методи, що дозволяють моделювати певний тип динамічного веб-сайту, шляхом періодичної генерації великої кількості

статичних сторінок, що дозволяє уникнути затрат часу на ініціювання динамічного процесу для кожного окремого користувача або з'єднання.

HTML5 є одним з найзручніших елементів для розробки динамічних веб-сайтів, оскільки він включає в себе вбудовану підтримку аудіо та відео без потреби використання додаткових плагінів. Крім того, сучасні веб-браузери надають вбудовану підтримку JavaScript, що дозволяє розробникам відправляти код веб-браузеру для динамічного змінення вмісту сторінок та взаємодії з веб-сервером при необхідності. Внутрішня модель документа браузера, відома як DOM (Document Object Model), використовується для представлення структури і вмісту веб-сторінок. Для створення динамічних елементів на сторінках використовується технологія, відома як динамічний HTML.

WebGL (Web Graphics Library) - є технологією, що дозволяє відображати 3D-графіку безпосередньо в браузері. Вона базується на мові програмування JavaScript і використовує апаратне забезпечення комп'ютера для прискорення графічних обчислень. Завдяки WebGL розробники можуть створювати захоплюючі 3D-візуалізації та ігри, які працюють безпосередньо у веб-браузері.

У сучасному веб-дизайні широко поширюється концепція "адаптивного дизайну". Адаптивний дизайн відноситься до підходу у веб-розробці, коли веб-сайт динамічно пристосовується до різних розмірів екранів та пристроїв, на яких він відображається. Цей підхід дозволяє забезпечити оптимальний користувацький досвід незалежно від того, чи відкривається веб-сайт на настільному комп'ютері, смартфоні чи планшеті. Адаптивний дизайн використовує різні техніки, такі як гнучкі сітки, медіа-запити та респонсивні зображення, щоб забезпечити оптимальне відображення та взаємодію змісту сайту на різних пристроях і екранах.

Статичні веб-сайти і інтерактивні веб-сайти представляють два різні підходи до веб-розробки і мають відмінності щодо взаємодії з користувачем та оновлення вмісту. Статичні веб-сайти є основними формами веб-презентацій. Вони складаються зі сторінок, які містять фіксований вміст, який залишається незмінним незалежно від відвідувачів чи контексту. Статичні веб-сайти зазвичай створюються за допомогою простого HTML-коду, який описує структуру та вигляд сторінок.

Вони можуть містити текст, зображення, відео та посилання, але цей вміст не змінюється автоматично. Для оновлення статичного веб-сайту потрібно вручну редагувати HTML-код.

З іншого боку, інтерактивні веб-сайти надають користувачам можливість взаємодіяти з вмістом сайту. Вони забезпечують більш динамічний та змінний досвід для користувачів. Інтерактивні веб-сайти використовують різні технології, для динамічного змінення вмісту сторінок під час взаємодії користувача зі сайтом. Вони можуть мати функціональні елементи, такі як введення даних, анімація, форми, валідація даних та інші елементи, які відповідають на дії користувача.

Деякі веб-сайти можуть бути включені в одну або декілька з цих категорій. Наприклад, деякі веб-сайти великих компаній, таких як Facebook, Yahoo!, Microsoft і Google, мають складну інфраструктуру, що включає в себе багато серверів та обладнання для збалансування навантаження.

Одним з поширених методів для забезпечення ефективного розподілу навантаження на таких веб-сайтах є використання комутаторів служб вмісту Cisco. Ці комутатори виконують функцію розподілу трафіку між різними серверами, розташованими у різних місцях. Вони дозволяють розподіляти навантаження відвідувачів між цими серверами, забезпечуючи оптимальну швидкість та доступ до вмісту.

Цей підхід до розподілу навантаження дозволяє веб-сайтам витримувати великі обсяги трафіку та забезпечувати стабільну роботу для великої кількості користувачів. Комутатори служб вмісту Cisco є популярним рішенням для таких масштабних веб-сайтів, оскільки вони забезпечують ефективне розподілення навантаження та підвищують продуктивність та доступність ресурсу.

Розробка веб-сайтів - це процес створення та впровадження інтернет-ресурсу, який може бути доступний через веб-браузер. Цей процес включає в себе розробку дизайну, структури, функціональності та контенту веб-сайту.

Розробка веб-сайтів зазвичай включає такі етапи:

- ***Збір вимог:*** Розробник збирає вимоги до веб-сайту від клієнта або визначає їх самостійно. Це включає визначення мети веб-сайту, цільової аудиторії, функціональних вимог та дизайну.
- ***Планування:*** Розробник складає план роботи, визначає структуру веб-сайту, створює макети і робочі процеси.
- ***Дизайн:*** Розробник створює веб-дизайн, включаючи графічні елементи, кольори, шрифти та інші візуальні аспекти веб-сайту. У цьому етапі також можуть використовуватись інструменти для створення прототипів, щоб продемонструвати функціональність та структуру веб-сайту.
- ***Розробка:*** На цьому етапі розробник виконує програмування та розробку функціональних частин веб-сайту. Використовуються різні мови програмування та технології, такі як HTML, CSS, Java, PHP, бази даних тощо. Розробка включає створення інтерактивності, форм, збереження та отримання даних, роботу з базами даних та інші функціональні можливості.
- ***Тестування:*** Веб-сайт проходить процес тестування, щоб переконатися в його правильному функціонуванні та сумісності з різними браузерами та пристроями. Виявлені помилки та проблеми виправляють.

З введенням комерціалізації Інтернету, галузь веб-розробки швидко набула значного росту. Багато компаній виявили бажання використовувати веб-сайти для реклами та продажу товарів і послуг споживачам.

У сучасній веб-розробці використовуються різноманітні інструменти з відкритим вихідним кодом, такі як BerkeleyDB, GlassFish та стек LAMP (Linux, Apache, MySQL, PHP), а також Perl/Plack. Використання таких інструментів дозволяє знизити витрати на навчання веб-розробки. Значний розквіт галузі також сприяє появі простих у використанні програм для веб-розробки WYSIWYG, наприклад Adobe Dreamweaver, BlueGriffon та Microsoft Visual Studio. Ці програми дозволяють швидко створювати веб-сайти навіть без глибоких знань мови розмітки HyperText (HTML) або програмування.

Безпека є надзвичайно важливим аспектом веб-розробки, якому приділяють особливу увагу розробники. Вони зосереджуються на перевірці введених даних через форми, фільтрації вихідних даних та використанні шифрування. Для запобігання автоматичному заповненню форм та надходженню спаму використовуються різні заходи захисту, включаючи captcha.

Окрім цього, важливим аспектом є захист веб-серверів від потенційних вторгнень та безпечна передача інформації по мережі Інтернет. Використання сертифікатів TLS (або "сертифікатів SSL") та різних методів шифрування є невід'ємною частиною заходів, що допомагають запобігти шахрайству та зберегти конфіденційну інформацію в безпеці. Крім того, постійне оновлення програмного забезпечення є важливим елементом для усунення виявлених проблем безпеки та захисту веб-додатків від нових загроз.

1.5. Висновок до розділу 1

Отже, в першому розділі було детально розглянуто види програмного забезпечення, основні принципи веб-розробки, глобальне поняття про веб-додаток та інформаційну систему. Вище перераховані теми є основою для складання плану розробки веб-додатків та їх реалізації. Уже існує досить велика кількість веб-додатків, які покращують та полегшують життя у багатьох галузях. Безперечно можна сказати, що допоміжні веб-додатки є актуальними зараз та будуть актуальними у майбутньому, адже потреба у вдосконаленні навичок та досягненні нових успіхів є пріоритетом для кожної цілеспрямованої людини. Окрім цього, додатки для сфери дозвілля з легкістю допоможуть знайти людей, які мають аналогічні інтереси, спілкуватися з ними, що покращить мотивацію та бажання здобувати максимальні висоти та задоволення від вашого хобі.

На просторах інтернету є велика кількість веб-додатків, які певним чином допомагають людям розвиватись у їхніх дозвіллях. Щоденний ріст завантажень та пошукових запитів свідчить про актуальність та високу потребу у якісних та зручних допоміжних додатках.

Проаналізувавши предметну область, було обрано для використання мову програмування Java та фреймворк Spring. Для роботи з базою даних буде використано СКБД MySQL та модуль JSP.

РОЗДІЛ 2
ПРОЕКТУВАННЯ СИСТЕМИ

2.1. Вибір архітектури

Model-View-Controller (MVC) є визнаним стандартом у розробці програмного забезпечення [1], що забезпечує ефективну організацію та розподіл обов'язків між компонентами системи. Шаблон вирішує складні завдання, пов’язані із взаємодіючим інтерфейсом користувача з логічною програмою та забезпечує підтримку модульності, багаторазового використання та легкість провідності.[2]

Модель виступає як центральний елемент в архітектурі MVC, відповідаючи за логічне представлення даних і бізнес-логіки програми. Вона забезпечує незалежність відображення та управління, інкапсулюючи логіку та структуру даних у своєму внутрішньому представленні. Модель гарантує цільність даних і забезпечує доступ до них для інших компонентів системи.

Model-View-Controller (MVC) був розроблений для створення захоплюючих та інтуїтивно зрозумілих графічних інтерфейсів користувача на настільних платформах. Однак його естетика та ефективність знайшли своє застосування і в розробці сучасних веб-додатків [4].

Модель

Виступає як центральний елемент в архітектурі MVC, відповідаючи за логічне представлення даних і бізнес-логіки програми. Вона забезпечує незалежність відображення та управління, інкапсулюючи логіку та структуру даних у своєму внутрішньому представленні. Модель гарантує цільність даних і забезпечує доступ до них для інших компонентів системи.

Огляд

Він забезпечує репрезентацію інформації в різних форматах і виглядах. Цей компонент відкриває безмежні можливості для візуалізації даних, надаючи користувачам різноманітні способи сприйняття та розуміння інформації.

Огляд може включати такі елементи, як діаграми, графіки, таблиці та інші форми представлення даних. Наприклад, графіки можуть бути корисними для керівництва, оскільки вони дають візуальні виявлення про тенденції та показники, які можуть бути важливими для прийняття рішень.

Огляд в рамках MVC надає можливість перегляду інформації в різних видах, що дозволяє кожному користувачеві зберігати найбільш зручний спосіб сприйняття даних. Це допоможе забезпечити ефективне використання інформації та прийняття обґрунтованих рішень.

Контролер

Контролер активно сприймає вхідні дані від користувача та регулює їх зміни, виконуючи відповідні дії. Він виконує програму логіки розуміння та забезпечує зручну спільну взаємодію між моделлю, що містить дані та бізнес-логіку, і поданням, яке відповідає візуалізації та взаємодії з користувачем.

Головна мета контролера здійснюється в забезпеченні ефективного потоку даних та подій між моделлю та поданням. Він координує взаємодію між цими двома компонентами, передаючи відповідні дані та реагуючи на зміни стану. Таким чином, контролер повинен забезпечити гладку та злагоджену роботу програми, що дозволяє їй відповідати на потреби користувачів і виконувати заплановані завдання.

Контролер виконує важливу роль у системі MVC, дозволяючи моделі та подачі працювати разом, забезпечуючи оптимальне використання ресурсів та розподіл функцій між компонентами. Його присутність сприяє підвищенню ефективності та легкості розробки програмних додатків, а також забезпечує зручну взаємодію з користувачем.

Використання у веб-додатках

На початку розвитку Інтернет-архітектури MVC була реалізована на стороні сервера, але з часом більша частина логіки була передана клієнтській стороні за

допомогою загальних даних та технологій, таких як XMLHttpRequest. Це дозволяє частково оновлювати сторінки та підвищує швидкість завантаження додатків.

Використання шаблону Model-View-Controller (MVC) у веб-додатках дає безліч переваг. Воно дозволяє зберегти окремі компоненти (модель, подання, контролер) незалежними один від одного, спрощує розробку та розширення додатка, полегшує тестування та підтримку. Крім того, MVC дозволяє чітко розділити обов'язки та логічну структуру додатків, що сприяє покращенню його якості та обслуговування.

У сучасних веб-фреймворках, таких як Ruby on Rails, Django та Laravel, використання MVC є стандартним підходом до розробки веб-додатків. Це дозволяє розробникам швидко та ефективно будувати потужні та масштабовані добавки, забезпечуючи чітку організацію та структуру коду.

Загалом, використання шаблону MVC у веб-додатках є потужним і ефективним підходом, який разом із розробниками створює високоякісні та легкі у підтримці додатків.

2.2. Аналіз варіантів діяльності

В першу чергу необхідно спроектувати майбутній функціонал системи.

Системні вимоги можна ефективно охопити шляхом моделювання поведінки системи за допомогою діаграм варіантів використання в UML.

Щоб окреслити функції та межі системи, діаграми варіантів використання пропонують огляд верхнього рівня. Вони також окреслюють інтерфейс між системою та її акторами. Варіанти використання та актори на діаграмах варіантів використання деталізують дії системи та те, як актори її використовують, але не внутрішню роботу системи.

Діаграми варіантів використання служать для ілюстрації та визначення контекстуальних і основних вимог усієї системи або її критичних компонентів. Ці діаграми можна використовувати для моделювання складної системи в цілому або створення кількох діаграм для представлення різних частин системи. Як правило,

діаграми варіантів використання розробляються на ранніх етапах проекту і служать орієнтиром протягом усього процесу розробки.

Зокрема, діаграми варіантів використання корисні при:

- ***Визначенні функцій системи:*** Вони дозволяють ідентифікувати та описати основні функції та можливості системи з точки зору користувачів.
- ***Аналізі вимог:*** Вони допомагають з'ясувати, які вимоги мають бути враховані при проектуванні системи, шляхом ідентифікації різних сценаріїв взаємодії та варіантів використання.
- ***Виявленні потенційних проблем:*** Діаграми варіантів використання дозволяють виявити можливі проблеми або неоднозначності у вимогах та взаємодії з системою ще на ранніх етапах розробки.
- ***Плануванні тестування:*** Вони можуть служити основою для розробки тестових сценаріїв та планування тестування системи з урахуванням різних варіантів використання.
- ***Взаємодії зі зацікавленими сторонами:*** Діаграми варіантів використання допомагають встановити спільне розуміння між розробниками, замовниками та іншими зацікавленими сторонами щодо функціональності та вимог до системи.

У наступних розділах будуть розглянуті елементи моделі в діаграмах варіантів використання:

Варіанти використання

Варіанти використання описують конкретні сценарії або функції, які можуть бути виконані акторами у системі. Вони відображають послідовність дій та взаємодію з системою для досягнення певної мети. Кожен варіант використання описує певну функціональність та поведінку системи у конкретному контексті.

Актори

Актори представляють зовнішніх агентів або ролі, які взаємодіють з системою. Вони можуть бути користувачами, іншими системами, зовнішніми

сервісами або навіть автоматизованими процесами. Актори визначають, хто із зовнішніх сутностей має вплив на систему та які функції вони виконують.

Підсистеми

Підсистеми в моделях UML використовуються для організації незалежних поведінкових одиниць в системі. Вони допомагають у групуванні пов'язаних компонентів та функціональностей, спрощуючи розробку та управління великими системами. Підсистеми використовуються в діаграмах класів, компонентів і варіантів використання для створення структурованих моделей і підвищення рівня абстракції. Вони дозволяють ізолювати та організовувати компоненти в логічні групи, спрощуючи розуміння системи.

Зв'язки між варіантами використання

Зв'язки між варіантами використання показують залежності та взаємозв'язки між різними сценаріями. Вони допомагають уявити, які варіанти використання можуть бути викликані іншими або які можуть мати спільні кроки. Зв'язки вказують на можливі альтернативи, розширення або включення між варіантами використання.

UML діаграма варіантів використання зображена на рисунку 2.1.

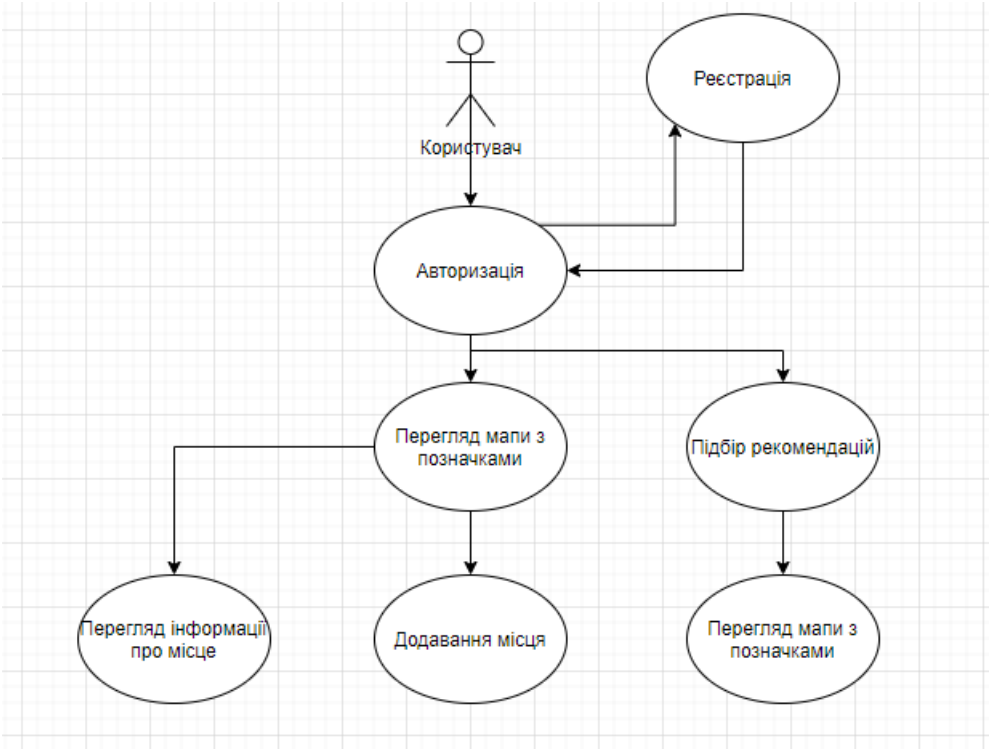

Рис.2.1. Діаграма варіантів використання

Діаграма показує усі можливі дії користувача в системі на поточний момент.

2.3. Проектування внутрішньої будови

Для проектування внутрішньої будови системи було розроблено UML діаграму класів, яка ілюструє внутрішню структуру проекту.

В межах стандарту UML (Unified Modeling Language), діаграми класів виступають одним із шести типів структурних діаграм. Вони є ключовим інструментом для моделювання об'єктів і статичної структури системи. Залежно від складності самої системи, можна використовувати одну діаграму класів для моделювання всієї системи або використовувати кілька діаграм для детального моделювання окремих компонентів.

Діаграми класів забезпечують візуалізацію системи або підсистеми, де можна описати об'єкти, встановити взаємозв'язки між ними та описати їх функціональні можливості та сервіси. Вони мають велике значення на різних етапах проектування системи. На етапі аналізу, діаграма класів допомагає розібратися у вимогах проблемної області та ідентифікувати її компоненти. Під час проектування об'єктно-орієнтованого програмного забезпечення, діаграми класів стають основою для створення фактичних класів та об'єктів у програмному коді. Пізніше, діаграми класів допомагають уточнити аналіз та концептуальні моделі, вони показують конкретні компоненти системи, інтерфейси користувача та логічні реалізації. Діаграми класів надають детальний опис того, як система функціонує, взаємозв'язки між компонентами на різних рівнях та плани реалізації.

Такі діаграми класів можна використовувати для візуалізації, визначення та документування структурних аспектів моделей. Наприклад, під час аналізу та проектування, діаграми класів допомажуть вам виконувати такі функції:

- зберіть і визначте структуру класів та інших класифікаторів;
- визначте зв’язки між класами та класифікаторами;
- проілюструйте структуру моделі за допомогою атрибутів, операцій і сигналів;

– дайте огляд загальних ролей та відповідальностей класифікатора, які визначають поведінку системи;

– показати класи реалізації в пакеті;

– покажіть структуру та поведінку одного чи кількох класів;

– показати ієрархію успадкування серед класів і класифікаторів;

– покажіть працівників і сутності як моделі бізнес-об’єктів.

На етапі впровадження циклу розробки програмного забезпечення ви можете використовувати діаграми класів для перетворення ваших моделей у код і для перетворення вашого коду в моделі.

У наступних темах описуються елементи моделі в діаграмах класів:

Заняття

У мові UML, класи є абстрактними представниками об'єктів або груп об'єктів з спільною структурою та поведінкою. Вони визначають атрибути та методи, які характеризують об'єкти даного класу.Класи або екземпляри класів є звичайними елементами моделі в діаграмах UML.

Об'єкти

У мові UML об'єкти виконують роль елементів моделі, які символізують конкретні існуючі екземпляри класів або набори класів. Ви можете додавати об’єкти до своєї моделі для представлення конкретних і прототипних екземплярів. Конкретний екземпляр представляє реальну особу чи річ у реальному світі. Наприклад, конкретний екземпляр класу Customer представляє реального клієнта. Прототипний екземпляр класу Customer містить дані, які представляють типового клієнта.

Пакети

Пакети групують пов’язані елементи моделі всіх типів, включаючи інші пакунки.

Сигнали

У моделях UML сигнали є елементами моделі, які не залежать від класифікаторів, які їх обробляють. Сигнали визначають односторонній асинхронний зв’язок між активними об’єктами.

Перерахування

У моделях UML перерахування — це елементи моделі в діаграмах класів, які представляють визначені користувачем типи даних. Переліки містять набори іменованих ідентифікаторів, які представляють значення переліку. Ці значення називаються літералами перерахування.

Типи даних

У діаграмах UML типи даних — це елементи моделі, які визначають значення даних. Типи даних зазвичай використовуються для представлення примітивних типів, таких як цілі чи рядкові типи, і перерахувань, таких як типи даних, визначені користувачем.

Артефакти

У UML-моделях артефакти — це елементи моделі, які представляють фізичні сутності в програмній системі. Артефакти являють собою фізичні одиниці реалізації, такі як виконувані файли, бібліотеки, програмні компоненти, документи та бази даних.

Зв'язки в діаграмах класів

В діаграмах класів UML, зв'язки відіграють важливу роль у визначенні взаємозв'язків між елементами моделі. Зв'язок - це концептуальна залежність між класами, яка вказує на існування асоціацій, агрегацій, композицій, спадкувань та інших типів залежностей між об'єктами. Відношення UML додають семантику та деталізують структуру та поведінку системи, виражаючи зв'язки між класами та їх ролі у взаємодії. Ці зв'язки допомагають зрозуміти спосіб взаємодії між класами, передати потрібну інформацію та забезпечити належну функціональність системи. Використання зв'язків у діаграмах класів дозволяє розробникам чітко моделювати взаємозв'язки та відношення між класами, що сприяє розумінню системи та успішній реалізації проектів програмного забезпечення.

Бінарні асоціації

В UML кваліфікатори є властивостями бінарних асоціацій і є необов’язковою частиною кінців асоціації. Кваліфікатор містить список атрибутів асоціації, кожен з

яких має назву та тип. Атрибути асоціації моделюють ключі, які використовуються для індексування підмножини екземплярів зв’язку.

Для системи розроблено діаграму класів (рис.2.2.)

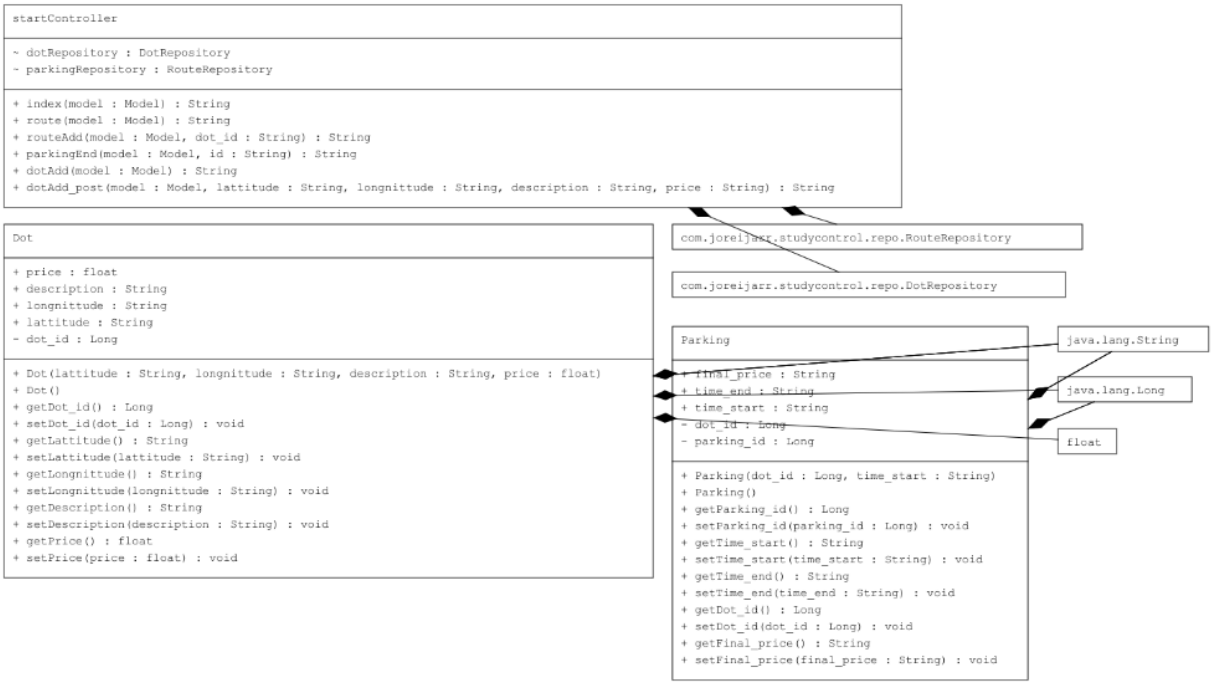

Рис.2.2. Діаграма класів системи

Система містить в собі 1 клас-контролер, 2 класи-сутності, 2 класи-репозиторії, які відповідають за взаємодію між класами-сутностями і БД.

2.4. Висновок до розділу 2

У другому розділі було розглянуто шаблон проектування програмного забезпечення Model-View-Controller.

Було спроектовано майбутній функціонал системи за допомогою діаграм варіантів використання в UML.

Для проектування внутрішньої будови системи було розроблено UML діаграму класів, яка ілюструє внутрішню структуру проекту. Для цього було визначено структуру та зв’язки класів та інших класифікаторів, проілюстровано

структуру моделі за допомогою атрибутів, операцій і сигналів, показано загальні ролі та обов’язки класифікатора, які визначають поведінку системи, класи реалізації в пакеті, структуру класів, ієрархію успадкування серед класів і класифікаторів.

РОЗДІЛ 3
ПРОЕКТУВАННЯ ТА РОЗРОБКА СИСТЕМИ

3.1. Вибір інструментальних засобів розробки

3.1.1. Огляд мови програмування

Мова програмування Java є заснованою на об'єктно-орієнтованому підході та володіє високим рівнем абстракції. Її створення було спрямоване на зниження залежності від конкретної реалізації та розробку загальнопризначених програм. Одним з незаперечних переваг Java є можливість написання програм один раз і запуску їх на будь-якій платформі, що підтримує Java, без необхідності в перекомпіляції.

У процесі компіляції, програми Java перетворюються на байт-код, який може бути виконаний на будь-якій віртуальній машині Java (JVM), незалежно від архітектури комп'ютера. Синтаксис Java подібний до мов програмування C і C++, проте вона надає менше низькорівневих можливостей. Однак, однією з основних переваг Java є можливість динамічного виконання, такого як відображення та модифікація коду під час виконання, що рідко зустрічається у інших компільованих мовах.

Мова програмування Java стала надзвичайно популярною, особливо у сфері веб-розробки клієнт-серверних додатків. За даними, наданими GitHub у 2019 році, близько 9 мільйонів розробників активно використовують Java. Історія Java починається у 1995 році, коли Джеймс Гослінг та його команда розробили перші версії мови в компанії Sun Microsystems.

Початково компілятори Java, віртуальні машини та бібліотеки класів були поширювані під ліцензіями, наданими саме Sun Microsystems.

У травні 2007 року було проведено переліцензування більшості Java-технологій згідно з Java Community Process, і тепер вони опубліковані під ліцензією GPL-2.0 компанією Sun. Хоча Oracle розробляє свою власну віртуальну машину HotSpot Java, основною реалізацією є відкрита вихідна реалізація OpenJDK JVM, яка є безкоштовним програмним забезпеченням і використовується більшістю розробників, а також є стандартною віртуальною машиною для багатьох дистрибутивів Linux.

Станом на березень 2022 року, остання доступна версія Java - Java 18, тоді як Java 17, 11 і 8 є версіями з довгостроковою підтримкою (LTS). Oracle випустила безкоштовне оновлення Java 8 LTS у січні 2019 року для комерційного використання, після чого надалі надає загальнодоступні оновлення Java 8 для особистого використання безстроково. Інші постачальники також надають безкоштовні збірки OpenJDK 8 і 11 з оновленнями безпеки та іншими вдосконаленнями.

Oracle і різні джерела настійно рекомендують користувачам видалити застарілі та непідтримувані версії Java через можливі проблеми з безпекою, які можуть виникнути у старих версіях. Oracle радить переходити на підтримувані версії, зокрема на одну з версій LTS (8, 11, 17).

Java є мовою програмування, яка забезпечує переносимість програм завдяки своїй особливості компіляції коду в байт-код Java. Цей байт-код виконується на віртуальній машині Java (JVM), яка доступна на різних апаратних платформах з відповідною підтримкою JVM. Це означає, що програми Java можуть працювати на різних операційних системах без необхідності перекомпіляції.

Байт-код Java є універсальним проміжним форматом, який може бути компільований в машинний код в процесі виконання програми за допомогою компілятора Just-in-time або GraalVM. Це дозволяє покращити продуктивність програми, оптимізувати виконання коду та забезпечити ефективне використання ресурсів.

Раніше Java мала деяку репутацію як мова з високим споживанням пам'яті та низькою швидкодією. Однак завдяки вдосконаленням у віртуальних машинах Java, зокрема HotSpot, які використовуються для виконання коду, та додаванню нових функцій і пакетів, таких як java.util.concurrent, продуктивність Java значно покращилася. Кожна нова версія Java принесла з собою покращення у швидкодії та функціональності, що сприяє подальшому зростанню її продуктивності.

Java є мовою програмування, яка використовує автоматичне управління пам'яттю для полегшення життя програмістів. Замість ручного виділення та звільнення пам'яті, Java використовує механізм зборки сміття. Середовище виконання Java відповідає за визначення та видалення недоступних об'єктів, тим самим автоматично відновлюючи пам'ять.

Збірка сміття виявляє об'єкти, які більше не використовуються, і звільняє пам'ять, яку вони займають. Однак, витоки пам'яті можуть виникати, якщо програміст зберігає посилання на непотрібні об'єкти, зокрема в контейнерах, які продовжують бути використовуваними. Це може призвести до виникнення винятків нульового покажчика, коли намагається отримати доступ до неіснуючого об'єкта.

Основна ідея автоматичного управління пам'яттю в Java полягає в тому, що програмісти звільняються від потреби ручного керування пам'яттю. Це відрізняється від деяких інших мов програмування, де програміст самостійно відповідає за виділення та звільнення пам'яті. Використання автоматичного управління пам'яттю допомагає уникнути витоків пам'яті, але не вирішує проблему логічних витоків пам'яті, коли пам'ять все ще посилається, але не використовується.

Автоматична збірка сміття в Java відбувається у різні моменти, зазвичай, коли програма неактивна. Вона автоматично активується, коли доступна пам'ять у купі не вистачає для нових об'єктів, що може призвести до тимчасової зупинки програми. Одна з переваг Java полягає у відсутності арифметики вказівників, як у мовах C/C++, що дозволяє збірнику сміття переміщувати об'єкти і забезпечує безпеку типів.

Java має різні типи збирачів сміття, і за замовчуванням використовується збирач сміття "Garbage First" (G1GC) з Java 9. Але також є інші збирачі сміття, які

можна використовувати залежно від вимог програми. Більшість програм успішно працюють з G1GC. Раніше, у Java 8, використовувався паралельний збирач сміття.

Не зважаючи на те, що автоматичне управління пам'яттю вирішує проблему виділення та звільнення пам'яті для об'єктів, програміст все ще повинен належним чином керувати іншими ресурсами, такими як мережеві з'єднання, бази даних або файлові дескриптори, особливо у випадку виникнення винятків.

Jakarta Servlet (раніше відомий як Java Servlet) - це компонент програми на Java, який розширює функціональність сервера. Сервлети зазвичай використовуються для розробки веб-контейнерів, що дозволяють розміщувати веб-додатки на веб-серверах. Це веб-API, яке надає можливості сервлетів на стороні сервера, подібно до того, як PHP або ASP.NET забезпечують динамічний веб-контент.

Сервлет Jakarta - це клас Java, який обробляє або зберігає запити в рамках API сервлетів Jakarta EE [1]. Сервлети зазвичай використовуються для обробки запитів HTTP і дозволяють розробникам додавати динамічний вміст на веб-сервери за допомогою платформи Java. Вони можуть генерувати HTML, XML або JSON відповіді. Сервлети можуть також зберігати стан між запитами за допомогою файлів cookie або зіставлення URL-адрес.

API Jakarta Servlet було доповнено двома іншими технологіями Java для веб-сервісів: JAX-RS 2.0 (для RESTful веб-служб) та JAX-WS (для веб-служб SOAP).

Для використання сервлетів необхідно використовувати веб-контейнер (контейнер сервлетів), який взаємодіє з сервлетами. Веб-контейнер відповідає за керування життєвим циклом сервлетів, маршрутизацію запитів до відповідних сервлетів та забезпечення безпеки.

API сервлетів (javax.servlet) визначає очікувані взаємодії між веб-контейнером та сервлетом, забезпечуючи потужний механізм для створення динамічних веб-сторінок. Ці два елементи, сервлети і JavaServer Pages (JSP), спільно працюють, допомагаючи розробникам створювати веб-додатки зі зручними і ефективними інструментами.

Сервлети використовуються для обробки запитів і генерації відповідей на серверному боці. Вони можуть вбудовувати HTML-код безпосередньо у свій Java-код, дозволяючи детально настроювати відповідь, що повертається клієнту. З іншого боку, JSP дозволяють вбудовувати Java-код у HTML-сторінки, що полегшує створення динамічного вмісту.

За допомогою патерну "Модель 2" або Model-View-Controller (MVC), сервлети і JSP можуть використовуватись разом для створення потужних веб-додатків. У цій архітектурі, сервлети беруть на себе обробку запитів і взаємодію з базою даних, забезпечуючи високорівневу логіку. З іншого боку, JSP відповідають за красиве відображення даних на сторінці, дозволяючи розробникам легко управляти виглядом і макетом веб-додатку.

Актуальна версія Servlet 5.0 містить ряд технологій, серед яких варто виділити Jakarta Server Pages (JSP), який дозволяє розробникам створювати динамічні веб-сторінки на основі різноманітних типів документів, таких як HTML, XML, SOAP та інші. JSP був розроблений компанією Sun Microsystems у 1999 році і є схожим до PHP і ASP, проте використовує мову програмування Java.

Для використання Jakarta Server Pages необхідний веб-сервер з контейнером сервлетів, наприклад Apache Tomcat або Jetty, які забезпечують розгортання та запуск JSP. Архітектурно JSP можна розглядати як високорівневу абстракцію сервлетів Java. Під час виконання JSP перетворюється в сервлет, що дозволяє кешувати його і повторно використовувати, поки вихідний JSP не зміниться.

JSP можна використовувати як самостійно, так і в поєднанні з іншими компонентами, такими як JavaBeans і сервлети Java, у структурі Model-View-Controller (MVC) або як компонент відображення в архітектурі модель-вигляд-контролер. За допомогою JSP розробники можуть змішувати Java-код з передвизначеними діями та статичним вмістом веб-розмітки, таким як HTML. Сторінки JSP компілюються і виконуються на сервері, а результат доставляється клієнту. Вони містять байт-код Java, який виконується на віртуальній машині Java (JVM), що забезпечує абстрактне середовище, незалежне від платформи.

Хоча JSP зазвичай використовуються для генерації документів HTML і XML, вони також можуть доставляти інші типи даних за допомогою OutputStream. Веб-контейнер автоматично створює неявні об'єкти JSP, такі як запит, відповідь, сесія, програма, конфігурація, сторінка, pageContext, вихід і виняток. Компілятор JSP аналізує JSP-сторінки і перетворює їх у виконувані сервлети Java. Зазвичай компілятор вбудовується в сервер додатків і автоматично запускається при першому доступі до JSP. Крім того, сторінки можуть бути попередньо скомпільовані для поліпшення продуктивності або включені до процесу збірки для перевірки наявності помилок. Деякі контейнери JSP надають можливість налаштувати періодичність перевірки змін у JSP-файлі для оновлення сторінки.

3.1.2. Огляд середовища розробки

IntelliJ IDEA представляє собою програмне середовище розробки (IDE), що написане на мові Java для створення комп'ютерного програмного забезпечення. Це продукт, розроблений компанією JetBrains (раніше відомий як IntelliJ) і доступний як безкоштовна Apache 2-ліцензована спільнота версія, а також приватна комерційна версія. Обидва варіанти можуть використовуватися для комерційного розвитку.

Історично, в січні 2001 року вийшов IntelliJ IDEA - одна з перших інтегрованих середовищ розробки (IDE) для Java, яка мала в собі революційні функції розширеної навігації та рефакторингу коду. У 2010 році журнал InfoWorld визнав IntelliJ IDEA найкращим з чотирьох найпопулярніших інструментів для розробки на мові Java.

У грудні 2014 року компанія Google оголосила про випуск версії 1.0 Android Studio - відкритої IDE для розробки програм під платформу Android, яка базується на безкоштовній версії IntelliJ IDEA - Community Edition. Крім цього, компанія JetBrains, яка стоїть за розробкою IntelliJ IDEA, також представляє інші популярні IDE, засновані на цій платформі. Серед них AppCode, CLion, DataGrip, GoLand, PhpStorm, PyCharm, Rider, RubyMine, WebStorm і MPS.

Ці IDE мають різноманітні функціональні можливості, що спрощують процес розробки. Вони надають автодоповнення на основі контексту, зручну навігацію по коду, інструменти для рефакторингу, засоби налагодження коду, лінтинг і автоматичне виправлення помилок. Крім цього, ці IDE мають вбудовані інструменти та підтримують інтеграцію з зовнішніми системами контролю версій і базами даних.

Інтегровані середовища розробки IntelliJ також підтримують широку екосистему плагінів, які дозволяють додавати додаткові функціональні можливості. Плагіни можна знайти та встановити з веб-сайту сховища плагінів IntelliJ, або використовувати вбудовану функцію для пошуку і встановлення плагінів. Кожна версія IntelliJ (Community та Ultimate) має своє власне сховище плагінів, і на даний момент їх кількість перевищує 3000 для кожної версії.

3.1.3. Огляд додаткового інструментарію

Мова розмітки гіпертексту (HTML) є незамінним стандартом для створення документів, які відображаються у веб-браузерах. Ця мова надає можливість використовувати каскадні таблиці стилів (CSS) та скриптові мови, такі як JavaScript, для покращення вигляду та функціональності веб-сторінок.

Веб-браузери відправляють запити на отримання HTML-документів до веб-серверів або локальних сховищ і перетворюють ці документи в мультимедійні веб-сторінки, які ми бачимо на екрані. HTML забезпечує семантичний опис структури сторінок і надає вказівки щодо їх зовнішнього вигляду.

Елементи HTML є невід'ємною складовою створення блоків на веб-сторінках. Вони надають можливість вставляти зображення та інші об'єкти, такі як інтерактивні форми, у створені сторінки. Крім того, HTML надає інструменти для створення структурованих документів, які описують структурну семантику тексту, таку як заголовки, абзаци, списки, посилання та інші елементи. Елементи HTML визначаються за допомогою тегів, які знаходяться між кутовими дужками. Деякі теги, наприклад *<img />* і *<input />*, вставляють вміст безпосередньо на сторінку, тоді як інші теги, наприклад *<p>*, оточують текстовий контент і можуть містити

інші теги як дочірні елементи. Браузери використовують HTML-теги для інтерпретації вмісту сторінок, приховуючи їх від користувача.

HTML - це мова розмітки, яка включає в себе різноманітні можливості для створення веб-сторінок. Один з цих можливостей - використання скриптів, таких як JavaScript, які дозволяють змінювати поведінку та вміст сторінок. Крім того, для визначення зовнішнього вигляду та макету веб-сторінок застосовується CSS. Значимим кроком у розвитку HTML та CSS було уведення стандартів Консорціумом World Wide Web Consortium. Вони рекомендують використовувати CSS для визначення стилів замість безпосереднього опису в HTML. З'явлення HTML5 було ще одним проривом, оскільки воно дозволило відтворювати відео та аудіо на сторінках за допомогою нового елементу <canvas> в поєднанні з JavaScript. Таким чином, HTML продовжує розвиватися і надавати все більше можливостей для створення багатофункціональних веб-сторінок.

Мова HTML має різноманітні компоненти, що включають теги з атрибутами, символьні типи даних, посилання на символи та посилання на сутності. Теги HTML зазвичай використовуються в парах, наприклад, *<h1>* і *</h1>*, де перший тег є початковим, а другий - закриваючим тегом. Проте, деякі теги є самозакриваючими, наприклад *<img>*. Це означає, що вони не мають кінцевого тегу. Використання правильної пари тегів дозволяє визначити структуру та ієрархію елементів на веб-сторінці, що є важливим для коректного відображення та інтерпретації вмісту.

Оголошення типу документа HTML встановлює стандартний режим відтворення для веб-сторінки. Нижче наведено приклад класичної програми "Привіт, світ!" на HTML:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Моя перша сторінка</title>
  </head>
  <body>
```

```
    <div>
      <h1>Привіт, світ!</h1>
      <p>Це моя перша веб-сторінка.</p>
    </div>
  </body>
</html>
```

Текст між тегами <*html*> і </*html*> описує веб-сторінку, а текст між тегами <*body*> і </*body*> є видимим вмістом сторінки. Тег <*title*> визначає назву сторінки, яка відображається на вкладках браузера та заголовках вікна. Тег <*div*> використовується для розділення сторінки на частини для полегшення дизайну.

В HTML5 для визначення типу документа використовується розмітка <*!DOCTYPE html*>. Цей тег дозволяє браузерам автоматично переключатися в режим "чудового" відтворення (standards mode), якщо включений режим сумісності (compatibility mode) не встановлений. В режимі "чудового" відтворення, браузери спираються на останні стандарти HTML5 і здатні правильно відображати та інтерпретувати вміст сторінки згідно зі специфікацією. Це забезпечує більшу сумісність між різними браузерами та впевненіше відтворення сторінок.

Документи HTML складаються зі вкладених елементів, які представлені тегами HTML, розташованими між кутовими дужками. Ці теги використовуються для створення різних елементів на веб-сторінках. Зазвичай, елементи використовуються у парах, наприклад, використовується початковий тег <*p*>, а потім вміст елемента розташовується між ним і відповідним кінцевим тегом </*p*>.

Теги можуть також містити додаткові атрибути, які надають додаткову інформацію про елемент, таку як його ідентифікатор, клас або посилання на зображення. Це дозволяє краще впорядковувати і стилізувати елементи на сторінці.

Деякі елементи, як, наприклад, <*br*>, є самозакриваючими тегами, що не мають закриваючого тегу і не допускають вмісту всередині. Інші елементи, наприклад <*p*>, можуть мати необов'язковий закриваючий тег, оскільки браузери можуть визначити кінець елемента на основі контексту.

На веб-сторінках використовуються також спеціальні теги для встановлення структурних маркерів, таких як заголовки *<h1>* до *<h6>*, які вказують рівень важливості тексту. Презентаційні теги, як *<b>* для жирного тексту або ** для курсиву, використовуються для визначення вигляду тексту, незалежно від його семантики.

HTML5 підтримує використання CSS для стилізації вмісту, дозволяючи більш гнучке керування візуальним представленням елементів.

Гіпертекстова розмітка в HTML дозволяє створювати посилання на інші документи. Елементи прив'язки, позначені тегом *<a>*, використовуються для створення гіперпосилань у документі. Атрибут href визначає цільову URL-адресу, до якої буде вести посилання. Наприклад, з використанням такого коду:

Вікіпедія,

можна створити гіперпосилання з текстом "Вікіпедія", яке веде до веб-сторінки Google.

Щоб зробити зображення гіперпосиланням, необхідно використати елемент *<a>* як обгортку навколо елементу *<img>*. Наприклад:

**

<img src="image.gif"

0alt="текст опису"

width="50"

height="50"

border="0">.

В цьому випадку, зображення "image.gif" стане гіперпосиланням, яке веде до веб-сторінки "https://example.org". Атрибути width і height визначають розміри зображення, а атрибут alt встановлює текст опису для зображення.

У HTML багато елементів мають атрибути, які складаються з пари ім'я-значення і записуються після імені елемента у вихідній розмітці. Ці атрибути можуть бути включені у лапки (одинарні або подвійні), але також допускається використання безлапкового варіанту (за винятком XHTML). Проте, використання безлапкових значень атрибутів вважається небезпечним. Деякі атрибути впливають

на елементи, лише своєю присутністю в розмітці елемента, без явного значення. Наприклад, атрибут ismap у елемента img.

HTML надає можливість використовувати різноманітні загальні атрибути, які є придатними для багатьох елементів:

- ***Атрибут id*** присвоює унікальний ідентифікатор елементу в межах всього документа. Це дозволяє змінювати його стилі за допомогою таблиць стилів, а також здійснювати зміни, анімацію або видалення його вмісту та презентації за допомогою скриптів. При додаванні ідентифікатора до URL-адреси сторінки він створює глобально унікальний ідентифікатор для елемента, зазвичай для певного розділу сторінки. Наприклад, ідентифікатор "Attributes" у *https://en.wikipedia.org/wiki/HTML#Attributes*.
- ***Атрибут class*** дозволяє класифікувати схожі елементи. Він може використовуватись для семантичних або презентаційних цілей. Наприклад, у HTML-документі можна використовувати клас "notation", щоб вказати, що всі елементи з цим класом є підпорядкованими до основного тексту документа. З презентаційної точки зору, такі елементи можуть бути зібрані разом і відображені як виноски на сторінці, замість того, щоб з'являтися там, де вони зустрічаються у вихідному HTML. Класи також використовуються для семантичного розмічення в мікроформатах. Можна вказувати кілька значень класів; наприклад, <*class="notation important"*> позначає, що елемент належить як до класу "notation", так і до класу "important".
- ***Атрибут style*** дозволяє визначити презентаційні властивості конкретного елемента. Рекомендується використовувати ідентифікатор елемента або класові атрибути для вибору елемента з таблиць стилів, хоча іноді використання цього атрибута може бути зручним для простих, конкретних або спеціальних стилів.
- ***Атрибут title*** використовується для додавання підказки до елемента. Більшість браузерів відображають цей атрибут як підказку, коли курсор наведений на елемент.

- ***Атрибут lang*** визначає мову, в якій викладений контент елемента і яка може відрізнятися від мови решти документа. Наприклад, у англомовному документі: * Ну що ж, c'est la vie, як кажуть у Франції. *.

Семантичний HTML є методом написання HTML, який акцентує на значущості закодованої інформації відносно її представлення. З самого свого виникнення HTML включав елементи семантичної розмітки, а також елементи презентації, такі як теги <font>, <i> і . Крім цього, існують семантично нейтральні теги span і div. Проте з початку 1990-х років, коли більшість браузерів почали підтримувати каскадні таблиці стилів, веб-авторам рекомендували уникати використання HTML для презентації та розділяти презентацію від вмісту.

Основна ідея семантичного HTML полягає в тому, щоб використовувати відповідні теги для правильного вираження значення контенту. Замість використання загальних або неправильних тегів, семантичний HTML надає можливість використовувати спеціалізовані теги, такі як <header>, <nav>, <article>, <section> і багато інших, що чітко вказують на семантику контенту.

Цей підхід забезпечує кращу доступність та розуміння веб-сторінок браузерами, пошуковими системами та іншими технологіями. Використання семантичного HTML сприяє поліпшенню SEO, структурованості вмісту та розробці більш доступних та універсальних веб-додатків.

Каскадні таблиці стилів (CSS) - це мова, призначена для спрощення процесу створення захопливих веб-сторінок. Вони дозволяють застосовувати стилі до веб-сторінок незалежно від HTML-коду, з якого складається кожна сторінка. CSS визначає зовнішній вигляд веб-сторінки, включаючи кольори, шрифти, відступи та інше. Використовуючи CSS, можна налаштувати веб-сайт так, щоб він виглядав саме так, як потрібно. Ця мова надає розробникам і дизайнерам повний контроль над відображенням елементів, включаючи їх розташування в браузері.

CSS використовується для створення стильових правил, які застосовуються до HTML-елементів. Це дозволяє відокремити структуру веб-сторінки від її

представлення, що сприяє більшій гнучкості та повторному використанню стилів. За допомогою CSS можна створювати складні макети, адаптивні дизайни, анімації та багато іншого, щоб зробити веб-сайт привабливим та зручним для користувачів.

Spring Framework був розроблений з великою кількістю цілей на увазі, забезпечуючи розробникам зручність і простоту у створенні корпоративних програм на платформі Java. Цей фреймворк дозволяє використовувати мови програмування Java, Groovy і Kotlin для реалізації рішень, орієнтованих на підприємництво, на платформі Java Virtual Machine (JVM). Окрім того, Spring Framework забезпечує гнучкість для створення різних архітектур, що відповідають унікальним вимогам кожної програми.

Spring Framework дозволяє використовувати інверсію управління (Inversion of Control, IoC) і внедрення залежностей (Dependency Injection, DI), що спрощує розробку програм і підвищує їхню модульність та тестируваність. Він також надає широкий набір функціональності, включаючи обробку транзакцій, кешування, обробку подій та безліч інших компонентів, що допомагають забезпечити ефективність, масштабованість та безпеку програм.

Наявність широкого спектру сценаріїв застосування робить Spring Framework надзвичайно популярним серед великих підприємств. Цей фреймворк дозволяє розробникам створювати програми, які без проблем працюватимуть на різних версіях JDK (Java Development Kit) і серверах, незалежно від їхнього циклу оновлення. Безліч сценаріїв використання Spring Framework включають в себе вбудовування програм в банківські системи або їхнє розгортання в хмарних середовищах, а також створення автономних програм, які не потребують постійного з'єднання з сервером.

Spring Framework відомий своєю відкритістю і має дуже активну спільноту розробників, що сприяє його постійному розвитку та вдосконаленню протягом тривалого періоду. Завдяки цій спільноті, розробники мають змогу обмінюватись досвідом, знаходити рішення для реальних проблем і отримувати зворотний зв'язок, що сприяє покращенню фреймворку. Ця активна спільнота забезпечує необхідну

підтримку та допомогу розробникам у використанні Spring у різних сценаріях розробки програмного забезпечення.

Spring Framework надає багато важливих функціональних можливостей, включаючи керування життєвим циклом об'єктів, інверсію управління (Inversion of Control, IoC) і внедрення залежностей (Dependency Injection, DI), що спрощують розробку програм і полегшують підтримку. Крім того, Spring Framework надає підтримку для обробки транзакцій, безпеки, кешування і багатьох інших аспектів, що допомагають розробникам створювати потужні та надійні програми.

Наслідки модулів JDK 9 («Jigsaw») також були враховані в Spring Framework. Фреймворк постачається з записами маніфесту "Automatic-Module-Name" для використання з програмами, які підтримують Jigsaw. Ці записи визначають стабільні назви модулів на рівні мови, незалежно від назви артефактів JAR. Це дозволяє Spring Framework працювати як на JDK 8, так і на 9+ без проблем.

Філософія дизайну Spring Framework зосереджена на простоті використання, гнучкості, модульності та відкритому коді. Ці основні принципи є невід'ємною частиною розробки фреймворку і допомагають розробникам створювати ефективні та масштабовані програми, що відповідають високим стандартам корпоративного середовища. Завдяки їм, Spring Framework став одним з провідних рішень у сфері розробки програмного забезпечення.

Унікальність Spring Framework базується на низці принципів, які його відрізняють від інших рішень. Основні з них включають:

- *Принцип вибору на кожному рівні:* Spring дозволяє відкладати прийняття важливих дизайнерських рішень до пізнішого етапу. Наприклад, ви можете легко змінити постачальників постійності через конфігурацію, не залучаючи код. Цей принцип поширюється на багато інших аспектів інфраструктури та інтеграції зі сторонніми API.
- *Урахування різноманітних точок зору:* Spring пропонує підтримку різних підходів і перспектив. Він підтримує широкий спектр потреб додатків і дозволяє розробникам вибирати підходи, які найкраще відповідають їхнім потребам.

– *Забезпечення сильної зворотної сумісності:* Spring прагне зберігати сильну зворотну сумісність між версіями. Він дбає про те, щоб критичні зміни були мінімізовані, що полегшує обслуговування програм і бібліотек, що залежать від Spring. Крім того, Spring підтримує різні версії JDK і сторонніх бібліотек, щоб задовольнити потреби різних проектів.

– *Розробка дизайну API:* Команда Spring надає особливу увагу створенню зрозумілих і стабільних API, які залишаються сумісними протягом багатьох версій і років. Вони вкладають значні зусилля у проектування API, що полегшує його використання розробниками.

– *Високі стандарти якості коду:* Spring Framework встановлює високі стандарти якості коду. Великий акцент приділяється написанню зрозумілого, актуального і точного javadoc. Важливо зазначити, що проект має чисту структуру коду без циклічних залежностей між пакетами, що сприяє легкості розуміння та підтримки коду.

3.2. Розробка графічного інтерфейсу користувача

GUI (графічний інтерфейс користувача) — це система інтерактивних візуальних компонентів для комп'ютерного програмного забезпечення. GUI відображає об'єкти, які передають інформацію та представляють дії, які може виконати користувач. Об'єкти змінюють колір, розмір або видимість, коли користувач взаємодіє з ними.

Графічний інтерфейс користувача був вперше розроблений у Xerox PARC Аланом Кеєм, Дугласом Енгельбартом та групою інших дослідників у 1981 році. Пізніше Apple представила комп'ютер Lisa з графічним інтерфейсом користувача 19 січня 1983 року.

Огляд GUI

Графічний інтерфейс включає об'єкти графічного інтерфейсу, як-от значки, курсори та кнопки. Ці графічні елементи іноді покращуються звуками або

візуальними ефектами, такими як прозорість і тіні. Використовуючи ці об’єкти, користувач може користуватися комп’ютером без знання команд.

Які існують елементи графічного інтерфейсу користувача?

Щоб зробити GUI максимально зручним для користувача, існують різні елементи та об’єкти, які користувач використовує для взаємодії з програмним забезпеченням. Нижче наведено список кожного з них із коротким описом.

– ***Кнопка*** – графічне представлення кнопки, яка при натисканні виконує дію в програмі

– ***Діалогове вікно*** – тип вікна, яке відображає додаткову інформацію та запитує користувача про введення.

– ***Піктограма*** – маленьке графічне зображення програми, функції або файлу.

– ***Меню*** - список команд або варіантів, які пропонуються користувачеві через панель меню.

– ***Панель меню*** – тонка горизонтальна панель, що містить написи меню.

– ***Стрічка*** — заміна меню файлів і панелі інструментів, які групують дії програм.

– ***Вкладка*** – область, яку можна натиснути у верхній частині вікна, яка відображає іншу сторінку чи область.

– ***Панель інструментів*** - ряд кнопок, часто у верхній частині вікна програми, який керує функціями програмного забезпечення.

– ***Вікно*** - прямокутна частина дисплея комп'ютера, яка показує програму, що використовується в даний момент.

Як працює графічний інтерфейс користувача (GUI)?

Інтерфейс користувача включає різноманітні графічні елементи, такі як вікна, піктограми, меню тощо, які дозволяють виконувати різні команди, наприклад, відкривати, видаляти та переміщувати файли. У операційних системах з графічним інтерфейсом навігацію в основному здійснюють за допомогою миші, проте також

існує можливість використання клавіатури за допомогою комбінацій клавіш або стрілок.

Графічне взаємодія з користувачем створює зручне та інтуїтивно зрозуміле середовище, де він може легко взаємодіяти зі своїми файлами та програмами. Візуальні елементи, такі як кнопки, панелі та поля введення, допомагають користувачам швидко зорієнтуватися та здійснювати потрібні дії.

Комбінація графічного інтерфейсу та різноманітних способів навігації надає користувачам багато можливостей для взаємодії зі своїм комп'ютером, забезпечуючи зручність і продуктивність в роботі.

Наприклад, для відкриття програми в операційній системі з графічним інтерфейсом користувача, достатньо навести вказівник миші на піктограму програми та двічі клацнути на ній. У випадку інтерфейсу командного рядка, необхідно знати команди для переходу до потрібного каталогу, виведення списку файлів та запуску файлу.

Які переваги надає графічний інтерфейс користувача (GUI)?

Графічний інтерфейс користувача вважається зручнішим, ніж текстовий інтерфейс командного рядка, такий як MS-DOS або Unix-подібні оболонки. Операційні системи з графічним інтерфейсом користувача легші для вивчення та використання, оскільки не потрібно запам'ятовувати команди. Крім того, користувачам не потрібно знати мови програмування. Завдяки простоті використання та сучасному зовнішньому вигляду, операційні системи з графічним інтерфейсом користувача стали домінуючими на сучасному ринку.

Як користувач взаємодіє з графічним інтерфейсом користувача (GUI)?

Для взаємодії з графічним інтерфейсом користувача використовується вказівний пристрій, такий як миша, яка дозволяє взаємодіяти зі всіма елементами GUI. Сучасні та мобільні пристрої також використовують сенсорні екрани для взаємодії з графічним інтерфейсом.

При початку роботи з системою користувач потрапляє на головну сторінку, яка містить мапу з відмітками місць риболовлі (рис. 3.1.), при натисканні на будь-який з маркерів з’являється інформаційне вікно (рис. 3.2.).

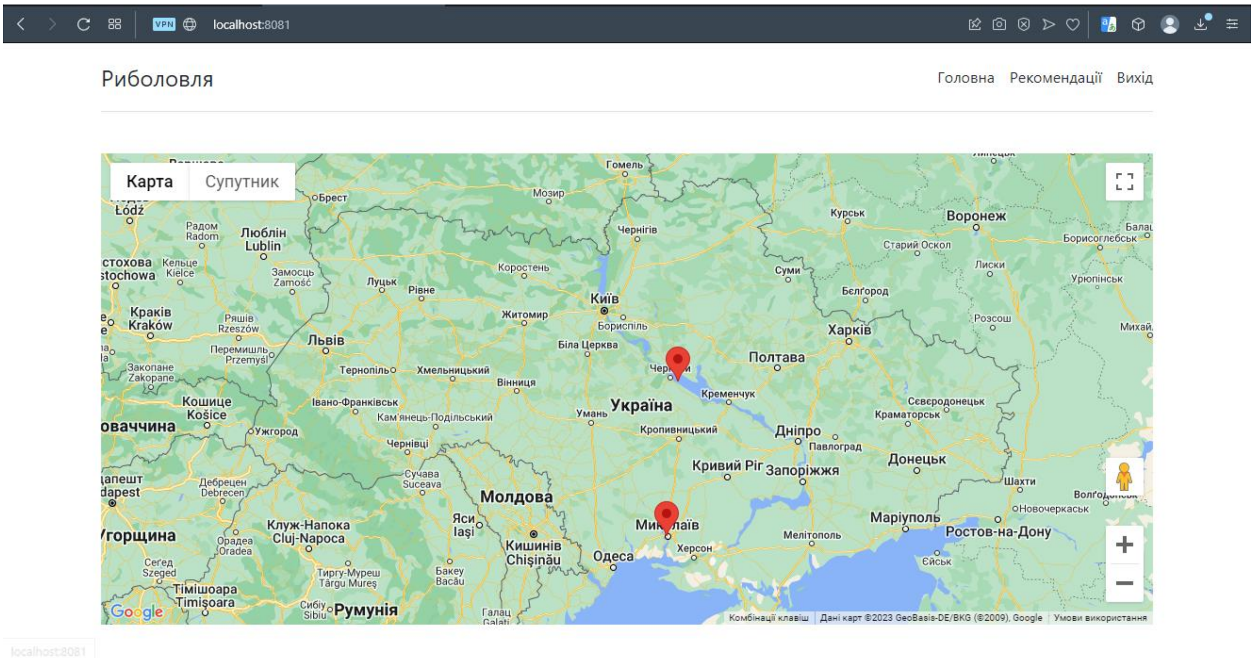

Рис.3.1. Головна сторінка сайту

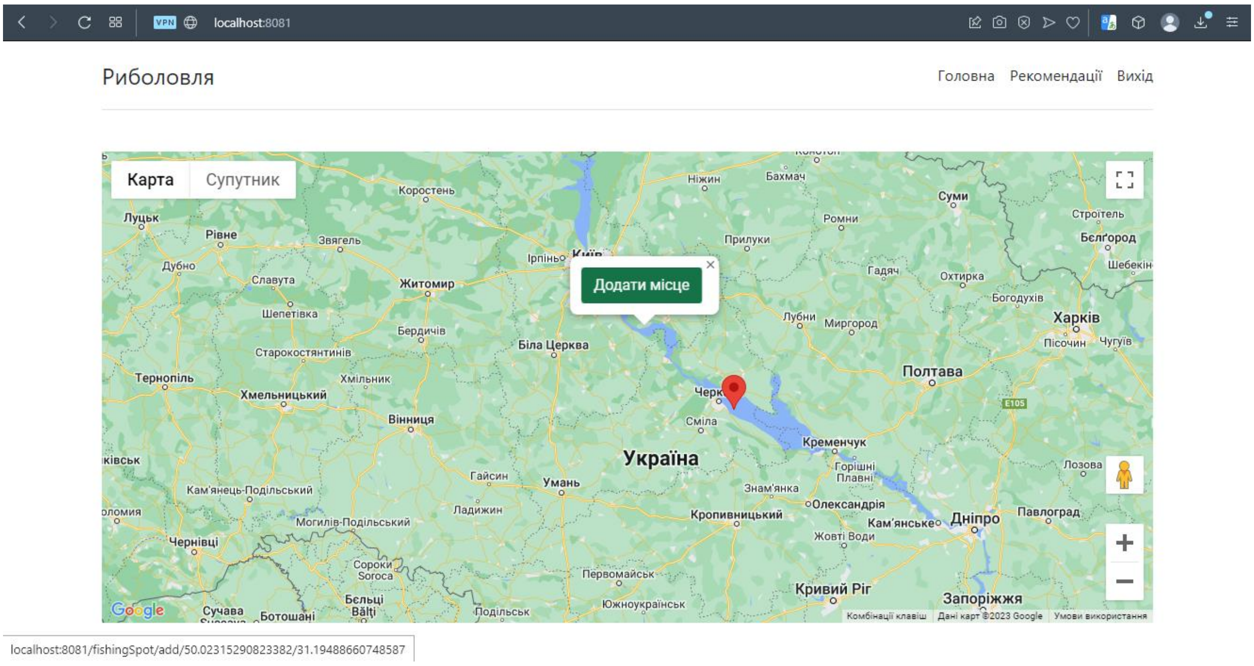

Рис.3.2. Інформаційне вікно

При натисненні на кнопку «Додати місце» відкривається форма додавання нового місця (рис. 3.3.).

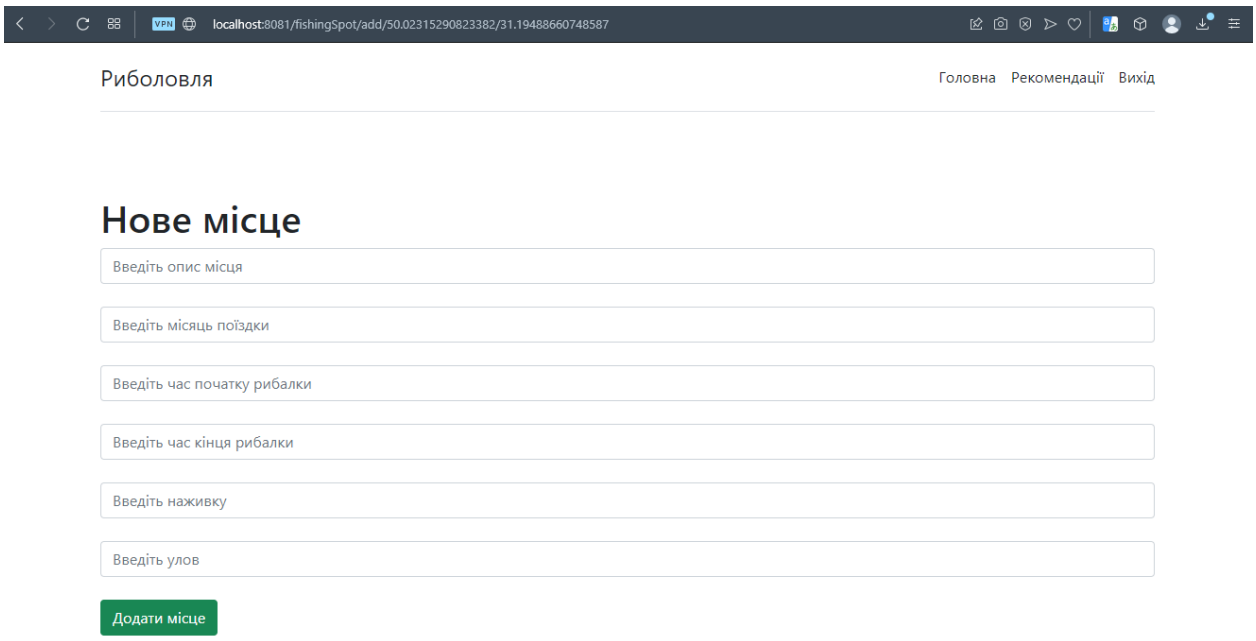

Рис.3.3. Сторінка нового місця

При натисканні кнопки «Рекомендації» з’являється форма, яка запитує користувача про час поїздки (рис. 3.4.).

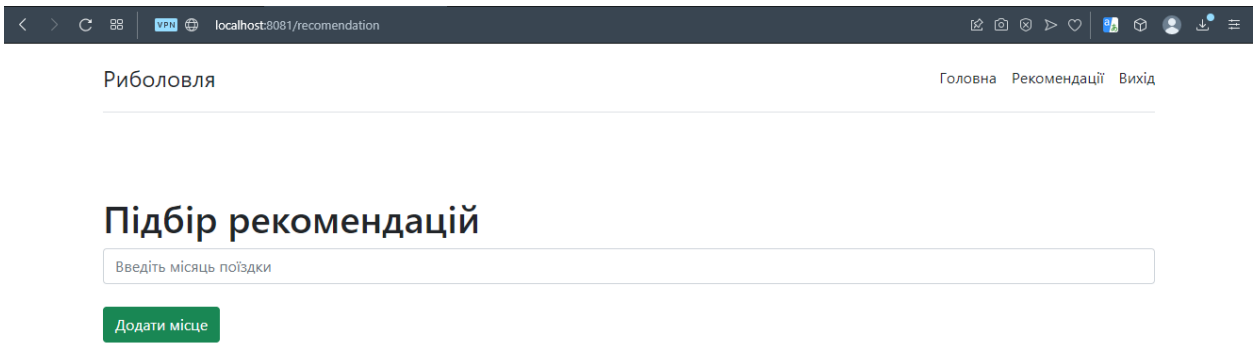

Рис.3.4. Сторінка рекомендацій

Після того, як місяць поїздки обрано, система відкриває мапу, на якій позначки відфільтровані таким чином, щоб залишилися лише ті, що підходять по місяцю (рис. 3.5.).

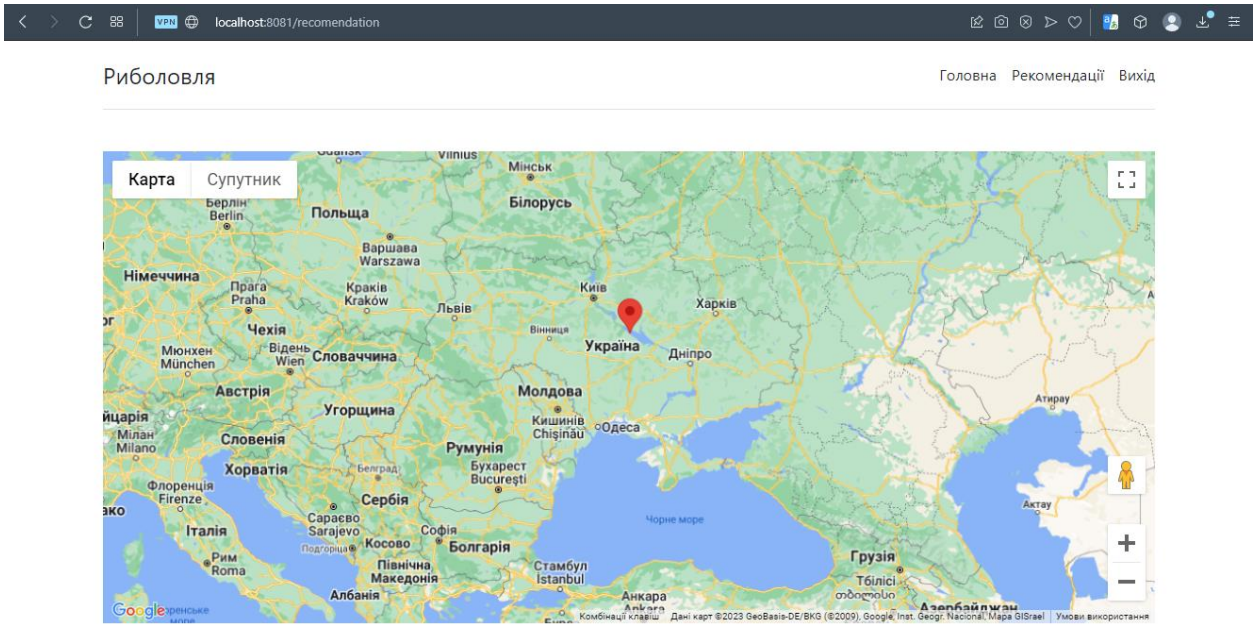

Рис.3.5. Сторінка рекомендацій (мапа)

3.3. Тестування системи

Тестування програмного забезпечення - це процес, який включає перевірку артефактів та поведінки програмного забезпечення з метою підтвердження його функціональності та якості. Цей процес дозволяє отримати незалежну та об'єктивну оцінку програмного забезпечення, щоб виявити можливі ризики та проблеми, пов'язані з його використанням. Існує безліч методів тестування, які можуть бути використані, але деякі з них включають:

- Аналіз вимог продукту, щоб забезпечити їх повноту, точність та відповідність. Це включає розгляд різних контекстів, таких як галузеві вимоги, бізнес-потреби, зручність використання, продуктивність, безпека та інфраструктурні вимоги.

- Перегляд архітектури та дизайну продукту для забезпечення його ефективності та якості. Це може включати огляд коду, аналіз структури та взаємодії компонентів системи.
- Співпраця з розробниками для покращення практик кодування, використання дизайн-шаблонів та написання тестів. Це допомагає забезпечити виявлення та усунення помилок на ранніх етапах розробки.
- Виконання програми або скрипту з метою перевірки його правильності та відповідності очікуваному результату. Це може включати тестування функціональності, реакції на помилки, продуктивності та безпеки.
- Перевірка інфраструктури розгортання та виконання скриптів автоматизації. Це включає перевірку наявності необхідних середовищ, конфігурацію серверів, налаштування мережі та інших компонентів інфраструктури.
- Застосування методів моніторингу та спостереження для участі у виробничих процесах. Це може включати аналіз логів, метрик продуктивності та використання ресурсів.

Тестування програмного забезпечення надає користувачам або спонсорам незалежну та об'єктивну інформацію щодо якості програмного забезпечення та ризиків, пов'язаних з його неполадками.

Помилки

Помилки та неполадки програмного забезпечення виникають внаслідок такого процесу: розробник допускає помилку (дефект, ваду), яка впливає на вихідний код програмного забезпечення. Якщо ця помилка залишається непоміченою, у певних ситуаціях система може видавати некоректні результати, що може привести до збоїв.

Не всі недоліки обов'язково викликають збої. Наприклад, помилки у безжиттєвому коді ніколи не викликають збоїв. Дефект, який не проявляє збоїв, може призвести до збою при зміні оточення. Наприклад, зміни у оточенні можуть

включати запуск програмного забезпечення на новій апаратній платформі комп'ютера, зміну вихідних даних або взаємодію з іншим програмним забезпеченням. Один недолік може призвести до широкого спектру відмов.

Тестування програмного забезпечення розкриває не тільки помилки в коді, але і прогалини у вимогах, що є одним із поширених джерел дефектів. Прогалини вимог виникають тоді, коли розробник не розпізнає або пропускає певні вимоги, що призводить до помилок у програмі. Ці прогалини вимог можуть стосуватися нефункціональних аспектів, таких як тестуваність, масштабованість, підтримка, продуктивність, безпека та інші. При виявленні таких прогалин необхідно вносити корективи в вимоги та відповідно адаптувати розробку програмного забезпечення для забезпечення його якості та відповідності вимогам.

Статичне, динамічне та пасивне тестування

У тестуванні програмного забезпечення існує широкий спектр підходів, включаючи статичне, динамічне та пасивне тестування. Статичне тестування вимагає проведення оглядів, аналізу чи перевірок без фактичного виконання програмного коду. Наприклад, це може включати перевірку структури вихідного коду або синтаксичний аналіз за допомогою спеціальних інструментів або компіляторів. Динамічне тестування, натомість, відбувається під час активного виконання програми з використанням заданого набору тестових випадків. Цей підхід дозволяє перевірити реальну поведінку програмного забезпечення та виявити помилки, що можуть виникнути під час його роботи. У свою чергу, пасивне тестування займається аналізом поведінки системи без активної взаємодії з програмним продуктом, використовуючи системні журнали та трасування. Кожен з цих підходів має свої переваги та застосування, і їх комбінація дозволяє досягти максимальної ефективності тестування програмного забезпечення.

Підхід, який передбачає дослідження

На додаток, можна зазначити про науковий підхід до тестування програмного забезпечення, який включає в себе дослідження, розробку тестів та їх виконання як єдине ціле. Цей підхід передбачає, що тестувальник самостійно вчиться, розробляє тести та виконує їх з метою постійного вдосконалення якості своєї роботи. Він

включає в себе аналіз результатів і навчання протягом усього проекту, що надає тестувальнику особисту свободу і відповідальність за процес тестування.

Підходи "білий ящик" і "чорний ящик"

У методології тестування програмного забезпечення широко використовуються два основних підходи: "білий ящик" і "чорний ящик". Підхід "білий ящик" передбачає, що тестувальник має повний доступ до внутрішньої структури та реалізації програми. Це дає йому можливість ефективно розробляти тести, враховуючи всі внутрішні деталі. З іншого боку, підхід "чорний ящик" базується на зовнішньому спостереженні поведінки програми без знання її внутрішньої структури. Тестувальник виконує тести, основуючись лише на вхідних та вихідних данних, та спостерігає за реакцією програми. Також існує гібридний підхід, відомий як "тестування сірого ящика", який поєднує елементи обох підходів.

Загалом, використання різних підходів до тестування програмного забезпечення дозволяє розширити охоплення тестування та забезпечити виявлення різноманітних дефектів у програмному забезпеченні.

Для тестування даного додатку обрано підхід "чорний ящик", що означає, що тестувальник не має доступу до програмного коду.

Процес тестування системи проілюстровано в таблиці 3.1.

Таблиця 3.1.

Тестування додатку

№	Тест-кейс	Очікуваний результат	Отриманий результат
1	Створення місця з пустими полями	При спробі створення місця з пустими полями система повідомляє користувача про те, що ці поля необхідно заповнити.	При спробі створення місця з пустими полями система повідомляє користувача про те, що ці поля необхідно заповнити.

Продовження табл. 3.1.

2	Спроба переглянути існуючи місця при відсутності місць	При спробі переглянути існуючі місця при відсутності місць система відображає пусту сторінку.	При спробі переглянути існуючі місця при відсутності місць система відображає пусту сторінку.
3	Спроба переглянути рекомендацій при відсутності місць	При спробі переглянути рекомендацій при відсутності місць система відображає пусту сторінку.	При спробі переглянути рекомендацій при відсутності місць система відображає пусту сторінку.

Результати тестування показують, що система повністю функціональна і готова до використання в реальних умовах.

3.4. Висновок до розділу 3

У третьому розділі було обрано та використано інструментальні засоби розробки такі як мова програмування Java, технології Jakarta Server Pages, компілятор сторінок JavaServer. Також оглянувши, проаналізувавши всі наявні переваги та недоліки програм, як середовище розробки було обрано IntelliJ IDEA.

Також було використано ряд додаткових інструментів, серед яких мова розмітки гіпертексту HTML, каскадні таблиці стилів CSS та фреймворк Spring. Ці інструменти виявилися незамінними в розробці та реалізації програмного продукту. Використання мови розмітки гіпертексту HTML дозволило створити структуровану та доступну веб-сторінку, яка забезпечує відображення інформації для користувача. Каскадні таблиці стилів CSS були використані для стилізації веб-сторінки, забезпечення її привабливого та сучасного вигляду. Фреймворк Spring, в свою чергу, став незамінним інструментом для розробки розширених функціональних можливостей програмного продукту. Spring надав можливість реалізації інверсії керування, контролю транзакцій, обробки помилок та інших важливих аспектів.

Було активно використано графічний інтерфейс користувача (GUI) для розробки програмного забезпечення. GUI виявився незамінним інструментом у спрощенні взаємодії користувача з системою, забезпечивши зручний та інтуїтивно зрозумілий спосіб взаємодії з функціональністю, реалізованою у розробленому програмному продукті.

У процесі виконання було приділено велику увагу тестуванню програмного забезпечення з метою перевірки його якості, надійності та функціональності. Результати тестування були детально проаналізовані, помилки виявлені та усунуті, а також зроблені необхідні виправлення.

ВИСНОВКИ

У ході виконання дипломної роботи було розроблено допоміжний веб-додаток, який має на меті покращити досвід рибалки та забезпечити користувачам зручну та інформативну платформу для планування та організації рибальських виїздів.

Веб-додаток надає рибалкам можливість отримувати актуальну та достовірну інформацію про риболовні місця, рекомендації щодо найбільш ефективних методів та знарядь лову. Користувачі можуть створювати свої профілі, обмінюватися досвідом та враженнями.

Загальний результат дипломної роботи демонструє ефективність та корисність допоміжного веб-додатка для рибальства. Його розробка відповідає потребам сучасних рибалок, які шукають зручний інструмент для покращення свого риболовного досвіду.

Використання сучасних технологій та фреймворків у розробці додатка дозволило забезпечити його швидку та стабільну роботу, а також зручний інтерфейс для користувачів. Додаток надає користувачам цінну інформацію та функціонал, що полегшують їх риболовний процес та допомагають досягати кращих результатів.

Завдяки чіткому виконанню завдань, поставлених на початку роботи, в результаті виконання роботи було отримано повноцінну систему, що здатна виконувати закладений в неї функціонал та готова до використання в реальних умовах.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. What is Application Software? (With Examples) [Електронний ресурс] – Режим доступу: https://www.simplilearn.com/tutorials/programming-tutorial/what-is-application-software (дата звернення 17.05.2023). – Назва з екрана.

2. Coward D. Java Servlet Specification, v2.2 / D. Coward, J. Duncan Davidson. – Palo Alto: Wyd-wo Sun Microsystems,1999. – 79s. – Bibliogr.: s. 42-46.

3. What Does AJAX Even Stand For? [Електронний ресурс] – Режим доступу: https://thehistoryoftheweb.com/what-does-ajax-even-stand-for/ (дата звернення 25.05.2023). – Назва з екарна.

4. Dhana S. College Time Table Management System using Android Application / S. Dhana, K. Hemanth. - Chittoor: Wyd-wo Sun IJSRCSEIT, 2018. – 5s. – Bibliogr.: s. 1–5.

5. Web application [Електронний ресурс] – Режим доступу: https://codedocs.org/what-is/web-application (дата звернення 26.05.2023). – Назва з екрана.

6. Top Web Frameworks to Learn and Tips to Build a Career in Web Development [Електронний ресурс] – Режим доступу: https://www.simplilearn.com/top-web-frameworks-and-career-tips-in-web-development-article (дата звернення 27.05.2023). - Назва з екрана.

7. Web Development [Електронний ресурс] – Режим доступу: https://www.techopedia.com/definition/23889/web-development#:~:text=Web%20development%20refers%20in%20general,hosting%20via%20intranet%20or%20internet (дата звернення 27.05.2023). - Назва з екрана.

8. Campbell J. Web Design / Jennifer Campbell. – Farnborough: Wyd-wo Cengage Learning, 2017. – 234s. – Bibliogr.: s. 75–84.

9. Websites [Електронний ресурс] – Режим доступу: http://confocal-manawatu.pbworks.com/w/page/113890672/Websites (дата звернення 31.05.2023). - Назва з екрана.

10. Java Learning Library [Електронний ресурс] – Режим доступу: https://apexapps.oracle.com/pls/apex/f?p=44785:141:10183554927715::::P141_PAGE_ID,P141_SECTION_ID:15,86 - (дата звернення 31.05.2023). - Назва з екрана.

11. Exceptions in Java [Електронний ресурс] – Режим доступу: https://www.artima.com/articles/exceptions-in-java (дата звернення 01.06.2023). - Назва з екрана.

12. HTML5 specification finalized, squabbling over specs continues [Електронний ресурс] – Режим доступу: https://arstechnica.com/information-technology/2014/10/html5-specification-finalized-squabbling-over-who-writes-the-specs-continues/ (дата звернення 03.06.2023). - Назва з екрана.