Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: «Розробка предметно-орієнтованого інформаційного додатку для використання на ОС Android».

Виконав: студент IV курсу, групи СП-43

спеціальності 121 – Інженерія програмного забезпечення
(шифр і назва спеціальності)

	(підпис)	Шевчук В.Ю. (прізвище та ініціали)
Керівник	(підпис)	Яворська Є.Б. (прізвище та ініціали)
Нормоконтроль	(підпис)	Стоянов Ю.М. (прізвище та ініціали)
Завідувач кафедри	(підпис)	Петрик М.Р. (прізвище та ініціали)
Рецензент	(підпис)	Матійчук Л.П. (прізвище та ініціали)

Тернопіль 2025

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра програмної інженерії
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Петрик М.Р.
(підпис) (прізвище та ініціали)
«_____» _____ 2025 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавр
(назва освітнього ступеня)

за спеціальністю 121 – Інженерія програмного забезпечення
(шифр і назва спеціальності)

студенту Шевчук Владислав Юрійович
(прізвище, ім'я, по батькові)

1. Тема роботи «Розробка предметно-орієнтованого інформаційного додатку для використання на ОС Android».

Керівник роботи Яворська Євгенія Богданівна, к.т.н., доц.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «18» квітня 2025 року № 4/7-335

2. Термін подання студентом завершеної роботи «___» червня 2025 р.

3. Вихідні дані до роботи: Завдання на розробку системи

4. Зміст роботи (перелік питань, які потрібно розробити)
Обрати напрям дослідження та об'єкт, сформулювати, погодити та затвердити тему кваліфікаційної роботи; розробити та затвердити завдання;
проаналізувати завдання, зробити огляд та проаналізувати бібліографічні матеріали щодо
стану справ та тенденій в обраному напрямку дослідження;
сформулювати завдання, обґрунтувати цілі дослідження;
розробити та спроєктувати;
описати технології, етапи розробки та сам продукт,
розробити план тестування програмного продукту; оформити допоміжну документацію;
проаналізувати роботу щодо питань з дотримання положень про безпеку життєдіяльності та
основи охорони праці; оформити пояснювальну записку; зробити відповідні висновки
за результатами виконаної роботи, представити роботу на розгляд ЕК.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
Ілюстративна частина кваліфікаційної роботи оформляється у вигляді слайдів, висвітлює
основні розділи та етапи роботи та містить:
тему роботи, мету, предмет та об'єкт дослідження,
сформульовані завдання, перелічено етапи виконання кваліфікаційної роботи, короткий
аналіз актуальності теми, варіант архітектури,
представлено опис програмного забезпечення; опис програмного забезпечення та тестування.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності,			
основи охорони праці			
Нормоконтроль	Стоянов Ю.М.		

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Вибір та погодження теми з керівником. Створення та затвердження завдання	27.01-02.02.25	виконано
2.	Затвердження теми КР наказом ректора	18.04.2025	виконано
3.	Аналіз технічного завдання, підбір та аналіз	20.04.2025	виконано
	бібліографічних матеріалів необхідних для виконання		
	кваліфікаційної роботи		
4.	Проєктування та розробка програмного продукту	11.05.2025	виконано
5.	Тестування та дослідна експлуатація програмного	19.05.2025	виконано
	забезпечення		
6.	Створення допоміжної документації. Написання та	25.05.2025	виконано
	перевірка на відповідність вимогам пояснювальної		
	записки		
7.	Формування записки кваліфікаційної роботи	28.05.2025	виконано
8.	Перевірка програмою антиплагіат	29.05.2025	виконано
9.	Безпека життєдіяльності, основи охорони праці	.06.2025	виконано
10.	Нормоконтроль	.06.2025	виконано
11.	Попередній захист	09.06.2025	виконано
12.	Захист кваліфікаційної роботи	.06.2025	

Студент _____ Шевчук В.Ю.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Яворська Є.Б.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота ОР «бакалавра» складається з вступу, основної частини, висновку, та містить в собі: 70 с., 10 рис., 2 табл, 45 літ. джер., 2 дод.

Ключові слова: Кваліфікаційна робота, освітній рівень, бакалавр, програмний продукт, інформаційний довідник, пошук, критерії, вимоги, концептуальну модель, розробка, Android, MySQL, Java, PHP, тестування.

В роботі представлено основні етапи створення програмного забезпечення довідникового змісту стосовно об’єктів визначеної території та систему вдосконаленого пошуку на основі довільних критеріїв. Сформульовано завдання та описано вимоги. Проаналізовано підходи та обрано оптимальний. Створено схему бази даних, для чого попередньо, виокремлено основні сутності, побудована концептуальну модель бази даних, UML діаграми класів, модель архітектури системи.

Оскільки система призначена для мобільного пристрою, то розробка розрахована для одного типу користувачів, які здійснюють пошук за заданими критеріями.

Система призначена для ОС Android, при розробці використано парадигми об’єктно орієнтованого підходу, мова Java.

ABSTRACT

Key words: qualification work, educational level, bachelor, software product, information reference, search, criteria, requirements, conceptual model, development, Android, MySQL, Java, PHP, testing.

The paper presents the main stages of creating software for reference content regarding objects of a specific territory and an advanced search system based on arbitrary criteria. The task is formulated and the requirements are described. The approaches are analyzed and the optimal one is selected. A database schema is created, for which the main entities are previously identified, a conceptual database model, UML class diagrams, and a system architecture model are built.

Since the system is designed for a mobile device, the development is designed for one type of user who searches according to the specified criteria.

The system is designed for the Android OS, the paradigms of the object-oriented approach and the Java language were used in the development.

ЗМІСТ

ВСТУП

Кожне покоління інше, всі більше зусиль прикладається на автоматизацію та зменшення фізичної праці, натомість вже почалась ера роботизованої «розумної» техніки. Моє покоління за цифровою обізнаністю та можливостями далеко перевершує вже навіть попереднє, натомість тут я бачу і проблему, коли людина не має можливості скористуватись телефоном чи іншим гаджетом вона стає безпомічною. Не дивлячись ні на що в цьому я вважаю і один з моментів прогресу, і на часі нові винаходи, коли вже не потрібно буде мати фізично навіть телефон чи інший гаджет. Проте, якого б рівня техніка не була, як би тісно штучний інтелект не зайшов в наше життя, я вважаю, що ІТ спеціальності як і всі, де, окрім знань, потрібна креативністсь, кмітливість та інші навики притаманні інженерам ще довго будуть актуальними і роботонезамінними в повній мірі.

Для своєї роботи я одразу прийняв рішенняя розробити щось, що дасть змогу зменшити час ,потрібний для комунікації, адже час – то є важко а подекуди не відновлюваний ресурс і його заощадження – це в першу чергу можливість дослідити щось нове, чи просунутись вперед.

Сьогодні вже фактично всі побутового рівня речі можна значно швидше та ефективніше вирішити засобами зв’язку. Звичайно, фізичне спілкування для людини є незамінним, проте в час інтернету інформація між людьми розповсюджується дуже швидко, час, потрібний для комунікації скорочується, можливість прийняття рішення може бути втрачено. Інформація яка в цю хвилину вартує мільйони, через хвилину вже нікому може бити не цікава. Швидкість та вчасність сьогодні є ключовими критеріями успішного бізнесу.

Отже, інформацію можна отримувати, сприймати та розглядати по різному і одним з варіантів, який я пропоную розглянути в роботі детальніше є тип довідника, який би був більш інформативним та доступним в будь-який момент часу, також було б корисно завжди мати цей довідник при собі. Для того, щоб не множити ґаджети, адже це мінімум не зручно, такого типу потреби можна легко задовольнити, втіливши в життя відповідну програму, призначену для

використання в смартфонах та планшетних комп’ютерах, при умові підтримання інформації в максимально актуальному стані. Зрештою, функціонал такої програми може надавати можливість і самим користувачам допомагати в цьому. Зараз майже кожен має смартфон під управлінням одної з популярних ОС – Android, iOS чи Windows Phone, які дозволяють встановлювати різноманітні сторонні програми. Очевидно, що такий довідник в кишені кожного більш ніж потрібен. Також варто закцентувати увагу на тому, яким чином повинен проводитись пошук в такого роду програмному забезпеченні. Він повинен максимально задовольняти користувача. Для цього можна користуватися різного роду критеріями. Розробка такої системи пошуку потребує детального аналізу вибраної предметної області для виявлення лише доцільних критеріїв і відсіювання зайвих. В цьому можуть допомогти описи вже готових подібних програмних рішень пошукових систем та довідників. Отже, якщо програмний продукт буде мати зручний та інтуїтивний для пересічного користувача інтерфейс, адекватну швидкість роботи і час відгуку та буде працювати на більшості Android-пристроїв, то він буде користуватись популярністю і зможе бути однозначно корисним.

1 РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ

1.1 Аналіз вимог до програмної системи

1.1.1 Аналіз предметної області

Обираючи тему, я виходив в критеріїв повноти та, за підказкою керівника роботи, почитав освітню програму за якою я навчався і виявив, що в принципі тема є такою важливою, головне, щоб вчасно завершити роботу, вона була актуальною, та дала мені можливість проявити себе. Особисто я найчастіше в телефоні використовую функцію пошуку, і не важливо чого. Одного разу я спізнився на важливу зустріч, оскільки заплутався з локаціями, адже на сьогодні існує багато довідників, але по відношенню до конкретних місць, міст чи територій вони містять недостатньо інформації [1-5]. Дуже для багатьох характерно, що вони працюють в межах світу, з можливістю деталізації, тобто нема конкретної орієнтовності на певне місто. З іншого боку є і місцеві ресурси, але вони виконані в якості сайтів, а це несе за собою в більшості довгий час завантаження зі смартфона, не завжди швидкість є задовільною ну і смартфони є різного класу. Це пояснюється тим, що при завантаженні кожної сторінки браузер смартфона повинен отримувати багато зайвого, що не стосується конкретного запиту користувача (релевантність запиту). Крім того, на невеликому дисплеї інформативність таких сайтів різко знижується, немає цілісної картинки або вона надто мала, важко зорієнтуватись, особливо якщо нервуєш чи спішиш, і стає ще важче знайти потрібну інформацію. Таким чином вузькоспеціалізована програма, конкретно для смартфонів і з конкретним містом чи локацією (не деталізація світова) матиме значну перевагу насамперед в точності, актуальності і зручності.
Перш за все це інформативність – елементи на екрані розташовані таким чином, щоб максимально доступно подати ту інформацію, яка потрібна користувачеві. Інша, не менш важлива деталь, це швидкість відображення інформації. В програмі вже безпосередньо є шаблони представлення і все, що залишається зробити – це завантажити з сервера лише текстову інформацію і лиш при потребі картинки.

Такий процес відбувається в рази швидше, ніж завантаження сайтів, і ідеально підходить для українських військових реалій.

1.1.2 Постановка задачі

Для створення програмного забезпечення Розробка предметно-орієнтованого інформаційного додатку для використання на ОС Android, мовою Java, в середовищі Eclipse [6-9] та реалізації відповідної автоматизованої системи подачі інформації необхідно скласти список завдань, а саме:

1) створення багатокористувацького програмного продукту;

2) автоматизована система повинна підтримувати один тип користувача;

3) у продукті буде враховано можливість захисту від неправильного використання системи; також в системі повинен бути передбачений пошук місця, закладу чи установи по його назві;

4) у системі повинна бути реалізована категоризація закладів, установ тощо;

5) у системі повинен бути передбачений пошук місця, закладу чи установи по одній чи декількох категоріях;

6) передбачено можливість ознайомитись детально з інформацією про знайдений об’єкт: номер телефону; адресу; e-mail; графік роботи; загальний опис; категорію; розташування на карті; фотографії об’єкту.

Для ефективного та зручного використання автоматизована система повинна підтримувати ряд наступних функцій:

- моментальне завантаження інформації, як тільки зміниться текст в полі пошуку («інтелектуальний пошук»);
- поступове завантаження пошукової видачі;
- якщо жодної категорії не вибрано, пошук по всіх, і додатково сортування по рейтингу.

1.1.3 Пошук актантів та варіантів використання

Варіанти використання в більшості випадків складаються з набору сценаріїв. Так як варіант використання визначає мету операцій, то сценарій описує спосіб досягнути поставлену мету. Після детального аналізу сценаріїв використання системи було виділено основного актанта – користувача [10-18]. Користувач матиме доступ до інформації, яка міститься в базі даних, через відповідну програму на мобільному пристрої. Мається на увазі наступне:

1) номер телефону об’єкта;

2) адреса об’єкту;

3) e-mail об’єкту;

4) графік роботи об’єкту;

5) загальний опис об’єкту;

6) категорія об’єкту;

7) розташування об’єкту на карті;

8) фотографії об’єкту.

Для зображення усіх відношень необхідно побудувати діаґраму варіантів використання, або по іншому діаграму прецендентів, врахувавши усі сценарії використання, можна скласти діаграму варіантів використання, результат представлено на діаграмі як на 1.1

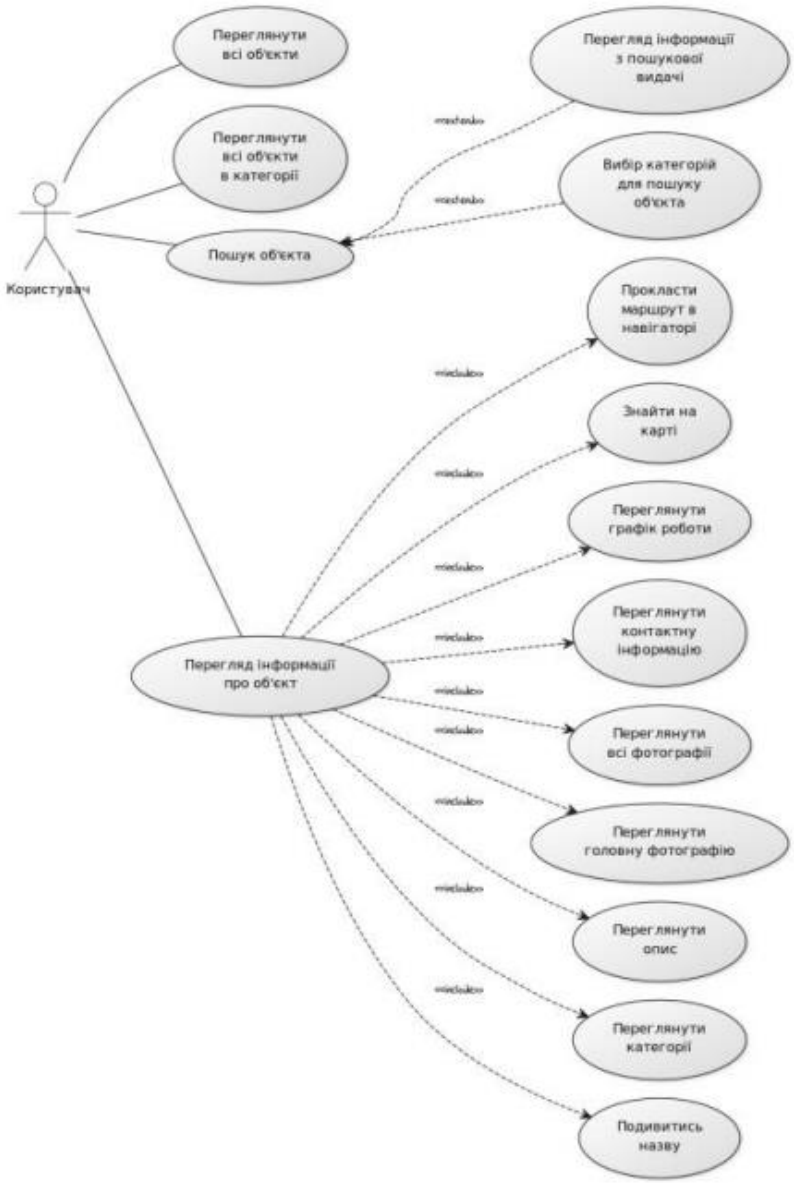

Рисунок 1.1 — Діаґрама варіантів використання

1.1.4 Опис ключових варіантів використання

Після перегляду та аналізу усіх ролей та варіантів використання, я прийняв рішення скоротити схему варіантів використання через її громіздкість та малу функціональність деяких сценаріїв [19-23]. Таким чином, я прийняв рішення

усунути деякі розширені ролі користувача, оскільки вони відображаються на одному екрані, при одному й тому ж стані програми. На рисунку 1.2 я показав скорочений варіант діаграми оптимізувавши функціональність

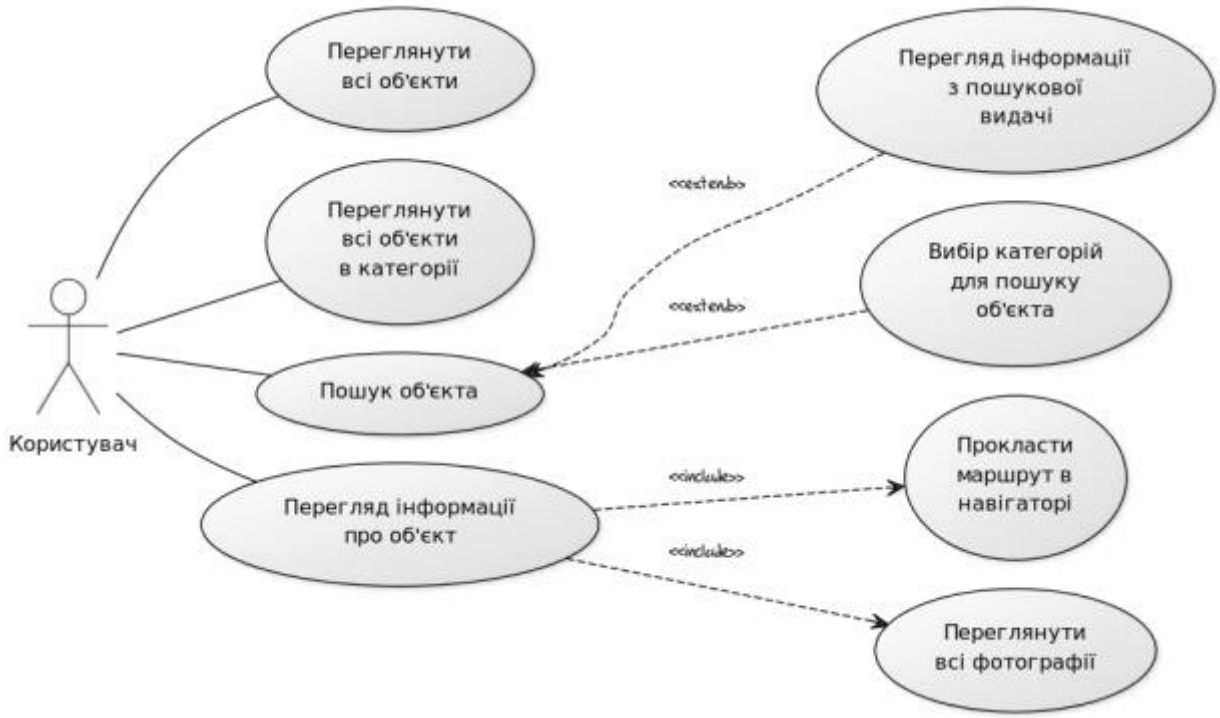

Рисунок 1.2 — Оптимізоване представлення діаграми

1.1.5 Моделювання словника системи

Глосарій містить описи термінів, що використовуються при проектуванні інформаційного довідника. Таким чином я визначаю основні поняття, які напряму пов’язані із розробкою програмного забезпечення. Глосарій створений в рамках проекту та містить такі основні терміни: Об’єкт; Категорія; Пошукова видача; Багатокористувацька система; Запит; База даних; Серіалізація; Десереалізація; Тестування.

1.2 Проектування програмної системи

1.2.1 Вибір процесу розробки

В даній роботі я прийняв рішення реалізувати механізм з врахуванням парадингм об’єктно-орієнтованого підхода, перевагою якого є швидкість і надійність, а також можливість для гнучкої модифікації. І саме це і стало вирішальним при побудові систем інформаційного типу вже як окремого напрямку, який виділився та інтенсивно розвивається [24, 25].

В загальному варта зазначити, що при застосуванні, для реалізації, об’єктно-орієнтованого підходу задачу слід поділити на етапи, а саме: етап аналізу вимог, створення моделі роботи (моделювання важливий етап, оскільки спрощує розробку, аналіз та реалізацію її оскільки робиться акцент на основні особливості розроблюваного продукту. Таким чином, модель є адекватною щодо представлення структури проєктованої системи, а також визначальною по функціональним залежностям поміж об’єктами в цілому, етап розробки архітектури системи, етап проєктування проґрамних модулів і бібліотек, етап реалізації вже безпосередньо проєкту, етапи тестування і впровадження системи.

Якщо говорити про об’єктну модель, то за її допомогою представляється проектований додаток як сукупність об’єктів, і кожен з цих об’єктів є абстракцією або чимось з окресленими границями, які значущі в контексті задачі. Кожен об’єкт, в свою чергу, відповідає якимось особливостям, а також реалізовує специфічні функції. В моделі представляються об'єкти, які є важливими в розроблюваному додатку, якими власне і визначається прагматика досліджуваної системи.

Отже, метою розробки об’єктної моделі переслідується опис елементів, які становлять проєктовану систему та встановлення взаємозалежностей між ними.

Також я проаналізував основні підходи до проєктування програмних засобів на основі обраного об’єктно-орієнтованого підхода, застосування парадигм якого без особливих складностей дає змогу створити систему на базі запропонованої моделі і забезпечити механізм її модифікації через розширення

функцій або зміни алгоритму візуалізації. Завдячуючи оригінальності структури запропоновано підходи, які дають можливість реалізовувати ширші функціональні можливості. Також, такий підхід є досить простим у використанні, високофункціональним під час роботи з даними та забезпечує і високу гнучкість, і легку розширюваність в процесі створення складних спеціалізованих систем.

1.2.2 Побудова схеми бази даних

Проаналізувавши дану предметну область, я виділив основні сутності для зберігання даних [26, 27].

Для початку в БД повинна містити інформацію про усі об’єкти, оскільки вони є основою для розробки даного програмного забезпечення. Також потрібна сутність для збереження категорій об’єктів. Оскільки один і той самий об’єкт може відноситись одразу до декількох категорій, потрібно виділити сутність для зв’язку категорій з об’єктами. Інформація ж про додаткові фотографії об’єктів буде міститись в ще одній сутності. Крім цього, в системі передбачено деякі дії користувачів, тому, звичайно, для початку потрібно виділити окрему сутність користувачів. Далі, враховуючи, що користувачі повинні мати можливість поставити рейтингову оцінку об’єкту, а також позначити його як улюблений, а також те, що кожен перегляд кожної сторінки повинен враховуватись, можемо винести ще три таблиці для збереження цих даних відповідно. Також, на основі цих оцінок повинна формуватись пошукова видача індивідуальна для кожного користувача, тому потрібно виділити ще одну сутність “критерії”, щоб була можливість зберігати значення ваги кожного критерія на постійній основі, крім того, унікально для кожного користувача. Отже, враховуючи вищенаписане, стає можливо спроектувати базу даних, в якій будуть присутні наступні сутності:

- об’єкти (місця);
- фотографії;
- категорії;

- користувачі;

- критерії;

- візити;

- рейтинг;

- закладки.

Після розгляду усіх наведених сутностей стає можливим створити між ними зв’язки. Таблиці “місця” та “фотографії” мають зв’язок один-до-багатьох, оскільки одне місце може мати кілька фотографій, проте одна фотографія належить лише одному місцю. В свою чергу між сутностями “місця” та “категорії” встановлюється зв’язок багато-до-багатьох. Це зроблено тому, що одне місце може належати одразу до декількох категорій, а категорії в свою чергу унікальні і можуть бути прив’язані до багатьох місць. Таблиця критеріїв повинна мати відношення один до одного з таблицею користувачів, оскільки кожному користувачеві відповідає один набір оцінки критеріїв. Також очевидно, що таблиці “візити”, “рейтинг” та “закладки” повинні мати відношення багато до одного з таблицями “місця” та “користувачі”.

Врахувавши усі зв’язки стає можливим побудувати схему сутностей бази даних, тобто ER-діаграму, яка зображена на рисунку 1.3. Сутність “місця” повинна містити інформацію об’єкти або локації та містити наступні атрибути:

- назва – назва закладу, підприємства чи установи;

- опис – загальний опис конкретного об’єкта;

- контакти — контактна інформація об’єкта: email, номер телефону, посилання на сторінки в соціальних мережах;

- адреса;

- координати – GPS координати об’єкта для знаходження і відображення його на карті;

- графік – графік роботи.

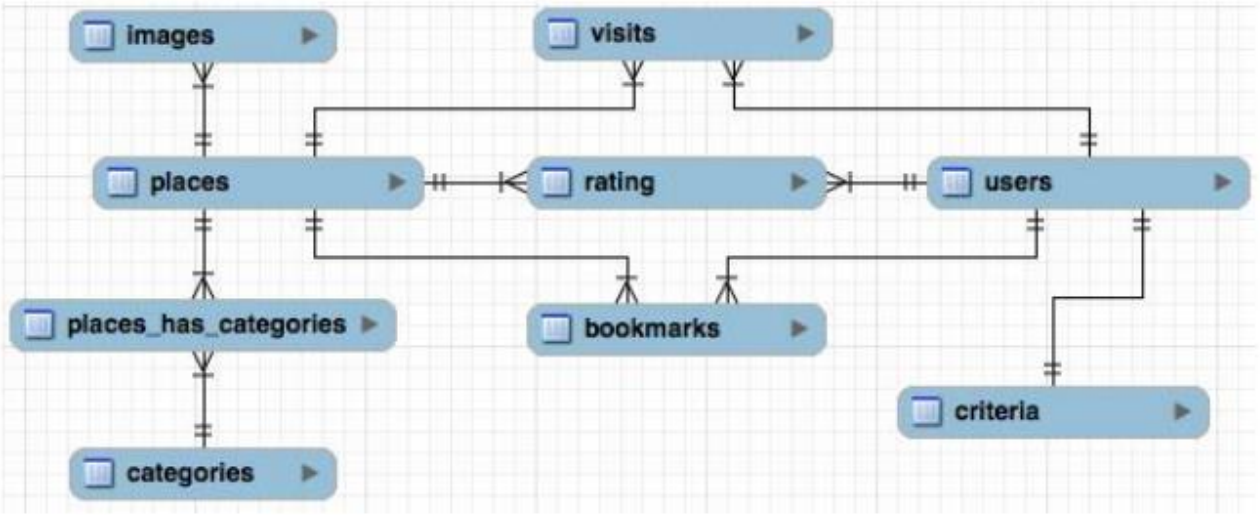

Рисунок 1.3 — Скорочена ER-діаграма бази даних

Сутність “фотографії” буде містити додаткові фотографії об’єкта і мати наступні атрибути:

- назва;

- URL-посилання на фотографію. Наступна сутність “категорії” міститиме всі можливі категорії об’єктів та буде містити поле назви категорії.

Сутність “місця-категорії” буде містити інформацію про те, які об’єкти до яких категорій належать.

Крім того, всі ці сутності повинні мати унікальний ідентифікатор, id, для унікальності кожного запису.

Сутність “користувачі” будуть містити мінімальну інформацію про користувача:

- id — унікальний ідентифікатор пристрою;

- пристрій — назва пристрою, з якого користуються програмою;

- операційна система — назва операційної системи на пристрої;

- версія операційної системи;

- IP — IP адреса, з якої останній раз користувались програмою.

Сутність “критерії” відображає вагу по кожному критерію для конкретного користувача, тому в ній повинні бути поля для ідентифікації користувача, а також комірки для виставлення коефіцієнту по кожному критерію:

- id;

- id відповідного запису в таблиці користувачів;

- візити — коефіцієнт важливості кількості переглядів даного об’єкта;

- рейтинг — коефіцієнт важливості загального рейтингу об’єкта;

- закладки — коефіцієнт важливості для об’єктів, доданих в закладки;

- відстань — коефіцієнт важливості для відстані до об’єкта.

Сутності “візити”, “рейтинг” та “закладки” мають ідентичні поля:

- id;

- id відповідного запису в таблиці об’єктів;

- id відповідного запису в таблиці користувачів;

- дата моменту виконаної відповідної дії. Врахувавши усі атрибути та їх типи, можливо спроектувати EERдіаграму бази даних (див. рисунок 1.4).

Після побудови схеми реляційної бази даних потрібно провести детальний аналіз з метою перевірки, чи знаходиться вона в третій нормальній формі [28]. Для того, щоб довести, що відношення приведені до третьої нормальної форми, спочатку потрібно довести, що вони приведені до І-ої нормальної форми, а після цього до другої НФ. Нормалізація сама по собі є покроковим процесом, який по своїй суті є розбиттям одного відношення на кілька відношень, який виконується відповідно до алгоритма нормалізації,на базі функціональних залежностей.

В свою чергу перша ІНФ створює підгрунття для структурованої схеми баз даних і включає такі фактори, як: наявність основного ключа для кожної таблиці; без повторність груп - категорій даних; атомарність, коли для кожного атрибуту присвоюється єдине (не множина) значення [29-31].

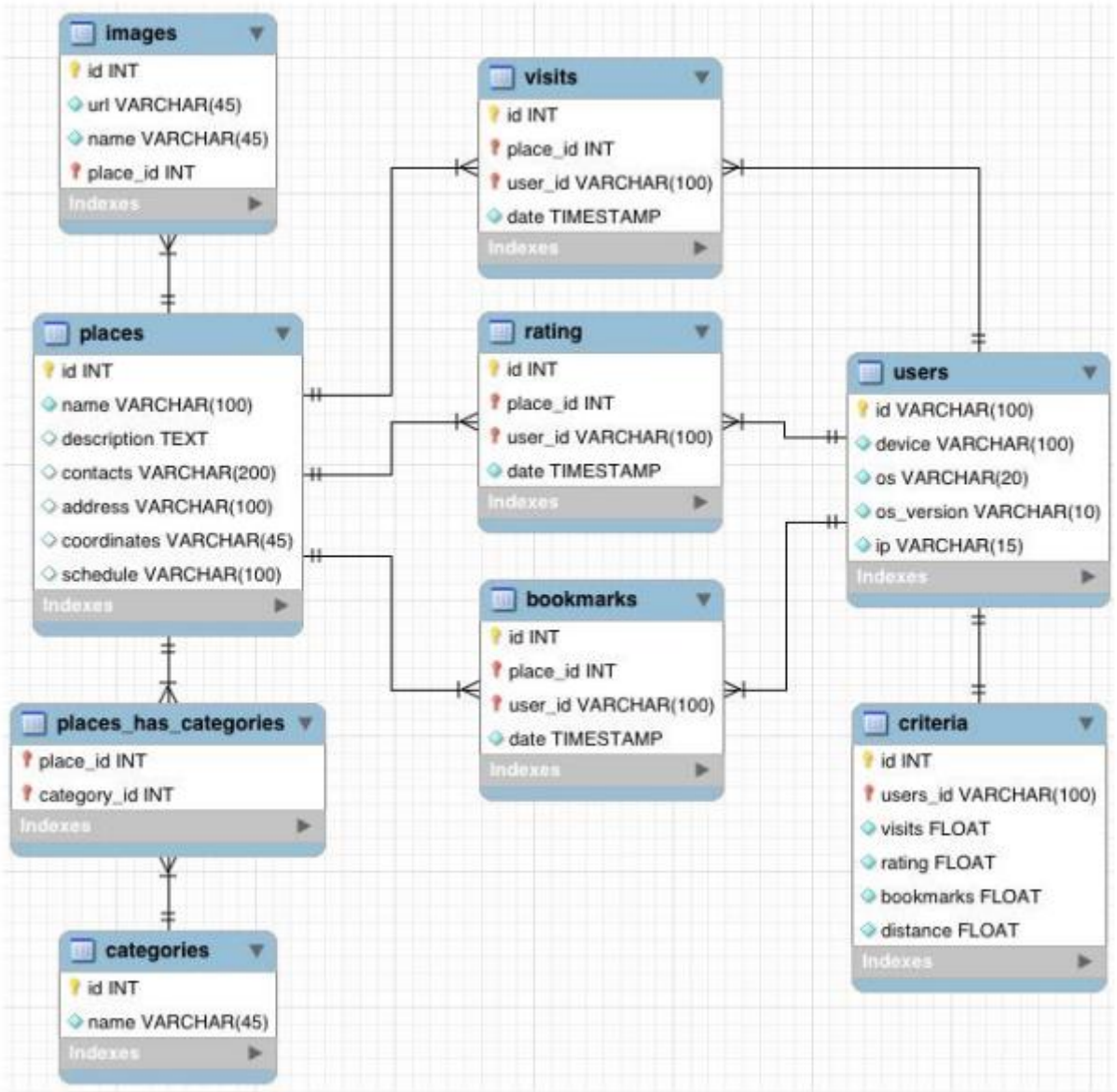

Рисунок 1.4 — EER-діаграма бази даних

Проаналізувавши кожну таблицю я зробив висновок, про те, що кожна таблиця володіє основним ключем, дані не повторюються, а кожному атрибуту присвоєне єдине значення. Таким чином, завершено процес приведенні бази даних приведена до І-ої НФ.

Для ІІ-ої норм-ної форми характерно, щоби данні, які знаходяться на зберіганні в таблицях із ключом композитного принципу, не залежали лиш від частки ключа, та була відповідність таким критеріям, як відповідність схеми бази даних до вимог ІНФ; дані, які повторюються в кількох колонках виносяться до окремої таблиці. Таким чином роблю висновок, що про схему бази даних можна говорити як про таку, що відповідає ІНФ.

Проаналізувавши дані у таблицях я роблю висновок, про те, що дані не дублюються в декількох колонках, а отже можу стверджувати, що представлену базу даних приведено до ІІ НФ.

Щоб говорити про ІІІ НФ потрібно щоби всі данні, які є таблиці були залежними виключно від ключа основного, та щоб схема бази була у відповідності до всіх вимог ІІ НФ; та щоби всі поля, яке залежать як від ключа основного так і від якогось іншого поля, були винесені до окремої таблиці.

Проаналізувавши дані у таблицях я роблю висновок, про те, що кожне табличне поле залежить лише від одного ключа, а отже можу стверджувати, що база приведена до ІІІ НФ.

Для того, щоб взаємов’язати усі таблиці певним видом зв’язку, використовують зовнішні ключі, атрибути чи їх набори, які перебувають в певному відношенні, та які відповідають первинному ключеві вже іншого відношення чи навіть і того ж самого відношення. Для реляційних баз даних зовнішній ключ як правило задають обмеженням FOREIGN KEY. Для того, щоб утворити взаємозв’язок один до багатьох, потрібно створити в таблиці, від якої буде йти зв'язок багато, зовнішній ключ з вказанням назви іншої таблиці та поля, по якому буде йти зв’язок (зазвичай, це основний ключ). У таблиці “місця-категорії” створено два зовнішніх ключа place_id та category_id для первинних ключів таблиць places та categories відповідно, тим самим згенерували зв’язки багато до одного. У відношенні images було створено зовнішній ключ place_id, що вказує на первинний ключ таблиці places, а саме id. Цим ініціалізовано між таблицями images та places зв’язок типу багато до одного.

1.2.3 Побудова UML-діаграми класів

Спочатку, для побудови UML діаграми [32], потрібно спроектувати головний клас для усіх об’єктів міста. Назвемо його Place. Він повинен віддзеркалювати відповідний йому запис в базі даних і має містити такі атрибути

як: назву, опис, фотографію, контакти, адресу, GPS-координати, графік роботи та рейтинг. Всі поля, крім координат і рейтингу, будуть стрічками. Рейтинг повинен бути чисельного типу, а координати будуть реалізовані як об’єкт, який має в собі два чисельні атрибути – X та Y. Тут варто зауважити, що координати виділяються в окремий клас, який буде атрибутом об’єкту Place. Крім того, об’єкт Place повинен мати ще один атрибут – масив фотографій конкретного об’єкту певної локації. Категорії теж варто виділити в окремий клас, оскільки кожна категорія має унікальний ID та назву.

Потрібен також і допоміжний клас Tools, який міститиме допоміжні методи, котрі будуть використовуватись для різних цілей в розроблюваному програмному забезпеченні, і буде зберігати в собі статичну інформацію, як наприклад список об’єктів категорій, які, доречі, будуть зберігатись в локальній базі даних SQLite. Крім того, буде реалізовано клас DataManager, який буде постачати дані в рівень представлення.

Для взаємодії з локальною базою даних необхідно створити клас DB, який буде реалізовувати основну логіку роботи з нею. В свою чергу, для отримання даних з віддаленого сервера відповідатиме клас Server, який буде підтримувати з’єднання з сервером і формувати запити до нього. Клас DataManager повинен використовувати класи DB і Server в своїй роботі для розділення рівня логіки і представлення.

Розглянувши усі класи та продумавши їх основну функціональність стає можливим побудувати UML діаграму класів (рисунок 1.5).

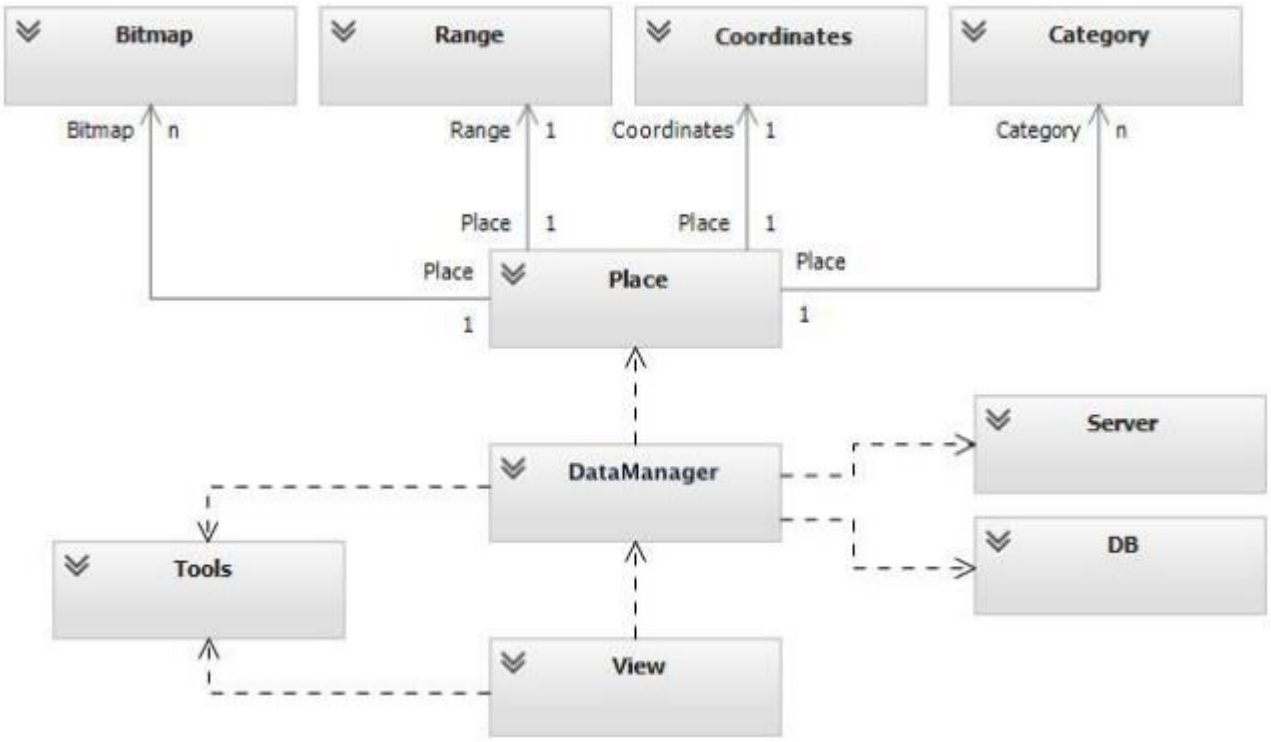

Рисунок 1.5 — UML-діаграма класів

Таким чином, рисунок 1.5 повністю відображає архітектуру класів, яка повинна бути реалізовано. Тут не включені внутрішні класи ОС Android, які будуть використовуватись в програмному забезпеченні, по причині нагромаджуваності діаграми і надлишкової інформації в ній.

1.2.4 Моделювання архітектури системи

Я вважаю, що одразу після вимог, поняття архітектури програмного забезпечення є ключовим в розрізі моделювання програмних систем [33]. Архітектура програмної системи це ціла множина рішень, які стосуються організації програмної системи, в сенсі вибору базових елементів, які складають систему, та також рішень щодо конструювання з підсистем, як складових, але більш простих елементів; інтерфейсів для окремих елементів системи; моделювання поведінки елементів системи, у тому числі в поєднаннях з іншими такими елементами. Важливе розуміння того, що архітектурою програмної

системи охоплюються не лише структурні та поведінкові аспекти, а й архітектурна функціональність, її продуктивність, гнучкість, можливості для використання, а також її повторного застосування, також вона має стосунок до економічних та технологічних обмежень, та пряме відношення до вимог. Традиційно архітектуру описує множина її представлень чи виглядів/views. Кожне з представлень архітектури відображає якийсь із аспектів, яким займаються фахівці ,які є членами команди яка працює над проєктом. Всі хто має стосунок до проекту чи то замовники, цільова аудиторія (користувачі), системні аналітики, менеджери проекту, розробники, менеджери та інші — переслідують свої цілі і тому бачить розроблювану систему «зі своєї позиції». Представленнями фіксуються головні рішення щодо проектованої структури, вказуючи, з яких компонент складається та як між собою пов’язані компоненти розроблюваної системи. Зазвичай ці рішення випливають з вимог до функціональності та низки інших факторів. Проте прийняті рішення в свою чергу встановлюють і подальші обмеження щодо вимог, так само як і на майбутні проектні рішення нижнього рівня.

Провівши аналіз та виходячи з потреб для розробки системи я обрав архітектуру клієнт-серверного типу, беручи до уваги як її гнучкість так і зручність. Такого типу архітектура представляє собою архітектурний шаблон програми та фактично є переважальною концепцією для створення розподільчих мережевих застосунків. Також цим типом архітектури передбачено взаємодію й обмін даними поміж ними. В ній передбачено головні компоненти, зокрема набір серверів, якими направляється інформація для програм, якими до них відправляються звернення; множину клієнтів, що користуються сервісами, що доступні на серверах; мережа, котрою забезпечується взаємодія поміж клієнтами і сервером/ами. Сервер/а незалежні одне від одного, стосовно клієнтів – вони працюють паралельно та також не залежно одне від одного. Жорстка прив'язка між клієнтами та сервером/ами відсутня. Часто може складатись так, що одним сервером одночасно опрацьовуються запити отримані від тих самих клієнтів. А з іншої сторони, клієнт має можливість звертатись до різних серверів за бажанням.

Хочу акцентувати увагу, на що чи хто розуміється під поняттям “клієнт”. Це може бути, наприклад, комп’ютер клієнта, від якого йде звертання до других комп’ютерних пристроїв, або якась із програм (клієнтська чи серверна), або люди, яким потрібна інформація, яку вони намагатимуться здобути при допомозі відповідного чи програмного чи апаратного забезпечень. Тобто загальноприйнятним є розуміння, що і клієнт і сервер перш за все програмними модулями. Зазвичай їх розташовують на різних комп’ютерах чи пристроях, проте бувають випадки коли як клієнтська, так і серверна програми фізично розташовані на одному комп’ютері, тому й сервер тоді вже називають локальним.

Модель клієнт-серверного типу взаємодії визначається поділом обов’язків поміж клієнтами і серверами. В цілому варта означити операції на трьох рівнях:

1-надання данних, є користувацьким інтерфейсом у відповідальності якого надання данних користувачеві а також введення отриманих від нього скеровуючих команд;

2-прикладне спрямування, де реалізується основна логіка програми. Також на цьому рівні виконується необхідна обробка інформації;

3- керування даними, де забезпечується збереження данних та доступ до цих даних.

Отже, для розроблюваної системи обрано дво-рівневе архітектурне представлення (рисунок 1.6).

Рисунок 1.6 – Дворівневе архітектурне представлення

Дворівневе клієнт-серверне архітектурне представлення має на увазі можливість взаємодії між двома програмними модулями (клієнт і сервер). А вже в залежності від того, як саме між ними розподіляються функції вирізняють модель типу «тонкого» клієнта (тут передбачено всю логіку застосунка), а керування даними зосереджене на сервері. Клієнтською ж програмою забезпечують лише функції рівню представлення; натомість модель по типу «товстого клієнта», у якій передбачено, що сервером лише керують даними, а вже робота з інформацією та користувацький інтерфейс зосереджено на боці у клієнта. До «товстих» клієнтів часто відносять пристрої обмеженої потужності, наприклад стільникові телефони, міні комп’ютери.

1.3 Конструювання програмної системи

1.3.1 Вибір мови та середовища розробки

Оскільки програмний продукт передбачено під платформу Android, для розробки було вибрано середовище розробки Eclipse, оскільки Google передбачили зручне SDK для цієї IDE, яке є у вільному відкритому доступі, як і код самої операційної системи [6-9]. Оскільки для клієнтської програми потрібен

сервер, який буде взаємодіяти із базою данних і постачатиме їй інформацію, я прийняв рішення написати його мовою PHP у тому ж самому середовищі для розробки Eclipse.

1.3.1.1 Обґрунтування вибору середовища для розробки Eclipse

З метою розробки клієнтської та серверної частин я прийняв рішення використати середовище для розробки Eclipse, яке є вільним модульним інтегрованим середовищем для етапу розробки програмного забезпечення. Написане в основному мовою Java, та може використовуватись з метою розробляти застосунки на Java і, при використанні різних плаґінів, іншими мовами програмування. Випущена на умовах Public License, вказане середовище розробки є безкоштовним загальнодоступним програмним забезпеченням та представляє собою фреймворк для розробки модульних крос-платформових застосунків з певними особливостями, зокрема можливістю розробки ПЗ багатьма мовами програмування (при цьому рідною є мова Java); є крос-платформна; є модульною, з призначенням для наступного розширення незалежними розробниками; володіє відкритим сирцевим кодом; з можливістю розробки і підтримки фондом Eclipse, до якого входять різні інші постачальники ПЗ. В першу ж чергу середовище Eclipse повноцінне Java IDE, націлене на групову розробку, з наявними засобами для роботи із системами для контроля версій.

Другим призначенням Eclipse є, що воно служить платформою для нових розширень. Множина розширень доповнює середовище Eclipse менеджерами по роботі із базами данних, серверами застосунків тощо.

Оскільки Eclipse написано мовою Java, середовище є платформо-незалежним продуктом, але це не стосується бібліотеки під графічний інтерфейс SWT, яку потрібно розробляти окремо під більшість найпоширеніших платформ. Бібліотекою SWT використовуються графічні засоби платформи, і це забезпечує швидкість і привичний користувацький зовнішній вигляд інтерфейсу.

Основою для середовища Eclipse являється платформа по типу розширеного клієнта RCP, яка складається з наступних компонент: ядра платформи (загрузка середовища Eclipse, запускання модулів); OSGi (є стандартним середовищем для поставки комплектів); SWT (є стандартним інструментарієм віджетів); JFace є файловими буферами, роботою із текстами, текстовими редакторами); саме робоче середовище для розробуи Eclipse (панель, редактор, проєкція, майстер).

З призначенням користувоцького інтерфейсу Eclipse також є залежним від проміжного шара GUI, який називають JFace, за допомогою якого спрощується побудова користувацького інтерфейса, який базується наSWT.

Eclipse є також доволі гнучким, і цю характеристику забезпечують модулі, що підключаю. Завдяки гнучкості є можливість здійснювати розробку не лише на Java, а й використовувати інші мови.

Під середовище Eclipse наявна ціла множина як безкоштовних так і платних модулів. Наявні і модуля призначеням яких є створення самих графічних інтерфейсів. Також важливо, що в середовище Eclipse інтегровано функцію для установки та оновлення модулів при використанні інтернет мережі..

1.3.1.2Пояснення вибору мови програмування Java

Насправді обгрунтування вибору достатньо просте. Справа в тому, що програмне забезпечення під Android можна писати всього двома мовами C++ чи Java [22-24, 35,36]. Для своєї розробки я вирішив зупинити свій вбір на другій мові Java, яка відноситься до мов програмування з об'єктно-орієнтованою парадигмою, перший випуск якої датовано 1995 роком. Синтаксис мови Для синтаксису Java властиве походження від мов C та C++, зокрема як основа, в модифікованій версії, запозичена об'єктна модель C++. Однак, важливо, що була усунута можливість для появи певних конфліктних ситуацій, які могли виникати, наприклад, в наслідок помилки програміста, також було полегшено і безпосередню процедуру розробки програм об'єктно-орієнтованого типу, а цілу

множину які в СтаС++ реалізовувались програмістами, передоручено для виконання віртуальною машиною. Оскільки, в першу чергу, мову Java розробляли як платформно- незалежну мову, тому їй властива наявність меншої кількості низькорівневих можливостей для роботи із апаратним забезпеченням. При потребі цих таких дій Java надає можливість викликати під програми, які було написано іншими мовами програмування.

Метою створення мови Java поставлено було наступні початкові цілі, зокрема, передбачалось, щоб синтаксис мови був достатньо простий (написана на Java програма працюватиме на любій платформі, яка підтримується апаратній або системній, без необхідності вносити зміни в початковий код чи проводити повторну компіляцію), очевидно, що об'єктно-орієнтованим, ну і просто звичним; реалізацію плановано зробити щоб була безвідмовна та безпечна, та високопродуктивна; заплановано, що повинні зберегтися такі властивості як незалежність від архітектури іа портативність; мова повинна бути динамічна, інтерпретована та підтримувати можливість мульти опрацьовування.

Сьогодні вже для більшості існуючих процесорів та операційних систем є віртуальні машини Java. Загальний доступ до платформозалежних особливостей зазвичай забезпечується стандартними бібліотеками, зокрема мова йде про про такі функції як обробка графіки та робота з мережами. Для деяких версій доступна можливість компілювання байт кода в машинний код до чи навіть при виконанні програми. Основною перевагою використання байт-кода є його портативність. Однак, наявність додаткових витрат на інтерпретацію значає, що інтерпретовані програми майже завжди працюватимуть повільніше, ніж ті, які було скомпільовано в машинний код. Завдяки цьому Java отримала репутацію досить повільної. Однак, розрив суттєво скоротили після введення кількох методів оптимізації до сучасних реалізацій JVM.

З часу випуску ранніх версій швидкість передбачену для офіційної версії віртуальної машини Java суттєво покращилась, також виявлено, що продуктивність передбачена для JIT компіляторів, порівнюючи з звичними компіляторами в машинний код, фактично зрівнялась. Однак, справедливо беду сказати, що ефективність компіляторів не у всіх випадках свідчить саме про

швидкість виконання зкомпільованого коду, оскільки лише при детальному тестуванні можливо виявити реальну ефективність даної системи.

1.3.1.3 Операційна система Android

Свою роботу я вирішив розробити під операційну систему Android, яка одночасно є і платформою для мобільних гаджетів і планшетів від компанії Google з ядром Linux о основі.

В ОП Android за базовий елемент прийнято реалізацію Dalvik, віртуальна машина Java, і відповідно усе програмне забезпечення і застосунки опираються на реалізацію Java.

Програми під Android обгрунтовано вважаються програмами з нестандартномим байт-кодом для віртуальної машини Dalvik. Розробка додатків під Android може вестися мовою Java. Наявний плагін під Eclipse, відомий як ADT («Android Development Tools»). Також існує плагін під IntelliJ IDEA, за допомогою якого полегшується розробка додатків під Android.

Окрім вище наведеного є ще MotodevStudio під Android – є комплексним середовищем розробки, розроблене на базі Eclipse, яке дає змогу працювати безпосередньо із Google SDK.

Після порівняння з додатками під Linux, для додатків під Android існують додаткові правила, зокрема ContentProviders можливість обміну данними між додатками; ResourceManager можливість доступу до ресурсів, файлиXML,PNG,JPEG; Notification anager можливість доступу до рядку стана; ActivityManager можливість керування активними додатками.

Також під ОП Android було розроблено формат призначений для інсталяційних пакетів типу .apk.

1.3.2 Вибір СУБД, її фізична модель

Основна роль СУБД полягає в тому, що через неї надається доступ до даних усім користувачам, при чому їм не потрібно навіть розуміти нічого стосовно фізичного розміщення данних в пам'яті та їх описи; можливих механізмів для пошуку даних на запит; проблем, які можуть виникати в ситуаціях одночасного запиту по одним і тих самим даним від багатьох користувачів (прикладних програмах); способів забезпечення захиста даних від не коректних відновлень та/чи не санкціонованого доступа; підтримки баз даних в актуальному стані та цілої низки інших не менш важливих функцій СУБД. В процесі реалізації основних функцій, СУБД використовує різноманітні описи даних.

Проектування бази даних необхідно розпочинати із детального аналізау предметної області та виявлення і формулювання вимог до неї користувачів, для яких власне вона і створюється. Об'єднавши, отримані в результаті опитування користувачів, окремі уявлення стосовно вмісту бази даних, зі своїми баченнями про дані, які можуть бути корисними при розробці майбутніх додатків, на початку розробником створюється узагальнений не формальний опис бази даних, яка розробляється. Зазвичай, такий опис - інфологічна модель даних [], виконується із використанням природньої мови, математичних представлень за допомогою формул, використовуючи таблиці, графіки та інші засоби, які є зрозумілими усім хто працює над проєктуванням баз даних.

Під час роботи над проєктом, я прийняв рішення використовувати СУБД MySQL з відкритим кодом [38]. Для нього я схилився завдяки цілого ряду її переваг відносно інших СУБД, зокрема це стосується в першу чергу швидкодії опрацювання великих масивів даних; простоти процесу встановлення та використання; підтримки необмеженої кількості користувачів, які можуть одночасно працювати з БД; кількості рядків в таблицях, яка може складати 50 млн.; високої швидкості виконання команд; простої та максимально ефективної системи забезпечення безпеки.

Обрана MySQL є однією із найпоширеніших систем для управління базами, яку, завдяки підтримці з боку різних мов програмування, перш за все, доцільно використовувати саме для створення динамічних web-сторінок. Розміщення даних в зовнішній пам’яті визначається фізичною моделлю. Часто фізичну модель можуть називати «внутрішня модель» системи, а вже форма її представлення в основному може залежати від вибраної системи керування БД, зокрема у випадку якщо перевага надана такій системі керування, яка підтримує реляційну модель даних, то необхідно таблиці із атрибутами і між табличними зв’язками переносити в середовище системи управління із необхідністю врахувати вимоги стосовно певних об’єктів БД. Таким чином, ідентифікатори-(імена таблиць та полів) повинні відповідати вимогам системи керуванняБД, також у відповідність до прийнятних для даної СКБД повинні приводитись як типи данних, так і розмір полів, а також обмеження.

Наступним кроком є необхідність визначитись із стратегіями для індексування, взаємозв’язками між таблицями, первинними та зовнішніми ключами, беручи до уваги визначені у даталогічній моделі та враховуючи способи їх задання у обраній СКБД.

На етапі вже фізичного проєктування варта зосередитись на забезпеченні цілісності БД. Зазвичай цілісність даних забезпечують її обмеженнями, яку досягають формулюванням цілої низки правил, якими регламентуються допуски для даних зі встановленими між ними зв’язками. Далі отримуємо варіант контрольного прикладу заповненої БД із врахуванням вже визначених і встановлених обмежень щодо її цілісності.

Отже, наступність дій в результаті яких буде створено фізичну модель в основі якої буде реляційна модель даних матиме наступну послідовність:

1 - проаналізувати та обрати СКБД;

2- розроблення структури (фізичної моделі) бази данних з використанням засобів обраної системи управління врахувавши типи даних та обмеження які стосуються цілісності атрибутів;

3- розробити схеми взаємозв’язків бази даних із використанням засобів обраної СКБД врахувавши обмеження для доступної цілісності.

1.3.3 Реалізація основних класів та методів

Немає потреби класи найнижчого рівня абстракції детально описувати, так як вони є нічим іншим як сутностями, якими оперує програма в своїй роботі, мають поля згідно бази даних та методи для їх отримання чи присвоєння.

Клас DataManager працює як міст між рівнем представлення і логіки. Деякі дані ним отримуються із локальної бази даних при допомозі класа DB, інші ж з віддаленого сервера через API, при використанні класу Server. В самому ж класі логіку не передбачено.

Далі пропоную розглянути клас DB. При створенні він наповнює локальну базу даних об’єктами категорій, щоб програма мала постійний, швидкий доступ до них, і не було потреби щоразу виходити в інтернет. В лістингу 1.1 я навів метод для отримання масиву категорій.

```
     public ArrayList<Category> GetCategories()
     {
          ArrayList<Category> categories=new
ArrayList<Category>();
          String query = "SELECT * FROM " + TABLE_CATEGORIES;
          SQLiteDatabase db = this.getReadableDatabase();

     Cursor c = db.rawQuery(query, null);

     if (c.moveToFirst()) {
          do {
               categories.add(new Category(
                    c.getInt(c.getColumnIndex(KEY_CAT_ID)),
                    c.getString(c.getColumnIndex(KEY_NAME))
               ));
          } while (c.moveToNext());
     }
     return categories;
}
```

Лістинг 1.1 – Метод для вибірки категорій з локальної бази даних

Далі пропоную перейти до класу Server. Він має публічні методи які повертають готову інформацію, таку як одне місце, набір місць для пошукової

видачі, набір фотографій місця. Отримання даних з сервера забезпечує метод, наведений в лістинзі 1.2.

Варта зазначити, що сервер з клієнтом «спілкуються» через JSON. Він передбачає серіалізацію даних в відповідну стрічку, яку потім на іншій стороні можна буде відновити як об’єкт.

Зазвичай, серіалізація використовується як спосіб передачі об’єктів через мережу так і з метою зберігати їх у файлах. Для прикладу, якщо виникла необхідність створення розподіленого додатку, в якому передбачено, що різні його складові будуть мінятися складноструктурованими даними. У такому разі під типи даних, які будуть передаватися, пишуть код, основним призначенням якого є виконання серіалізації і де-серіалізації. Об'єкти заповнюють потрібною інформацією, після чого викликають код вже для безпосередньо серіалізації. Далі, отриманий в підсумку результат серіалізації передають на приймаючу сторону, це можна зробити по e-mail чи засобами HTTP. Додатком-отримувачем створюються об'єкти того ж типа і викликаєтьс де- серіалізаційний код, в внаслідок роботи якого можна отримати об'єкт із аналогічним наповненням даними, які належали і об’єкту програми відправника. Серіалізація дає можливість отримати кілька, доволі корисних, можливостей, зокрема до таких можна віднести метод який передбачає виконання реалізації з метою збереження об'єкту/ів, який буде більш ручним, аніж варіант запису їхніх властивостей до файлу на диску з текстовим форматом представлення та дублюючий збір об'єктів зчитуванням файлів; метод здійснення реалізації віддалених процедурних викликів; метод призначений для розповсюджування об'єктів, зокрема при використанні технології компоненто - орієнтованого програмування; а також метод призначений для виявлення змін в наповненні даними, які зазнають часових змін.

Для максимально ефективного застосунку наведених можливостей потрібно дотримуватись принципу незалежності від самої архітектури. Для прикладу, потрібно знайти можливість надійного відтворення серіалізованого потоку даних, яке б не залежало від порядку байтів, який використовується цією архітектурою. Це значить, що найпростіша і найшвидша процедура суть якої полягає в

безпосередньому копіюванні ділянки пам’яті, на котрій розташована структура даних, і вона не може працювати одинаково якісно та надійно для усіх видів наявних архітектур. Процес серіалізації структур даних до архітектурно не залежного формата значить, що не буде жодних проблем щоб могли виникати внаслідок відмінностей по порядку в проходженні байтів, механізму розприділення пам’яті чи відмінності притаманні різним мовам програмування стосовно варіантів представлення структурованих даних. Для будь-котрої із серіалізаційних схем притаманно, що і наслідковий за визначенням процес кодування данних, а також вибірка любої з частин з серіалізованої структури даних потребує, щоби весь повністю об'єкт було зчитано від самого початка і до самого кінця і було відновлено. Для великої кількості програм така характерна лінійність є достатньо позитивним моментом, оскільки дає змогу зберігати і передавати стани об'єкту, користуватись звичайними інтерфейсами типу введення/ виведення, які мають загальне призначення.

```
    private String getDataFromUrl(String strurl) {
        String res = "";
        try {
            StrictMode.ThreadPolicy policy = new StrictMode.
              ThreadPolicy.Builder().permitAll().build();
            StrictMode.setThreadPolicy(policy);
            URL = new URL(strurl);
            HttpURLConnection con = (HttpURLConnection)
url.openConnection();
            res = readStream(con.getInputStream());

        } catch (Exception e) {
            e.printStackTrace();
        }
        return  res;
    }
```

Лістинг 1.2 – Метод для отримання даних з сервера

Крім того, в методі getDataFromUrl використовується приватний метод readStream, який забезпечує отримання вхідного потоку в стрічку (див. Лістинг 1.3). Він читає JSON-дані (відповідь сервера) в стрічку, допоки не закінчиться відповідь, і потім повертає стрічку, після чого її можна розпарсити в потрібний об’єкт.

```
    private String readStream(InputStream in) {
        BufferedReader reader = null;
        String result = "";
        try {
            reader = new BufferedReader(new
InputStreamReader(in));
            String line = "";
            while ((line = reader.readLine()) != null) {
                result += line;
            }
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (reader != null) {
                try {
                    reader.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
        return result;
    }
```

Лістинг 1.3 – Метод для направлення вхідного потоку даних в стрічку

Для отримання фотографій з сервера використовується метод getImageByUrl. Варто зауважити, що картинки з сервера передаються у вигляді base64 і на стороні клієнта, при отриманні, перекодовуються назад в Bitmap формат (див. Лістинг 1.4), після чого їх можна встановити, як ресурс для android-віджета ImageView

```
    public Bitmap getImageByUrl(String url){
        String requestUrl =
this.DOMEN+"?"+"method="+this.METHOD_GET_IMAGE +"&"+"url="+url;

        String response = getDataFromUrl(requestUrl);
        byte[] decodedString = Base64.decode(response,
Base64.DEFAULT);
        Bitmap decodedByte =
BitmapFactory.decodeByteArray(decodedString, 0,
decodedString.length);
        return decodedByte;
    }
```

Лістинг 1.4 – Метод для отримання картинок з сервера

Як описувалось вище, в програмі реалізовано real-time пошук. В лістингах 1.5 та 1.6 наведено уривки з коду, які відповідають за оновлення даних в списку пошукової видачі, якщо змінився текст в полі пошуку.

```
    public void onTextChanged(CharSequence s, int start, int
before, int count) {
        searchText = editText.getText().toString();
        if (msg!=null){
            handler.removeMessages(MSG_EVENT);
            msg = null; }
        if(searchText != ""){
            msg = handler.obtainMessage(MSG_EVENT);
            handler.sendMessageDelayed(msg, TIMER_DELAY); }
    }
```

Лістинг 1.5 – Код ініціювання оновлення пошукової видачі

```
public void handleMessage(Message msg) {
    Tools.PopupToastFast("Зачекайте...");
    Tools.searchText = searchText;
    Tools.rangeToSearch = null;
    Tools.SetCurrentPage(1);

    PlacesGetter = new PlacesGetter();
    Thread = new Thread(placesGetter);
    thread.start();
}
```

Лістинг 1.6 – Код оновлення пошукової видачі

З лістингу 1.5 видно, що логіка цієї особливості зав’язана на відкладеному виконанні методу. Якщо текст вводився і перестав вводитись більш, ніж хвилину назад, то це означає, що треба здійснити пошук по введеному тексту. Для цього ініціюється повідомлення з затримкою, яке після спрацювання викликає метод, описаний в лістингу 1.6.

Цей метод в свою чергу запускає новий потік (лістинг 1.7), в якому отримуються нові дані за допомогою DataManager і оновлюється масив місць в класі Tools. Цей масив пов'язаний зі списком в представленні, і haveToUpdate Handler оповіщає адаптера, який відповідає за відображення цього списку в представленні, що дані змінились.

```
class PlacesGetter implements Runnable {
   public void run() {
     ArrayList<Place> newPlaces =
DataManager.getInstance().GetPlaces(Tools.searchText,
Tools.GetSearchCategories(), Tools.rangeToSearch);
     if(Tools.rangeToSearch == null){
          Tools.foundPlaces.clear();
     }
     if(newPlaces != null){
          Tools.foundPlaces.addAll(newPlaces);
     }
     SearchScreenActivity.haveToUpdate.sendMessage(new
Message());
   }
}
```

Лістинг 1.7 – Код отримання нових даних

Новий потік потрібен для того, щоб не блокувати основний потім програми і його графічний інтерфейс, бо в іншому випадку програма просто «замерзне» на час, поки буде відбуватись отримання даних від сервера. Після того, як відпрацював потік призначенням якого є отримання нових даних з сервера, він посилає повідомлення основному потоку, що дія виконана і той в свою чергу оновлює список пошукової видачі (лістинг 1.8).

```
public static Handler haveToUpdate = new Handler(){
     @Override
     public void handleMessage(Message msg) {

          if(Tools.foundPlaces.size() > 0){
               textCount.setText("Показано: "
+Tools.foundPlaces.size()+" шт.");
          } else {
               Tools.PopupToastFast("Нічого не знайдено");
               textCount.setText("Показано: 0 шт.");

          }
          lvAdapter.notifyDataSetChanged();
     }
};
```

Лістинг 1.8 – Код оновлення пошукової видачі в представленні

З поміж інших варто виділити метод, за який відповідає кнопка “Знайти ще” (лістинг 1.9). Вона працює майже так само, як і метод для стрічки пошуку, але

разом з тим має одну суттєву відмінність. При вибірці нових даних тут вказується проміжок, на якому вибирати. Це зроблено для того, щоб зняти навантаження з мобільної мережі, а також для пришвидшення процесу отримання результату, коли пошукова видача передбачає велику кількість елементів. При кожному натисканні цієї кнопки буде додаватись нових 10 елементів, допоки вони не закінчаться в базі даних сервера.

```
btUpdate.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View arg0) {
                Tools.PopupToastFast("Зачекайте...");
                Range   =   new   Range(Tools.GetCurrentPage()*10,
Tools.GetCurrentPage()*10+9);
                Tools.rangeToSearch = range;
                Tools.SetCurrentPage(Tools.GetCurrentPage()+1);

                PlacesGetter = new PlacesGetter();
                Thread = new Thread(placesGetter);
                thread.start();
        }
});
```

Лістинг 1.9 – Додаткова вибірка пошукових даних

На екрані вибору категорій є кнопка для вибору одразу всіх. Вона працює таким чином, що проходить по всіх чекбоксах і включає їх (див. лістинг 1.10).

```
public void onClick(View arg0) {
        // select all
        for(int i=0; i < lv.getChildCount(); i++){
                LinearLayout itemLayout =
(LinearLayout)lv.getChildAt(i);

                CheckBox cb =
(CheckBox)itemLayout.findViewById(R.id.categoriesElement_CheckBo
x);
                cb.setChecked(true);
        }
}
```

Лістинг 1.10 – Кнопка вибору всіх категорій

Одна з досить цікавих функціональностей, це знаходження місця на карті і прокладання маршруту (див. лістинг 1.11). Всі роботи по даній частині

виконуються вбудованим системним навігатором. Пріоритетно системою вибирається Google Maps, але якщо його нема, то пропонується вибрати інший.

```
     public void onClick(View arg0) {
          Place = Tools.GetLastPlace();
          String uri = String.format(Locale.getDefault(),
"http://maps.google.com/maps?&daddr=%f,%f (%s)",
                    place.getCoordinates().getX(),
place.getCoordinates().getY(), place.getName());
          Intent = new Intent(Intent.ACTION_VIEW,
Uri.parse(uri));
          intent.setClassName("com.google.android.apps.maps",
"com.google.android.maps.MapsActivity");
          try {
               startActivity(intent);
          } catch(ActivityNotFoundException ex) {
               try {
               Intent unrestrictedIntent = new
          Intent(Intent.ACTION_VIEW, Uri.parse(uri));
                    startActivity(unrestrictedIntent);
               } catch(ActivityNotFoundException innerEx) {
                    Tools.PopupToast("Програма \"Карти\"
відсутня");
               }
          }
     }
```

Лістинг 1.11 – Перенаправлення на вбудований навігатор

В списку категорій на кожен чекбокс зав’язана подія, по якій дана категорія додається або видаляється з масиву категорій, в яких потрібно шукати і які знаходяться у вигляді статичного масиву в класі Tools (див. лістинг 1.12).

```
     chk.setOnCheckedChangeListener(new OnCheckedChangeListener()
{
          public void onCheckedChanged(CompoundButton buttonView,
boolean state) {
               int id = (Integer) buttonView.getTag();
               if(state == true)
                    Tools.AddCategoryToSearch(id);
               else Tools.RemoveCategoryFromSearch(id);
               tv.setEnabled(state);
          }
     });
```

Лістинг 1.12 – Вибір категорії для пошуку

В представленому розділі описані основні методи логіки роботи інформаційного довідника.

1.4 Використання програмної системи

1.4.1 Розгортання програмної системи та системні вимоги

1.4.1.1 Розгортання програмної системи

До процесу розгортки програмного продукту (далі – процесу) відносяться ті дії, виконання яких призводить до того, що програмна система набуває готовності до її використання. Процес є однією з важливих частин всього життєвого циклу будь-якого програмного продукту. Якщо розглядати в цілому, то вище згаданий процес містить в собі декілька між собою взаємнопов'язаних дій з передбаченими можливостями здійснення між ними переходів. Така активність зазвичай відбувається і як з боку розробника так і зі сторони кінцевого користувача. Так як кожен розроблений програмний продукт апріорно вважається унікальним, то в процесі розгортання доволі складно спрогнозувати усі можливі процеси чи процедури. Саме з цієї причини, процес розгортки варта все таки сприймати як більшій мірі загальний з відповідністю до вимог та характеристик. Розгортка може здійснюватись як самим програмістом так і під час самого процесу розробки програмного продукту. До активностей пов’язаних із процесом розгортки зазвичай відносять наступні процеси: випуску, установки, активації, де-активації, адаптації, оновлення, монтування, відстеження за версіями, видалення, вилучання з обігу.

Для розгортання моєї системи перш за все потрібно

1- визначити системні вимоги, які представлені далі,

2- порівняти системні вимоги з характеристиками цільового апарату,

3- встановити сервер бази даних MySQLl Server на серверній машині, де буде розгортатись база даних,

4- розгорнути БД на сервері та наповнити її необхідними даними,

5- інсталювати саму програмну систему на кожний цільовий апарат.

1.4.2 Типові схеми по використанню системи

Отже, запустивши систему, отримаємо виведений екран початку роботи. При переході до наступного стану користувач бачить головний екран програми з доступним для нього функціоналом. Тут є поле для пошуку, кнопка для вибору категорій і список, в якому виводяться результати пошуку. Варто зауважити, що пошук відбувається автоматично, коли користувач щось вводить чи змінює в полі пошуку. Результати пошуку показуються частково, по 10 одиниць, для зменшення трафіку і підвищення швидкості отримання результату, що особливо помітно з використанням доступу до інтернету через EDGE. Якщо користувач дійшов до кінця списку і хоче подивитись наступні результати, потрібно натиснути на відповідну кнопку внизу списку. Результати пошуку для кожного місця містять наступну інформацію: назва; адреса; категорії, до яких належить місце; основна фотографія. Кнопка «Категорії» переводить користувача на екран вибору категорій. Тут знаходиться повний список категорій місць, з чекбоксами для їх вибору. Якщо користувач нічого не вибрав, вважається що шукати потрібно по всіх категоріях. Після вибору закладу з пошукової видачі користувач переходить на екран з детальною інформацією про дане місце. Тут він може побачити таку інформацію: - назву; - категорії; - всі фотографії; - графік роботи; - контакти; - опис; - адресу. Крім того, з цього екрану можна знайти місце на карті та прокласти маршрут до даного місця за допомогою вбудованого в систему навігатора.

1.4.3 Верифікація програмної системи

Провівши верифікацію я зможу зробити висновки стосовно коректності створеної програмної системи під час її проєктування, та вже після закінчення процесу її розробки. Під час процесу верифікації виконують задачі на перевірку умов контракту, коректність самого процесу, відповідність вимогам, інтеграцію, код та документацію [39] .

Верифікацію програмної системи я провів згідно наступних етапів:

1) Формулювання питань.

За результатами ознайомлення з матеріалами та аналізу сформулював наступні питання: а) яким чином відбуватиметься пошук? б) які критерії розміщення об’єктів в пошуковій видачі? в) об’єкти чи місця якого типу будуть знаходитись в базі даних? г) яким чином буде прокладатись маршрут до вибраного місця? д) на скільки високим буде трафік для даної програми?

2) Загальна оцінка вимог. Для зручності оперування всі вимоги зведені в таблицю 1.1. Введена наступна типізація вимоги (поле “тип”): UC – функціональна, у формі прецеденту; F – функціональна; U – не функціональна (застосовуваність); R - не функціональна (надійність); P - не функціональна (продуктивність); S – не функціональна (щодо придатності до експлуатації).

3) Кількісне оцінювання вимог. Властивості вимог були оцінені за кількісною шкалою [0,1]: 0 – дуже низька якість, 1 – дуже висока якість. Для властивостей повноти та ясності шкала – дробова, для властивостей коректності та верифікованості – бінарна (0 або 1). У результуючої таблиці для кожної з властивостей представлені чотири оцінки, так як це показано у таблиці 1.1.

Таблиця 1.1 — Реєстр вимог з кількісною оцінкою

Код	Найменування	Тип	Повнота	Ясність	Коректність	Верифікованість
1	2	3	4	5	6	7
A1	Пошук місця	UC	0,9	0,8	1	1
A2	Перегляд всіх місць	UC	0,8	0,9	1	1

Продовження таблиці 1.1

A3	Перегляд детальної інформації про місце	UC	0,7	0,8	1	1
A4	Перегляд фотографій місця	UC	0,7	0,7	1	1
A5	Прокладання маршруту в навігаторі	UC	0,7	0,8	1	1
F2	Миттєвий пошук підчас вводу тексту	F	0,8	0,7	1	1
U1	Зручність використання	U	0,7	0,6	1	1
R1	Доступність	R	0,7	0,7	1	1
R2	Наробки на відмову	R	0,5	0,5	1	1
R3	Норма дефектів	R	0,6	0,6	1	1
P1	Одночасно працюючі користувачі	P	0,7	0,7	1	1
P2	Час відклику	P	0,8	0,8	1	1
S1	Масштабованість	S	0,9	0,6	1	1
S2	Оновлення версій	S	0,8	0,5	1	1
X1	Використовувані стандарти	O	0,8	0,8	1	1
X2	Вимоги до середовища для виконування	O	0,8	0,7	1	1
X3	Вимоги до СУБД та доступ до даних	O	1	1	1	1

4) Оцінка варіантів використання.

Всі вимоги, представлені у формі варіантів використання, були оцінені по ряду параметрів: автономність і закінченість; наявність мети (вимірюваного значення); правильний вибір рівня абстракції; повнота опису альтернативних сценаріїв; повнота опису не функціональних вимог; структурованість.

5) Кількісні оцінки прецедентів Оцінювання проводилося на основі використання шкали [0,1]. Результати зведені в таблицю 1.2.

Таблиця 1.2 — Реєстр існуючих прецедентів; кількісна оцінка

Властивості прецеденту/Код прецеденту	A1	A2	A3	A4	A5
Автономність і закінченість	1	1	1	0,8	0,7
Наявність мети (вимірюваного значення)	0,9	0,8	0,8	1	1
Правильний вибір рівня абстракції	0,8	0,7	0,7	0,8	0,8
Повнота опису альтернативних сценаріїв	0,6	0,7	0,6	0,6	1
Повнота опису нефункціональних вимог	0,8	0,9	0,9	0,7	0,8
Структурованість	0,7	0,8	0,8	0,8	0,9

1.5 Тестування програмної системи

В процесі тестування програмного забезпечення будь-якого типу та складності здійснюється технічне дослідження на предмет виявлення інформації стосовно якості програмного продукту відповідно до вимог, наповнення та контексті в межах його передбачене його використання.

Як правило поняття «якість» пояснюється поняттями коректності, надійності, практичності, безпечності, проте є й інші технічні вимоги, які регламентуються стандартом ISO9126. Наповненість та зміст супровідної документації всього етапу тестування визначено стандартом IEEE829–1998 Standard for SoftwareTestDocumentation.

1.5.1 Розробка тестів

Після детального аналізу та перевірки було вирішено провести кілька модульних тестів для клієнтської програми. Оскільки вона не виконує ніяких обчислень, а лише візуалізує інформацію, яка прийшла з сервера, було прийнято рішення створити тест для метода, який отримує з сервера об’єкти міста. Ми відправляємо до сервера запит на знаходження місць, назва яких містить текст «карма». В базі даних є лише один такий запис, що відповідає цій умові і його ID рівне 1. Відповідно по цьому параметру і будемо перевіряти правильність отриманих даних. Для тестування використовуються стандартна бібліотека JUnit. Код тесту наведено у лістингу 1.13.

```
public class GetPlacesTest extends TestCase {

     Server;

     protected void setUp() throws Exception {
          super.setUp();
          server = new Server();
     }

     public void test(){
          ArrayList<Place> places =
     server.GetPlaces("карма", new ArrayList<Integer>(1),
     null);
          boolean condition = false;
          for (Place : places) {
               if(place.getId() == 1){
                    condition = true;
                    break;
               }
          }
          assertTrue(condition);
     }

     protected void tearDown() throws Exception {
          super.tearDown();   }    }
```

Лістинг 1.13 – Тестування метода GetPlaces

Тест успішно пройдено (рисунок 1.13).

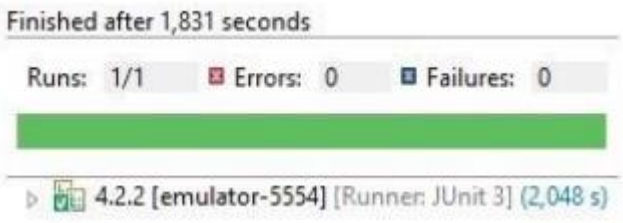

Рисунок 1.13 – Результат модульного тесту

Щоб протестувати графічний інтерфейс користувача, потрібно скористатись другорядними бібліотеками, тому що в стандартному наборі бібліотек є модулі лише для unit-тестування коду.

Для даної задачі добре підходить відома бібліотека Robotium. З нею можна дуже легко створювати методи, які будуть натискати по кнопках, вводити дані в поля та ін.

В графічному інтерфейсі буде доцільно звернути увагу на екран пошуку. Наступний тест (лістинг 1.14) повинен перевірити спочатку чи коректно загрузився початковий екран, а потім і екран пошуку, перевіривши наявність всіх елементів, які мають бути на екрані.

```
     public class GUITest extends
ActivityInstrumentationTestCase2< LandingScreenActivity> {

          private Solo;

          public GUITest() {
              super(LandingScreenActivity.class);
          }

          @Override
          protected void setUp() throws Exception {
              super.setUp();
              solo = new Solo(getInstrumentation(),
getActivity());
          }

          public void testOpenLandingPage(){
               solo.assertCurrentActivity("Check on Landing
page", LandingScreenActivity.class);
               Button btGo = (Button) solo.getView(
"landingScreen_ButtonStart");
```

```
                assertNotNull(btGo);
            }

            public void testOpenSearchPage(){
                solo.clickOnButton("Вперед");

                Button btGo = (Button) solo.getView(
"searchScreen_ButtonCategories");
                Button btUpdate = (Button) solo.getView(
"searchScreen_ButtonUpdate");
                EditText et = (EditText) solo.getView(
"searchScreen_TextSearch");
                ImageView img = (ImageView) solo.getView(
"searchScreen_ImageSearch");
                TextView tvItemCount = (TextView) solo.getView(
"searchScreen_TextItemsCountFound");
                ListView lv = (ListView) solo.getView(
"searchScreen_ListView");

                assertNotNull(btGo);
                assertNotNull(btUpdate);
                assertNotNull(et);
                assertNotNull(img);
                assertNotNull(tvItemCount);
                assertNotNull(lv);
            }
        }
```

Лістинг 1.14 – Тестування метода GetPlaces

Як видно з коду, тестування складається з декількох етапів: testOpenLandingPage; testOpenSearchPage.

Обидва етапи пройдені успішно (рисунок 1.14).

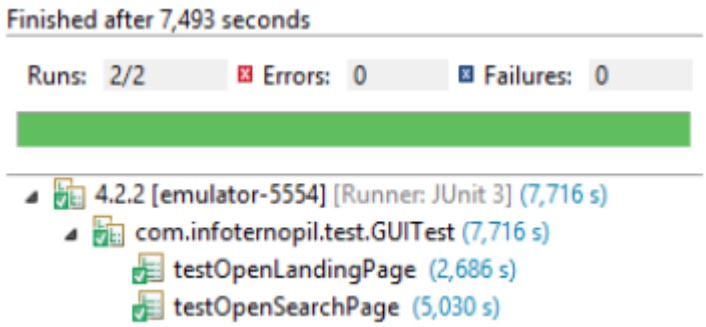

Рисунок 1.14 – Результат тесту графічного інтерфейсу

Оскільки система є багатокористувацькою, то дуже важливо, щоб сервер завжди працював стабільно і був доступним для всіх навіть при великих

навантаженнях. Для тесту було вибрано такі параметри: - 50 підключень; - кожне підключення відправляє запит в циклі 100 разів.

Сумарно виходить 5000 запитів до сервера, які, в результаті проходження тесту, він відпрацював на протязі ~100 секунд. Запити обробляються відносно швидко, оскільки в серверному коді немає складних обрахунків і в основному йдуть лише звернення до бази даних, тому частково тут швидкість залежить і від сервера БД. Як бачимо з графіку на додатку В, це в середньому, 1019 мілісекунд.

Отож, згідно отриманих результатів тестування можна стверджувати, що дана система добре витримує навантаження великої кількості одночасно активних програм-клієнтів.

2 ОПИС АЛГОРИТМА ПОШУКУ

2.1 Загальний опис алгоритму

Інформаційний довідник призначений для того, щоби допомагати в пошуку та отримувати детальну інформацію на запит. Система складається з клієнтського та серверного додатків та слугує для пошуку потрібної інформації на запит , та призначена для власників гаджетів, для яких використовується ОС Android).

Всю інформацію, яку призначено для закриття запиту передбачено зберігати в структурованій базі реляційного типу даних на стороні сервера. Власне проблематичним є питання вибірки найбільше адуквотної запиту інформації, яка була б якомога точніше та відповідала потребам та очікуванням користувачів.

Найпростішим з алгоритмів додатків такого напрямку можна вважати пошук за назвою. Ще одним за рахунок підтримки категоризації об’єктів, є визначення співпадання за категорією, а після цього представлення знайденого за алфавітом без згадування тексту зі змістом пошукового запита. Також можна застосовувати і такий критерій як, наприклад, оприлюднені рейтинги або ж кількості відвідин.

Власне мій задум полягає в здійсненні інтелектуального пошуку , тобто в розробці високоефективного алгоритму пошуку, релевантність запитів якого була б максимально високою.

Відштовхуючись від існуючих функціональних можливостей сервісу, критерії пошуку місць можуть бути абсолютно різними, повторюваними. Також в якості критеріїв можуть бути додаткові вподобання користувачів, щодо відбору, зокрема, наприклад, можливість перебувати з тваринами, наявність бомбосховища, можливість парковки для відвідувачів, наявність аніматорів та дитячої кімнати, тощо.

Помічними також буде можливість використати історію попередніх запитів та їх результатів,.

Головне питання розробки даного пошукового двигуна полягає в імплементації деякого механізма, до якого можна під’єднати довільну кількість модулів. Кожний таких модуль «відповідатиме» за якийсь критерій пошуку та фіналізувати його у вигляді конкретного результату з врахуванням важливості цього критерію відповідно до інших, які застосовувались попередньо.

Важливо, що ваговий коефіцієнт «важливості» по кожному з критеріїв можна міняти, причому самим користувачем, коли це потрібно. Отже, система буде завжди стійкою щодо релевантності відповідей на запит.

2.2 Реалізація алгоритму

Так як програмні рішення для мобільних пристроїв пріоритетно, щоби були максимально швидкими та не енергозатратними, було б дуже не доречним реалізовувати алгоритм на стороні клієнтської програми. Тим більше, у випадку, якщо фільтрувати інформацію на клієнтській стороні, то потрібно передавати дуже великі масиви не підготовлених даних з сервера, що потягне за собою значні недоцільні затрати і в першу чергу інтернет трафіка, а також оперативної пам’яті та процесорного часу.

Отже, виходячи з вище наведених міркувань, алгоритм набагато буде ефективніше реалізовувати на серверній частині додатка. Так як, для серверного комп’ютера відсутній ліміт по енерго затратах, які, власне, залежать від швидкості та ефективності роботи програм, крім цього серверні станції, як правило, є достатньо потужними і можуть без проблем одночасно обслуговувати тисячі користувачів.

Тепер думаю варта перейти до опису серверного додатку в цілому та алгоритма зокрема.

Отже, основна робота виконується при допомозі наступних таких класів: Criteria, SearchEngine, Service, APIMessage, DB. Крім цього присутні інтерфейси ICriteria та IAPIMessage.

UML діаграма класів представлено на рисунку 2.1.

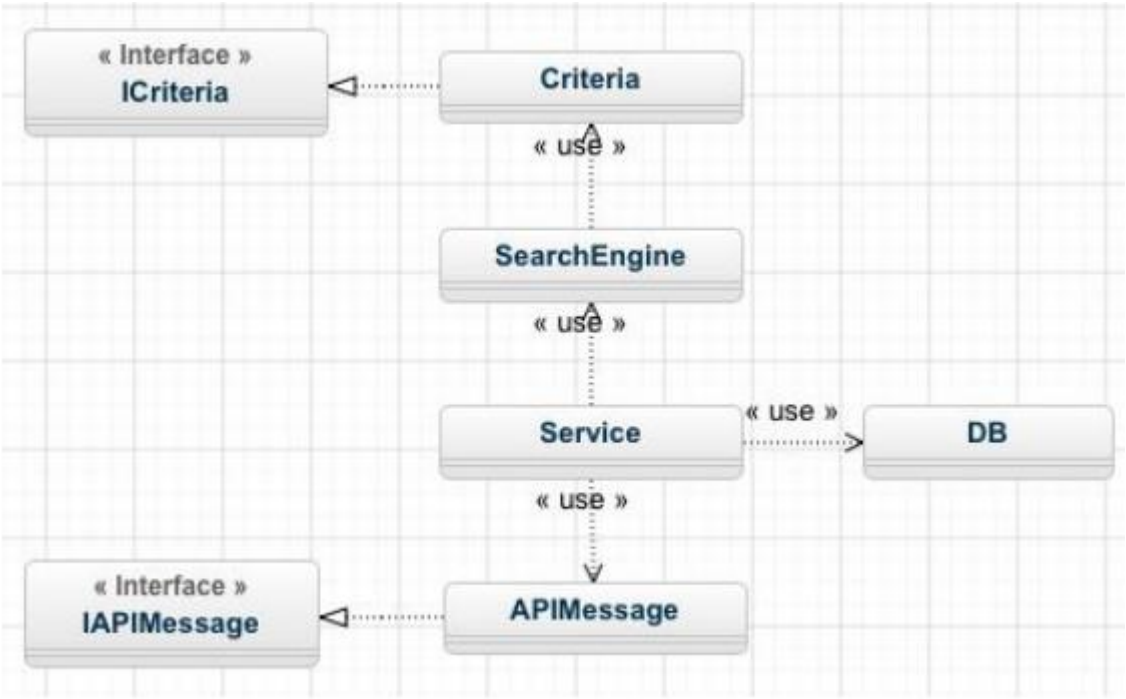

Рисунок 2.1 — UML діаграма класів серверу

В першу чергу варто звернути увагу на інтерфейс ICriteria. Він описує як має виглядати кожен критерій в коді. В ньому знаходяться всі основні методи, які повинні бути реалізовані класом Criteria, для коректної взаємодії з іншими компонентами програми. Кожен критерій, який використовується для пошуку та фільтрування результатів вибірки з бази даних, повинен бути об’єктом класу Criteria. Окремо один від одного критерії нічого не варті — для бажаного результату з ними потрібно звідкись взаємодіяти.

Клас SearchEngine містить все необхідне для коректної та ефективної роботи з критеріями (клас Criteria) і з усіма аспектами, що стосуються пошуку. Таким чином він забезпечує абстрагування пошукового процесу від всіх інших компонентів системи.

Клас DB забезпечує безперебійну роботу з базою даних. На кожен запит клієнтської програми створюється окреме підключення до бази даних, що забезпечує вищу стабільність та швидкодію при роботі. Крім того, окремо завжди існує одне резервне підключення, яке використовується в сервісних цілях – для оновлення даних всередині серверної програми, а також для моніторингу статусу підключення до бази даних.

Інтерфейс IAPIMessage створює зразок як повинно виглядати повідомлення, яке використовується для обміну даними між серверною та клієнтською частиною програми. В свою чергу клас APIMessage реалізовує цей інтерфейс, що забезпечує серверу “розуміння” того, як потрібно працювати з повідомленнями зовнішнього програмного інтерфейсу програми.

І на останок, клас Service. Це ядро сервера. Він управляє всіма основними процесами програми — взаємодіє з базою даних, клієнтською програмою, і з пошуковим двигуном. Добре налагоджені процеси та результат хорошого проектування дозволяють серверній програмі ефективно взаємодіяти з клієнтською та повністю задовольняти потреби.

На рисунку 2.2 зображено EER-діаграму бази даних серверної частини програми. Як вже було описано в попередньому розділі, в базі даних планується зберігати інформацію про візити користувачів в певний заклад харчування (чи інший об’єкт інфраструктури міста), рейтинг цього об’єкта, а також кількість додавань його в закладки користувачами цього програмного забезпечення. Крім того, при користуванні даною розробкою на мобільному пристрої користувача використовуються дані таких модулів, як GPS, AGPS та LBS для визначення координат широти і довготи поточного місцезнаходження користувача.

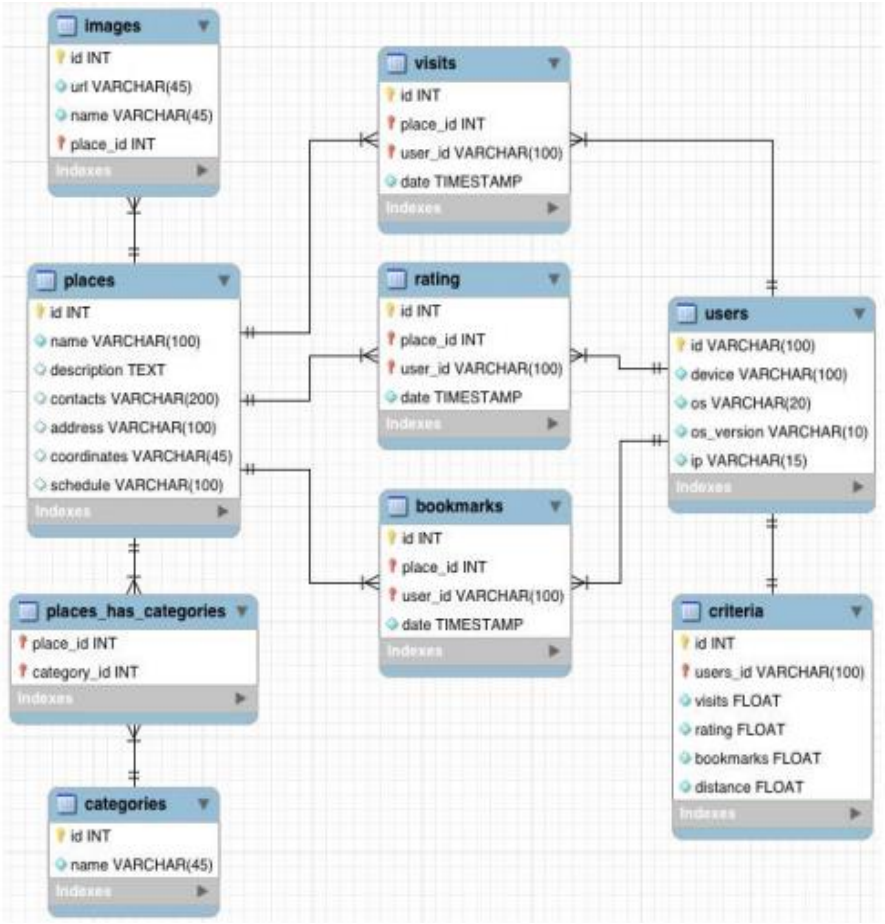

Рисунок 2.2 — EER-діаграма бази даних сервера

За використання технології A-GPS можна прискорити так званий «холодний» старт для GPS-приймача.

Отже, використовуючи спроектовану і розроблену систему, підчас проведення пошуку для забезпечення максимально релевантних результатів стає можливим використання таких критеріїв як: категорія об’єкту; кількість відвідувань об’єкта; рейтинг об’єкту; кількість додавань об’єкту в закладки; відстань від користувача до об’єкту.

Також варто було б розуміти, що в першу чергу в будь-якому випадку фільтрація результатів повинна відбуватись по категорії, оскільки користувач сам вибирає по яких конкретно (чи по всіх) категоріях потрібно провести пошук. Далі одним з вагомих критеріїв є відстань до об’єкту, після проходження якого можна віддавати дані на обробку по всіх інших критеріях. Гнучкість системи заключається в тому, що можна в будь-який момент змінити вагу потрібного критерія, і таким чином змінювати алгоритм і результат пошуку.

3 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

3.1 Безпека в надзвичайних ситуаціях

3.1.1 Запобігання виникненню надзвичайних ситуацій

Запобігання виникненню надзвичайних ситуацій– це підготовка та реалізація комплексу правових, соціально-економічних, політичних, організаційно-технічних, санітарно-гігієнічних та інших заходів, спрямованих на регулювання безпеки, проведення оцінки рівнів ризику, завчасне реагування на загрозу виникнення надзвичайної ситуації на основі даних моніторингу (спостережень), експертизи, досліджень та прогнозів щодо можливого перебігу подій з метою недопущення їх переростання у надзвичайну ситуацію або пом'якшення її можливих наслідків [40-42].

Зазначені функції запобігання надзвичайним ситуаціям техногенного та природного характеру в нашій країні виконує Єдина державна система запобігання надзвичайним ситуаціям техногенного і природного характеру і реагування на них, затверджена Постановою Кабінету Міністрів України.

Єдина державна система запобігання надзвичайним ситуаціям техногенного і природного характеру і реагування на них (ЄДСЗР) включає в себе центральні та місцеві органи виконавчої влади, виконавчі органи рад, державні підприємства, установи та організації з відповідними силами і засобами, які здійснюють нагляд за забезпеченням техногенної та природної безпеки, організують проведення роботи із запобігання надзвичайним ситуаціям техногенного та природного походження і реагування у разі їх виникнення з метою захисту населення і довкілля, зменшення матеріальних втрат.

Основною метою створення ЄДСЗР є забезпечення реалізації державної політики у сфері запобігання і реагування на надзвичайні ситуації, забезпечення цивільного захисту населення.

ЄДСЗР складається з постійно діючих функціональних і територіальних підсистем і має чотири рівні: загальнодержавний, регіональний, місцевий та об'єктовий.

Функціональні підсистеми створюються міністерствами та іншими центральними органами виконавчої влади для організації роботи, пов'язаної із запобіганням надзвичайним ситуаціям та захистом населення і територій від їх наслідків.

Кожний рівень ЄДСЗР має координуючі та постійні органи управління щодо розв'язання завдань у сфері запобігання надзвичайним ситуаціям, захисту населення і територій від їх наслідків, систему повсякденного управління, сили і засоби, резерви матеріальних та фінансових ресурсів, системи зв'язку та інформаційного забезпечення.

До складу ЄДСЗР входять відповідні сили і засоби функціональних і територіальних підсистем, а також недержавні (добровільні) рятувальні формування, які залучаються для виконання відповідних робіт.

У виняткових випадках, коли стихійне лихо, епідемія, епізоотія, аварія чи катастрофа ставить під загрозу життя і здоров'я населення і потребує термінового проведення великих обсягів аварійно-рятувальних і відновлювальних робіт, Президент України може залучати до виконання цих робіт у порядку, визначеному Законом України «Про надзвичайний стан», спеціально підготовлені сили і засоби Міноборони.

На базі існуючих спеціалізованих служб і підрозділів (будівельних, медичних, хімічних, ремонтних та інших) в областях, районах, населених пунктах, підприємствах, установах та організаціях утворюються позаштатні спеціалізовані формування, призначені для проведення конкретних видів невідкладних робіт у процесі реагування на надзвичайні ситуації. Ці формування проходять спеціальне навчання, періодично залучаються до участі у практичному відпрацюванні дій з ліквідації надзвичайних ситуацій.

У виконанні робіт, пов’язаних із запобіганням надзвичайним ситуаціям і реагуванням на них, можуть брати участь також добровільні громадські об'єднання за наявності у представників цих об'єднань відповідного рівня підготовки, підтвердженого в атестаційному порядку. Свої дії вони повинні узгоджувати з територіальними органами та уповноваженими з питань

надзвичайних ситуацій та цивільного захисту населення, а роботи виконувати під їх керівництвом.

Надзвичайний стан по всій території України або в окремих її місцевостях вводиться постановою Верховної Ради України з негайним повідомленням Президента України або Указом Президента України, який підлягає затвердженню Верховною Радою України.

Під час надзвичайного стану держава може вживати заходів, передбачених Законом «Про надзвичайний стан», відступаючи від своїх зобов'язань за Конституцією лише настільки, наскільки це вимагається гостротою стану, за умови, що такі заходи не є несумісними з іншими зобов'язаннями за міжнародним правом і не тягнуть за собою дискримінації на основі національності, мови, статі, релігії чи соціального походження.

З метою ліквідації наслідків надзвичайної ситуації у мирний час може проводитись цільова мобілізація. У виняткових випадках допускається залучення працездатного населення і транспортних засобів громадян для виконання невідкладних аварійно-рятувальних робіт за умови обов'язкового забезпечення безпеки праці. При Цьому забороняється залучення неповнолітніх, а також вагітних жінок до робіт, які можуть негативно вплинути на стан їхнього здоров'я.

3.1.2 Аварії з викидом шкідливих речовин

Аварії з викидом сильнодіючих отруйних речовин (СДОР) і зараженням навколишнього середовища виникають на підприємствах хімічної, нафтопереробної, целюлозно-паперової, м'ясо-молочної та харчової промисловості (що мають холодильні установки і які використовують у них в якості холодоагенту речовини типу аміак), водопровідних і очисних спорудах (що використовують хлор), а також при транспортуванні СДОР по залізниці і автомобільних дорогах [42].

Безпосередніми причинами викиду СДОР є порушення правил зберігання і транспортування, недотримання техніки безпеки, вихід з ладу агрегатів, механізмів, трубопроводів, ушкодження ємностей тощо. При цьому можливе ураження робітників, службовець і інших категорій населення, що знаходяться в районах викиду СДОР.

Сильнодіючими отруйними речовинами називаються хімічні сполуки, які в певних кількостях, що перевищують ГДК, шкідливо впливають на людей, сільськогосподарських тварин, рослини, викликаючи у них ураження різного ступеня.

СДОР можуть бути елементами технологічного процесу (аміак, хлор, сірчана й азотна кислоти, фтористий водень) і можуть утворюватись при пожежах на об'єктах господарства (оксид вуглецю, оксид азоту, хлористий водень, сірчистий газ).

У народному господарстві великого поширення набуло використання таких СДОР, як хлор (для знезараження води; при виробництві целюлози на виробництво 1 т целюлози потрібно 40 кг хлору), аміак (при виробництві добрив; як холодоагент у холодильних установках), сірководень, сірковуглець та ін.

Об'єкти, які виробляють СДОР, використовують їх у процесі виробництва, здійснюють їх зберігання, поділяють на 3 ступені хімічної небезпеки. Ступінь хімічної небезпеки визначається видом СДОР та його сумарною кількістю.

Шкідлива дія СДОР на організм людини може проявлятися як в результаті потрапляння таких речовин у краплиннорідкому вигляді на шкіру, так і в результаті вдихання їх парів. За токсичним властивостям СДОР в основному відносяться до групи речовин загально отруйної та задушливої дії. Симптомами отруєння ними в більшості випадків є головний біль, запаморочення, потемніння в очах, шум у вухах, наростаюча слабкість, задишка, нудота, блювота, а при сильних отруєннях - втрата свідомості, судоми і навіть смерть.

Внаслідок аварій із СДОР утворюється зона хімічного — зараження та осередок хімічного зараження.

Зона хімічного зараження СДОР включає територію, на яку поширюється хмара СДОР. Площі хімічного зараження СДОР визначаються напрямком і швидкістю вітру та іншими параметрами.

Осередок хімічного ураження включає територію, на якій відбулися масові ураження людей, тварин та рослин.

У населених пунктах, лісах, виробничих приміщеннях, підвалах і комунікаційних тунелях стійкість зараження СДОР буде вище, ніж на відкритій місцевості, оскільки вплив вітру проявляється у меншій мірі.

При цьому характер токсичної дії деяких СДОР може змінюватися, наприклад, пари хлору в концентрації 0,1-0,2 мг / л смерть людини можуть викликати при вдиханні їх протягом не менше як 1 години, а пари в концентрації 10-15 мг / л викличуть рефлекторну зупинку дихання при 1-2 вдихах. Деякі з СДОР при високих концентраціях їх парів здатні викликати шкірні ураження людини (з утворенням пухирів).

В першу чергу необхідно захистити органи дихання від подальшої дії сильнодіючих отруйних речовин. На потерпілого необхідно надіти протигаз або ватяну марлеву пов’язку, попередньо змочивши її при отруєнні хлором водою або 5 % розчином питної соди, а при отруєнні аміаком – водою або 2% розчином лимонної кислоти, і винести (вивести) його із зони ураження. У випадках отруєння сильнодіючими отруйними речовинами потерпілому забезпечити спокій і тепло. При виході із зони зараження вимийте очі і відкриті ділянки тіла і зверніться за допомогою до медичного працівника або до медичного закладу.

При отруєнні аміаком винесіть потерпілого з зони зараження, шкіряні покрови, очі, ніс, рот вимийте водою. В очі закапайте дві три краплі 30% розчину альбуциду, в ніс оливкове масло.

При отруєнні хлором винесіть потерпілого із зони зараження. При зупинці дихання зробіть штучне дихання. Шкіряні покрови, рот, ніс вимийте 5 % розчином питної соди або водою.

При отруєнні метаном винесіть потерпілого із зони зараження. При зупинці дихання зробіть потерпілому штучне дихання.

При отруєнні чадним газом винесіть потерпілого із зони зараження, розстебніть комірець одягу. При необхідності зробіть штучне дихання. При необхідності зверніться за допомогою до медичного працівника або відправте потерпілого в медичний заклад.

При отруєнні СДОР дійте у відповідності з отриманими розпорядженнями управління (відділу) з питань надзвичайних ситуацій та цивільного захисту населення області (міста обласного підпорядкування, району).

У системі цивільної оборони розроблена «Методика прогнозування масштабів зараження СДОР при аваріях». Методика дає змогу розрахувати можливу площу хімічного зараження та оцінити можливі втрати людей.

3.2 Основи охорони праці

Темою кваліфікаційної роботи є Розробка програми для створення та редагування ігрових моделей з використанням Autodesk 3D Max, Blue. При використанні ПЗ, яке є результатом даної розробки, як і при використанні будь-якого іншого ПЗ, необхідно дотримуватися рекомендацій та вимог з охорони праці. В цьому розділі пропоную розглянути основні нормативні документи, в яких зазначені вимоги до робочих місць та приміщень при використанні ПК.

При розробці системи потрібно запобігти негативному впливу виробничих факторів, а саме дотримуватися правил та норм щодо приміщень де знаходитиметься робоче місце, відповідно до «Правил охорони праці під час експлуатації електронно-обчислювальних машин» та ДСанПіН «Гігієнічні вимоги до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [43,44].

Відповідно до ДСанПіН робоче місце працівника не можна розміщувати у підвалах та цокольних поверхах. Площа робочого місця для виконання розробки повинна становити не менше 6 $м^2$, а об’єм 20 $м^3$. Приміщення, в якому працюють на ПК повинно бути забезпечене природнім та штучним освітлення згідно з

нормами ДБН. Природне освітлення здійснюється крізь світлові отвори, направлені на північний схід і вони забезпечують коефіцієнт природньої освітленості (КПО) не менший ніж 1,5%. Розраховується КПО за методикою, що міститься у ДБН. В приміщеннях де проводиться розробка та міститься ПК, штучне освітлення здійснюється системою загального рівномірного освітлення. Джерелом штучного освітлення слугують люмінесцентні лампи. Віконні отвори приміщення обладнані вертикальними жалюзями та зовнішніми козирками. У даному приміщенні функціонує система опалення. Відповідно до санітарних норм та правил у даному приміщенні знаходиться аптечка першої медичної допомоги. Також, у даному приміщенні щоденно проводиться вологе прибирання.

Згідно правил НПАОП обладнання та організація робочого місця працюючого з ПК має відповідати ергономічним вимогам, а саме:

- оптимальна робоча поза користувача ПК забезпечується конструкцією робочого місця;
- робоче місце розташоване так відносно світлових отворів, що природне світло падає збоку зліва;
- відповідно вимогам, робочий стілець є підйомно-поворотним з регульованою висотою;
- будова робочого стола відповідає вимогам ергономіки та дозволяє оптимально розміщувати на робочій поверхні ПК, допоміжне обладнання та документи.

Нормовані параметри мікроклімату, іонного складу повітря, вмісту шкідливих речовин мають відповідати вимогам Санітарних норм;

Мікрокліматичні умови приміщення повинні відповідати нормальним значенням таких показників:

- температура повітря;
- відносна вологість повітря;
- швидкість руху повітря;
- інтенсивність теплового (інфрачервоного) опромінення.

За ступенем впливу на тепловий стан людини мікрокліматичної умови поділяють на оптимальні та допустимі.

Для робочої зони приміщення встановлюються оптимальні та допустимі мікрокліматичні умови з урахуванням складності виконуваної роботи та періоду року. При виконанні роботи на ПК, що пов’язано з нервово-емоційним напруженням, у приміщенні потрібно підтримувати температура повітря $+22^0$С – $+24^0$С , відносну вологість 60-40%. Дані вимоги до параметрів мікроклімату містяться у санітарних нормах ДСН.

Оскільки розробка системи комплексної обробки подій здійснюються за ПК, необхідно проводити перерви по 15 хвилин через кожну годину. Якщо, виробничі обставини не дозволяють здійснювати часті перерви, тривалість роботи з ПК не повинна бути більшою чотирьох годин.

Відповідно до основних правил електробезпеки під час використання ПК потрібно дотримуватися таких вимог:

- при використанні ліній електромережі необхідно запобігти виникненню електричного джерела займання внаслідок короткого замикання чи перевантаження мережі;

- до електромережі ПК підключається тільки з а допомогою штепсельних з’єднань і електророзеток;

- спеціальні контакти для підключення нульового захисного провідника повинні міститися у штепсельних з’єднаннях та електророзетках. При цьому, під’єднання нульового захисного провідника відбувається швидше, ніж під’єднання фазового та нульового робочого провідника;

- підключення живлення ПК до двопровідної електромережі є недопустимим, навіть з використанням перехідних пристроїв. Тому здійснюється підключення по трьохпровідній мережі.

Згідно основних вимог до пожежної безпеки приміщення, у яких знаходиться ПК, мають бути оснащені переносними вуглекислотними або аерозольно-водопінними вогнегасниками. Підходи до засобів пожежогасіння повинні бути вільними. Також, згідно вимог НАПБ «Перелік однотипних за призначенням об'єктів, які підлягають обладнанню автоматичними установками пожежогасіння та пожежної сигналізації», у приміщенні де здійснюється робота з

ПК, робочі місця повинні бути обладнанні системою автоматичної пожежної сигналізації з димовим пожежним сповіщувачем.

Таким чином, в даному підрозділі виконано огляд основних законодавчих актів та нормативів при роботі з ПК. Так як, дипломна робота магістра спрямована на розробку системи комплексної обробки подій та її реалізація для сфери алгоритмічної торгівлі, то було наведено основні стандарти і правила щодо влаштування робочих місць де використовують комп’ютерну техніку. Також було наведено вимоги, які стосуються приміщення де знаходиться робоче місце працівника. Крім того, розглянуті санітарні норми та правила щодо мікроклімату у даному приміщенні. Наведенні правила електробезпеки під час роботи з ПК. Також поданні основні вимоги до пожежної безпеки приміщення. При дотриманні даних правил гарантуються безпечні умови праці на ПК та запобігання шкідливих виробничих факторів.

ВИСНОВКИ

В роботі представлено основні етапи створення програмного забезпечення довідникового змісту стосовно об’єктів визначеної території та систему вдосконаленого пошуку на основі довільних критеріїв. Сформульовано завдання та описано вимоги. Проаналізовано підходи та обрано оптимальний. Створено схему бази даних, для чого попередньо, виокремлено основні сутності, побудована концептуальну модель бази даних, UML діаграми класів, модель архітектури системи.

Сьогодні вже фактично всі побутового рівня речі можна значно швидше та ефективніше вирішити засобами зв’язку. Звичайно, фізичне спілкування для людини є незамінним, проте в час інтернету інформація між людьми розповсюджується дуже швидко, час, потрібний для комунікації скорочується, можливість прийняття рішення може бути втрачено. Інформація яка в цю хвилину вартує мільйони, через хвилину вже нікому може бити не цікава. Швидкість та вчасність сьогодні є ключовими критеріями успішного бізнесу.

Отже, інформацію можна отримувати, сприймати та розглядати по різному і одним з варіантів, який я пропоную розглянути в роботі детальніше є тип довідника, який би був більш інформативним та доступним в будь-який момент часу, також було б корисно завжди мати цей довідник при собі. Для того, щоб не множити ґаджети, адже це мінімум не зручно, такого типу потреби можна легко задовольнити, втіливши в життя відповідну програму, призначену для використання в смартфонах та планшетних комп’ютерах, при умові підтримання інформації в максимально актуальному стані. Зрештою, функціонал такої програми може надавати можливість і самим користувачам допомагати в цьому. Зараз майже кожен має смартфон під управлінням одної з популярних ОС – Android, iOS чи Windows Phone, які дозволяють встановлювати різноманітні сторонні програми. Очевидно, що такий довідник в кишені кожного більш ніж потрібен. Також варто закцентувати увагу на тому, яким чином повинен проводитись пошук в такого роду програмному забезпеченні. Він повинен

максимально задовольняти користувача. Для цього можна користуватися різного роду критеріями. Розробка такої системи пошуку потребує детального аналізу вибраної предметної області для виявлення лише доцільних критеріїв і відсіювання зайвих. В цьому можуть допомогти описи вже готових подібних програмних рішень пошукових систем та довідників. Отже, якщо програмний продукт буде мати зручний та інтуїтивний для пересічного користувача інтерфейс, адекватну швидкість роботи і час відгуку та буде працювати на більшості Android-пристроїв, то він буде користуватись популярністю і зможе бути однозначно корисним.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Дистанційний курс «Кваліфікаційні роботи бакалаврів» сайту дистанційного навчання ТНТУ [Електронний ресурс]. – Режим доступу: URL: https://dl.tntu.edu.ua/bounce.php?course=5329

2. Інформаційні технології видобутку даних (Data mining, високопродуктивні обчислення у складних системах): навчальний посібник ІВ Бойко, МР Петрик, Г Цуприк - 2020

3. Дискретні структури (Алгебраїчні та числові системи, комбінаторний аналіз) : навчально-методичний посібник для студентів спеціальності 121 «Інженерія програмного забезпечення», аспірантів та викладачів вищих навчальних закладів / Укладач : Бойко І.В., Петрик М.Р., Цуприк Г.Б. – Тернопіль : Тернопільський національний технічний університет імені Івана Пулюя, 2017 – 64 с.

4. Бойко І.В., М.Р. Петрик, Г.Б. Цуприк Моделювання та видобуток даних (висопродуктивні обчислення у великих алгебраїчних та числових системх, комбінаторному аналізі): навчальний посібник. Тернопіль: : ТНТУ 2019 – 62 с

5. Pasika DR, Tsupryk HB (2023) Rozrobka odnostorinkovoho veb-zastosunku z vykorystanniam VUE. JS i REACT: porivnialnyi analiz produktyvnostib korystuvatskoho dosvidu ta dotsilnosti [Developing a single-page web application using VUE. JS and REACT: a comparative analysis of user experience performance and expediency]. IMSTT (Tern., 13-14 December 2023), pp. 218-219

6. Офіційний сайт [Електронний ресурс] / Wikipedia – URL: https://www.android.com/

7. Електронна енциклопедія: Вікіпедія. — Режим доступу : http://uk. wikipedia.org/wiki/Android . — Android.

8. Електронна енциклопедія: Вікіпедія. — Режим доступу : http://uk. wikipedia.org/wiki/Java . — Java.

9. Електронна енциклопедія: Вікіпедія. — Режим доступу : http://uk. wikipedia.org/wiki/Eclipse . — Eclipse.

10. Android-development-tools-eclipse Електронний ресурс] – Режим доступу : URL : https://marketplace.eclipse.org/content/android-development-tools-eclipse

11. Список_версій_Android [Електронний ресурс] – Режим доступу : URL : https://uk.wikipedia.org/wiki/Список_версій_Android

12. Step-by-step-guide-to-Android-development-with-Eclipse [Електронний ресурс] – Режим доступу: URL: https://www.theserverside.com/tutorial/Step-by-step-guide-to-Android-development-with-Eclipse

13. Android SDK [Електронний ресурс] – Режим доступу: URL: http://androidfanclub.ru/programming/%D0%BE%D0%B1%D0%B7%D0%BE%D1%80-android-sdk .

14. Компоненти Android [Електронний ресурс] – Режим доступу: URL: http://developer.alexanderklimov.ru/android/android1.php.

15. Developer.android [Електронний ресурс] – Режим доступу: URL: https://developer.android.com/about/versions/11/setup-sdk

16. Developer.android [Електронний ресурс] – Режим доступу: URL: https://developer.android.com/about/versions/11/setup-sdk

17. Developer.android [Електронний ресурс] – Режим доступу: URL: https://developer.android.com/about/versions/11/setup-sdk

18. Instrumenti-dlja-profesijnoi-rozrobki-pid- android [Електронний ресурс] – Режим доступу: URL:https://www.zapptreinamentos.com.br/novo/2020/07/17/instrumenti-dlja-profesijnoi-rozrobki-pid-android/

19. Сутність_—_зв'язок [Електронний ресурс] – Режим доступу : URL : https://uk.wikipedia.org/wiki/Модель_«сутність_—_зв'язок»

20. James Rumbaugh, Ivar Jacobson, Grady Booch (1999). The unified modeling language reference manual (англ.). Addison Wesley Longman Inc. ISBN 0-201-30998-X.

21. Основні_поняття_моделі_«сутність_–_зв’язок» [Електронний ресурс] – Режим доступу: URL: https://wiki.cuspu.edu.ua/index.php/Модель_«сутність – зв’язок». Основні поняття моделі «сутність – зв’язок»: сутності, зв’язки, атрибути та їх класифікація

22. Життєвий_цикл_програмного_забезпечення [Електронний ресурс] – Режим доступу: URL: https://uk.wikipedia.org/wiki/ Життєвий_цикл_програмного_забезпечення

23. Unified_Modeling_Language [Електронний ресурс] – Режим доступу: URL: https://uk.wikipedia.org/wiki/Unified_Modeling_Language

24. Armstrong, Deborah J. (February 2006). The Quarks of Object-Oriented Development. Communications of the ACM. **49** (2): 123—128. ISSN 0001-0782. Процитовано 8 серпня 2006.

25. Meyer, Bertrand (1997). Object-Oriented Software Construction[en]. Prentice Hall. ISBN 0-13-629155-4.

26. Організація баз даних: практичний курс: Навч. посіб. для студ. / А. Ю. Берко, О. М. Верес; Нац. ун-т «Львів. політехніка». — Л., 2003. — 149 с.

27. Резніченко, В.А. (2021). 60 років базам даних. Проблеми програмування (№ 3). doi:10.15407/pp2021.03.040. ISSN 1727-4907

28. «An Introduction to Database Systems» C. J. Date. ISBN 0-321-19784-4

29. Введення в нормалізацію баз даних [Електронний ресурс] – Режим доступу: URL: https://web.archive.org/web/20110606025027/http://dev.mysql.com/tech-resources/articles/intro-to-normalization.html

30. Базові знання з нормалізації баз даних [Електронний ресурс] – Режим доступу: URL: https://www.lifewire.com/database-normalization-basics-1019735

31. Нормалізація [Електронний ресурс] – Режим доступу: URL: https://web.archive.org/web/20100106115112/http://www.utexas.edu/its/archive/windows/database/datamodeling/rm/rm7.html

32. Підручник з Umbrello UML Modeller [Електронний ресурс] – Режим доступу: URL: https://docs.kde.org/stable5/uk/umbrello/umbrello/index.html

33. Архітектура програмного забезпечення [Електронний ресурс] – Режим доступу: URL: https://wezom.com.ua/ua/blog/arhitektura-programmnogo-obespecheniya

34. PHP [Електронний ресурс] – Режим доступу: URL: http://php.net/manual/ua/.

35. Бйорн Страуструп . Мова програмування С++. Короткий курс - 2019. - 320 с. - ISBN 978-5-907144-12-5 .

36. Бйорн Страуструп . Програмування: принципи та практика з використанням С++. - 2016. - 1328 с. - ISBN 978-5-8459-1949-6 .

37. Інфологічна модель даних [Електронний ресурс] – Режим доступу: URL: https://studfile.net/preview/8965876/page:2/

38. Офіційна сторінка [Електронний ресурс] – Режим доступу: URL: https://www.mysql.com/

39. Kaner, Cem; Falk, Jack; Nguyen, Hung Quoc (1999). Testing Computer Software, 2nd Ed. New York, et al: John Wiley and Sons, Inc. c. 480. ISBN 0-471-35846-0.

40. Розробка автоматизованої інформаційної системи обробки даних / Казмірчук В.М., Цуприк Г.Б./ Матеріали VIII Науково-технічної конференції «Інформаційні моделі, системи та технології» – Тернопіль 9-10 грудня 2020. — Т. : ТНТУ, 2020. — С. 148.

41. Запобігання виникненню надзвичайних ситуацій [Електронний ресурс] / Wikipedia – URL: https://uk.wikipedia.org/wiki/Запобігання_виникненню_надзвичайних_ситуацій_характеру.

42. Ураження сильнодіючими отруйними речовинами [Електронний ресурс] / URL: http://www.wikidocs.ru/preview/44941.

43. Дистанційний курс «Основи охорони праці» сайту дистанційного навчання ТНТУ [Електронний ресурс]. – Режим доступу: URL: http://dl.tntu.edu.ua/index.php

44. Охорона праці : Навчальний посібник [Текст] Геврик Є.О. - Ельга: Ніка-Центр. - 2003. – 279 с.

45. Методичні вказівки до виконання дипломної роботи освітнього рівня ―бакалавр студентами усіх форм навчання для напряму підготовки 121 – Інженерія програмного забезпечення‖ / Укладачі : Петрик М.Р., Михалик Д.М., Кінах Я.І., Гладьо С.В., Цуприк Г.Б. – Тернопіль : Вид-во ТНТУ імені Івана Пулюя, 2016 – 28 с.

ДОДАТКИ

Додаток А. Тези конференції

комп'ютерних систем, SCADA-системи мають інші вимоги до безпеки та низькі обчислювальні можливості. Крім того, механізми безпеки не завжди враховуються в специфікаціях пристрою або протоколу, потенційно через високу вартість реалізації або низьку обчислювальну здатність пристрою. Ще одним важливим фактором безпеки є те, що життєвий цикл SCADA-систем набагато довший, ніж стандартних комп'ютерних систем. Забезпечення безпеки для величезної кількості мережевих пристроїв, які часто розгорнуті в широких географічних зонах, є складним завданням, яке потребує вивчення та розробки сценарію реагування на кіберінциденти в SCADA системах. Крім того, фізичний доступ до цих пристроїв може бути необмеженим, що може дозволити зловмиснику скомпрометувати всю мережу.

Література

1. Pliatsios, D., Sarigiannidis, P., Lagkas, T. and Sarigiannidis, A.G. (2020) A Survey on SCADA Systems: Secure Protocols, Incidents, Threats and Tactics. Communications Surveys and Tutorials, 22, pp. 1942-1976.

УДК 004.41

В.Ю.Шевчук, Є.Б.Яворська, к.т.н., доц..
Тернопільський національний технічний університет імені Івана Пулюя, Україна

ПРЕДМЕТНО-ОРІЄНТОВАНИЙ ПІДХІД ЯК ОПТИМАЛЬНИЙ ДЛЯ РОЗРОБКИ СУЧАСНИХ, МОБІЛЬНИХ ОБ'ЄКТНО-ОРІЄНТОВАНИХ ІНФОРМАЦІЙНИХ СИСТЕМ

V.Yu.Shevchuk, Ye.B.Yavorska, Ph.D., Assoc. Prof.

OBJECT-ORIENTED APPROACH AS OPTIMAL FOR DEVELOPING MODERN, MOBILE OBJECT-ORIENTED INFORMATION SYSTEMS

Кожне покоління інше, всі більше зусиль прикладається на автоматизацію та зменшення фізичної праці, натомість вже почалась ера роботизованої «розумної» техніки. Моє покоління за цифровою обізнаністю та можливостями далеко перевершує вже навіть попереднє, натомість тут я бачу і проблему, коли людина не має можливості скористуватись телефоном чи іншим ґаджетом вона стає безпомічною. Не дивлячись ні на що в цьому я вважаю і один з моментів прогресу, і на часі нові винаходи, коли вже не потрібно буде мати фізично навіть телефон чи інший ґаджет. Проте, якого б рівня техніка не була, як би тісно штучний інтелект не зайшов в наше життя, я вважаю, що ІТ спеціальності як і всі, де, окрім знань, потрібна креативністсь, кмітливість, винахідливість, тобто навики притаманні саме інженерам, ще довго будуть актуальними і роботонезамінними в повній мірі.

Все що я роблю завжди пов'язано з розробками, що дають змогу економити один із основних ресурсів – час, адже то є важко, а подекуди не відновлюваний ресурс і його заощадження – це в першу чергу можливість дослідити щось нове, чи просунутись вперед, розвиватись.

Розробники інформаційних систем, доволі часто, на початку своєї роботи, вагаються який підхід обрати, і з різних міркувань, буває, що приймають рішення не на користь об'єктно-орієнтованого підходу, зумовлюючи свій вибір відповідною складністю в тому числі при моделюванні. Однак, реалізації механізмів з врахуванням парадигм об'єктно-орієнтованого підходу має значно більше переваг ніж недоліків, важливими з яких є швидкість і надійність, а також можливість для гнучкої модифікації. І саме це і стало вирішальним при побудові систем інформаційного типу вже як окремого напрямку, який виділився та інтенсивно розвивається.

В загальному, для отримання успішного, очікуваного результату, варта зазначити, що при застосуванні, для реалізації, об'єктно-орієнтованого підходу задачу слід поділити на етапи, а саме: етап аналізу вимог, створення моделі роботи (моделювання важливий етап, оскільки спрощує розробку, аналіз та реалізацію її оскільки робиться акцент на основні особливості розроблюваного продукту). Таким чином, модель є адекватною щодо представлення структури проєктованої системи, а також визначальною по функціональним залежностям поміж об'єктами в цілому, етапом розробки архітектури системи, етапом проєктування програмних модулів і бібліотек, етапом реалізації вже безпосередньо проєкту, етапами тестування і впровадження системи.

Якщо говорити про об'єктну модель, то за її допомогою представляється проектований додаток як сукупність об'єктів, і кожен з цих об'єктів є абстракцією або чимось з окресленими границями, які значущі в контексті задачі. Кожен об'єкт, в свою чергу, відповідає якимось особливостям, а також реалізовує специфічні функції. В моделі представляються об'єкти, які є важливими в розроблюваному додатку, якими власне і визначається прагматика досліджуваної системи.

Отже, метою розробки об'єктної моделі переслідується опис елементів, які становлять проєктовану систему та встановлення взаємозалежностей між ними.

Також, аналіз основних підходів до проєктування програмних засобів на основі обраного об'єктно-орієнтованого підхода, дав змогу зробити висновки про те, що застосування його парадигм без особливих складностей дає змогу створити систему на базі запропонованої моделі і забезпечити механізм її модифікації через розширення функцій або зміни алгоритму візуалізації. Завдячуючи оригінальності структури також можна запропонувати підходи, які дають можливість реалізовувати ширші функціональні можливості. Також, такий підхід є досить простим у використанні, високофункціональним під час роботи з даними та забезпечує і високу гнучкість, і легку розширюваність в процесі створення складних спеціалізованих систем.

Література

1. Armstrong, Deborah J. (February 2006). The Quarks of Object-Oriented Development. Communications of the ACM. 49 (2): 123—128. ISSN 0001-0782

2. Meyer, Bertrand *(1997)*. Object-Oriented Software Construction[en]. *Prentice Hall*. ISBN 0-13-629155-4.

3. Methods of constructing algorithms for comparative test statistical verification of mathematical models of bioobject responses to low-intensity stimuli / Bohdan Yavorskyy, Evhenia Yavorska, Halyna Tsupryk, Roman Kinash // Scientific Journal of TNTU. — Tern.: TNTU, 2023. — Vol 112. — No 4. — P. 82–90.

Додаток Б. Диск з роботою