Форма № Н-9.02

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ
імені адмірала Макарова

Херсонський навчально-науковий інститут

Кафедра інформаційних технологій та фізико-математичних дисциплін

"Допущений до захисту"
Завідувач кафедри
д.т.н., доцент Гучек П.Й.

"16" грудня 2025 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи

на здобуття ступеня вищої освіти "магістр"

на тему: «Дослідження методів багатокритеріального вибору та розробка системи підтримки прийняття рішень для вибору технологічного стеку вебзастосунку»

Виконала: студентка VI курсу, групи 6157зм спеціальності
122 – "Комп'ютерні науки", ОПП – "Інформаційні управляючі системи та технології"
Чорна Л.О.
(прізвище та ініціали)

Керівник к.ф.-м.н., доцент Латанська Л.О.
(прізвище та ініціали)

Рецензент к.т.н., доцент Макарова Л.М.
(прізвище та ініціали)

Херсон - 2025 р.

Форма № Н-9.01

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ
імені адмірала Макарова
ХЕРСОНСЬКИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ

Кафедра інформаційних технологій та фізико-математичних дисциплін
Освітній ступень магістр
Галузь знань 12 – "Інформаційні технології"
(шифр і назва)
Спеціальність 122 – " Комп'ютерні науки"
(шифр і назва)
Освітньо-професійна програма "Інформаційні управляючі системи та технології"

ЗАТВЕРДЖУЮ
Гарант освітньої програми
_____ Гучек П.Й.
"23" жовтня 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття ступеня вищої освіти «магістр»

Студенту Чорній Лілії Олександрівні
(прізвище, ім'я, по батькові)
1. Тема кваліфікацій роботи: «Дослідження методів багатокритеріального вибору та розробка системи підтримки прийняття рішень для вибору технологічного стеку вебзастосунку»
Керівник роботи Латанська Л.О., к.ф.-м.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Тема затверджена розпорядженням по інституту від "17" жовтня 2025 року № 20
2. Термін подання студентом роботи 12.12.2025 р.
3. Вихідні дані до роботи:

4. Зміст та структура розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):

4.1 Титульний аркуш, завдання на магістерську роботу, анотація (українською, російською, англійською), зміст, перелік умовних означень, символів, одиниць та термінів (при необхідності).
4.2 Вступ.
4.3 Дослідження технологій проєктування інформаційних систем та аналіз їх особливостей.
4.4 Формування вимог до інформаційної системи, розробка її концепції і постановка задачі.
4.5 Розробка проєктних рішень по інформаційній системі
4.6 Реалізація проєкту інформаційної системи (технічна документація по проєкту).
4.7 Розділ з економіки
4.8 Розділ з охорони праці та безпеки в надзвичайних ситуаціях (охорони оточуючого середовища).
4.9 Висновки.
4.10 Список використаної літератури.
4.11 Додатки (технічне завдання, загальний опис ІС, інструкція користувача, текст програми).

5. Перелік презентаційних матеріалів

5.1	Актуальність теми. 5.2 Мета і завдання дослідження.
5.3	Об’єкт і предмет дослідження. 5.4 Методи дослідження.
5.5	Практичне значення одержаних результатів.
5.6	Апробація результатів досліджень та публікації.
5.7	Аналіз сучасного стану проблеми вибору технологічного стеку вебзастосунків.
5.8	Методи багатокритерійного вибору. 5.9 Метод аналізу ієрархій (MAI).
5.10	Вимоги до функціональних характеристик. 5.11 Загальносистемні рішення.
5.12	Рішення з інформаційного забезпечення.
5.13	Рішення з програмного забезпечення. Діаграма класів.
5.14	Рішення з програмного забезпечення. Діаграма компонентів.
5.15	Рішення з програмного забезпечення. Засоби розробки. 5.16 Екранні форми.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
4.7	Ломоносов Дмитро Анатолійович		
4.8	Щедролосєв Олександр Вікторович		

7. Дата видачі завдання 27 жовтня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк виконання етапів роботи	Примітка
1.	Вивчення і аналіз предметної галузі	28.10.2025	
2.	Розробка концепції ІС	30.10.2025	
3.	Розробка проєктних рішень по ІС	31.10.2025	
4.	Реалізація проєкту, розробка робочої документації	03.11.2025	
5.	Вирішення питань з економіки	06.11.2025	
6.	Вирішення питань охорони праці та безпеки в надзвичайних ситуаціях (охорони оточуючого середовища)	12.11.2025	
7.	Виконання графічної частини роботи	22.11.2025	
8.	Оформлення пояснювальної записки	06.12.2025	
9.	Подання роботи на попередній захист	12.12.2025	
10.	Подання роботи рецензенту	20.12.2025	
11.	Подання роботи на державний захист	29.12.2025	

Студент _____ (Підпис) Чорна Л.О. (Прізвище та ініціали)

Керівник роботи _____ (Підпис) Латанська Л.О. (Прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота присвячена дослідженню методів багатокритеріального вибору та розробці системи підтримки прийняття рішень для вибору технологічного стеку вебзастосунку.

У сучасних умовах стрімкого розвитку вебтехнологій вибір оптимального технологічного стеку для розробки вебзастосунків є складним і багатокритеріальним завданням. Існує велика кількість мов програмування, фреймворків, бібліотек та інструментів, які відрізняються за продуктивністю, масштабованістю, безпекою, складністю підтримки, вартістю розробки та іншими характеристиками. Прийняття рішення щодо вибору технологічного стеку часто ґрунтується на суб’єктивному досвіді розробників, що може призводити до неефективних або ризикованих рішень.

У зв’язку з цим актуальним є застосування формалізованих методів багатокритеріального вибору, які дозволяють враховувати кількісні і якісні критерії та зменшувати вплив суб’єктивного чинника

Робота представлена на 92 сторінках друкованого тексту та містить 8 рисунків, 13 таблиць, 5 додатків та список використаної літератури з 17 найменувань.

Ключові слова: система підтримки прийняття рішень, технологічний стек, вебзастосунок, альтернативи, методи багатокритеріального вибору.

ABSTRACT

The qualification work is devoted to the study of multi-criteria selection methods and the development of a decision support system for choosing a technological stack for a web application.

In today's conditions of rapid development of web technologies, choosing the optimal technological stack for developing web applications is a complex and multi-criteria task. There are a large number of programming languages, frameworks, libraries and tools that differ in performance, scalability, security, complexity of support, development cost and other characteristics. Making a decision on choosing a technological stack is often based on the subjective experience of developers, which can lead to ineffective or risky decisions.

In this regard, the application of formalized multi-criteria selection methods is relevant, which allow taking into account quantitative and qualitative criteria and reducing the influence of the subjective factor.

The work is presented on 92 pages of printed text and contains 8 figures, 13 tables, 5 appendices, and a list of 17 references.

Keywords: decision support system, technology stack, web application, alternatives, methods of multi-criteria choice.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасних умовах стрімкого розвитку вебтехнологій вибір оптимального технологічного стеку для розробки вебзастосунків є складним і багатокритеріальним завданням. Існує велика кількість мов програмування, фреймворків, бібліотек та інструментів, які відрізняються за продуктивністю, масштабованістю, безпекою, складністю підтримки, вартістю розробки та іншими характеристиками. Прийняття рішення щодо вибору технологічного стеку часто ґрунтується на суб'єктивному досвіді розробників, що може призводити до неефективних або ризикованих рішень [1–3].

У зв'язку з цим актуальним є застосування формалізованих методів багатокритеріального вибору, які дозволяють враховувати кількісні і якісні критерії та зменшувати вплив суб'єктивного чинника.

Мета і завдання дослідження. Метою роботи є дослідження методів багатокритеріального вибору та розробка системи підтримки прийняття рішень для вибору технологічного стеку вебзастосунків шляхом порівняння альтернативних рішень за сукупністю наявних критеріїв.

Завдання дослідження:

– Проаналізувати сучасний стан проблеми вибору технологічного стеку вебзастосунків.

– Дослідити методи багатокритеріального вибору альтернатив.

– Обрати метод багатокритеріального вибору для розробки системи підтримки прийняття рішень.

– Виконати постановку задачі на розробку системи підтримки прийняття рішень для вибору технологічного стеку вебзастосунку.

– Розробити систему підтримки прийняття рішень для вибору технологічного стеку вебзастосунку.

Об'єктом дослідження є процес прийняття рішень щодо вибору технологічного стеку вебзастосунків.

Предметом дослідження є методи багатокритеріального вибору, які використовуються для обґрунтованого вибору технологічного стеку вебзастосунків.

Методи дослідження. При дослідженні використовувалися методи теорії прийняття рішень, математичної статистики, математичного моделювання, об’єктно-орієнтованого програмування.

Практичне значення одержаних результатів. Практичне значення одержаних результатів полягає у розробці системи підтримки прийняття рішень для вибору технологічного стеку вебзастосунку.

Апробація результатів дослідження. Основні положення і результати досліджень пройшли апробацію на VIII Всеукраїнській науково-практичній інтернет конференції студентів, аспірантів та молодих вчених «СУЧАСНІ ІНФОРМАЦІЙНІ СИСТЕМИ І ТЕХНОЛОГІЇ» (24 листопада 2025 р., м.Херсон, м.Хмельницький).

Публікації. За результатами досліджень опубліковано одну наукову працю – тези конференції.

1 ДОСЛІДЖЕННЯ МЕТОДІВ БАГАТОКРИТЕРІАЛЬНОГО ВИБОРУ ТА ПОСТАНОВКА ЗАДАЧІ НА РОЗРОБКУ СППР ДЛЯ ВИБОРУ ТЕХНОЛОГІЧНОГО СТЕКУ ВЕБЗАСТОСУНКУ

1.1 Дослідження методів багатокритеріального вибору альтернатив

Задачі вибору оптимального рішення за наявності кількох, часто суперечливих критеріїв, є типовими для інформаційних технологій. У процесі прийняття рішень щодо вибору технологій, архітектурних рішень або програмних платформ необхідно враховувати сукупність технічних, економічних та експлуатаційних показників, що зумовлює доцільність застосування методів багатокритеріального аналізу.

Методи багатокритеріального вибору альтернатив спрямовані на формалізацію процесу прийняття рішень шляхом упорядкування множини альтернатив відповідно до заданих критеріїв та їхньої відносної важливості. Основною перевагою таких методів є можливість урахування кількох показників одночасно та зменшення суб’єктивного впливу особи, яка приймає рішення.

Серед найбільш поширених методів багатокритеріального вибору альтернатив можна виділити метод зваженої суми, метод ELECTRE, метод TOPSIS, метод PROMETHEE, а також метод аналізу ієрархій (MAI). Кожен із зазначених методів має свої особливості, переваги та обмеження, що визначають сферу його практичного застосування [4–7].

Метод зваженої суми ґрунтується на нормалізації значень критеріїв та обчисленні інтегрального показника як зваженої суми часткових оцінок. Попри простоту реалізації, даний метод вимагає точного задання вагових коефіцієнтів та не завжди коректно відображає взаємозв’язки між критеріями.

Методи сімейства ELECTRE та PROMETHEE належать до так званих методів переваг і дозволяють здійснювати попарне порівняння альтернатив. Вони є ефективними для складних задач прийняття рішень, проте характеризуються

відносною складністю інтерпретації результатів та потребують значної кількості вхідних параметрів.

Метод TOPSIS базується на визначенні альтернативи, яка має мінімальну відстань до ідеального рішення та максимальну відстань до антиідеального. Цей підхід є наочним та широко застосовується в інженерних задачах, однак вимагає попередньої нормалізації даних і чутливий до вибору методу нормалізації.

Особливе місце серед методів багатокритеріального вибору займає метод аналізу ієрархій, запропонований Т. Сааті. Даний метод дозволяє представити задачу прийняття рішення у вигляді ієрархічної структури, що складається з мети, критеріїв та альтернатив. Основною перевагою MAI є можливість урахування як кількісних, так і якісних критеріїв шляхом попарних порівнянь та подальшого обчислення вагових коефіцієнтів.

Метод аналізу ієрархій передбачає перевірку узгодженості суджень експерта, що підвищує надійність отриманих результатів. Завдяки цьому MAI широко застосовується у задачах вибору технологічних стеків, оцінювання програмних рішень, планування проєктів та розробки систем підтримки прийняття рішень.

У багатьох практичних задачах прийняття рішень значна частина критеріїв має якісний або експертний характер. Такі критерії складно формалізувати за допомогою чітких числових значень, що зумовлює доцільність застосування методів багатокритеріального вибору, заснованих на апараті нечіткої логіки [8].

Методи нечіткого багатокритеріального аналізу базуються на теорії нечітких множин і дозволяють оперувати лінгвістичними змінними, такими як «високий рівень», «середня складність», «низька надійність» тощо. Це забезпечує більш адекватне відображення невизначеності та суб’єктивності експертних оцінок у процесі прийняття рішень.

До найбільш поширених підходів належать методи нечіткого логічного вибору, методи на основі нечіткого відношення переваги, а також нечіткі модифікації класичних багатокритеріальних методів, наприклад нечіткий ELECTRE.

Методи нечіткого логічного вибору ґрунтуються на використанні нечітких правил типу *IF-THEN*, які формують базу знань експертної системи. У таких методах альтернативи оцінюються на основі набору нечітких продукційних правил, а результат вибору визначається шляхом агрегування ступенів істинності відповідних правил. Перевагою даного підходу є висока інтерпретованість результатів, однак складність побудови та валідації бази правил обмежує його застосування у задачах з великою кількістю альтернатив і критеріїв.

Методи, засновані на нечіткому відношенні переваги, передбачають побудову матриці попарних порівнянь альтернатив, елементи якої задаються у вигляді нечітких чисел або функцій належності. Такі методи дозволяють формалізувати ступінь переваги однієї альтернативи над іншою в умовах неповної або неточної інформації. Подальший вибір оптимальної альтернативи здійснюється на основі аналізу домінування, індексів переваги або агрегованих показників.

Окрему групу становлять нечіткі модифікації класичних методів багатокритеріального вибору, зокрема нечіткий метод аналізу ієрархій (Fuzzy MAI). У даному підході парні порівняння задаються нечіткими трикутними або трапецієподібними числами, що дозволяє зменшити жорсткість експертних суджень і підвищити стійкість результатів до невизначеності. Аналогічним чином реалізуються нечіткі версії методів TOPSIS та ELECTRE, у яких етапи нормалізації та агрегування виконуються з використанням нечітких операцій [8].

Застосування нечітких методів багатокритеріального вибору є особливо доцільним у задачах оцінювання якості програмного забезпечення, вибору технологічних платформ і архітектурних рішень, коли частина критеріїв має експертний, евристичний або слабо формалізований характер. Водночас складність математичного апарату та підвищені вимоги до обчислювальних ресурсів можуть ускладнювати їх інтеграцію у прикладні системи підтримки прийняття рішень.

Таким чином, нечіткі методи багатокритеріального вибору є ефективним інструментом для роботи в умовах невизначеності, однак у межах даної кваліфікаційної роботи перевага надається класичному методу аналізу ієрархій,

який забезпечує достатній рівень формалізації, простоту реалізації та прозорість інтерпретації результатів, що є важливим для розробки практично орієнтованих інформаційних систем.

1.2 Опис методу аналізу ієрархій

Для вибору кращої альтернативи в даній роботі буде використовуватися метод аналізу ієрархій (MAI). Розглянемо суть даного методу.

MAI – це структурована методика для прийняття складних рішень, заснована на математиці та психології. Вона використовується, коли потрібно вибрати найкращий варіант з кількох альтернатив, враховуючи множину (часто конфліктуючих) критеріїв.

Розглянемо спрощений алгоритм методу:

1) Побудова ієрархії: Проблема розкладається на ієрархію: ціль (найвищий рівень), критерії та підкритерії (середній рівень), альтернативи рішень (нижчий рівень).

2) Парні порівняння: Експерт порівнює елементи одного рівня попарно, оцінюючи, наскільки один важливіший за інший за шкалою відносної важливості Сааті) [7].

Шкала відносної важливості представлена в таблиці 1.1.

Таблиця 1.1 – Шкала відносної важливості Сааті

Значення	Вербальна інтерпретація	Пояснення
1	**2**	**3**
1	Рівна важливість	Обидва елементи мають однаковий вплив на досягнення мети
3	Помірна перевага	Один елемент незначно переважає інший
5	Сильна перевага	Один елемент істотно важливіший за інший

Кінець таблиці 1.1

1	2	3
7	Дуже сильна перевага	Перевага одного елемента є домінуючою
9	Абсолютна перевага	Один елемент має надзвичайно високу важливість
2, 4, 6, 8	Проміжні значення	Використовуються для компромісних оцінок між основними рівнями
1/3, 1/5, 1/7, 1/9	Обернені значення	Використовуються, якщо другий елемент переважає перший

Розглянемо приклад заповнення матриці попарних порівнянь.

Припустимо, ми порівнюємо три критерії: Вартість (В), Якість (Я), Термін (Т).

Експерт вважає, що:

– Якість *помірно важливіша* за Вартість → 3.

– Якість *сильно важливіша* за Термін → 5.

– Вартість *трохи важливіша* за Термін → 2 (проміжне значення між 1 і 3).

Тоді матриця (рядки: що порівнюється, стовпці: з чим порівнюється) буде мати такий вигляд (Таблиця 1.2):

Таблиця 1.2 – Матриця попарних порівнянь

	Вартість	Якість	Термін
Вартість	1	1/3	2
Якість	3	1	5
Термін	1/2	1/5	1

Пояснення таблиці:

– Діагональ завжди 1 (елемент порівнюється сам із собою).

– Значення, яке симетричне відносно діагоналі, є оберненими (3 та 1/3, 5 та 1/5, 2 та 1/2).

Ця матриця є вихідними даними для подальшого математичного розрахунку вектору пріоритетів у методі MAI (власний вектор, що відповідає максимальному власному значенню).

3) Обчислення пріоритетів: На основі цих порівнянь для кожного елемента (критерію, альтернативи) обчислюється числова «вага» (пріоритет), що показує його відносну важливість.

4) Синтез результатів: Ваги альтернатив множаться на ваги критеріїв, що дозволяє отримати загальну оцінку (пріоритет) для кожної альтернативної опції та вибрати найкращу.

Метод аналізу ієрархій використовується для розв’язання ряду задач, як наприклад:

– Стратегічне планування та вибір напрямів розвитку організацій.

– Прийняття управлінських рішень в умовах багатокритеріальності та невизначеності;

– Оцінювання та ранжування альтернатив за сукупністю якісних і кількісних критеріїв;

– Вибору оптимальних проєктів, технологій або інвестиційних рішень;

– Розподілу ресурсів між конкуруючими варіантами;

– Аналізу ризиків та пріоритизації факторів впливу;

– Підтримки прийняття рішень у технічних, економічних та соціально-економічних системах.

1.3 Аналіз сучасного стану проблеми вибору технологічного стеку вебзастосунків

У сучасних умовах розвитку інформаційних технологій вибір технологічного стеку вебзастосунку є складною багатокритеріальною задачею, від якої істотно залежать якісні характеристики програмного продукту, зокрема його надійність, безпека, масштабованість та придатність до довгострокової

експлуатації. Зростання кількості мов програмування, фреймворків і серверних платформ призводить до ускладнення процесу прийняття рішень на етапі проєктування вебсистем.

Аналіз сучасних публікацій і практичних рішень у галузі веброзробки свідчить, що найбільш поширеними підходами до створення серверної частини вебзастосунків є технології, засновані на платформах Java, .NET, Python та JavaScript. Кожна з цих платформ має власні архітектурні особливості, інструментальну підтримку та область доцільного застосування, що зумовлює необхідність їх порівняльного аналізу з позицій ключових критеріїв якості.

За результатами проведеного аналізу в роботі виділено такі альтернативні технологічні стеки для подальшого багатокритеріального оцінювання [1-3, 9]:

–A_1: Java + JSP (Servlet API) – класичний підхід до побудови серверної частини вебзастосунків, що характеризується високою стабільністю, зрілістю та широким застосуванням у корпоративних системах.

– A_2: Java + Spring Boot + Thymeleaf – сучасний Java-орієнтований стек, який забезпечує швидку розробку, зручну конфігурацію та інтеграцію з іншими корпоративними сервісами.

– A_3: .NET (ASP.NET Core) – кросплатформне рішення для веброзробки з високою продуктивністю та розвиненими засобами безпеки.

– A_4: Python (Django) – фреймворк високого рівня абстракції, орієнтований на швидку розробку та підтримку, з широкими можливостями розширення.

– A_5: JavaScript (Node.js + Express) – подієво-орієнтована платформа, що забезпечує ефективну обробку великої кількості одночасних запитів і популярна у розробці масштабованих вебсервісів.

Для обґрунтованого порівняння наведених альтернатив було сформовано систему критеріїв оцінювання, яка відображає основні вимоги до корпоративних і інформаційно-управляючих вебзастосунків:

– C_1: Надійність та стабільність, що характеризує здатність системи до безвідмовної роботи протягом тривалого часу.

– C_2: Безпека, яка визначає рівень захищеності вебзастосунку від типових загроз та атак.

– C_3: Масштабованість, що відображає можливість ефективного функціонування системи за зростання навантаження.

– C_4: Зрілість та підтримка екосистеми, яка включає наявність документації, інструментів, спільноти та довгострокову підтримку технологій.

– C_5: Простота супроводу, що характеризує трудомісткість внесення змін, розширення та підтримки програмного забезпечення.

Таким чином, сучасний стан проблеми вибору технологічного стеку вебзастосунків характеризується наявністю кількох конкурентних альтернатив, оцінювання яких потребує застосування формалізованих методів багатокритеріального аналізу. Це обґрунтовує доцільність використання методів підтримки прийняття рішень, зокрема методу аналізу ієрархій, для вибору оптимального технологічного стеку вебзастосунку.

У даній роботі кожен технологічний стек розглядається як альтернатива у задачі багатокритеріального вибору та подається у вигляді формалізованої моделі, що описується сукупністю критеріїв.

1.4 Постановка задачі на розробку СППР для вибору технологічного стеку вебзастосунку

Виходячи з вище викладеного, сформулюємо наступну постановку задачі. В даній кваліфікаційній роботі необхідно розробити систему підтримки прийняття рішень для вибору технологічного стеку вебзастосунку.

Для реалізації мети СППР повинна забезпечувати можливість виконання нижче перерахованих функцій:

- додавання оцінки відношення за критерієм;
- видалення оцінки відношення за критерієм;
- перегляд оцінок відношення за критерієм;
- додавання критеріїв;
- видалення критеріїв;
- перегляд критеріїв;
- додавання моделі (технологічного стеку);

- видалення моделі (технологічного стеку);
- перегляд моделей (технологічних стеків);
- додавання оцінки важливості критеріїв;
- видалення оцінки важливості критеріїв;
- перегляд оцінок важливості критеріїв;
- перегляд кінцевих результатів.

Для забезпечення коректної та стабільної роботи програмного продукту рекомендовано використовувати апаратні засоби з такими характеристиками:

- роздільна здатність екрана – не менше 1366×768 px;
- обсяг оперативної пам’яті – не менше 2 ГБ;
- підключення до мережі Інтернет зі швидкістю не менше 10 Мбіт/с.

Програмний продукт висуває такі мінімальні вимоги до програмного середовища користувача:

– операційна система: Microsoft Windows 7 та вище або сучасні дистрибутиви Linux;

– інтернет-браузер: Google Chrome, Mozilla Firefox, Microsoft Edge або Safari – останні стабільні версії.

Детальні вимоги до СППР визначені у документі «Технічне завдання», який наведено в додатку А.

2 РОЗРОБКА ПРОЄКТНИХ РІШЕНЬ ДЛЯ СППР ВИБОРУ ТЕХНОЛОГІЧНОГО СТЕКУ ВЕБЗАСТОСУНКУ

2.1 Загальносистемні рішення для СППР

2.1.1 Вибір засобів моделювання для СППР

Проєктування системи підтримки прийняття рішень є складним етапом розробки, що потребує формалізованого опису структури системи, її функціональних можливостей, поведінки та взаємодії між компонентами. З метою забезпечення наочності, однозначності інтерпретації та узгодженості проєктних рішень у даній роботі для моделювання СППР обрано уніфіковану мову моделювання UML (Unified Modeling Language) [10].

UML є загальноприйнятим стандартом, який широко використовується для аналізу, проєктування та документування програмних систем різного рівня складності. Використання UML дозволяє формалізувати вимоги до системи, описати логіку функціонування та взаємодію між користувачами й компонентами системи незалежно від конкретної технології реалізації.

До основних переваг використання UML у процесі розробки СППР належать:

– можливість комплексного опису системи з різних точок зору (функціональної, структурної та поведінкової);

– підтримка об’єктно-орієнтованого підходу, що відповідає сучасним принципам розробки програмних систем;

– зручність комунікації між розробниками, аналітиками та замовниками завдяки уніфікованій нотації;

– можливість подальшої реалізації моделі у вигляді програмного коду.

У межах даної роботи UML використовується для побудови діаграми варіантів використання, яка відображає функціональні вимоги до СППР, діаграми

класів для представлення статичної структури системи, діаграми кооперацій для моделювання взаємодії, діаграми компонентів та діаграми розміщення.

Таким чином, вибір UML як основного засобу моделювання є обґрунтованим та доцільним, оскільки забезпечує системний підхід до проєктування СППР, підвищує якість проєктних рішень і сприяє ефективній реалізації системи.

2.1.2 Розробка діаграми варіантів використання для СППР

В процесі розробки загальносистемних рішень для СППР була побудована модель прецедентів та розроблені специфікації до кожного з прецедентів. Модель прецедентів призначена для формалізованого опису функціональних вимог до системи з точки зору взаємодії з користувачами та іншими зовнішніми сутностями (акторами) [11, 12].

Вона використовується для:

- визначення переліку функцій, які система повинна виконувати для кожного актора;
- опису сценаріїв використання системи у вигляді прецедентів (use cases);
- встановлення меж системи та її взаємодії із зовнішнім середовищем;
- узгодження вимог між замовником, користувачами та розробниками;
- виявлення та уточнення функціональних вимог на ранніх етапах проєктування;
- забезпечення основи для подальшого проєктування, тестування та документування системи.

Діаграма прецендентів приведена на рисунку 2.1.

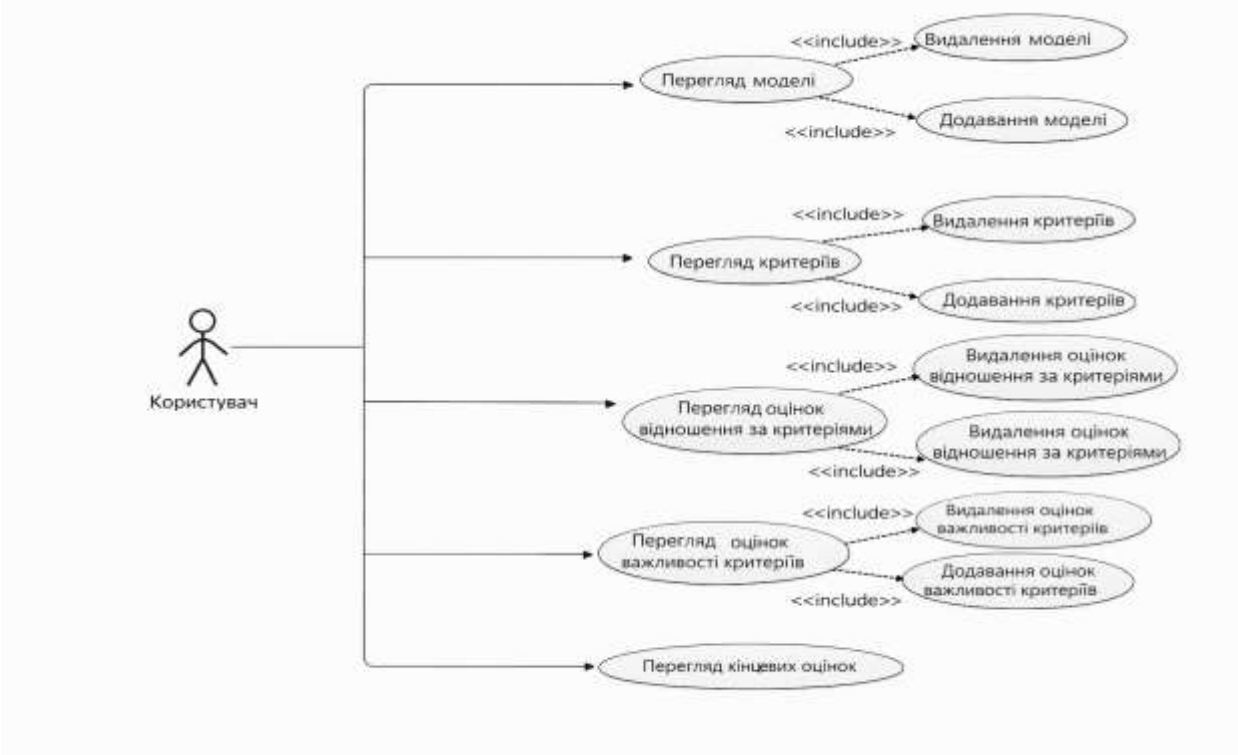

Рисунок 2.1– Діаграма варіантів використання

На наступному етапі будуть розроблені специфікації варіантів використання. Вони відіграють визначальну роль у формуванні уявлення про функціональні можливості системи з позиції кінцевого користувача. Саме вони задають напрям і логіку процесу розроблення на всіх його етапах – від початкового аналізу до завершення проєктування [11, 12].

2.1.3 Специфікації варіантів використання для СППР

Специфікація варіанту використання «Додавання моделі»

1) Назва: «Додавання моделі».
2) Призначення: дозволяє додати модель в БД.
3) Актори: активний суб’єкт «Користувач».
4) Попередня дія: перейти на посилання «Моделі», ввести модель в пусту строку перед доданням.
5) Стартова дія: ініціюється активним суб’єктом «Користувач».
6) Потоки подій:

Користувачу буде надане вікно, в якому він зможе додати модель, після введення якої користувач може натиснути на кнопку «додати».

7) Результати: користувач додав нову модель в БД, після успішного додання модель з’явиться у списку переліку моделей.

Специфікація варіанту використання «Додавання критеріїв»

1) Назва: «Додавання критеріїв».
2) Призначення: дозволяє додати критерій до БД.
3) Актори: активний суб’єкт «Користувач».
4) Попередня дія: перейти на посилання «Критерії», ввести критерій у строку перед доданням.
5) Стартова дія: ініціюється активним суб’єктом «Користувач».
6) Потоки подій:

Користувачу буде надане вікно, в якому він зможе додати критерій, після введення критерію користувач може натиснути на кнопку «додати».

7) Результати: користувач додав новий критерій в БД, після успішного додання критерій з’явиться у списку переліку критеріїв.

Специфікація варіанту використання «Додавання оцінок відношення за критеріями»

1) Назва: «Додавання оцінок відношення за критеріями».
2) Призначення: дозволяє додати оцінку відношення за критеріями до БД.
3) Актори: активний суб’єкт «Користувач».
4) Попередня дія: перейти на посилання «Важливість критеріїв», ввести оцінку в строку перед доданням.
5) Стартова дія: ініціюється активним суб’єктом «Користувач».
6) Потоки подій:

Користувачу буде надане вікно, в якому він зможе додати оцінку відношення за критеріями, після введення оцінки користувач може натиснути на кнопку «додати».

7) Результати: користувач додав нову оцінку відношення за критеріями в БД, після успішного додання оцінка з’явиться у списку оцінок відношення за критеріями.

Специфікація варіанту використання «Видалення оцінок відношення за критеріями»

1) Назва: «Видалення оцінок відношення за критеріями».
2) Призначення: дозволяє видалити оцінку відношення за критеріями з БД.
3) Актори: активний суб’єкт «Користувач».
4) Попередня дія: перейти на посилання «Важливість критеріїв».
5) Стартова дія: ініціюється активним суб’єктом «Користувач».
6) Потоки подій:

Користувачу буде надане вікно, в якому він зможе видалити оцінку відношення за критеріями, після знаходження оцінки, котру він бажає видалити, він натискає на кнопку «вилучити».

7) Результати: користувач видалив оцінку відношення за критеріями з БД, після успішного видалення оцінка зникне зі списку оцінкок відношення за критеріями.

Специфікація варіанту використання «Видалення моделі»

1) Назва: «Видалення моделі».
2) Призначення: дозволяє видалити модель з БД.
3) Актори: активний суб’єкт «Користувач».
4) Попередня дія: перейти на посилання «Моделі».
5) Стартова дія: ініціюється активним суб’єктом «Користувач».
6) Потоки подій:

Користувачу буде надане вікно, в якому він зможе видалити модель, після знаходження моделі, яку користувач бажає видалити, він натискає на кнопку «вилучити».

7) Результати: користувач видалив модель з БД, після успішного видалення модель зникне зі списку переліку моделей.

Специфікація варіанту використання «Видалення критеріїв»

1) Назва: «Видалення критеріїв».
2) Призначення: дозволяє видалити критерій з БД.
3) Актори: активний суб’єкт «Користувач».
4) Попередня дія: перейти на посилання «Критерії».
5) Стартова дія: ініціюється активним суб’єктом «Користувач».
6) Потоки подій:

Користувачу буде надане вікно, в якому він зможе видалити критерій, після знаходження критерію, який користувач бажає видалити, він натискає на кнопку «вилучити».

7) Результати: користувач видалив критерій з БД, після успішного видалення критерій зникне зі списку переліку критеріїв.

Специфікація варіанту використання «Додавання оцінки важливості критеріїв»

1) Назва: «Додавання оцінки важливості критеріїв».
2) Призначення: дозволяє додати оцінку важливості критеріїв до БД.
3) Актори: активний суб’єкт «Користувач».
4) Попередня дія: перейти на посилання «Важливість критеріїв».
5) Стартова дія: ініціюється активним суб’єктом «Користувач».
6) Потоки подій:

Користувачу буде надане вікно, в якому він зможе додати оцінку важливості критерію, після введення оцінки користувач може натиснути на кнопку «додати».

7) Результати: користувач додав оцінку в БД, після успішного додання оцінка з’явиться у списку переліку важливості критеріїв.

Специфікація варіанту використання «Видалення оцінки важливості критеріїв»

1) Назва: «Видалення оцінки важливості критеріїв».

2) Призначення: дозволяє видалити оцінку важливості критеріїв з БД.
3) Актори: активний суб’єкт «Користувач».
4) Попередня дія: перейти на посилання «Важливість критеріїв».
5) Стартова дія: ініціюється активним суб’єктом «Користувач».
6) Потоки подій:

Користувачу буде надане вікно, в якому він зможе видалити оцінку важливості критеріїв, після знаходження оцінки важливості критеріїв, яку користувач бажає видалити, він натискає на кнопку «вилучити».

8) Результати: користувач видалив оцінку з БД, після успішного видалення оцінка зникне зі списку переліку оцінок важливості критеріїв.

Специфікація варіанту використання «Перегляд оцінок важливості критеріїв»

1) Назва «Перегляд оцінок важливості критеріїв»
2) Призначення: дозволяє переглянути оцінки важливості критеріїв.
3) Актори: активний суб’єкт «Користувач».
4) Попередня дія: перейти на посилання «Перегляд оцінок важливості критеріїв».
5) Стартова дія: ініціюється активним суб'єктом «Користувач».
6) Потоки подій:

Користувачу буде надано вікно із оцінками важливості критеріїв.

7) Результати: користувачу надається список оцінок важливості критеріїв.

Специфікація варіанту використання «Перегляд моделей»

1) Назва «Перегляд моделей»
2) Призначення: дозволяє переглянути список усіх моделей.
3) Актори: активний суб’єкт «Користувач».
4) Попередня дія: перейти на посилання «Моделі».
5) Стартова дія: ініціюється активним суб'єктом «Користувач».
6) Потоки подій:

Користувачу буде надано вікно із списком моделей.

7) Результати: користувачу надається список моделей.

Специфікація варіанту використання «Перегляд оцінок відношення за критеріями»

1) Назва «Перегляд оцінок відношення за критеріями»
2) Призначення: дозволяє переглянути список оцінок відношення за критеріями.
3) Актори: активний суб’єкт «Користувач».
4) Попередня дія: перейти на посилання «Оцінки».
5) Стартова дія: ініціюється активним суб'єктом «Користувач».
6) Потоки подій:

Користувачу буде надано вікно із списком оцінок відношеня за критеріями.

7) Результати: користувачу надається список оцінок відношення за критеріями.

Специфікація варіанту використання «Перегляд кінцевих оцінок»

1) Назва «Перегляд кінцевих оцінок»
2) Призначення: дозволяє переглянути кінцеві оцінки по кожному технологічному стеку.
3) Актори: активний суб’єкт «Користувач».
4) Попередня дія: перейти на посилання «Результати».
5) Стартова дія: ініціюється активним суб'єктом «Користувач».
6) Потоки подій:

Користувачу буде надане вікно, в якому він побачить таблиці з оцінками за його критеріями, а також кінцеву таблицю, котра покаже йому поточні оцінки по кожній моделі.

7) Результати: користувачу буде надане вікно, в якому він побачить таблиці з оцінками за його критеріями, а також кінцеву таблицю, котра покаже йому поточні оцінки по кожній моделі.

2.1.4 Діаграма станів СППР

Діаграма станів призначена для опису динамічної поведінки об’єкта або системи в часі шляхом відображення можливих станів та переходів між ними у відповідь на події.

Вона використовується для [11, 12]:

- моделювання життєвого циклу об’єкта від моменту створення до завершення його існування;
- визначення можливих станів системи або її окремих компонентів;
- опису умов і подій, що ініціюють переходи між станами;
- аналізу реакції системи на зовнішні та внутрішні впливи;
- виявлення логічних помилок у поведінці системи (зависання, недосяжні стани, некоректні переходи);
- уточнення вимог до керування станами під час проєктування програмного забезпечення.

У UML діаграма станів особливо доцільна для моделювання об’єктів зі складною поведінкою, стан яких суттєво впливає на виконання їхніх функцій, зокрема у подієво-орієнтованих та інтерактивних системах.

На основі аналізу варіантів використання була побудована діаграма станів СППР, яка приведена на рисунку 2.2, де зображено різні стани об’єкта під час його існування і події, які призводять до переходу об’єкта з одного стану в інший.

Робота програми складається зі станів, в яких перебуває програма, та з переходів, які призводять до утворення нового стану.

Робота програми починається з завантаження програмного продукту, який призводить до утворення стану «Відображення головної сторінки» в цьому стані відсутні можливості для роботи з БД.

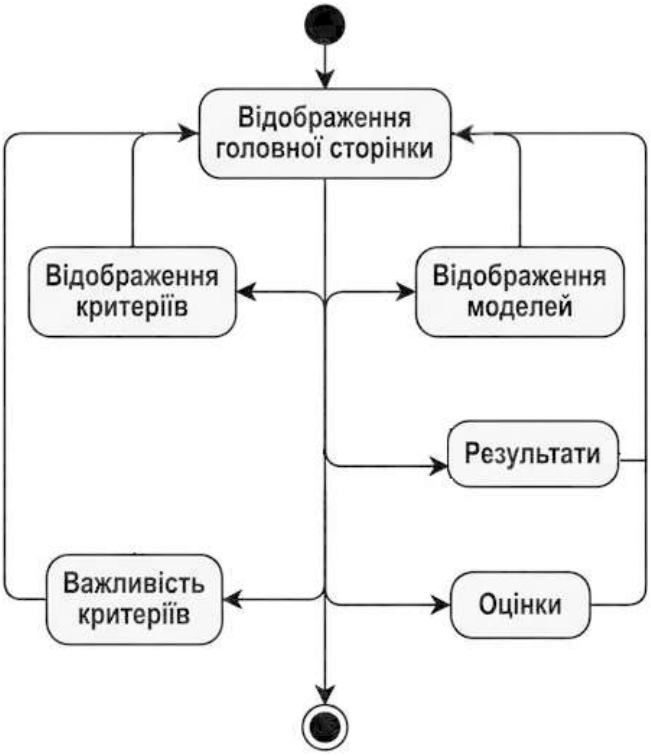

Рисунок 2.2 – Діаграма станів СППР

При відкритті сторінки «Оцінки», якщо відбулося вдале підключення до БД, утворюється новий стан «Відображення оцінок», при невдалому підключенні до БД сторінка не загрузиться, а при натисканні посилання «На головну» відбувається перехід до головної сторінки «Відображення головної сторінки». При відкритті сторінки «Результати», якщо відбулося вдале підключення до БД, утворюється новий стан «Відображення результатів», при невдалому підключенні до БД сторінка не загрузиться, а при натисканні посилання «На головну» відбувається перехід до головної сторінки «Відображення головної сторінки».

При відкритті сторінки «Моделі», якщо відбулося вдале підключення до БД, утворюється новий стан «Відображення моделей», при невдалому підключенні до БД сторінка не загрузиться, а при натисканні посилання «На головну» відбувається перехід до головної сторінки «Відображення головної сторінки». При відкритті сторінки «Критерії», якщо відбулося вдале підключення до БД, утворюється новий стан «Відображення критеріїв», при невдалому підключенні до БД сторінка не загрузиться, а при натисканні посилання «На головну» відбувається перехід до головної сторінки «Відображення головної сторінки». При відкритті сторінки «Важливість критеріїв», якщо відбулося вдале

підключення до БД, утворюється новий стан «Відображення важливості критеріїв», при невдалому підключенні до БД сторінка не загрузиться, а при натисканні посилання «На головну» відбувається перехід до головної сторінки «Відображення головної сторінки».

2.2 Рішення з інформаційного забезпечення для СППР

В рамках рішення з інформаційного забезпечення представлена логічна модель даних.

Логічна модель даних призначена для формалізованого опису структури даних інформаційної системи на логічному рівні без прив’язки до конкретної СУБД або фізичної реалізації.

Основне призначення логічної моделі даних полягає в такому:

- визначення сутностей предметної області та їх атрибутів;
- встановлення логічних зв’язків між сутностями (один-до-одного, один-до-багатьох, багато-до-багатьох);
- опис первинних і зовнішніх ключів, правил цілісності та обмежень даних;
- забезпечення єдиного розуміння структури даних між замовником, аналітиками та розробниками;
- перевірка повноти й узгодженості вимог до даних на етапі проєктування;
- створення основи для подальшої побудови фізичної моделі бази даних та реалізації в обраній СУБД.

Логічна модель даних відіграє ключову роль у розробленні надійних та масштабованих інформаційних систем.

Логічна структура даних СППР для вибору моделі розроблялася згідно даних від замовника та технічного завдання з використанням засобу моделювання бази даних InteliJ Idea.

Таблиця «model» містить перелік моделей (технологічних стеків). Таблиця «mark» містить перелік оцінок моделей, а також оцінки критеріїв. Зв’язок між «model» та «mark» становить один до багатьох. Оскільки зв'язок один до багатьох,

то ці таблиці потрібно якось розрізняти, щоб розрізняти існує маркер, в якому міститься позначення, за яким відрізняються дані таблиць. Таблиці «parameter», «parameterpriority» містять в собі дані про назву критеріїв та відповідно самі критерії. Їх винесено в окремі таблиці у зв’язку з тим, що вони мають невеликий перелік критеріїв, які повторюються в кожній таблиці «mark». Таблиці «parameter» і «mark» містять в собі дані про критерії та відповідно їх id. Щоб уникнути процесу дублювання даних і як наслідок зменшити розміри БД, було вирішено винести ці дані в окремі таблиці та встановити зв'язок один до багатьох. Детальна інформація по структурі міститься в таблицях 2.1–2.4. Логічна модель даних представлена на рисунку 2.3.

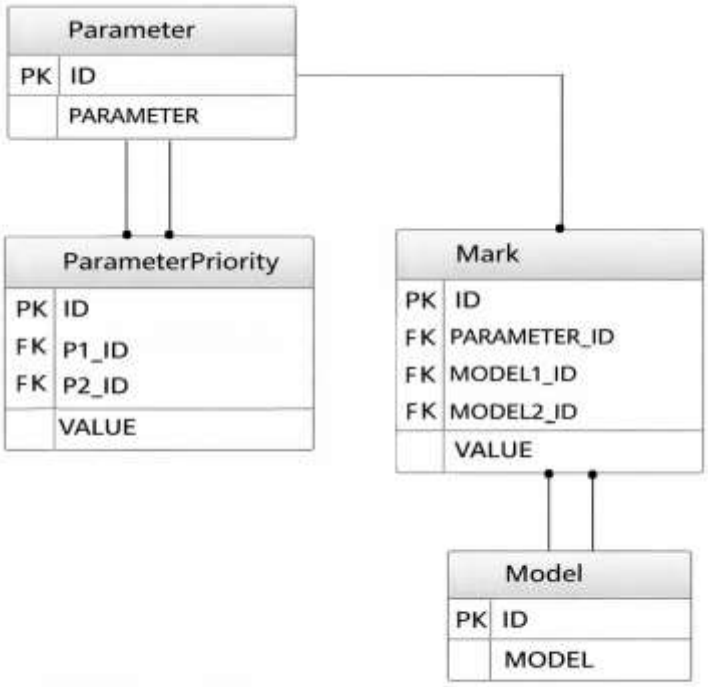

Рисунок 2.3 – Логічна модель даних

Таблиця 2.1 – Структура таблиці «Parameter»

Назва поля (українською)	Назва поля (англійською)	Тип даних	Ключ	Примітка
Id критерію	Id	N	PK	NOT NULL
Критерій	Parameter	S		NOT NULL

Таблиця 2.2 – Структура таблиці «ParameterPriority»

Назва поля (українською)	Назва поля (англійською)	Тип даних	Ключ	Примітка
Id оцінки	Id	N	PK	NOT NULL
Оцінка	Value	N		NULL
Id першого критерію	P1_Id	N	FK	NOT NULL
Id другого критерію	P2_Id	N	FK	NOT NULL

Таблиця 2.3 – Структура таблиці «Mark»

Назва поля (українською)	Назва поля (англійською)	Тип даних	Ключ	Примітка
Id оцінки	Id	N	PK	NOT NULL
Оцінка	Value	N		NULL
Id критерію	Parameter_Id	N	FK	NOT NULL
Id першої моделі	Model1_Id	N	FK	NOT NULL
Id другої моделі	Model2_Id	N	FK	NOT NULL

Таблиця 2.1 – Структура таблиці «Model»

Назва поля (українською)	Назва поля (англійською)	Тип даних	Ключ	Обов’язковість заповнення
Id моделі	Id	N	PK	NOT NULL
Модель	Model	S		NOT NULL

2.3 Рішення з програмного забезпечення для СППР

2.3.1 Вибір засобів розробки СППР

Для створення СППР щодо вибору моделей (технологічних стеків) було обрано мову програмування Java, оскільки однією з її ключових переваг є кросплатформеність, тобто можливість виконання однієї й тієї ж програми на

різних апаратних і програмних платформах. Реалізація цієї властивості забезпечується за рахунок використання віртуальної машини Java (JVM – Java Virtual Machine) [13].

Таким чином, технологія Java складається з двох основних компонентів:

- мови програмування Java;
- віртуальної машини Java.

Принципова відмінність компілятора Java від компіляторів мов програмування C/C++ та Pascal полягає в тому, що результатом компіляції є файл з байт-кодом – проміжним представленням програми, яке не прив’язане до конкретної архітектури процесора. Для виконання програми байт-код передається на вхід віртуальній машині Java, яка здійснює його інтерпретацію або компіляцію в машинні інструкції відповідного процесора. Завдяки цьому для забезпечення кросплатформенності достатньо реалізувати JVM для конкретної операційної системи, що на сьогодні виконано для всіх поширених ОС.

Однією з причин низької надійності програм, розроблених мовами низького рівня, є надання програмісту прямого доступу до оперативної пам’яті (наприклад, через покажчики у C/C++ або Pascal). У Java цей недолік усунено шляхом заборони прямого доступу до пам’яті, що істотно підвищує стабільність та безпечність програм. Разом із тим, така архітектура супроводжується певним зниженням продуктивності, зокрема через виконання додаткових перевірок (наприклад, перевірки виходу за межі масивів під час доступу до їх елементів).

Ще одним недоліком традиційних мов програмування є необхідність ручного керування пам’яттю, зокрема явного звільнення ресурсів для об’єктів, які більше не використовуються. Помилки на цьому етапі можуть призводити до витоків пам’яті під час виконання програми. У Java ця проблема вирішується за допомогою вбудованого механізму автоматичного керування пам’яттю – збирача сміття (garbage collector), який відстежує всі створені об’єкти та автоматично звільняє пам’ять, зайняту тими з них, що більше не використовуються. Таким чином, ручне видалення об’єктів у Java відсутнє, що зменшує ймовірність помилок та підвищує надійність програмного забезпечення.

Для розробки програмного забезпечення було використано інтегроване середовище розробки IntelliJ IDEA, яке забезпечує зручні засоби проєктування, налагодження та тестування програм, а також підтримує ефективне створення графічних інтерфейсів користувача [14]. У порівнянні з альтернативними середовищами, такими як Eclipse та NetBeans, IntelliJ IDEA вирізняється розвинутими інструментами аналізу коду та високою продуктивністю роботи розробника.

Для проєктування та моделювання бази даних було використано інструмент DataGrip від компанії JetBrains. Даний програмний продукт підтримує роботу з СУБД MySQL та забезпечує засоби побудови як логічної, так і фізичної моделей бази даних [15]. Основними перевагами DataGrip є зручний інтерфейс, підтримка широкого спектра систем керування базами даних, а також можливість безкоштовного використання в навчальних цілях, що робить його доцільним вибором для реалізації бази даних у межах даної роботи.

2.3.2 Фізична модель даних для СППР

Фізична модель даних призначена для конкретної реалізації структури даних інформаційної системи у вибраній системі управління базами даних (СУБД) та визначає, як дані будуть зберігатися, індексуватися на фізичному рівні.

Основне призначення фізичної моделі даних:

- відображення логічних сутностей та їх атрибутів у вигляді конкретних таблиць, полів та типів даних у СУБД;
- визначення ключів, індексів, унікальних обмежень та інших механізмів забезпечення цілісності даних;
- оптимізація зберігання та доступу до даних з точки зору продуктивності, швидкості запитів і ресурсів системи;
- планування фізичного розміщення даних на диску, кластерів, кешування;
- забезпечення підтримки резервного копіювання, відновлення та масштабування бази даних;

– перевірка реалістичності та ефективності обраної структури даних перед впровадженням у робоче середовище.

Фізична модель даних є завершальним етапом моделювання даних і забезпечує безпосередню реалізацію бази даних для конкретного програмного забезпечення.

Розробимо фізичну структуру кожної таблиці, що входить у базу даних. Типи даних будемо вказувати для СУБД MySQL, оскільки вона використовується для зберігання даних в системі [15].

При описі фізичних таблиць бази даних використані стандартні позначення ключів: PK (первинний ключ), FK (зовнішній ключ).

Усі таблиці бази даних знаходяться в 1 нормальній формі (НФ), оскільки вони є двовимірними, не містять полів, що включають кілька значень, і порожніх ключових полів (PK).

Усі таблиці бази даних знаходяться в 2 НФ, тому що всі їхній поля, що не входять у первинний ключ, зв'язані повною функціональною залежністю з первинним ключем.

Усі таблиці бази даних знаходяться в 3 НФ, оскільки немає неключових полів, що залежать від інших неключових полів.

На рисунку 2.4 представлено фізичну модель даних системи підтримки прийняття рішень по вибору моделей. В таблиці 2.5 представлені описи типів даних фізичної моделі.

Таблиця 2.5 – Типи даних для СУБД MySQL

Тип даних	Довжина
int	4 byte
varchar	length + 1 byte
double	8 byte

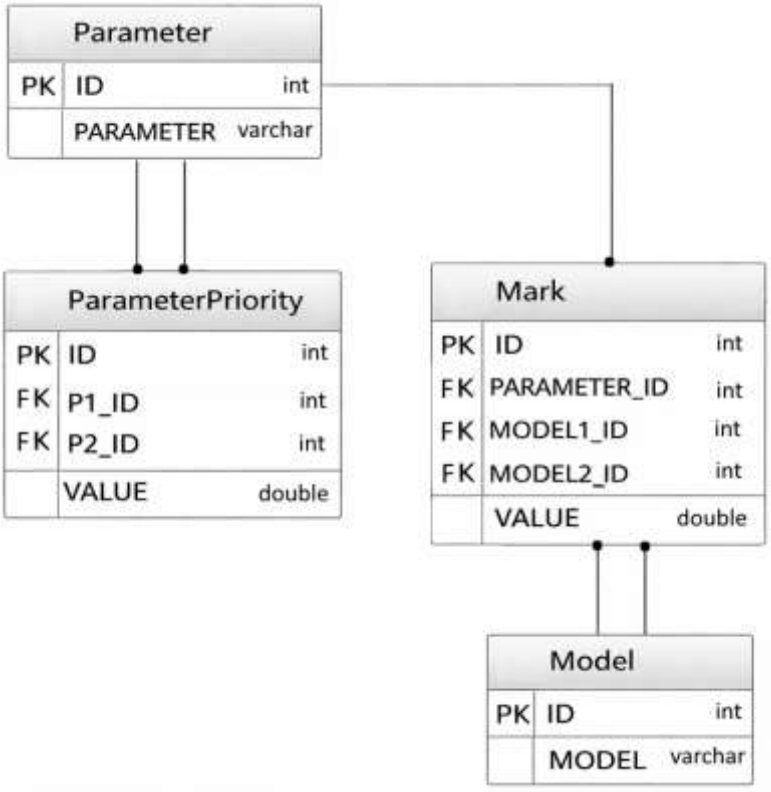

Рисунок 2.4 – Фізична модель даних

2.3.3 Модель класів для СППР

Статична модель системи призначена для опису структури системи та її компонентів у стані спокою, тобто без врахування змін у часі та динамічної поведінки.

Основне призначення статичної моделі системи [11, 12]:

- визначення основних елементів системи (класи, об’єкти, компоненти) та їх властивостей;
- опис зв’язків та залежностей між елементами системи (асоціації, агрегації, композиції, наслідування);
- забезпечення зрозумілого представлення архітектури системи для розробників, аналітиків та замовників;
- формалізація структури даних та програмних компонентів, що дозволяє планувати модульну побудову системи;
- служить основою для динамічного моделювання поведінки системи (через діаграми станів, послідовностей).

Статична модель системи у вигляді діаграми класів представлена на рисунку 2.5.

Точкою входу у веб-додаток та водночас головним класом являється клас «MainServlet», який надає можливість працювати з базою даних, а саме доповнювати, отримувати, редагувати та видаляти дані.

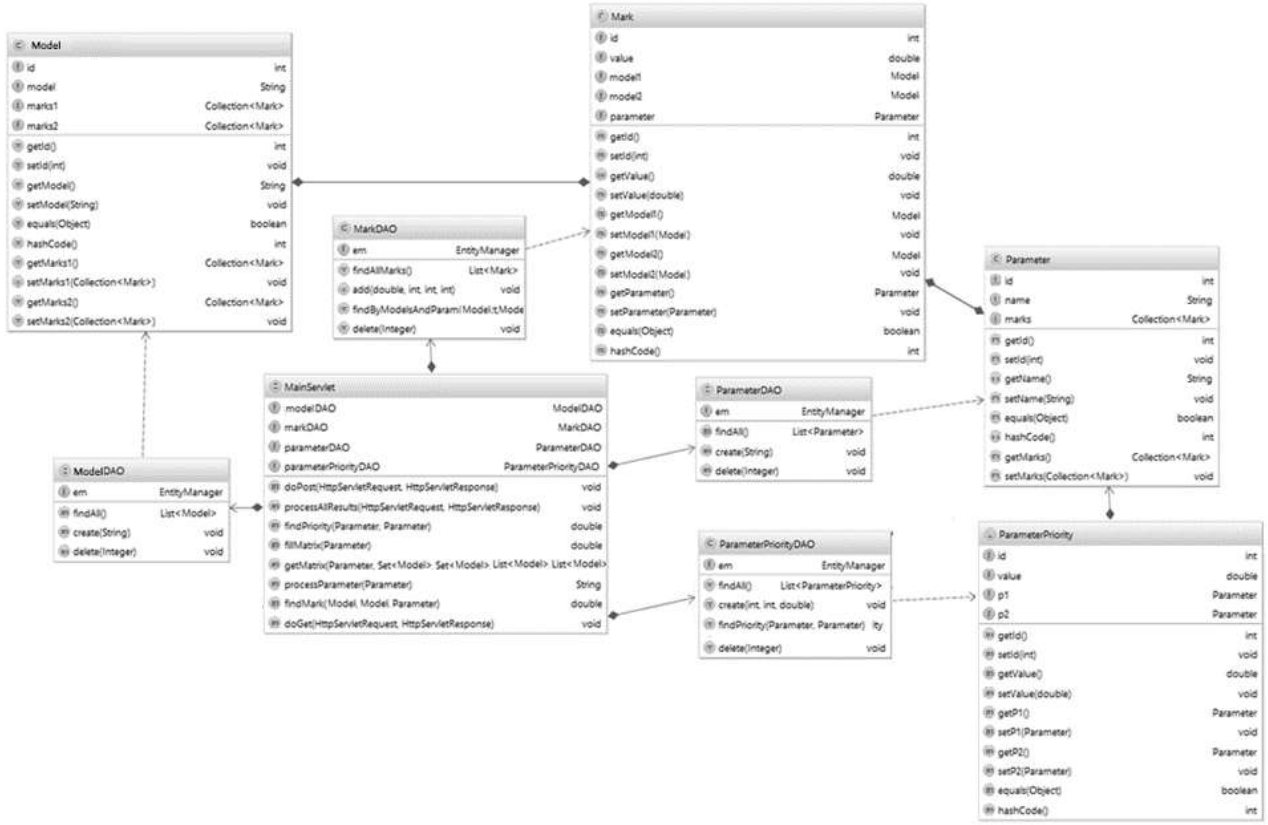

Рисунок 2.5 – Діаграма класів

Клас: ParameterDAO

Призначення: надає можливість редагувати дані, додавати та видаляти з критерії.

Атрибути:

em – об’єкт класу котрий видає нам інформацію почергово

Методи:

findAll() – надає список усіх критеріїв

create() – додає новий критерій

delete() – видаляє існуючий критерій

Клас: MarkDAO

Призначення: надає можливість редагувати дані, додавати та видаляти оцінки.

Атрибути:

em – об’єкт класу котрий видає нам інформацію почергово

Методи:

findAllMarks() – надає список оцінок

findByModelsAndParam() – надає список оцінок відносності параметрів один до одного за певним критерієм

add() – додає нову оцінку

delete() – видаляє існуючу оцінку

Клас: ModelDAO

Призначення: надає можливість редагувати дані, додавати та видаляти моделі.

Атрибути:

em – об’єкт класу котрий видає нам інформацію почергово

Методи:

findAll() – надає список усіх моделей

create() – додає нову модель

delete() – видаляє існуючу модель

Клас: ParameterPriorityDAO

Призначення: надає можливість редагувати дані, додавати та видаляти відношення важливостей критеріїв.

Атрибути:

em – об’єкт класу котрий видає нам інформацію почергово

Методи:

findAll() – надає список оцінок відношення важливостей критеріїв

findPriority() – надає випадаючий список критеріїв

create() – додає нову оцінку відношення важливостей критеріїв

delete() – видаляє існуючу оцінку відношення важливостей критеріїв

Клас: Model

Призначення: додає, видаляє моделі з БД

Атрибути:

id – особливий номер моделі

model – назва моделі

marks1 – оцінка першої моделі

marks2 – оцінка другої моделі

Методи:

getId() – виймає id моделі з БД

setId() – надає id моделі

getModel() – виймає модель з БД

setModel() – додає модель до БД

equals() – порівнює моделі

hashCode() – повертає хеш-код моделі

getMarks1() – дістає оцінки першої моделі

setMarks1() – надає оцінки першій моделі

getMarks2() – дістає оцінки другої моделі

setMarks2() – надає оцінки другої моделі

Клас: Mark

Призначення: надає можливість редагувати дані, додавати та видаляти оцінки моделі.

Атрибути:

id – особливий номер оцінки

value – оцінка

model1 – модель перша

model2 – модель друга

parameter – критерій

Методи:

getId() – виймає id моделі з БД

setId() – надає id моделі

getValue() – виймає оцінку з БД

setValue() – надає оцінку

getModel1() – надає список існуючих моделей з БД

setModel1() – вибирає модель

getModel2() – надає список існуючих моделей з БД

setModel2() – вибирає модель

getParameter() – надає список існуючих критеріїв з БД

setParameter() – вибирає критерій

equals() – порівнює моделі

hashCode() – повертає хеш-код моделі

Клас: Parameter

Призначення: надає можливість редагувати дані, додавати та видаляти критерії.

Атрибути:

id – особливий номер критерію

name – им’я критерію

marks – оцінка критеріїв

Методи:

getId() – виймає id критерію з БД

setId() – додає id критерію

getName() – виймає критерій з БД

setName() – додає критерій до БД

equals() – порівнює критерії

hashCode() – повертає хеш-код критеріїв

getMarks() – виймає оцінку з БД

setMarks() – надає оцінку

Клас: ParameterPriority

Призначення: надає можливість редагувати дані, додавати та видаляти оцінки відношень критеріїв.

Атрибути:

id – особливий номер оцінки

value – оцінка

p1 – перший критерій

p2 – другий критерій

Методи:

getId() – виймає id оцінки з БД

setId() – додає новий id

getValue() – виймає оцінку

setValue() – надає оцінку

getP1() – надає випадаючий список критеріїв

setPt1() – вибирає критерій з випадаючого списку

getP2() – надає випадаючий список критеріїв

setP2() – вибирає критерій з випадаючого списку

equals() – порівнює критерії

hashCode() – повертає хеш-код критеріїв

Клас: MainServlet

Призначення: надає можливість редагувати дані, додавати та видаляти моделі.

Атрибути:

modelDAO – атрибут котрий використовується для викликів методів

parameterDAO – атрибут котрий використовується для викликів методів

markDAO – атрибут котрий використовується для викликів методів

parameterPriorityDAO – атрибут котрий використовується для викликів методів.

Методи:

doPost() – головний метод програми який за введеними даними користувача визначає за якою послідовністю продовжувати роботу програми

processAllResults() – виводить всі результати на екран

findPriority() – вираховує

fillMatrix() – вираховує оцінку моделі за певним критерієм

getMatrix() – повертає матрицю

processParameter() – надає оцінку моделям за певним критерієм

findMark() – знаходить оцінку по зрівнянню моделей за критерієм

doGet() – головний метод програми, котрий використовує інші методи класів для видалення тієї чи іншої позиції.

2.3.4 Моделі взаємодії для СППР

Для опису динамічної поведінки СППР та аналізу процесів взаємодії між її складовими елементами використовуються моделі взаємодії UML. Дані моделі дозволяють відобразити порядок обміну повідомленнями між об’єктами системи у межах реалізації окремих функціональних сценаріїв, а також визначити логіку виконання операцій у часі [11, 12].

Побудова діаграм взаємодії забезпечує наочне представлення механізмів координації між компонентами системи, що сприяє уточненню вимог, виявленню можливих логічних неузгодженостей та підвищенню якості проєктних рішень. Діаграми використовуються для деталізації варіантів використання та слугують основою для подальшого проєктування системи.

Основними діаграмами взаємодії в UML являються такі діаграми:

– Діаграма послідовностей

Описує порядок обміну повідомленнями між об’єктами у часовій послідовності.

Використовується для:

- деталізації сценаріїв варіантів використання;

- аналізу логіки виконання операцій;
- проєктування методів класів.

– Діаграма комунікації

Показує взаємодію об’єктів з акцентом на їхні зв’язки та обмін повідомленнями.

Є альтернативою діаграмі послідовностей, але з меншим акцентом на час.

На основі побудованої діаграми класів та специфікацій варіантів використання розробимо динамічну модель СППР. Програмний продукт складається з двох частини. Перша частина – це база даних, в якій міститься інформація про моделі, критерії та оцінки. Друга частина – це безпосередньо графічний інтерфейс, який надає можливість маніпулювати даними, а саме поповнювати базу новою інформацією та видаляти існуючу інформацію.

Діаграма кооперації підключення до БД представлена на рисунку 2.6.

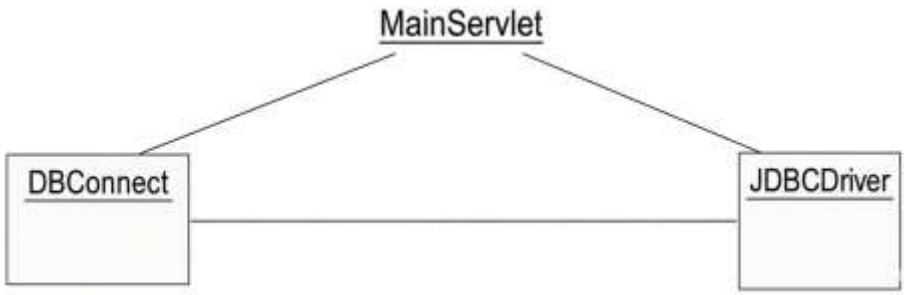

Рисунок 2.6 – Діаграма кооперації

Об’єктом «MainServlet» викликається конструктор «DBConnect», що призводить до утворення об’єкта «DBConnect», який через метод «connectDriver()» створює об’єкт «JDBCDriver», який здійснює підключення до БД.

2.3.5 Діаграма компонентів СППР

Діаграма компонентів UML призначена для відображення структури системи на рівні її логічних та фізичних компонентів, а також для опису взаємозв’язків між ними. Вона демонструє, з яких модулів складається система і яким чином ці модулі взаємодіють між собою [11, 12].

Діаграма компонентів дозволяє визначити межі відповідальності окремих компонентів, спростити аналіз залежностей між ними та забезпечити можливість повторного використання і заміни компонентів без порушення цілісності системи.

Діаграма компонентів є важливим інструментом на етапах аналізу та проєктування, оскільки сприяє узгодженню архітектурних рішень, полегшує командну розробку та підвищує супроводжуваність системи.

Проаналізувавши діаграму класів та діаграми кооперації, було побудовано діаграму компонентів яка представлена на рис. 2.7.

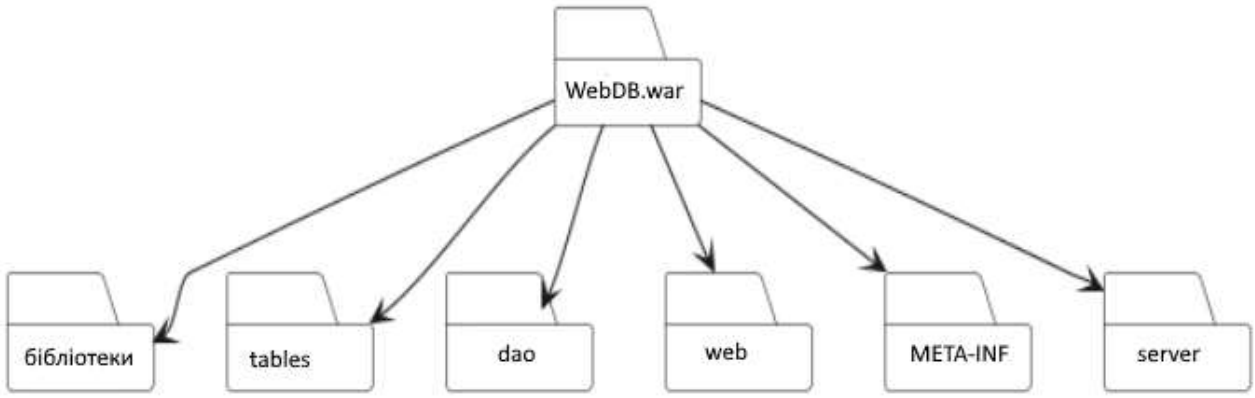

Рисунок 2.7 – Діаграма компонентів СППР по вибору технологічного стеку

В результаті компіляції програми, JDK створює файл «webdb.war» з так званим байт - кодом – проміжною мовою, який містить в собі наступні пакети з компонентами:

– «META-INF» – пакет містить в собі компонент «Persistence.xml», драйвер для підключення до бази даних «JDBCMySQL».

– «tables» – пакет містить в собі компоненти «Mark.java», «Parameter.java», «Models.java», «ParameterPriority.java» які відповідають за заповнення таблиць для відображення отриманих даних з БД;

– «dao» – пакет котрий містить в собі компоненти: «MarkDAO.java», «ParameterDAO.java», «ModelsDAO.java», «ParameterPriorityDAO.java», які відповідають за відображення графічного інтерфейсу користувача;

– «server» – пакет містить в собі компонент «MainServlet.java», який відповідає за підключення до БД;

–«Бібліотеки» – пакет містить в собі компонент драйвер для підключення до бази даних «JDBCMySQL».

2.3.6 Діаграма розміщення СППР

Діаграма розміщення використовується для опису фізичної конфігурації системи, зокрема апаратних вузлів, мережевих з'єднань між ними та розміщення програмних артефактів (виконуваних файлів, сервісів, баз даних) на відповідних обчислювальних ресурсах. Вона дає змогу наочно представити взаємозв'язок між програмними компонентами та фізичним середовищем їх функціонування, а також оцінити вимоги до інфраструктури, масштабованість і надійність системи [11, 12].

Діаграма розміщення є доцільною на етапах проєктування та впровадження програмного забезпечення, оскільки забезпечує узгодження програмної архітектури з апаратною платформою та умовами експлуатації системи.

Проаналізувавши технічні вимоги, діаграму компонентів, було розроблено програмне забезпечення та БД, які можуть бути встановлені на різних ПК, які об'єднані локальною мережею. Діаграма розміщення компонентів представлена на рис 2.8.

З діаграми компонентів видно, що ПЗ і база даних можуть знаходитися в різних місцях на різних ПК. Тобто клієнтська частина буде знаходитися у любому поміщені із доступом до інтернету на персональному комп'ютері, а сервер з БД

буде знаходитися на окремому комп’ютері, який здатен працювати з БД через локальну мережу. Також можливо встановити на один персональний комп’ютер і БД і програмний продукт webdb.war.

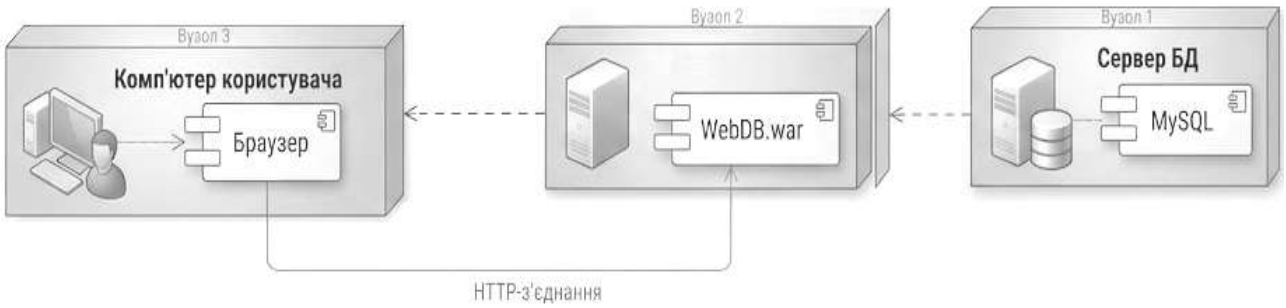

Рисунок 2.8 – Діаграма розміщення СППР по вибору технологічного стеку

2.3.7 Випробування СППР

Випробування СППР проводяться відповідно до програми та методики випробувань представниками замовника в присутності розробника. При наявності відповідної можливості, у процесі тестування можуть додатково брати участь програмісти, тестувальники від замовника, які не брали участі у розробці системи.

Перед початком (не пізніше 2 днів) до випробувань уточнюється кінцевий склад групи, що буде проводити випробування. Кожний член групи отримує необхідну документацію до системи: технічне завдання, загальний опис програмного забезпечення, керівництво користувача.

Порядок проведення випробувань:

– Виконуються контрольні функції в довільному порядку. Результати заносяться у відповідний журнал.

– Робиться аналіз отриманих результатів і встановлюється відповідність програми заданим вимогам.

При виявленні помилок у роботі програмного забезпечення для прийняття рішень по вибору моделей складається їхній перелік і обговорюється термін їхнього виправлення з розробником. Повний зміст програми та методики випробувань наведено у додатку В.

Випробування розробленої СППР проводили згідно описаної в додатку В програми. Отриманий результат представлений в таблиці 2.6.

Таблиця 2.6 – Результат випробування розробленої СППР

№	Функція, що перевіряється	Очікуваний результат	Отриманий результат
1	2	3	4
1	Перегляд головної сторінки	Після завантаження застосунку користувач перенаправляється на головну сторінку	Після завантаження застосунку користувач перенаправляється на головну сторінку
2	Додання моделі	Нова модель з’явиться у списку переліку моделей	Нова модель з’явиться у списку переліку моделей
3	Видалення моделі	Модель зникне зі списку переліку моделей	Модель зникне зі списку переліку моделей
4	Перегляд моделей	Надається список моделей	Надається список моделей
5	Додання критерію	Новий критерій з’явиться у списку переліку критеріїв	Новий критерій з’явиться у списку переліку критеріїв
6	Видалення критерію	Критерій зникне зі списку переліку критеріїв	Критерій зникне зі списку переліку критеріїв
7	Перегляд критеріїв	Надається список критеріїв	Надається список критеріїв
8	Додання оцінки	Нова оцінки важливості	Нова оцінки важливості

	важливості критеріїв	критеріїв з’явиться у списку оцінок важливості критеріїв	критеріїв з’явиться у списку оцінок важливості критеріїв
9	Видалення оцінки важливості критеріїв	Оцінка зникне зі списку переліку оцінок важливості критеріїв.	Оцінка зникне зі списку переліку оцінок важливості критеріїв.

Кінець таблиці 2.6

1	2	3	4
10	Перегляд оцінок важливості критеріїв	Надається список оцінок важливості критеріїв	Надається список оцінок важливості критеріїв
11	Додання оцінки відношення за критерієм	Нова оцінки відношення за критерієм з’явиться у списку оцінок відношення за критерієм	Нова оцінки відношення за критерієм з’явиться у списку оцінок відношення за критерієм
12	Видалення оцінки відношення за критерієм	Оцінка зникне зі списку оцінок відношення за критерієм	Оцінка зникне зі списку оцінок відношення за критерієм
13	Перегляд оцінок відношення за критерієм	Надається список оцінок відношення за критерієм	Надається список оцінок відношення за критерієм
14	Перегляд кінцевих результатів	Надаються таблиці з оцінками за критеріями, а також кінцева таблиця з оцінками по кожній моделі.	Надаються таблиці з оцінками за критеріями, а також кінцева таблиця з оцінками по кожній моделі.

3 ТЕХНІКО-ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ РОЗРОБКИ СППР

Доцільність розробки СППР можливо обґрунтувати з економічної точки зору, якщо ми розрахуємо собівартість системи та прибуток. Основним критерієм економічної доцільності витрат на розробку СППР буде річний економічний ефект [16].

3.1 Розрахунок витрат на створення й експлуатацію СППР

Витрати на розробку системи складаються з витрат на заробітну плату розробника, на амортизацію та експлуатацію ПК, на якому виконується розробка, на засоби розробки та витрати на матеріали і комплектуючі. Розробка СППР виконується спеціалістом, місячний оклад якого складає 19000 грн. Додаткова заробітна плата складає 20% від основної. Виходячи з цього, основна і додаткова заробітна плата розроблювача системи 22800 грн./міс, а вартість сучасного ПК складає 26000 грн. Вартість кіловат-години електроенергії дорівнює 4,32 грн. Витрати на допоміжні матеріали наведені в таблиці 3.1.

Таблиця 3.1 – Витрати на допоміжні матеріали

№	Пункти витрат	Сума, грн.
1	Папір для принтера	300
2	Картридж та тонер для принтера	300
3	Використання флеш-накопичувача	300
4	Непередбачувані витрати	1500
	Разом	2400

Вартість розробки системи розраховується за формулою:

$$С_{пр} = (З_{зп} + З_{зс} + З_{зг} + З_{е}) * Т + З_{м},$$

де Т – тривалість розробки у місяцях;

$З_{зп}$ – основна і додаткова заробітна плата, грн.;

$З_{зс}$ – відрахування на соціальні заходи (38% від основної і додаткової заробітної плати), грн.;

$З_{зг}$ – загальногосподарські витрати (10% від заробітної плати), грн.;

$З_{е}$ – витрати на електроенергію, грн.

$З_{м}$ – витрати на допоміжні матеріали, грн.

$З_{е}$ при витратах в 0,5 кВт з тривалістю роботи на місяць 22*8 = 176 годин та вартості електроенергії 4,32 грн. становить

$$З_{е} = 176 * 4{,}32 * 0{,}5 = 380{,}16 \text{ грн.} \quad (3.1)$$

Представимо всі поточні витрати на розробку системи в таблиці 3.2.

Таблиця 3.2 – Витрати на розробку системи

№	Пункти витрат	Одиниця	Кількість
1	Тривалість розробки (Т)	Міс.	2
2	Основна та додаткова заробітна плата (Ззп)	Грн.	22800
3	Відрахування на соціальні витрати (Ззс)	Грн.	8664
4	Загальногосподарські витрати (Ззг)	Грн.	2280
5	Витрати на допоміжні матеріали (Зм)	Грн.	2400
6	Витрати на електроенергію *(Зе)	Грн.	380,16

За формулою (3.1) виконаємо розрахунок вартості програми:

$$С_{пр} = (22800 + 8664 + 2280 + 380{,}16) * 2 + 2400 = 70648{,}32 \text{ грн.}$$

Амортизаційні відрахування на устаткування складають 60% балансової вартості в рік:

$$А_{об} = 26000 * 0{,}6 = 15600 \text{ грн.}$$

У масштабах підприємства річні витрати на основні і допоміжні матеріали визначаються в розмірі 5% вартості основного устаткування:

$$В_{м} = 26000 * 0{,}05 = 1300 \text{грн.}$$

Річний обсяг робіт ПК у годинах можна визначити як

$$Ф_{м} = 264{,}5 * Т_{з},$$

де Тз – середнє навантаження на устаткування в місяць (6 годин), 264,5 – середня кількість робочих днів на рік.

Отже, річний обсяг роботи ПК складає:

$$Ф_м = 264{,}5 * 6 = 1587 \text{ годин}$$

Витрати на електроенергію Зе при 1578 годинах роботи устаткування на рік становлять:

$$З_е = 1587 * 4{,}32 = 6555{,}84$$

Експлуатаційні витрати на ПК на рік становлять:

$$З_{зр} = 15600 + 1300 + 6555{,}84 = 23455{,}84$$

Таким чином витрати на розробку та експлуатацію системи в перший рік становлять:

$$З_р = 70648{,}32 + 23455{,}84 = 94104{,}16$$

3.2 Економічна ефективність розробки та впровадження системи

Визначити пряму економічну ефективність можна, ґрунтуючись на тому, що впровадження системи вивільняє одного працівника (за експертною оцінкою фахівців підприємства). Зарплата одного працівника в рік складає

$$З = 19000 * 12 * 1 = 228000 \text{ грн.}$$

Річний економічний ефект розраховується по формулі:

$$E_{рік} = \Delta C_n - E_n * k,$$

де ΔC_n – кошти, що буде вивільнено при впровадженні програмного забезпечення (228000) без експлуатаційних витрат (23455,84), тобто, 204544,16 грн.;

E_n – коефіцієнт ефективності (такий же як й коефіцієнт амортизації – 0,6)

k – одноразові витрати на впровадження програмного забезпечення (70648,32 грн.).

$$E_{рік} = 204544{,}16 - 0{,}6 * 70648{,}32 = 218149{,}16 - 28362{,}19 = 162155{,}94$$

Термін, за який програмне забезпечення окупиться наступний:

$$T = \frac{k}{\Delta C_n} = \frac{70648,32}{204544,16} = 0,34 \text{ року}$$

Отже термін, за який програмне забезпечення окупиться 4 місяці.

3.3 Висновки за розділом 3

У цьому розділі проведено розрахунки, спрямовані на створення та інтеграцію системи підтримки прийняття рішень для вибору технологічного стеку вебзастосунку. Також було оцінено економічну ефективність розробки та визначено термін її окупності. Згідно з розрахунками, окупність системи становить 4 місяці впродовж першого року використання. Це свідчить про високу рентабельність проєкту.

4 ОХОРОНА ПРАЦІ

4.1 Вступ

Охорона праці – це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних та лікувально-профілактичних заходів і засобів, спрямованих на збереження життя, здоров'я та працездатності людини в процесі трудової діяльності.

Умови праці – сукупність факторів виробничого середовища і трудового процесу, які впливають на здоров'я і працездатність людини під час виконання нею професійної діяльності.

Шкідливий виробничий фактор – виробничий фактор, вплив якого на працівника може призвести до погіршення стану здоров'я або зниження працездатності.

Охорона праці базується на законодавчих, директивних та нормативно-правових документах. Під час управління охороною праці не допускаються рішення та заходи, що суперечать чинному законодавству, державним нормативним актам про охорону праці, стандартам безпеки праці, правилам та нормам охорони праці [17].

До основних функцій управління охороною праці належать:

– прогнозування і планування робіт, їх фінансування;

– організація та координація робіт;

– облік показників стану умов і безпеки праці;

– аналіз та оцінка стану умов і безпеки праці;

– контроль за функціонуванням системи управління охороною праці.

Основні завдання управління охороною праці:

– навчання працівників безпечним методам праці та пропаганда питань охорони праці;

– забезпечення безпеки технологічних процесів, виробничого устаткування, будівель і споруд;

– нормалізація санітарно-гігієнічних умов праці;

– забезпечення працівників засобами індивідуального захисту;

– забезпечення оптимальних режимів праці та відпочинку;

– організація лікувально-профілактичного обслуговування;

– професійний добір працівників за певними професіями;

– удосконалення нормативної бази з питань охорони праці.

Охорона праці відіграє важливу роль як суспільний чинник, оскільки жодні трудові досягнення не можуть компенсувати втраченого здоров'я чи життя. Тому вивченню питань охорони праці приділяється значна увага. Представники багатьох наук досліджують проблеми створення безпечних та нешкідливих умов праці, адже саме за таких умов людина здатна працювати високопродуктивно, створюючи матеріальний добробут суспільства.

Основоположним законодавчим документом у сфері охорони праці є Закон України «Про охорону праці» [17]. Цей Закон поширюється на всі підприємства, установи та організації незалежно від форми власності та видів діяльності, на всіх громадян, які працюють або залучаються до праці на цих підприємствах. Закон визначає основні положення щодо реалізації конституційного права працівників на охорону їхнього життя і здоров'я в процесі трудової діяльності, на належні, безпечні та здорові умови праці, регулює відносини між роботодавцем і працівником з питань безпеки, гігієни праці та виробничого середовища, встановлює єдиний порядок організації охорони праці в Україні.

4.2 Аналіз потенційно небезпечних і шкідливих факторів у приміщенні комп'ютерної фірми

Класифікація шкідливих та небезпечних виробничих факторів наведена в таблиці 4.1.

Розглянемо основні вимоги до приміщень, де встановлені ПК. Залежно від орієнтації вікон рекомендується таке забарвлення стін і підлоги приміщення:

– вікна орієнтовані на південь – стіни зеленувато-блакитного або світло-блакитного кольору; підлога – зелена;

– вікна орієнтовані на північ – стіни світло-оранжевого або оранжево-жовтого кольору; підлога – червонувато-оранжева;

– вікна орієнтовані на схід – стіни жовто-зеленого кольору; підлога – зелена або червонувато-оранжева;

– вікна орієнтовані на захід – стіни жовто-зеленого або голубувато-зеленого кольору; підлога – зелена або червонувато-оранжева.

Освітлення приміщень з комп'ютерами повинно бути змішаним.

Вимоги до освітленості в приміщеннях, де встановлені комп'ютери: при виконанні зорових робіт високої точності загальна освітленість повинна становити 300 лк, комбінована –750 лк; для робіт середньої точності – 200 і 300 лк відповідно (згідно з ДБН В.2.5-28:2018 «Природне і штучне освітлення»).

Таблиця 4.1 – Класифікація шкідливих та небезпечних виробничих факторів

Види потенційно можливих небезпечних і шкідливих виробничих факторів	Наявність факторів
1	2
1. Фізичні небезпечні і шкідливі виробничі фактори:	
– шум,	+
– вібрація,	-
– підвищене виділення тепла,	-
– підвищена вологість,	-
– наявність електромагнітних полів,	+
– наявність електростатичних полів,	+
– небезпека вибуху,	-
– небезпека виникнення пожежі,	+
– небезпека ураження електрострумом,	+
– підвищена загазованість,	-
– відкрито рухаються й обертаються частини машин.	-
2. Хімічні небезпечні і шкідливі виробничі фактори:	
– токсичні речовини: пари ртуті, пари свинцю і його сполук та ін.,	-
– подразнюючі речовини: пральні засоби, кислоти, луги, роз-чинники, кислі гази,	-
– канцерогенні речовини: сполуки алюмінію, бензин, газ, 3,4-бензапирен (міститься у вихлопних газах автомобілів).	-

Кінець таблиці 4.1

1	2
3. Біологічні небезпечні і шкідливі виробничі фактори: – мікроби, віруси, бактерії, – комахи, – гриби, найпростіші, – інші мікроорганізми.	 + + - -
4. Психофізіологічні небезпечні і шкідливі виробничі фактори: – малорухомість роботи, – фізична утома, – нервова напруга, – напруга аналізаторів (зорового-слухового-тактильного)	 + + + +

4.2.1 Електробезпека

У приміщенні комп'ютерної фірми експлуатуються персональні комп’ютери та інші електротехнічні пристрої, монтаж і використання яких повинні здійснюватися відповідно до вимог Правил улаштування електроустановок (ПУЕ), затверджених наказом Міністерства енергетики України від 21.07.2017 № 476.

Норми електробезпеки щодо електронно-обчислювальних машин і персональних комп’ютерів регламентуються чинними нормативно-правовими актами, зокрема ПУЕ, Правилами пожежної безпеки в Україні (НАПБ А.01.001-2014), а також технічною та експлуатаційною документацією виробників обладнання.

Персональні комп’ютери, периферійні пристрої та інше електрообладнання мають відповідати класу приміщення за ПУЕ та бути оснащені засобами захисту від короткого замикання й інших аварійних режимів роботи.

Електроживлення комп’ютерної техніки здійснюється через окрему групову трипровідну мережу, що включає фазний провідник, нульовий робочий та нульовий захисний провідники. Нульовий захисний провідник призначений для заземлення (занулення) електроприймачів.

У разі експлуатації в приміщенні більше п’яти персональних комп’ютерів передбачається встановлення аварійного вимикача для оперативного знеструмлення електромережі, за винятком освітлювальних ліній.

Підключення персональних комп’ютерів до електромережі допускається виключно через справні штепсельні з’єднання промислового виготовлення, оснащені захисним контактом.

Забороняється використання двопровідних електромереж, саморобних подовжувачів, електропроводки з пошкодженою ізоляцією, а також застосування нестандартних або несправних електронагрівальних приладів.

4.2.2 Електробезпека

Для приміщень, у яких експлуатується комп’ютерна техніка, діють вимоги Правил пожежної безпеки в Україні (НАПБ А.01.001-2014), затверджені наказом Міністерства внутрішніх справ України від 30.12.2014 № 1417.

Згідно з класифікацією вибухопожежної небезпеки приміщення належить до категорії Д, що характеризується наявністю негорючих речовин у холодному стані.

Приміщення для зберігання інформаційних носіїв розташовуються окремо та виконуються з використанням негорючих будівельних матеріалів. Звукопоглинальні покриття застосовуються з негорючих або важкогорючих матеріалів.

Приміщення обладнується системою автоматичної пожежної сигналізації та первинними засобами пожежогасіння, зокрема вуглекислотними вогнегасниками з розрахунку два одиниці на кожні 20 м² площі.

Не рідше одного разу на квартал здійснюється очищення комп’ютерного та допоміжного обладнання від пилу з метою зменшення пожежної небезпеки.

Для підвищення рівня пожежної безпеки у приміщенні комп'ютерної фірми передбачено влаштування підлоги з негорючих матеріалів, зберігання паперової документації в металевих шафах, наявність вуглекислотних вогнегасників типу

ОУ-5 і димових пожежних сповіщувачів, а також регулярне проведення вологого прибирання та забезпечення ефективної вентиляції.

4.2.3 Освітлення

Освітлення приміщень, у яких експлуатуються персональні комп’ютери, організовується з використанням природного та штучного світла відповідно до вимог ДБН В.2.5-28:2018 «Природне і штучне освітлення». Допускається застосування змішаної системи освітлення.

Штучне освітлення виконується у вигляді системи загального рівномірного освітлення, при цьому за необхідності дозволяється використання комбінованого освітлення з додатковими місцевими світильниками.

Для загального освітлення передбачається встановлення світильників, оснащених розсіювачами світла та високочастотними пускорегулювальними апаратами; застосування світильників без розсіювачів не допускається.

Як джерела світла доцільно використовувати люмінесцентні лампи типу ЛБ, що забезпечують нормативний рівень освітленості на робочій поверхні в межах 300–500 лк.

Розміщення світильників здійснюється з бокової сторони відносно робочого місця та паралельно напрямку зору працівника з метою запобігання виникненню відблисків і засліплення.

4.2.4 Захист від шуму та вібрації

Підвищений рівень шуму негативно впливає на функціональний стан організму людини та може знижувати працездатність. Допустимі рівні шуму в приміщеннях, оснащених персональними комп’ютерами, регламентуються ДСН 3.3.6.037-99 і становлять 45–55 дБА для приміщень розумової праці та не більше 65 дБА під час роботи з комп’ютерною технікою.

Основними джерелами шуму є вентилятори системних блоків, принтери та інше периферійне обладнання. Для зменшення шумового навантаження

передбачається розміщення друкувальних пристроїв в окремих приміщеннях, використання звукоізолювальних екранів, а також застосування шумопоглинальних матеріалів, виконаних з негорючих або важкогорючих матеріалів.

Вібрація також належить до шкідливих фізичних факторів і може негативно впливати на стан здоров'я працівників. Допустимі рівні вібрації не повинні перевищувати нормативних значень, установлених ДСН 3.3.6.039-99.

З метою зниження впливу вібраційних навантажень застосовуються амортизувальні прокладки під обладнанням, а також раціональне планування розміщення робочих місць і технічних засобів.

4.2.5 Виробничий мікроклімат

Приміщення, у яких експлуатується комп'ютерна техніка, оснащуються системами опалення, кондиціювання повітря або припливно-витяжною вентиляцією з метою забезпечення нормативних параметрів повітряного середовища.

Показники мікроклімату на робочих місцях із використанням персональних комп'ютерів повинні відповідати вимогам ДСН 3.3.6.042-99 та ДСТ 12.1.005-88, що наведені в таблицях 4.2 і 4.3.

Таблиця 4.2 – Нормовані значення параметрів мікроклімату для приміщень з відеодисплейними терміналами та персональними комп'ютерами.

Пора року	Категорія робіт	Температура повітря, °С оптимальна	Відносна вологість повітря, % оптимальна	Швидкість руху повітря, м/с оптимальна
Холодна	Легка - 1а	22–24	40–60	0,1
Холодна	Легка - 1б	21–23	40–60	0,1
Тепла	Легка - 1а	23–25	40–60	0,1
Тепла	Легка - 1б	22–24	40–60	0,2

Таблиця 4.3 – Рівні іонізації повітря приміщень при роботі на ВДТ та ПК

Рівні	Кількість іонів в 1 см³ повітря n^+	n^-
Мінімально необхідні	400	600
Оптимальні	1500–3000	3000–5000
Максимально допустимі	50000	50000

Для підтримки норм необхідно використовувати установки зволоження, іонізації або кондиціювання.

4.2.6 Організація робочого місця

Робоче місце визначається як зона постійного або тимчасового перебування працівника під час виконання трудових обов'язків. Організація робочих місць із використанням персональних комп'ютерів повинна відповідати встановленим ергономічним вимогам.

Мінімальна площа, що відводиться на одне робоче місце з персональним комп'ютером, має становити не менше 6 м², а об'єм приміщення – не менше 20 м³.

Розміщення робочих місць здійснюється таким чином, щоб природне освітлення надходило збоку від працівника, переважно з лівої сторони.

Під час планування робочих місць необхідно дотримуватися таких нормативних відстаней:

– від стін із віконними прорізами – не менше 1 м;

– між бічними поверхнями відеомоніторів – не менше 1,2 м;

– між тильною поверхнею одного монітора та екраном іншого – не менше 2,5 м;

– ширина проходів між рядами робочих місць – не менше 1 м.

Висота робочої поверхні столу повинна регулюватися або перебувати в межах 680–800 мм. Робочий стілець має бути підйомно-поворотного типу з

можливістю регулювання висоти, кута нахилу сидіння та спинки, а також оснащений підлокітниками.

Екран монітора розміщується на відстані не менш як 600 мм від органів зору користувача. Клавіатура встановлюється на відстані 100–300 мм від краю стільниці з кутом нахилу в межах 5–15°.

4.3 Заходи щодо зменшення впливу шкідливих факторів роботи за персональним комп'ютером

Для зменшення впливу небезпечних і шкідливих факторів у приміщенні необхідно здійснити ряд заходів:

– Для правильної подачі світла і правильної освітленості робочих місць слід використовувати світильники типу УПМ. Кожний світильник комплектувати 2 лампами. Розташувати світильники двома паралельними рядами по три в кожному ряді і один світильник перпендикулярно двом паралельним усередині приміщення.

– Зниження впливу шуму на робітників відділу можна досягти шляхом більш раціональною розташування джерел шуму у приміщенні або зменшення рівня шуму у джерелі його виникнення, використання додаткової звукоізоляції і звуко погашення, впровадження малошумного обладнання(більш сучасного) і правильного планування режиму праці і відпочинку робітників.

– Зменшити ймовірність виникнення пожежі у відділі можна шляхом зменшення документообігу, наприклад, повністю автоматизувати обліковоаналітичний процес, при якому значна кількість документів буде залишатись в електронній формі.

– Для забезпечення безпечної для робітників відділу, безаварійної роботи електроприладів, необхідно поряд з положенням, виконанням, обладнанням так організувати їх експлуатацію, щоб повністю виключити будь-яку ймовірність помилок з боку робітників. Для організації безпечної експлуатації електроприладів необхідно підвищити технічну грамотність і свідому дисципліну робітників, які зобов'язані чітко дотримуватись організаційних і технічних заходів

у відповідності з Правилами технічної експлуатації електроприладів споживачами і Правилами техніки безпеки при експлуатації електроприладів споживачами.

– Знизити вплив електромагнітних випромінювань на співробітників при роботі на комп’ютері можна шляхом встановлення захисних екранів. Такий спосіб захисту є дуже ефективним і може бути досягнутий шляхом раціонального вибору конструкції екрану і матеріалу для екранування.

– Наявність пилу, озону, можна знищити шляхом обладнання робочого приміщення системою кондиціонування повітря.

Всі ці заходи у сукупності сприяють злагодженій, безперебійній роботі відділу розробки програмного забезпечення, дозволять підвищити продуктивність праці робітників.

5 ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

5.1 Вступ

В останні роки все частіше поставала проблема охорони навколишнього середовища. Поступово вона перетворилася на глобальну та з кожним днем набуває ще більшої актуальності. Причиною такого положення являються різноманітні антропогенні фактори. Це і демографічний вибух, і зростаюча швидкість урбанізації, і збільшення темпів та об'ємів виробництва та багато іншого. Великий вплив на навколишнє середовище здійснює людський фактор: викиди відходів, забруднення водоймищ та лісів, збільшення навантаження на орні землі тощо – усе це результати діяльності людини. Вже сьогодні у світі знищено більше 20 мільйонів гектарів орних земель. На атомних електростанціях накопичено тисячі тонн відпрацьованого ядерного палива, десятки тисяч кубометрів твердих і десятки мільйонів літрів рідких радіоактивних відходів. Понад 70 млн. $м^3$ радіоактивних відходів зосереджено у відвалах та сховищах уранової, гірничодобувної та переробної промисловості України. Особливу небезпеку для здоров'я людини викликає забруднення води побутовими та промисловими стоками. Окрім забруднення отруйними речовинами та важкими металами така вода містить збудники різноманітних захворювань та зовсім непридатна для постачання населенню.

Внаслідок цього були розроблені міри для збереження навколишнього середовища, які включають в себе комплекс заходів з охорони атмосфери, водних ресурсів, флори, фауни та геологічного середовища. Таким чином, охорона навколишнього середовища – це комплексні заходи з раціонального використання природних ресурсів, їх збереження та примноження, а також забезпечення екологічної безпеки. Основної їх метою є зведення до мінімуму шкідливого впливу людини на оточуючий світ.

5.2 Забруднення навколишнього середовища в процесі використання комп’ютерної техніки

Використання комп’ютерної техніки в процесі професійної діяльності супроводжується низкою факторів, що можуть опосередковано впливати на стан навколишнього середовища. До таких факторів належать електромагнітні випромінювання, які створюються персональними комп’ютерами та периферійними пристроями під час їх функціонування. Дослідження в цій галузі свідчать про необхідність урахування даного виду впливу в системі заходів з охорони навколишнього середовища.

Електромагнітні поля, що виникають у процесі роботи комп’ютерної техніки, поширюються в навколишньому просторі та можуть накопичуватися в приміщеннях із високою концентрацією електронного обладнання. За відсутності належної організації робочих місць і технічних засобів захисту це може призводити до підвищення рівня техногенного навантаження на виробниче та офісне середовище.

Окрім електромагнітного впливу, експлуатація комп’ютерної техніки пов’язана зі споживанням електроенергії, що опосередковано зумовлює збільшення викидів забруднюючих речовин в атмосферу на етапах її виробництва. Також важливим екологічним аспектом є утворення електронних відходів унаслідок морального та фізичного зношення комп’ютерного обладнання, які за неналежної утилізації можуть негативно впливати на ґрунти та водні ресурси.

З метою зменшення негативного впливу комп’ютерної техніки на навколишнє середовище доцільним є впровадження енергоефективних пристроїв, дотримання екологічних стандартів під час експлуатації обладнання, раціональна організація робочих приміщень, а також забезпечення збору та утилізації електронних відходів відповідно до вимог екологічного законодавства. Реалізація зазначених заходів сприятиме зниженню техногенного навантаження та забезпеченню сталого розвитку.

5.3 Заходи щодо зменшення забруднення навколишнього середовища під час використання персональних комп’ютерів

У сучасних умовах цифровізації практично всі сфери діяльності людини пов’язані з використанням персональних комп’ютерів. Хоча комп’ютерна техніка не належить до джерел прямого забруднення довкілля у традиційному розумінні, її масове застосування формує суттєвий опосередкований вплив на навколишнє середовище. Цей вплив проявляється через споживання електричної енергії, утворення електронних відходів, використання природних ресурсів на етапах виробництва обладнання, а також через локальне техногенне навантаження у приміщеннях із високою концентрацією електронних пристроїв.

У зв’язку з цим у розділі охорони навколишнього середовища доцільно розглянути комплекс заходів, спрямованих на зменшення негативного впливу персональних комп’ютерів на довкілля протягом усього життєвого циклу комп’ютерної техніки – від експлуатації до утилізації:

– Зменшення електромагнітного навантаження на навколишнє середовище

Під час роботи персональні комп’ютери та периферійні пристрої створюють електромагнітні поля, які поширюються в навколишньому просторі. У приміщеннях із великою кількістю комп’ютерної техніки можливе підвищення загального електромагнітного фону, що розглядається як один із чинників техногенного навантаження на середовище.

З метою зменшення електромагнітного впливу доцільно застосовувати сучасні комп’ютерні пристрої, які відповідають чинним міжнародним та національним стандартам електромагнітної безпеки. Важливе значення має правильне планування розміщення робочих місць, дотримання рекомендованих відстаней між комп’ютерами, а також раціональна організація кабельних систем живлення та передавання даних. Використання сертифікованих моніторів і блоків живлення сприяє зниженню рівня електромагнітних випромінювань та зменшенню негативного впливу на навколишнє середовище.

– Підвищення енергоефективності комп’ютерної техніки

Одним із ключових екологічних аспектів експлуатації персональних комп’ютерів є споживання електроенергії. Виробництво електричної енергії, особливо на теплових електростанціях, супроводжується викидами парникових газів та інших забруднюючих речовин в атмосферу. Таким чином, зменшення енергоспоживання комп’ютерної техніки опосередковано сприяє скороченню негативного впливу на довкілля.

Ефективним заходом є впровадження енергоощадних комп’ютерів і моніторів, що відповідають сучасним стандартам енергоефективності. Використання режимів енергозбереження, автоматичного переходу в сплячий режим у періоди простою, а також вимкнення обладнання після завершення роботи дозволяє суттєво знизити обсяги споживаної електроенергії. Раціональна експлуатація комп’ютерної техніки є важливою складовою екологічно відповідального використання інформаційних технологій.

– Раціональне використання матеріальних ресурсів

Виробництво персональних комп’ютерів пов’язане з використанням значної кількості природних ресурсів, зокрема металів, пластмас та рідкоземельних елементів. Тому подовження строку служби комп’ютерної техніки є важливим екологічним завданням.

Доцільним є своєчасне технічне обслуговування обладнання, модернізація окремих компонентів замість повної заміни комп’ютера, а також використання програмних засобів, що не потребують надмірних апаратних ресурсів. Такі заходи сприяють зменшенню обсягів виробництва нової техніки та, відповідно, зниженню навантаження на навколишнє середовище.

– Управління електронними відходами

Однією з найсерйозніших екологічних проблем, пов’язаних з використанням персональних комп’ютерів, є утворення електронних відходів. Відпрацьоване комп’ютерне обладнання містить небезпечні речовини, які за неналежної утилізації можуть забруднювати ґрунти, поверхневі та підземні води.

З метою зменшення негативного впливу електронних відходів необхідно забезпечити їх роздільний збір, передачу спеціалізованим підприємствам для переробки або утилізації, а також дотримання вимог екологічного законодавства.

Повторне використання окремих компонентів та вторинна переробка матеріалів сприяють збереженню природних ресурсів і зменшенню обсягів відходів.

– Організаційні та нормативні заходи

Важливу роль у зменшенні екологічного впливу комп’ютерної техніки відіграють організаційні заходи. До них належать розроблення внутрішніх інструкцій з екологічно безпечної експлуатації комп’ютерів, проведення інструктажів для працівників, а також контроль за дотриманням екологічних вимог.

Дотримання чинних нормативів і стандартів у сфері охорони навколишнього середовища, охорони праці та енергозбереження дозволяє забезпечити комплексний підхід до мінімізації негативного впливу персональних комп’ютерів на довкілля. Реалізація зазначених заходів сприяє формуванню екологічно відповідального підходу до використання інформаційних технологій та забезпеченню сталого розвитку.

ВИСНОВКИ

Кваліфікаційна робота присвячена вирішенню актуального завдання, а саме розробці системи підтримки прийняття рішень для вибору технологічного стеку вебзастосунку.

Метою роботи було дослідження методів багатокритеріального вибору та розробка системи підтримки прийняття рішень для вибору технологічного стеку вебзастосунків шляхом порівняння альтернативних рішень за сукупністю наявних критеріїв.

Для досягнення мети вирішено наступні завдання:

– Виконаний аналіз сучасного стану проблеми вибору технологічного стеку вебзастосунків.

– Досліджені методи багатокритеріального вибору альтернатив.

– Обраний метод багатокритеріального вибору для розробки системи підтримки прийняття рішень, а саме метод аналізу ієрархій.

– Виконана постановка задачі на розробку системи підтримки прийняття рішень для вибору технологічного стеку вебзастосунку.

– Розроблені проєктні ряшення для СППР вибору технологічного стеку вебзастосунку, а саме:

- розроблені загальносистемні рішення для СППР;
- розроблені рішення з інформаційного забезпечення для СППР;
- розроблені рішення з програмного забезпечення для СППР.

Також в кваліфікаційній роботі виконані розділи з економіки, охорони праці та охорони навколишнього середовища.

Мета роботи досягнута. Усі завдання виконано.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Doyle B., Lopes C.V. Survey of Technologies for Web Application Development // *arXiv preprint*. – 2008. URL: https://arxiv.org/abs/0801.2618 (дата звернення: 12.11.2025).
2. Технологічні стеки вебзастосунків // ІнфДев. URL: https://infdev.com.ua/docs/web-development/technology-stacks/ (дата звернення: 12.11.2025).
3. Full Stack Development: What It Is & How to Become a Full Stack Developer // Coursera. URL: https://www.coursera.org/articles/full-stack-developer (дата звернення: 10.11.2025).
4. Azhar N. A. та ін. Multi-criteria Decision Making: A Systematic Review // *Recent Advances in Electrical & Electronic Engineering*. – 2021. – Vol. 14. URL: https://eurekaselect.com/public/article/118613 (дата звернення: 12.11.2025).
5. Comparative Analysis of PROMETHEE, TOPSIS and ELECTRE // *Sciety preprint*. URL: https://sciety.org/articles/activity/10.21203/rs.3.rs-7870462/v1 (дата звернення: 12.11.2025).
6. Міхно О. М., Патракеєв І. М., Левінськова Н. М. Багатокритеріальний вибір альтернатив на основі методу аналізу ієрархій // Вісник Київського національного університету імені Тараса Шевченка. – 2023. – № 56. – С. 57–63. DOI: https://doi.org/10.17721/1728-2217.2023.56.57-63
7. Saaty T.L. The Analytic Hierarchy Process. – New York: McGraw-Hill, 1980. – 287 p.
8. Сачук В.В., Вавринюк К.І., Хиць Р.В. Застосування методів багатокритеріального аналізу для підтримки прийняття управлінських рішень в умовах невизначеності // Вісник Хмельницького національного університету. Серія: Технічні науки. – 2025. – Т. 359, № 6.2. – С. 142–148. URL: https://heraldts.khmnu.edu.ua/index.php/heraldts/article/view/2331/2267 (дата звернення: 10.11.2025).
9. Django та фреймворки для вебзастосунків // Селдон Новини. URL: https://surl.lt/psdatu (дата звернення: 12.11.2025).

10. Алхімова С.М. Об’єктно-орієнтоване програмування : навч. посіб. Ч. 2. – Київ : КПІ ім. Ігоря Сікорського, 2019. – 194 с.
11. Цибульник С.О., Барандич К.С. Технології розроблення програмного забезпечення. Ч. 1. Життєвий цикл програмного забезпечення: навч. посіб. – Київ: КПІ ім. Ігоря Сікорського, 2022. – 270 с.
12. Авраменко В.С., Авраменко А.С. Проєктування інформаційних систем. – Черкаси : ЧНУ ім. Богдана Хмельницького, 2017. – 433 с.
13. Eck D.J. Introduction to Programming Using Java : електронний підручник. URL: https://open.umn.edu/opentextbooks/textbooks/419 (дата звернення: 12.11.2025).
14. IntelliJ IDEA: інтегроване середовище розробки. URL: https://www.jetbrains.com/help/idea/discover-intellij-idea.html (дата звернення: 12.11.2025).
15. Фролов А., Фролов Г. Практика застосування Perl, PHP, Apache, MySQL для активних веб-сайтів. – Київ: Інтернет-технології, 2002. – 576 с.
16. Козловський В.О. Техніко-економічні обґрунтування та економічні розрахунки в дипломних проєктах і роботах. – Вінниця : ВДТУ, 2003. – 76 с.
17. Закон України «Про охорону праці» від 14.10.1992 № 2694-XII. URL: https://zakon.rada.gov.ua/laws/show/2694-12 (дата звернення: 10.11.2025).

ДОДАТОК А – Технічне завдання

Вступ

Повне найменування системи: Система підтримки прийняття рішень для вибору технологічного стеку вебзастосунку.

Коротка характеристика галузі застосування: дана система може бути застосована розробниками вебзастосунків при виборі технологічного стеку вебзастосунку.

1 Підстави для розробки

Розробка виконується на підставі завдання на кваліфікаційну роботу.

2 Призначення розробки

2.1 Функціональне призначення системи

Функціональним призначенням СППР є додавання, видалення, перегляд критеріїв, додавання, видалення, перегляд моделей (технологічних стеків), додавання, видалення, перегляд оцінок важливості критеріїв, додавання, видалення, перегляд оцінок відношення за критеріями, перегляд кінцевих результатів.

2.2 Експлуатаційне призначення системи

Експлуатаційним призначенням є спрощення обгрунтованого вибору технологічного стеку.

3 Вимоги до програмного виробу

3.1 Вимоги до функціональних характеристик

3.1.1 Вимоги до складу виконуваних функцій:

Система повинна забезпечувати можливість виконання нижче перерахованих функцій:

- додавання оцінки відношення за критерієм;

- видалення оцінки відношення за критерієм;
- перегляд оцінок відношення за критерієм;
- додавання критеріїв;
- видалення критеріїв;
- перегляд критеріїв;
- додавання моделі (технологічного стеку);
- видалення моделі (технологічного стеку);
- перегляд моделей (технологічних стеків);
- додавання оцінки важливості критеріїв;
- видалення оцінки важливості критеріїв;
- перегляд оцінок важливості критеріїв;
- перегляд кінцевих результатів.

3.1.2 Вимоги до організації вхідних та вихідних даних

До вхідних та вихідних даних відноситься інформація про моделі (технологічні стеки), оцінки важливості критеріїв, оцінки відношення за критеріями, критерії, ваги критеріїв, ваги моделей за критеріями, глобальні пріоритети.

3.2 Вимоги до надійності

Збереження бази даних повинно відбуватися за встановленим графіком на інший носій.

У разі введення користувачем некоректної інформації система повинна повідомити про помилку і надати можливість виправити її.

3.3 Вимоги до умов експлуатації

Кліматичні умови експлуатації, за яких забезпечується дотримання заданих характеристик, мають відповідати нормативним вимогам, встановленим для технічних засобів з урахуванням умов їх застосування.

3.4 Вимоги до складу та параметрів технічних засобів

Для забезпечення коректної та стабільної роботи системи рекомендовано використовувати апаратно-програмні засоби з такими характеристиками:

– роздільна здатність екрана – не менше 1366×768 px;
– обсяг оперативної пам’яті – не менше 2 ГБ;
– підключення до мережі Інтернет зі швидкістю не менше 10 Мбіт/с.

3.5 Вимоги до інформаційної та програмної сумісності

Система висуває такі вимоги до програмного середовища користувача:

– операційна система: Microsoft Windows 7 чи вище, або сучасні дистрибутиви Linux;

– інтернет-браузер: Google Chrome, Mozilla Firefox, Microsoft Edge або Safari – останні стабільні версії.

3.6 Вимоги до захисту інформації та системи

Вимоги до захисту інформації та системи не висуваються.

3.7 Вимоги до маркування та пакування

Вимоги до маркування та пакування не висуваються.

Поширення даного продукту на фізичних носіях не передбачається.

3.7 Вимоги до транспортування та зберігання

Вимоги до транспортування не висуваються у зв'язку з відсутністю фізичних носіїв.

4 Вимоги до системної документації

Системна документація повинна містити:

– технічне завдання;
– інструкцію користувача;
– програму та методику випробувань;

– текст системи;

– опис системи.

5 Техніко-економічні показники

Згідно з розрахунками, наведеними в 3 розділі роботи, окупність ситеми становить 4 місяці впродовж першого року використання. Це свідчить про високу рентабельність проєкту

6 Стадії та етапи розробки

Стадії та етапи розробки представлені нижче в таблиці А.1.

Таблиця А.1 – Стадії та етапи розробки СППР

Стадії розробки	Етапи розробки	Дата початку	Дата завершення
1 Технічне завдання	1.1 Обґрунтування необхідності розробки системи	01.09.2025	07.09.2025
	1.2 Розробка технічного завдання	08.09.2025	20.09.2025
	1.3 Затвердження технічного завдання	21.09.2025	25.09.2025
2 Загальносистемні рішення	2.1 Розробка загальносистемних рішень	26.09.2025	10.10.2025
	2.2 Затвердження загальносистемних рішень	11.10.2025	15.10.2025
3 Рішення з інформаційного забезпечення	3.1 Розробка рішень з інформаційного забезпечення	16.10.2025	05.11.2025
	3.2 Затвердження рішення з інформаційного забезпечення	06.11.2025	10.11.2025
4 Рішення з програмної системи	4.1 Розробка рішень з програмної системи	11.11.2025	30.11.2025
	4.2 Затвердження рішення з програмної системи	01.12.2025	06.12.2025
	4.3 Розробка програмної системи	07.12.2025	10.12.2025

7 Порядок контролю та приймання

Контроль за аналізом та проєктуванням кожної окремої складової СППР відбувається на кожному етапі враховуючи вимоги, що визначені в технічному завданні.

У випадку виявлення помилок під час процедури приймання системи підтримки прийняття рішень, складається акт щодо виявлених недоліків. Цей акт підписують представники замовника і розробника і затверджують керівники обох організацій – замовника та розробника. Розробник зобов'язаний усунути виявлені помилки протягом не більше ніж за 2 тижні і повідомити замовника про проведення повторної перевірки.

ДОДАТОК Б – Опис СППР

1 Загальні відомості

Найменування системи: Система підтримки прийняття рішень для вибору технологічного стеку вебзастосунку.

Коротка назва: СППР.

Застосування: СППР може бути застосована розробниками вебзастосунків при виборі технологічного стеку вебзастосунку.

2 Функціональне призначення

СППР повинна надавати наступні функціональні можливості:

- додавання оцінки відношення за критерієм;
- видалення оцінки відношення за критерієм;
- перегляд оцінок відношення за критерієм;
- додавання критеріїв;
- видалення критеріїв;
- перегляд критеріїв;
- додавання моделі (технологічного стеку);
- видалення моделі (технологічного стеку);
- перегляд моделей (технологічних стеків);
- додавання оцінки важливості критеріїв;
- видалення оцінки важливості критеріїв;
- перегляд оцінок важливості критеріїв;
- перегляд кінцевих результатів.

3 Вимоги до складу та параметрів технічних засобів

Для забезпечення коректної та стабільної роботи системи рекомендовано використовувати апаратні засоби з такими характеристиками:

- роздільна здатність екрана – не менше 1366×768 px;
- обсяг оперативної пам’яті – не менше 2 ГБ;

– підключення до мережі Інтернет зі швидкістю не менше 10 Мбіт/с.

4 Вимоги до організації вхідних та вихідних даних

До вхідних та вихідних даних відноситься інформація про моделі (технологічні стеки), оцінки важливості критеріїв, оцінки відношення за критеріями, критерії, ваги критеріїв, ваги моделей за критеріями, глобальні пріоритети.

ДОДАТОК В – Програма та методика випробувань

1 Об’єкт випробовування

Об’єктом для випробувань є Система підтримки прийняття рішень для вибору технологічного стеку вебзастосунку.

Сфера використання: дана система може бути застосована розробниками вебзастосунків при виборі технологічного стеку вебзастосунку.

2 Мета випробовування

Метою випробувань є перевірка працездатності системи та перевірка відповідності її характеристик і функціональних можливостей вимогам, визначеним у технічному завданні.

3 Вимоги до програми

Під час випробування СППР потрібно перевірити, чи відповідають її функціональні характеристики вимогам, які визначені та описані у розділі «Вимоги до функціональних характеристик» технічного завдання.

СППР повинна надавати наступні функціональні можливості:

- додавання оцінки відношення за критерієм;
- видалення оцінки відношення за критерієм;
- перегляд оцінок відношення за критерієм;
- додавання критеріїв;
- видалення критеріїв;
- перегляд критеріїв;
- додавання моделі (технологічного стеку);
- видалення моделі (технологічного стеку);
- перегляд моделей (технологічних стеків);
- додавання оцінки важливості критеріїв;
- видалення оцінки важливості критеріїв;
- перегляд оцінок важливості критеріїв;

– перегляд кінцевих результатів.

4 Вимоги до програмної документації

Програмна документація повинна містити наступні документи:

– технічне завдання;
– інструкцію користувача;
– програму та методику випробувань;
– текст системи;
– опис системи

5 Засоби та порядок випробувань

Програмні засоби випробувань

До програмних засобів, необхідних для випробувань належать: ОС Windows версії не нижче 7; інтернет браузер Google Chrome.

Апаратні засоби випробувань

Для проведення випробувань рекомендовано використовувати апаратні засоби з такими характеристиками:

– роздільна здатність екрана – не менше 1366×768 px;
– обсяг оперативної пам’яті – не менше 2 ГБ;
– підключення до мережі Інтернет зі швидкістю не менше 10 Мбіт/с.

Порядок проведення випробувань

Проведення випробування системи підтримки прийняття рішень повинно відбуватися в наступному порядку:

− виконуються інструкції до кожної функції;

− робиться аналіз отриманих результатів і встановлюється відповідність системи вимогам і системним документам.

При виявлені помилок у роботі системи складається їх перелік і обговорюється термін їх виправлення розробником. Після цього замовник проводить повторне тестування в повному обсязі.

6 Методика випробувань

Випробування проводиться за стратегією «чорного ящика». У таблиці В.1 наведено програму випробувань.

Таблиця В.1 – Програма випробувань

№	Функція, що перевіряється	Очікуваний результат
1	Додання оцінки відношення за критерієм	Нова оцінки відношення за критерієм з’явиться у списку оцінок відношення за критерієм
2	Видалення оцінки відношення за критерієм	Оцінка зникне зі списку оцінок відношення за критерієм
3	Перегляд оцінок відношення за критерієм	Надається список оцінок відношення за критерієм
4	Перегляд кінцевих результатів	Надаються таблиці з оцінками за критеріями, а також кінцева таблиця з оцінками по кожній моделі.
5	Додання критерію	Новий критерій з’явиться у списку переліку критеріїв
6	Видалення критерію	Критерій зникне зі списку переліку критеріїв
7	Перегляд критеріїв	Надається список критеріїв
8	Додання оцінки важливості критеріїв	Нова оцінки важливості критеріїв з’явиться у списку оцінок важливості критеріїв
9	Видалення оцінки важливості критеріїв	Оцінка зникне зі списку переліку оцінок важливості критеріїв.
10	Перегляд оцінок важливості критеріїв	Надається список оцінок важливості критеріїв
11	Перегляд головної сторінки	Після завантаження застосунку користувач перенаправляється на головну сторінку
12	Додання моделі	Нова модель з’явиться у списку переліку моделей
13	Видалення моделі	Модель зникне зі списку переліку моделей
14	Перегляд моделей	Надається список моделей

ДОДАТОК Г – Інструкція користувача

Відкривши веб-додаток на головній сторінці (рисунок Г.1), побачимо головне меню з 5 пунктів:

- Моделі
- Критерії
- Важливість критеріїв
- Оцінки
- Результати.

Рисунок Г.1 – Головна сторінка

Для того щоб додати новий критерій, або ж переглянути існуючі чи видалити, потрібно перейти на посилання «Критерії», рисунок Г.2, для додання написати в пустому рядку бажаний критерій, після чого натиснути «Додати». Для видалення треба натиснути кнопку «вилучити» навпроти вже існуючого критерію.

Щоб повернутись на головну сторінку натисніть «На головну».

Рисунок Г.2 – Сторінка «Критерії»

Для того щоб додати нову оцінку важливості критеріїв, або ж переглянути існуючі чи видалити, потрібно перейти на посилання «Важливість критеріїв», рисунок Г.3, для додання вибрати в пустому рядку бажані критерії, ввести оцінку та натиснути «Додати». Для видалення треба натиснути кнопку «вилучити» навпроти вже існуючої оцінки важливості критеріїв.

Щоб повернутись на головну сторінку натисніть «На головну».

Рисунок Г.3 – Сторінка «Важливість критеріїв»

Аналогічні дії відбуваються із іншими посиланнями окрім «Результати».

Щоб переглянути результати або ж проставлені оцінки по критеріям, треба перейти на посилання «Результати». Найкращий варіант вибору буде показаний в останній таблиці із найбільшим значенням навпроти моделі.

Виставлення оцінок

У таблиці Г.1 представлена шкала відносної важливості Сааті, по якій виставляються оцінки в методі аналізу ієрархій.

Таблиця Г.1 – Шкала відносної важливості Сааті

Значення	Вербальна інтерпретація	Пояснення
1	Рівна важливість	Обидва елементи мають однаковий вплив на досягнення мети
3	Помірна перевага	Один елемент незначно переважає інший
5	Сильна перевага	Один елемент істотно важливіший за інший
7	Дуже сильна перевага	Перевага одного елемента є домінуючою
9	Абсолютна перевага	Один елемент має надзвичайно високу важливість
2, 4, 6, 8	Проміжні значення	Використовуються для компромісних оцінок між основними рівнями
1/3, 1/5, 1/7, 1/9	Обернені значення	Використовуються, якщо другий елемент переважає перший

ДОДАТОК Д – Текст системи

Текст програми наводиться нижче.

```
table {
    background-color: gray;
}

th {
    background-color: aliceblue;
    text-align: left;
}

td {
    background-color: white;
}

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Пріоритети параметрів</title>
    <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
<ul>
    <li>
        <a href="index.jsp">На головну</a>
    </li>
</ul>
<table>
    <thead>
    <tr><th>Критерій 1</th><th>Критерій
2</th><th>відношення</th><th></th></tr>
    </thead>
    <tbody>
    <c:forEach items="${priorities}" var="priority">

<tr><td>${priority.p1.name}</td><td>${priority.p2.name}</td><td>${pr
iority.value}</td><td><a
href="delparameterpriority.html?id=${priority.id}">вилучити</a></td>
</tr>
    </c:forEach>
    </tbody>
</table>
<p></p>
```

```
<form action="newpriority.html" method="post">
    <select name="p1">
        <c:forEach items="${parameters}" var="parameter">
            <option
value="${parameter.id}">${parameter.name}</option>
        </c:forEach>
    </select>
    <select name="p2">
        <c:forEach items="${parameters}" var="parameter">
            <option
value="${parameter.id}">${parameter.name}</option>
        </c:forEach>
    </select>
    <input type="text" name="value">
    <input type="submit" value="Додати">
</form>
</body>
</html>

package server;

import dao.MarkDAO;
import dao.ParameterDAO;
import dao.ParameterPriorityDAO;
import dao.ModelDAO;

import javafx.scene.layout.Priority;
import tables.Mark;
import tables.Parameter;
import tables.ParameterPriority;
import tables.Model;

import javax.ejb.EJB;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.*;


@WebServlet(name = "MainServlet", urlPatterns = {"*.html"})
public class MainServlet extends HttpServlet {
    @EJB
    ModelDAO modelDAO;
    @EJB
    MarkDAO markDAO;
    @EJB
```

```
    ParameterDAO parameterDAO;
    @EJB
    ParameterPriorityDAO parameterPriorityDAO;


    protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
        if (request.getServletPath().endsWith("/addmodel.html")) {
            String modelname = request.getParameter("modelname");
            modelDAO.create(modelname);
            response.sendRedirect("models.html");
        } else if
(request.getServletPath().endsWith("/newmark.html")) {
            String paramname = request.getParameter("paramname");
            String model1 = request.getParameter("model1");
            String model2 = request.getParameter("model2");
            String strMark = request.getParameter("mark");
            int idParam = Integer.parseInt(paramname);
            int idModel1 = Integer.parseInt(model1);
            int idModel2 = Integer.parseInt(model2);
            double mark = Double.parseDouble(strMark);
            markDAO.add(mark, idParam, idModel1, idModel2);
            response.sendRedirect("addmark.html");
        } else if
(request.getServletPath().endsWith("/createparams.html")) {
            String paramname = request.getParameter("paramname");
            parameterDAO.create(paramname);
            response.sendRedirect("params.html");
        } else if
(request.getServletPath().endsWith("/newpriority.html")) {
            String sp1 = request.getParameter("p1");
            String sp2 = request.getParameter("p2");
            String svalue = request.getParameter("value");
            int idP1 = Integer.parseInt(sp1);
            int idP2 = Integer.parseInt(sp2);
            double value = Double.parseDouble(svalue);
            parameterPriorityDAO.create(idP1,idP2,value);
            response.sendRedirect("priority.html");
        }
    }

    private void processAllResults(HttpServletRequest request,
HttpServletResponse response) throws IOException, ServletException {
//
request.getRequestDispatcher("/test.jsp").forward(request,response);
        List<Parameter> parameters = parameterDAO.findAll();
        List<Model> models = modelDAO.findAll();
        List<String> strs = new ArrayList<>();
        for (Parameter p : parameters) {
            strs.add(processParameter(p));
        }
```

```
        List<double[][]> matrixes = new ArrayList<>();
        for (Parameter p: parameters) {
            matrixes.add(fillMatrix(p));
        }
        List<double[]> marksByParameter = new ArrayList<>();
        for (double[][] m : matrixes) {
            double[] res = new double[m.length];
            double s = 0;
            for (int i=0; i<m.length; i++) {
                double p = 1;
                for (int j=0; j<m[i].length; j++) {
                    p *= m[i][j];
                }
                res[i] = Math.pow(p,1.0/m[i].length);
                s += res[i];
            }
            for (int i=0; i<res.length; i++) {
                res[i] /= s;
            }
            marksByParameter.add(res);
        }
        int size = parameters.size();
        double[][] pw = new double[size][size];
        for (int i = 0; i < size; i++) {
            Parameter p1 = parameters.get(i);
            for (int j=0; j < size; j++) {
                Parameter p2 = parameters.get(j);
                if (i==j) {
                    pw[i][j] = 1;
                } else {
                    pw[i][j] = findPriority(p1,p2);
                }
            }
        }
        double[] sw = new double[pw.length];
        double ssw = 0;
        for (int i = 0; i < pw.length; i++) {
            sw[i] = 1;
            for (int j = 0; j < pw[i].length; j++) {
                sw[i] *= pw[i][j];
            }
            sw[i] = Math.pow(sw[i], 1.0/sw.length);
            ssw += sw[i];
        }
        for (int i = 0; i < sw.length; i++) {
            sw[i] /= ssw;
        }

        double[] sumMarks = new double[matrixes.get(0).length];

//        for (int i = 0; i < 3; i++) {
```

```
//             for (int j=0; j < 3; j++) {
//                 sumMarks[i] += pw[i][j]*sw[j];
//             }
//         }
        for (int i = 0; i < sumMarks.length; i++) {
            for (int j=0; j<sw.length; j++) {
                sumMarks[i] += sw[j] * marksByParameter.get(j)[i];
            }
        }


        response.setCharacterEncoding("UTF8");
        PrintWriter out = response.getWriter();
        out.println("<html>\n" +
                "<head>\n" +
                "    <title>Оцінки</title>\n" +
                "    <link rel=\"stylesheet\" type=\"text/css\"
href=\"styles.css\">\n" +
                "<meta http-equiv=\"content-type\"
content=\"text/html; charset=utf-8\" />" +
                "<meta http-equiv=\"content-language\"
content=\"ru\" />" +
                "</head>\n" +
                "<body>\n" +
                "<ul>\n" +
                "    <li>\n" +
                "        <a href=\"index.jsp\">На головну</a>\n" +
                "    </li>\n" +
                "</ul>");
        for (int i=0; i<strs.size(); i++) {
//         for (int i=0; i<1; i++) {
            out.println(strs.get(i));
        }
        out.println("<p></p>");
        out.println("<table>");
        out.println("<tr><th>Модель</th><th>Оцінка</th></tr>");
        for (int i = 0; i < sumMarks.length; i++) {
            out.println("<tr><td>" + models.get(i).getModel()
+"</td><td>"+sumMarks[i]+"</td></tr>");
        }
        out.println("</table>");
        out.println("</body>\n" +
                "</html>");
        out.flush();
    }

    private double findPriority(Parameter p1, Parameter p2) {
        ParameterPriority p =
parameterPriorityDAO.findPriority(p1,p2);
        return p.getValue();
    }
```

```
    private double[][] fillMatrix(Parameter p) {
        Collection<Mark> marks = p.getMarks();
        Set<Model> models1 = new
TreeSet<>(Comparator.comparing(Model::getId));
        Set<Model> models2 = new
TreeSet<>(Comparator.comparing(Model::getId));
        for (Mark m: marks) {
            models1.add(m.getModel1());
            models2.add(m.getModel2());
        }
        if (models1.size()!=models2.size()) {
            return null;
        }
        List<Model> prods1 = new ArrayList<>(models1);
        List<Model> prods2 = new ArrayList<>(models2);
        double[][] m = getMatrix(p, models1, models2, prods1,
prods2);
        return m;
    }

    private double[][] getMatrix(Parameter p, Set<Model> models1,
Set<Model> models2, List<Model> prods1, List<Model> prods2) {
        double[][] m = new double[models1.size()][models2.size()];
        for (int i=0; i<prods1.size(); i++) {
            Model prod1 = prods1.get(i);
            for (int j=0; j<prods2.size(); j++) {
                Model prod2 = prods2.get(j);
                if (i==j) {
                    m[i][j] = 1;
                } else {
                    m[i][j] = findMark(prod1, prod2, p);
                }
            }
        }
        return m;
    }

    private String processParameter(Parameter p) {
        Collection<Mark> marks = p.getMarks();
        Set<Model> models1 = new
TreeSet<>(Comparator.comparing(Model::getId));
        Set<Model> models2 = new
TreeSet<>(Comparator.comparing(Model::getId));
        for (Mark m: marks) {
            Models1.add(m.getModel1());
            models2.add(m.getModel2());
        }
        if (models1.size()!=models2.size()) {
            return null;
        }
```

```
        List<Model> prods1 = new ArrayList<>(models1);
        List<Model> prods2 = new ArrayList<>(models2);
        double[][] m = getMatrix(p, models1, models2, prods1,
prods2);
        StringBuilder sb = new StringBuilder();
        sb.append("<table>\n");
        sb.append("<tr><th>").append(p.getName()).append("</th>");
        for (Model p2: prods2) {
            sb.append("<th>").append(p2.getModel()).append("</th>");
        }
        sb.append("</tr>\n");
        for (int i = 0; i < prods1.size(); i++) {
            sb.append("<tr>");

sb.append("<th>").append(prods1.get(i).getModel()).append("</th>");
            for (int j=0; j<prods2.size(); j++) {
                sb.append("<td>").append(m[i][j]).append("</td>");
            }
            sb.append("</tr>\n");
        }
        sb.append("</table><br/><br/>\n");
        return sb.toString();
    }

    public double findMark(Model prod1, Model prod2, Parameter p) {
        Mark mark = markDAO.findByModelsAndParam(prod1, prod2, p);
        if (mark!=null) {
            return mark.getValue();
        } else return 0;
    }

    protected void doGet(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
        if (request.getServletPath().endsWith("/models.html")) {
            request.setAttribute("models", modelDAO.findAll());

request.getRequestDispatcher("/models.jsp").forward(request,
response);
        } else if
(request.getServletPath().endsWith("/delmodel.html")) {
            String sid = request.getParameter("id");
            Integer id = Integer.parseInt(sid);
            modelDAO.delete(id);
            response.sendRedirect("models.html");
        } else if
(request.getServletPath().endsWith("/addmark.html")) {
            request.setAttribute("marks", markDAO.findAllMarks());
            request.setAttribute("params", parameterDAO.findAll());
            request.setAttribute("models", modelDAO.findAll());

request.getRequestDispatcher("/marks.jsp").forward(request,
```

```
response);
        } else if
(request.getServletPath().endsWith("/params.html")) {
            request.setAttribute("params", parameterDAO.findAll());

request.getRequestDispatcher("/parameters.jsp").forward(request,
response);
        } else if
(request.getServletPath().endsWith("/delparameter.html")) {
            String sid = request.getParameter("id");
            Integer id = Integer.parseInt(sid);
            parameterDAO.delete(id);
            response.sendRedirect("params.html");
        } else if
(request.getServletPath().endsWith("/result.html")) {
            processAllResults(request,response);
        } else if
(request.getServletPath().endsWith("/priority.html")) {
            request.setAttribute("priorities",
parameterPriorityDAO.findAll());
            request.setAttribute("parameters",
parameterDAO.findAll());

request.getRequestDispatcher("/priorities.jsp").forward(request,resp
onse);
        } else if
(request.getServletPath().endsWith("/delparameterpriority.html")) {
            String sid = request.getParameter("id");
            Integer id = Integer.parseInt(sid);
            parameterPriorityDAO.delete(id);
            response.sendRedirect("priority.html");
        } else if
(request.getServletPath().endsWith("/delmark.html")) {
            String sid = request.getParameter("id");
            Integer id = Integer.parseInt(sid);
            markDAO.delete(id);
            response.sendRedirect("addmark.html");
        }
    }
}



<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Ez Analiz</title>
</head>
<body>
<ul>
    <li>
```

```
        <a href="models.html">Моделі</a>
    </li>
    <li>
        <a href="params.html">Критерії</a>
    </li>
    <li>
        <a href="priority.html">Важливість критеріїв</a>
    </li>
    <li>
        <a href="addmark.html">Оцінки</a>
    </li>
    <li>
        <a href="result.html">Результати</a>
    </li>
</ul>
</body>
</html>

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Моделі</title>
    <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
<ul>
    <li>
        <a href="index.jsp">На головну</a>
    </li>
</ul>
   <form action="addmodel.html" method="post">
    <table>
        <thead>
        <tr><th>id</th><th>Назва</th><th></th></tr>
        </thead>
        <tbody>
        <c:forEach items="${models}" var="Model">
            <tr><td>${model.id}</td><td>${model.model}</td><td><a
href="delmodel.html?id=${model.id}">вилучити</a></td></tr>
        </c:forEach>
        <tr><td>Новий</td><td><input type="text"
name="modelname"></td><td><input type="submit"
value="Додати"></td></tr>
        </tbody>
    </table>
   </form>

</body>
</html>
```

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Критерії</title>
    <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
<ul>
    <li>
        <a href="index.jsp">На головну</a>
    </li>
</ul>
<table>
<form action="createparams.html" method="post">
    <thead>
    <tr><th>id</th><th>Назва</th><th></th></tr>
    </thead>
    <tbody>
    <c:forEach items="${params}" var="parameter">
    <tr><td>${parameter.id}</td><td>${parameter.name}</td><td><a
href="delparameter.html?id=${parameter.id}">вилучити</a></td></tr>
    </c:forEach>
    <tr><td>Новий</td><td><input type="text"
name="paramname"></td><td><input type="submit"
value="Додати"></td></tr>
    </tbody>
</form>
</table>

</body>
</html>

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Оцінки</title>
    <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
<ul>
    <li>
        <a href="index.jsp">На головну</a>
    </li>
</ul>
    <table>
        <thead>
```

```
        <tr><th>критерій</th><th>Модель 1</th><th>Модель
2</th><th>оцінка</th><th></th></tr>
        </thead>
        <tbody>
        <c:forEach items="${marks}" var="mark">

<tr><td>${mark.parameter.name}</td><td>${mark.model1.model}</td><td>
${mark.model2.model}</td><td>${mark.value}</td><td><a
href="delmark.html?id=${mark.id}">вилучити</a></tr>
        </c:forEach>
        </tbody>
    </table>
    <p></p>
    <form action="newmark.html" method="post">
        <select name="paramname">
            <c:forEach items="${params}" var="par">
                <option value="${par.id}">${par.name}</option>
            </c:forEach>
        </select>
        <select name="model1">
            <c:forEach items="${models}" var="model">
                <option value="${model.id}">${model.model}</option>
            </c:forEach>
        </select>
        <select name="model2">
            <c:forEach items="${models}" var="model">
                <option value="${model.id}">${model.model}</option>
            </c:forEach>
        </select>
        <input type="text" name="mark">
        <input type="submit" value="Додати">
    </form>
</body>
</html>
```