МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проєктами
Кафедра програмного забезпечення автоматизованих систем
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма «Інженерія програмного забезпечення»

«Допущений до захисту»
Завідувач кафедри
_____ проф. Приходько С.Б.
«___» _____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти «магістр»

на тему: Нелінійна регресійна модель для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP, та її програмна реалізація

Виконав: студент групи 6151м
_____ Петраков С.О.
(підпис, ПІБ)

Керівник роботи:
доцент кафедри ПЗАС, к.т.н.
(посада, науковий ступень вчене звання)
_____ Пухалевич А.В.
(підпис, ПІБ)

Миколаїв – 2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проєктами
Кафедра програмного забезпечення автоматизованих систем
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма «Інженерія програмного забезпечення»

«ЗАТВЕРДЖУЮ»
Гарант освітньої програми
_____проф. С.Б.Приходько
(підпис)
« 27 » жовтня 2025 р.

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

Студенту Петракову Сергію Олександровичу
(Прізвище, ім'я, по батькові)

1. Тема роботи: Нелінійна регресійна модель для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP, та її програмна реалізація
Керівник роботи к.т.н., доц. Пухалевич А.В.
Затверджені наказом ректора № 1222-УЧ від « 15 » _____10_____ 2025 р.
2. Термін подання роботи: 08.12.2025 р.
3. Вихідні дані по роботі: _____

4. Перелік питань, що належать до розробки (найменування розділів) _____
Вступ (Актуальність теми. Зв'язок роботи з науковими програмами, планами, темами. Мета і завдання дослідження. Об'єкт дослідження. Предмет дослідження. Методи дослідження. Наукова новизна одержаних результатів. Практичне значення одержаних результатів. Особистий внесок здобувача. Апробація результатів досліджень**.** Публікації.); Огляд літератури та обґрунтування необхідності проведення досліджень за обраною темою; Викладення результатів власних досліджень з висвітленням того нового, що пропонується; Розділ з розробки програмного забезпечення; Організаційно-економічний розділ; Розділи з охорони праці; Висновки; Список використаних джерел; Додатки (технічне завдання, текст програми, опис програми, інструкція користувача, програма та методика випробувань програмного забезпечення)

5. Перелік презентаційних матеріалів: 1.Тема, 2. Мета та завдання дослідження. 3. Об'єкт та предмет дослідження, 4. Наукова новизна та практичне значення одержаних результатів, 5. Апробація результатів досліджень та публікації, 6. Аналіз існуючих методів та моделей для оцінювання розміру Java застосунків, 7. Обгрунтування необхідності проведення досліджень за обраною темою, 8. Трьохфакторна нелінійна регресійна модель для оцінювання розміру вебзастосунків на Java, що створюються з використанням технології JSP, 9. Порівняння однофакторної та трьохфакторної нелінійних регресійних моделей, 10. Архітектура програмного забезпечення, 11. Діаграма варіантів використання, 12. Діаграма діяльності, 13. Діаграма класів, 14. Діаграма послідовності, 15. Інтерфейс користувача, 16. Висновки. _____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 27.10.2025 р.

К А Л Е Н Д А Р Н И Й П Л А Н

Номер	Назва етапів роботи	Терміни виконання	Примітка
1.	Підготовка розділу ВСТУП	28.10.2025	ДП*
2.	Підготовка розділу з огляду літератури та обґрунтування необхідності проведення досліджень за обраною темою	30.10.2025	ДП*
3.	Підготовка розділу (ів) за результатами власних досліджень	01.11.2025	ДП*
4.	Підготовка розділу з розробки програмного забезпечення	26.11.2025	
5.	Підготовка організаційно-економічного розділу	28.11.2025	
6.	Підготовка розділу з охорони праці	02.12.2025	
7.	Підготовка розділу з охорони навколишнього середовища	03.12.2025	
8.	Підготовка розділу ВИСНОВКИ	04.12.2025	
9.	Оформлення списку використаних джерел та додатків	05.12.2025	
10.	Подання на кафедру ПЗАС тексту остаточного варіанту роботи, підписаного її керівником, у роздрукованому та електронному форматі разом із заявами щодо самостійності виконання роботи та ідентичності друкованої та електронної версії роботи (Додатки 1 і 2 «Порядку здійснення заходів з перевірки робіт на наявність текстових збігів/ідентичності/схожості із використанням програмно-технічних засобів», який введений в дію наказом ректора НУК за №20 від 20.01.2020 р.)	08.12.2025	не пізніше, ніж за 14 діб до захисту (згідно п.4.1 зазначеного Порядку)
11.	Підготовка презентації та доповіді	12.12.2025	
12.	Попередній захист роботи на засіданні кафедри ПЗАС	15.12.2025	
13.	Подання на кафедру ПЗАС електронних версії наступних документів у форматі pdf: кваліфікаційної роботи; файлу-опису кваліфікаційної роботи (згідно Додатку до наказу ректора НУК за №287-уч від 19.05.2020 р.); презентації доповіді	19.12.2025	

* - за результатами (розділ звіту) дослідницької практики (ДП), яка була з 01.09.2025 по 26.10.2025 р.

Студент _____ Петраков С.О.
(підпис) (ПІБ)

Керівник роботи _____ Пухалевич А.В.
(підпис) (ПІБ)

РЕФЕРАТ

Петраков Сергій Олександрович

«Нелінійна регресійна модель для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP, та її програмна реалізація»

Кваліфікаційна робота на здобуття освітнього рівня магістра зі спеціальності 121 – «Інженерія програмного забезпечення». Національний університет кораблебудування імені адмірала Макарова. Миколаїв, 2025 р.

Обсяг роботи: 101 стор., 8 табл., 11 рис., 18 використаних джерел, 5 додатків.

Актуальність теми. Раннє оцінювання розміру програмного проєкту є базовою складовою подальших розрахунків, зокрема визначення трудомісткості та, як наслідок, вартості виконання робіт. Помилки на цьому етапі, як правило, призводять до суттєвих відхилень у плануванні та часто стають причиною невдалого завершення проєкту.

Відомо, що моделі, сформовані з урахуванням специфіки конкретних технологій розроблення та типів програмного забезпечення, характеризуються вищою якістю та підвищеною достовірністю прогнозування результатів.

Відсутність нелінійних регресійних моделей для оцінювання розміру вебзастосунків з відкритим вихідним кодом, реалізованих мовою Java з використанням технології JSP, обґрунтовує актуальність даного дослідження.

Мета дослідження. Метою є підвищення достовірності оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP.

Об’єкт дослідження. Об’єктом є процес оцінювання розміру вебзастосунків з відкритим вихідним кодом, реалізованих мовою Java з використанням технології JSP.

Предметом дослідження є нелінійна регресійна модель для оцінювання розміру вебзастосунків з відкритим вихідним кодом, реалізованих мовою Java з використанням технології JSP.

Методи дослідження. Для розв’язання поставлених у роботі завдань застосовувались методи теорії ймовірностей і математичної статистики, методи лінійного та нелінійного багатовимірного регресійного аналізу, а також методи математичного програмування.

Наукова новизна одержаних результатів. Удосконалено трьохфакторну нелінійну регресійну модель для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP, за рахунок одновимірного нормалізуючого перетворення десяткового логарифму і багатовимірних викидів регресії. Запропонована нелінійна регресійна модель, порівняно з розробленою однофакторною нелінійною моделлю для оцінювання розміру вебзастосунків даного типу, а також з існуючою однофакторною нелінійною моделлю оцінювання розміру вебдодатків, реалізованих мовою Java, характеризується вищим значенням множинного коефіцієнта детермінації R^2, меншим значенням середньої відносної похибки *MMRE* та більшим відсотком прогнозованих результатів *PRED(0,25)*.

Практичне значення одержаних результатів. У межах кваліфікаційної роботи було виконано розробку програмного забезпечення для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP. Дане програмне забезпечення підвищує оперативність та точність оцінювання шляхом автоматизації відповідних розрахунків.

Апробація результатів досліджень. Представлені результати досліджень були апробовані на VI Міжнародній науково-практичній інтернет конференції «Інформаційні технології: моделі, алгоритми, системи – 2025» (Миколаїв, 16-17 листопада 2025 р.).

Публікації. За підсумками кваліфікаційної роботи підготовлено одну наукову публікацію, представлену у матеріалах Міжнародної науково-практичної інтернет конференції.

Ключові слова: вебзастосунки на Java, технології JSP, оцінювання розміру, регресійна модель, перетворення десятковий логарифм.

ABSTRACT

Petrakov Serhii Oleksandrovych

«A nonlinear regression model for assessing the size of open source Java JSP web applications, and its software implementation»

Qualification work for obtaining a master's degree in the specialty 121 – «Software Engineering» Admiral Makarov National University of Shipbuilding. Mykolaiv, 2025

Scope of work: 101 pages, 8 tables, 11 figures, 18 sources used, 5 appendices.

Relevance of the topic of the work: Early estimation of the size of a software project is a basic component of subsequent calculations, in particular, determining the labor intensity and, as a result, the cost of work. Errors at this stage, as a rule, lead to significant deviations in planning and often become the cause of unsuccessful completion of the project.

It is known that models formed taking into account the specifics of specific development technologies and types of software are characterized by higher quality and increased reliability of forecasting results.

The lack of nonlinear regression models for estimating the size of open source web applications implemented in Java using JSP technology justifies the relevance of this study.

Purpose and objectives of the study: The goal is to increase the reliability of estimating the size of open source Java web applications created using JSP technology.

The object of the study: The object is the process of estimating the size of open source web applications implemented in Java using JSP technology.

The subject of the study is a nonlinear regression model for estimating the size of open source web applications implemented in Java using JSP technology.

Research methods. To solve the tasks set in the work, methods of probability theory and mathematical statistics, methods of linear and nonlinear multivariate regression analysis, as well as methods of mathematical programming were used.

Scientific novelty of the results obtained. A three-factor nonlinear regression model for estimating the size of open-source Java web applications created using JSP technology has been improved by using a one-dimensional normalizing transformation of the decimal logarithm and multidimensional regression outliers. The proposed nonlinear regression model, compared to the developed one-factor nonlinear model for estimating the size of web applications of this type, as well as to the existing one-factor nonlinear model for estimating the size of web applications implemented in Java, is characterized by a higher value of the multiple coefficient of determination R^2, a lower value of the mean relative error *MMRE* and a higher percentage of predicted results *PRED(0.25)*.

Practical significance of the obtained results. As part of the qualification work, software was developed to estimate the size of open source Java web applications created using JSP technology. This software increases the efficiency and accuracy of the estimation by automating the relevant calculations.

Testing of the research results. The presented research results were tested at the VI International Scientific and Practical Internet Conference «Information Technologies: Models, Algorithms, Systems – 2025" (Mykolaiv, November 16-17, 2025).

Publications. Based on the results of the qualification work, one scientific publication was prepared, presented in the materials of the International Scientific and Practical Internet Conference.

Keywords. Java web applications, JSP technologies, size estimation, regression model, decimal logarithm conversion.

ЗМІСТ

ВСТУП

Актуальність теми. Раннє оцінювання розміру програмного проєкту є базовою складовою подальших розрахунків, зокрема визначення трудомісткості та, як наслідок, вартості виконання робіт. Помилки на цьому етапі, як правило, призводять до суттєвих відхилень у плануванні та часто стають причиною невдалого завершення проєкту [1].

Існуючі підходи до оцінювання розміру програмних продуктів, зокрема алгоритмічні, метрико-орієнтовані та статистичні, не завжди забезпечують достатній рівень достовірності результатів. Значна частина таких підходів не враховує специфіку конкретних технологій розроблення та типів програмного забезпечення, що негативно впливає на точність оцінювання.

Особливу увагу в сучасних дослідженнях приділяють використанню регресійних моделей, які дозволяють встановлювати кількісні залежності між характеристиками програмного продукту та його розміром. Водночас застосування регресійного аналізу супроводжується низкою обмежень, пов’язаних із припущеннями щодо розподілу даних, наявністю мультиколінеарності та впливом викидів. У цьому контексті використання нелінійних регресійних моделей у поєднанні з методами нормалізації даних є перспективним напрямом, що дозволяє врахувати складні залежності між факторами та підвищити обґрунтованість і точність оцінювання розміру програмних систем.

У процесі аналізу наукових публікацій за тематикою дослідження було виявлено низку регресійних моделей, призначених для оцінювання розміру програмного забезпечення, реалізованого мовою Java, зокрема представлених у працях [2-8]. Проте моделі для оцінювання розміру вебзастосунків з відкритим вихідним кодом, реалізованих мовою Java з використанням технології JSP, на даний момент відсутні. Тому обгрунтованою і актуальною є розробка нелінійної регресійної моделі, адаптованої до оцінювання розміру

вебзастосунків з відкритим кодом на Java, створюваних із використанням технології JSP.

Мета і завдання дослідження. Метою є підвищення достовірності оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP.

Мета досягається через наступні завдання дослідження:

– Аналіз методів та моделей оцінювання розміру вебзастосунків.

– Збір даних для побудови математичної моделі оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP.

– Перевірка даних на нормальність розподілу та мультиколінеарність факторів.

– Нормалізація даних та видалення викидів за потреби.

– Побудова нелінійної регресійної моделі оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP, із визначенням довірчих інтервалів і інтервалів прогнозування.

– Оцінювання якості отриманої нелінійної регресійної моделі.

– Розробка програмного забезпечення для реалізації побудованої регресійної моделі оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP.

Об’єктом дослідження. Об’єктом є процес оцінювання розміру вебзастосунків з відкритим вихідним кодом, реалізованих мовою Java з використанням технології JSP.

Предметом дослідження є нелінійна регресійна модель для оцінювання розміру вебзастосунків з відкритим вихідним кодом, реалізованих мовою Java з використанням технології JSP.

Методи дослідження. Для розв’язання поставлених у роботі завдань застосовувались методи теорії ймовірностей і математичної статистики, методи лінійного та нелінійного багатовимірного регресійного аналізу, а також методи математичного програмування.

Наукова новизна одержаних результатів. Удосконалено трьохфакторну нелінійну регресійну модель для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP, за рахунок одновимірного нормалізуючого перетворення десяткового логарифму і багатовимірних викидів регресії. Запропонована нелінійна регресійна модель, порівняно з розробленою однофакторною нелінійною моделлю для оцінювання розміру вебзастосунків даного типу, а також з існуючою однофакторною нелінійною моделлю оцінювання розміру вебдодатків, реалізованих мовою Java, характеризується вищим значенням множинного коефіцієнта детермінації R^2, меншим значенням середньої відносної похибки *MMRE* та більшим відсотком прогнозованих результатів *PRED(0,25)*.

Практичне значення одержаних результатів. У межах кваліфікаційної роботи було виконано розробку програмного забезпечення для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP. Дане програмне забезпечення підвищує оперативність та точність оцінювання шляхом автоматизації відповідних розрахунків.

Особистий внесок здобувача. Кваліфікаційна робота виконана автором самостійно і є результатом особистих наукових досліджень. У співавторській публікації [8] автору належить розробка нелінійної регресійної моделі для оцінювання розміру вебзастосунків з відкритим кодом на Java, створюваних із використанням технології JSP, яка базується на нормалізуючому перетворенні десяткового логарифму.

Апробація результатів дослідження. Представлені результати досліджень були апробовані на VI Міжнародній науково-практичній інтернет конференції «Інформаційні технології: моделі, алгоритми, системи – 2025» (Миколаїв, 16-17 листопада 2025 р.).

Публікації. За підсумками кваліфікаційної роботи підготовлено одну наукову публікацію, представлену у матеріалах Міжнародної науково-практичної інтернет конференції.

1 АНАЛІЗ СУЧАСНИХ ПІДХОДІВ ДО ОЦІНЮВАННЯ РОЗМІРУ JAVA ЗАСТОСУНКІВ

1.1 Основні метрики для оцінювання розміру програмних застосунків

Оцінювання розміру програмних застосунків є фундаментальним етапом у програмній інженерії. Точне визначення обсягу робіт безпосередньо впливає на планування проєкту, оцінку витрат, розподіл ресурсів, прогнозування термінів та управління якістю. Відмінною рисою метрик розміру є їхня об'єктивність та кількісний характер, що дозволяє мінімізувати суб'єктивність у процесі оцінювання. Розглянемо основні метрики розміру програмних застосунків:

– Найпростішими та найпоширенішими є метрики, що базуються на обсязі вихідного коду. До них належить метрика LOC (Lines of Code), яка визначає кількість рядків програмного коду в системі. Попри простоту обчислення, дана метрика має обмеження, пов’язані із залежністю від мови програмування, стилю кодування та рівня деталізації реалізації. Для зменшення впливу цих факторів часто використовується метрика SLOC (Source Lines of Code), що враховує лише логічні рядки коду без коментарів і порожніх рядків.

– Більш універсальними є функціональні метрики, які орієнтовані на оцінювання програмного забезпечення з точки зору реалізованої функціональності. Найбільш відомою серед них є метрика Function Points (FP), що дозволяє оцінити розмір програмної системи незалежно від мови програмування. Подібним підходом характеризується метрика Use Case Points (UCP), яка ґрунтується на аналізі діаграм варіантів використання та складності сценаріїв взаємодії користувача із системою. Дані метрики є доцільними на ранніх етапах життєвого циклу програмного забезпечення, проте потребують експертних оцінок, що може знижувати точність результатів.

– Для об’єктно-орієнтованих програмних систем широко застосовуються структурні метрики, зокрема метрики набору Chidamber-Kemerer. До ключових з них належать WMC (Weighted Methods per Class), яка характеризує складність

класів через кількість та складність їх методів, CBO (Coupling Between Objects), що відображає рівень зв’язності між класами, та RFC (Response For a Class), яка визначає кількість методів, що можуть бути викликані у відповідь на запит до класу. Зазначені метрики дозволяють кількісно оцінити структурну складність програмного забезпечення та часто використовуються як вхідні параметри в математичних моделях оцінювання розміру.

– Окрему групу становлять архітектурні метрики, до яких належать кількість класів, модулів або пакетів у програмній системі. Такі метрики надають узагальнену характеристику масштабу програмного забезпечення та можуть використовуватися для порівняльного аналізу різних програмних проєктів.

У сучасних дослідженнях дедалі більшого поширення набувають модельні та регресійні підходи, в яких розмір програмного забезпечення оцінюється на основі сукупності кількох метрик.

Таким чином, ключові метрики оцінювання розміру програмного забезпечення охоплюють як показники обсягу коду, так і функціональні та структурні характеристики програмних систем. Поєднання кількох метрик у межах математичних моделей дозволяє підвищити достовірність оцінювання та забезпечує більш обґрунтовані результати, особливо для складних об’єктно-орієнтованих програмних застосунків.

1.2 Дослідження методів та моделей для оцінювання розміру Java застосунків

Мова програмування Java має високу популярність серед розробників програмного забезпечення (відповідно до TIOBE Index [9]). Саме тому вже існують математичні моделі для оцінювання розміру таких застосунків, серед яких можна виділити регресійні моделі [2-8]. Розроблені як лінійні [2] так і нелінійні регресійні моделі [3-8]. Ці моделі відрізняються різним набором факторів, типом програмного забезпечення, на якому вони були побудовані, типом нормалізуючих перетворень.

У роботі [2] представлена трьохфакторна модель лінійної регресії, що розроблена для оцінювання кількісті рядків коду застосунків, розроблених мовою Java. Як незалежні змінні використано наступні метрики: кількість класів, сумарна кількість зв'язків між класами та загальна кількість атрибутів класів. Водночас застосування лінійної регресії можливе лише за виконання низки статистичних припущень, зокрема щодо нормального розподілу залишків, яке на практиці виконується лише в окремих випадках. У зв'язку з цим у більшості досліджень перевага надається нелінійним регресійним моделям, що будуються із застосуванням лінеаризуючих і нормалізуючих перетворень або методів перебору. Серед нормалізуючих перетворень найбільш поширеним є десятковий логарифм, який широко використовується під час розробки регресійних моделей.

У дослідженні [3] побудовано трьохфакторну нелінійну регресійну модель для прогнозування розміру у кількості рядках коду промислових інформаційних систем на мові Java. Як фактори використано кількість класів, загальну кількість зв'язків між класами та середню кількість атрибутів на клас. Для нормалізації даних застосовано багатовимірне перетворення Джонсона сімейства S_B.

Автори роботи [4] запропонували однофакторні моделі нелінійної регресії для прогнозування розміру вебзастосунків на Java, із використанням нормалізуючого перетворення Джонсона та десяткового логарифмічного перетворення. У цих моделях залежною змінною є кількість рядків коду, тоді як незалежною змінною виступає загальна кількість класів.

У роботі [5] розроблено трьохфакторну модель нелінійної регресії для прогнозування розміру застосунків з відкритим вихідним кодом Java. Дана модель була розроблена із застосуванням багатовимірного нормалізуючого перетворення Джонсона сімейства S_B. Побудова моделі ґрунтується на таких метриках програмного коду, як кількість рядків коду, загальна кількість класів, кількість зв'язків між класами та середнє число атрибутів на клас.

У дослідженні [6] представлено модель нелінійної регресії, що має чотири фактори та розроблена для прогнозування кількості рядків коду у

застосунках з відкритим кодом на Java. При побудові вказаної моделі використовувалося п’ятивимірне нормалізуюче перетворення Джонсона сімейства S_B. Модель розроблена на наступних метриках: кількість класів, кількість статичних методів, показник згуртованості методів класу та кількість унікальних викликів методів у класі.

Таким чином, у процесі аналізу наукових публікацій за тематикою дослідження було виявлено низку регресійних моделей, призначених для оцінювання розміру програмного забезпечення, реалізованого мовою Java, проте моделі для оцінювання розміру вебзастосунків з відкритим вихідним кодом, реалізованих мовою Java з використанням технології JSP, на даний момент відсутні.

Розглянемо основні співвідношення лінійного та нелінійного регресійних аналізів.

Якщо вихідні дані нормально розподілені, то модель лінійної регресії має вигляд [10]:

$$Y = \hat{Y} + \varepsilon = b_0 + b_1 X_1 + b_2 X_2 + \ldots + b_k X_k + \varepsilon, \qquad (1.1)$$

де Y – залежна змінна;

X_i $(i = \overline{1,k})$ – незалежні змінні;

$\hat{Y}$ – результат прогнозування за рівнянням лінійної регресії;

b_i $(i = \overline{0,k})$ – параметри лінійного рівняння регресії;

ε – нормально розподілена величина (залишки);

k – кількість незалежних змінних.

Нормальний розподіл ε – одна їз умов застосування нелінійної регресії.

При нормально розподілених залишках довірчий інтервал лінійної регресії розраховується згідно формули [10]:

$$\hat{Y} \pm t_{\alpha/2,\mathrm{v}} S_Y \left\{ \frac{1}{N} + (\mathbf{x}^+)^T [(\mathbf{X}^+)^T \mathbf{X}^+]^{-1} (\mathbf{x}^+) \right\}^{1/2}, \qquad (1.2)$$

а прогнозний інтервал за формулою [10]:

$$\hat{Y} \pm t_{\alpha/2,\mathrm{v}} S_Y \left\{1 + \frac{1}{N} + (\mathbf{x}^+)^T [(\mathbf{X}^+)^T \mathbf{X}^+]^{-1} (\mathbf{x}^+)\right\}^{1/2}, \quad (1.3)$$

де $S_Y^2 = \frac{1}{\mathrm{v}} \sum_{i=1}^{N} \left(Y_i - \hat{Y}_i\right)^2$;

$\mathrm{v} = N - k - 1$;

N – кількість точок багатовимірних даних;

$t_{\alpha/2,\mathrm{v}}$ – квантіль t-розподілу Стьюдента з $\alpha/2$ рівнем значимості та v степенями вільності;

$(\mathrm{X}^+)^T \mathrm{X}^+$ – $k \times k$ матриця [10]:

$$(\mathrm{X}^+)^T \mathrm{X}^+ = \begin{pmatrix} S_{X_1X_1} & S_{X_1X_2} & \dots & S_{X_1X_k} \\ S_{X_1X_2} & S_{X_2X_2} & \dots & S_{X_2X_k} \\ \dots & \dots & \dots & \dots \\ S_{X_1X_k} & S_{X_2X_k} & \dots & S_{X_kX_k} \end{pmatrix}, \quad (1.4)$$

де $S_{X_qX_r} = \sum_{i=1}^{N} \left[X_{q_i} - \bar{X}_q\right]\left[X_{r_i} - \bar{X}_r\right]$, $q, r = \overline{1,k}$.

За умови, що випадкові величини $Y, X_1, X_2, \dots, X_k$ не підпорядковуються гаусівському закону розподілу, доцільно застосовувати нелінійні регресійні моделі. На практиці для побудови таких моделей широко використовується метод нормалізуючих перетворень, який передбачає виконання низки вказаних кроків [10]:

1) вибір нормалізуючого перетворення;

2) нормалізація вихідних даних;

3) перевірка на наявність викидів (якщо викиди знайдено – вилучаємо найбільший викид і повертаємося на крок 2);

4) побудова рівняння лінійної регресії;

5) перевірка залишків регресії (ящо залишки не розподілені нормально – вилучаємо найбільший по модулю залишок і повертаємося на крок 2);

6) побудова довірчого інтервалу і інтервалу прогнозування лінійної регресії;

7) перехід від лінійної регресії і її інтервалів до нелінійної з її інтервалами (застосування зворотного нормалізуючого перетворення);

8) перевірка належності залежних емпіричних даних інтервалу прогнозування нелінійної регресії (якщо є залежні дані за межами інтервалу – вилучаємо всі відповідні набори даних і повертаємося на крок 2);

9) аналіз якості рівняння нелінійної регресії і перевірка адекватності рівняння емпіричним даним.

Нормалізуюче перетворення випадкових величин $Y, X_1, X_2, \ldots, X_k$, що не підпорядковуються нормальному закону, у гаусівські величини $Z_Y, Z_1, Z_2, \ldots, Z_k$ позначимо як ψ. Тоді [10]:

$$\mathrm{T} = \psi(\mathrm{P}), \tag{1.5}$$

де $\mathrm{P} = \{Y, X_1, X_2, \ldots, X_k\}^T$ – випадковий вектор, що не підпорядковується нормальному розподілу;

$\mathrm{T} = \{Z_Y, Z_1, Z_2, \ldots, Z_k\}^T$ – нормально розподілений випадковий вектор;

$\psi = \{\psi_Y, \psi_1, \psi_2, \ldots, \psi_k\}^T$ – нормалізуюче перетворення.

Лінійна регресійна модель для k+1 нормально розподілених величин $Z_Y, Z_1, Z_2, \ldots, Z_k$ визначається як [10]:

$$Z_Y = \hat{Z}_Y + \varepsilon = b_0 + b_1 Z_1 + b_2 Z_2 + \ldots + b_k Z_k + \varepsilon, \tag{1.6}$$

інтервал довіри знаходиться згідно формули [10]:

$$\hat{Z}_Y \pm t_{\frac{\alpha}{2},\nu} S_{Z_Y} \left\{ \frac{1}{N} + (z_X^+)^T [(Z_X^+)^T Z_X^+]^{-1} (z_X^+) \right\}^{1/2}, \tag{1.7}$$

а прогнозний інтервал визначається за співвідношенням [10]:

$$\hat{Z}_Y \pm t_{\frac{\alpha}{2},v} S_{Z_Y} \left\{1 + \frac{1}{N} + (z_X^+)^T [(Z_X^+)^T Z_X^+]^{-1} (z_X^+)\right\}^{1/2}, \quad (1.8)$$

де $S_{Z_Y}^2 = \frac{1}{\mathrm{v}} \sum_{i=1}^{N} \left(Z_{Y_i} - \hat{Z}_{Y_i}\right)^2$; $\mathrm{v} = N - k - 1$; $(Z_X^+)^T Z_X^+ - k \times k$ матриця S_Z

$$(Z_X^+)^T Z_X^+ = \begin{pmatrix} S_{Z_1 Z_1} & S_{Z_1 Z_2} & \dots & S_{Z_1 Z_k} \\ S_{Z_1 Z_2} & S_{Z_2 Z_2} & \dots & S_{Z_2 Z_k} \\ \dots & \dots & \dots & \dots \\ S_{Z_1 Z_k} & S_{Z_2 Z_k} & \dots & S_{Z_k Z_k} \end{pmatrix}, \quad (1.9)$$

де $S_{Z_q Z_r} = \sum_{i=1}^{N} \left[Z_{q_i} - \bar{Z}_q\right]\left[Z_{r_i} - \bar{Z}_r\right]$, $q, r = \overline{1, k}$.

Застосувавши зворотнє перетворення ψ^{-1} до співвідношення (1.5), отримаємо [10]:

$$\mathrm{P} = \psi^{-1}(\mathrm{T}). \quad (1.10)$$

Через зворотнє перетворення (1.7) отримуємо інтервал довіри множинної нелінійної регресії:

$$\Psi_Y^{-1} \left(\hat{Z}_Y \pm t_{\frac{\alpha}{2},v} S_{Z_Y} \left\{ \frac{1}{N} + (\boldsymbol{z}_X^+)^T [\mathbf{Z_X}]^{-1} (\boldsymbol{z}_X^+) \right\}^{1\backslash 2} \right), \quad (1.11)$$

а через зворотнє перетворення (1.8) отримуємо інтервал прогнозування множинної нелінійної рнгресії:

$$\psi_Y^{-1} \left(\hat{Z}_Y \pm t_{\frac{\alpha}{2},v} S_{Z_Y} \left\{ 1 + \frac{1}{N} + (\boldsymbol{z}_X^+)^T [\mathbf{Z_X}]^{-1} (\boldsymbol{z}_X^+) \right\}^{1\backslash 2} \right). \quad (1.12)$$

У даному дослідженні для нормалізації даних застосовано перетворення на основі десяткового логарифма.

1.3 Обґрунтування необхідності проведення дослідження за обраним напрямом

Ефективне управління процесом розробки програмного забезпечення критично залежить від точної оцінки його розміру та складності. Існуючі методи та моделі демонструють обмежену ефективність при аналізі сучасних веб-застосунків, особливо тих, що базуються на технологіях, що поєднують логіку сервера та представлення даних, як-от JSP.

Нелінійна природа залежностей між різними компонентами таких систем, складність взаємодії між серверними скриптами, об'єктами та шаблонами відображення роблять застосування стандартних лінійних моделей прогнозування недостатньо точним. Окрім цього, для використання лінійної регресійної теорії повинні виконуватися певні припущення, одним з яких є нормальний розподіл залишків, що рідко справедливо для емпіричних даних. Це змушує дослідників звертатися до нелінійних регресійних моделей. При аналізі публікацій за темою дослідження не було виявлено регресійної моделі для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP.

Виходячи з вище викладеного, розробка спеціалізованої нелінійної регресійної моделі, адаптованої до оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються із використанням технології JSP, є обґрунтованою та актуальною з низки причин. По-перше, JSP як технологія, що інтегрує Java-код у HTML-сторінки, генерує специфічні шаблони коду, де традиційне розмежування між логікою та інтерфейсом розмивається, що потребує нових підходів до оцінювання. По-друге, аналіз проєктів з відкритим кодом на Java надає унікальну можливість вивчити реальні, масштабні практики розробки, що сформовані спільнотою, та отримати надійний емпіричний базис для моделі. Таким чином, дане дослідження спрямоване на подолання наявного розриву та надання практичного інструментарію для більш ефективного керування життєвим циклом сучасних веб-орієнтованих Java-проєктів.

1.4 Висновки за розділом 1

У даному розділі було проведено детальний аналіз основних метрик для оцінювання розміру програмних застосунків та підкреслено, що в сучасних дослідженнях дедалі більшого поширення набувають модельні та регресійні підходи, у яких розмір програмного забезпечення оцінюється на основі сукупності кількох метрик.

Також у процесі аналізу наукових публікацій за тематикою дослідження виявлено низку регресійних моделей для оцінювання розміру Java застосунків. Серед них є лінійні та нелінійні моделі. Ці моделі відрізняються різним набором факторів, типом програмного забезпечення, на якому вони були побудовані. Нелінійні моделі відрізняються і типом нормалізуючих перетворень.

У розділі підкреслено, що нелінійна природа залежностей між різними компонентами вебзастосунків з відкритим вихідним кодом, реалізованих мовою Java з використанням технології JSP, роблять застосування стандартних лінійних моделей прогнозування недостатньо точним.

Проте моделі для оцінювання такого типу застосунків на даний момент відсутні. Тому розробка спеціалізованої нелінійної регресійної моделі, адаптованої до оцінювання розміру вебзастосунків з відкритим кодом на Java, що ствоюються із використанням технології JSP, є обґрунтованою та актуальною.

2 ПОБУДОВА МОДЕЛІ НЕЛІНІЙНОЇ РЕГРЕСІЇ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ВЕБЗАСТОСУНКІВ З ВІДКРИТИМ КОДОМ НА JAVA, ЩО СТВОРЮЮТЬСЯ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЇ JSP

2.1 Попередня обробка даних для побудови моделі нелінійної регресії для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP

Для побудови моделі нелінійної регресії для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються із застосуванням технології JSP, було проаналізовано 42 програмні проєкти, відібрані з репозиторію GitHub [11]. Дані отримані за допомогою інструменту CK. Для кожного з проєктів отримано дані з наступних метрик: обсяг коду в тисячах рядків (KLOC), кількість класів (Classes), середня кількість методів на один клас (Methods by Class) та глибина дерева успадкування (DIT). Описова статистика вихідного набору даних наведена в таблиці 2.1.

Таблиця 2.1 – Описова статистика вихідного набору даних

Метрика	Min	Max	Середнє	СКВ
KLOC (Y)	0,10	8,67	1,111	1,447
Classes (X1)	1,00	210,00	28,881	40,987
Methods by Class	0,96	8,18	3,974	2,019
DIT	1,00	2,00	1,594	0,266

Для оцінювання відповідності багатовимірного набору даних нормальному закону розподілу, у роботі застосовано тест Мардіа, який базується на аналізі коефіцієнтів багатовимірної асиметрії та ексцесу [12]:

$$\beta_{1,k} = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\left[(X_i - \bar{X})^T S_N^{-1}\left(X_j - \bar{X}\right)\right]^3, \qquad (2.1)$$

$$\beta_{2,k} = \frac{1}{N}\sum_{i=1}^{N}[(X_i - \bar{X})^T S_N^{-1}(X_i - \bar{X})]^2, \quad (2.2)$$

де N – кількість рядків даних;

X – вектор випадкових величин ($X = (X_1, X_2, \dots, X_k)^T$);

$\bar{X}$ – вектор вибіркових середніх ($\bar{X} = (\bar{X}_1, \bar{X}_2, \dots, \bar{X}_k)^T$);

k – кількість випадкових величин;

S_N – вибіркова коваріаційна матриця, яку можна представити у вигляді [10]:

$$S_N = \frac{1}{N}\sum_{i=1}^{N}(X_i - \bar{X})(X_i - \bar{X})^T .$$

Даний підхід дозволяє здійснити статистичну перевірку гіпотези про багатовимірну нормальність вибірки з урахуванням взаємозв'язків між змінними.

Відомо, що при багатовимірному нормальному розподілі очікуваний ексцес Мардіа обчислюється за формулою $k(k+2)$ і при $k = 3$ ексцес дорівнює 15.

Для $\beta_{1,k}$ розраховується тестова статистика

$$\frac{N}{6}\beta_{1,k}, \quad (2.3)$$

яка має наближений розподіл χ^2 з $k(k+1)(k+2)/6$ ступенями свободи та рівнем значущості α.

Перевірка гіпотези про нормальність за асиметрією здійснюється шляхом порівняння значення зазначеної статистики з відповідним квантилем χ^2 розподілу. Оцінювання відхилення розподілу від нормального за ексцесом виконується на основі стандартизованої статистики, що має нормальний розподіл з математичним сподіванням $k(k+2)$ та дисперсією $\frac{8k(k+2)}{N}$. Порівняння отриманого значення зі стандартним нормальним квантилем рівня 1− α дозволяє зробити висновок щодо наявності або відсутності багатовимірної

нормальності досліджуваних даних. При перевірці відхилення розподілу від нормального рекомендують брати α рівним 0,005.

Для зібраних даних отримали:

– Розраховане за (2.1) значення асиметрії: 34,6804; статистика (2.3): 242,7630; квантиль: 39,9968.

– Розраховане за (2.2) значення ексцесу: 58,1431; квантиль: 35,7752.

Таким чином і по розрахованому значенню асиметрії і по розрахованому значенню ексцесу отримали підтвердження негаусівського розподілу.

При побудові регресійної моделі на підготовчому етапі дослідження виконано перевірку потенційних предикторів на наявність мультиколінеарності. Мультиколінеарність означає наявність лінійної залежності між двома або більше предикторами регресійної моделі та є небажаним явищем у множинному регресійному аналізі. Мультиколінеарність ускладнює або унеможливлює коректну інтерпретацію внеску окремих предикторів у формування залежної змінної та свідчить про високий рівень взаємозв'язку або кореляції між незалежними змінними.

Один із способів виявлення мультиколінеарності – це розрахунок коефіцієнтів впливу дисперсії (VIF), які відповідають діагональним елементам оберненої коваріаційної матриці предикторів у моделі множинної лінійної регресії. Значення VIF, що перевищують 10, зазвичай розглядаються як показник значної мультиколінеарності, тоді як значення від 1 до 5 свідчать про її відсутність.

Для трьох предикторів – кількості класів (Classes), середньої кількості методів на клас (Methods by Class) та глибини дерева успадкування (DIT) – за зібраними даними отримано значення коефіцієнтів VIF, що становлять відповідно 1,265; 1,424 та 1,659. Оскільки всі значення є меншими за 5, то можна зробити висновок про відсутність мультиколінеарності у досліджуваній вибірці.

Отже, результати застосування тесту Мардіа підтвердили негаусівський характер розподілу досліджуваних даних, що обґрунтовує доцільність використання нелінійної регресійної моделі. Водночас аналіз коефіцієнтів

впливу дисперсії (VIF) засвідчив відсутність мультиколінеарності серед предикторів, що підтверджує коректність їх використання у побудові майбутньої регресійної моделі.

На наступних кроках будемо реалізовувати ітераційний процес побудови нелінійної регресійної моделі для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP.

2.2 Побудова моделі нелінійної регресії для оцінювання розміру

Для побудови нелінійної регресійної моделі для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP, використали ітераційний метод нормалізуючих перетворень.

Обираємо нормалізуюче перетворення десятковий логарифм.

Перша ітерація. Виконуємо нормалізацію даних. Для нормалізованих даних обчислюємо квадрат відстані Махаланобіса (SMD) згідно формули [10]:

$$d_i^2 = (Z_i - \bar{Z})^T S_N^{-1} (Z_i - \bar{Z}), \quad (2.4)$$

де Z_i – i-та точка багатовимірного нормалізованого вектора Z;

$\bar{Z}$ – вектор вибіркових середніх,

S_N – вибіркова коваріаційна матриця, яку можна представити у вигляді [10]:

$$S_N = \frac{1}{N} \sum_{i=1}^{N} (Z_i - \bar{Z}) (Z_i - \bar{Z})^T .$$

Для багатовимірного нормального розподілу значення d_i^2 близькі до випадкової величини $\chi_{m,\alpha}^2$, де $\chi_{m,\alpha}^2$ – квантиль розподілу χ^2 для рівня значущості α. В якості α обираємо 0,005; m=4.

Точка нормалізованих даних, для якої d_i^2 перевищить $\chi_{4;0,005}^2$, буде вважатися викидом. З набору даних необхідно вилучати точку з максимальним викидом.

Для нормалізованого набору даних із 42 рядків вилучається рядок **2** по SMD: $\chi^2_{4;0,005} = 14{,}8603;\ \max \text{SMD} = 18{,}2500 > \chi^2_{4;0,005}$.

На другій ітерації із 41 рядка даних вилучається рядок **19** по SMD:

$\chi^2_{4;0,005} = 14{,}8603;\ \max \text{SMD} = 16{,}4944 > \chi^2_{4;0,005}$.

На третій ітерації із 40 рядків даних вилучається рядок **25** по SMD:

$\chi^2_{4;0,005} = 14{,}8603;\ \max \text{SMD} = 15{,}7897 > \chi^2_{4;0,005}$.

На четвертій ітерації по 39 рядках даних викиди по SMD відсутні. Будуємо лінійну регресію для нормалізованих даних згідно (1.6) і перевіряємо залишки на нормальний розподіл. Отримали $\chi^2_{\text{кр}}$=7,8147; χ^2=9,0002>$\chi^2_{\text{кр}}$. Тобто залишки не нормально розподілені. Вилучаємо рядок даних **32** з максимальним по модулю залишком і переходимо до наступної ітерації побудови нелінійної регресійної моделі, яка починається з нормалізації вихідних даних.

На п`ятій ітерації із 38 рядків даних вилучається рядок **12** по SMD:

$\chi^2_{4;0,005} = 14{,}8603;\ \max \text{SMD} = 15{,}6021 > \chi^2_{4;0,005}$.

На шостій ітерації по 37 рядках даних викиди по SMD і залишкам регресії відсутні. Тому за формулами (1.7), (1.8) обчислюємо межі довірчого та прогнозного інтервалів для лінійної регресії та, використовуючи зворотнє до нормалізуючого перетворення, отримуємо нелінійну регресійну модель та її інтервали. По інтервалу прогнозування нелінійної регресії вилучаємо 1 викид - рядок **10**.

Залишається 36 рядків даних. Викиди по SMD відсутні.

Будуємо лінійну регресію (**b0 = -1,6057; b1 = 0,9512; b2 = 0,4244; b3 = 0.2656**) і перевіряємо залишки. Залишки нормально розподілені.

Обчислюємо межі довірчого та прогнозного інтервалів для лінійної регресії (1.7), (1.8).

Використовуючи зворотнє до нормалізуючого перетворення, отримуємо нелінійну регресійну модель та її інтервали (1.10), (1.11), (1.12). Викиди по інтервалу прогнозування відсутні.

Таким чином остаточна нелінійна регресійна модель побудована на 36 рядках даних. Всього вилучено 6 викидів (рядки 2, 19, 25, 32, 12, 10).

Лінійна регресійна модель для 36 рядків нормалізованих даних має вигляд (1.6):

$$Z_Y = \hat{Z}_Y + \varepsilon = -1{,}6057 + 0{,}9512Z_1 + 0{,}4244Z_2 + 0{,}2656Z_3 + \varepsilon.$$

Матриця S_Z (1.9) для отримання довірчих інтервалів та інтервалів передбачення за (1.7) і (1.8) має вигляд:

$$S_Z = \begin{pmatrix} 5{,}5574 & 1{,}2095 & -0{,}4126 \\ 1{,}2095 & 2{,}4329 & -0{,}3877 \\ -0{,}4126 & -0{,}3877 & 0{,}1690 \end{pmatrix}.$$

Остаточна нелінійна регресійна модель для 36 рядків нормалізованих даних має вигляд (1.10):

$$Y = 10^{\varepsilon - 1{,}6057} X_1^{0{,}9512} X_2^{0{,}4244} X_3^{0{,}2656}.$$

Для даної моделі визначено оцінки якості згідно [10]: $R^2 = 0{,}9602$, ($R^2 > 0{,}75$); $MMRE = 0{,}1322$ ($MMRE \leq 0{,}25$); $PRED(0{,}25) = 0{,}8611$ ($PRED(0{,}25) > 0{,}75$). В дужках вказані норми значень відповідних показників.

Для порівняння, для досліджуваного набору даних побудовано однофакторну нелінійну регресійну модель з кількістю класів у якості предиктора. В результаті вилучили 7 викидів (2, 19, 25, 10, 32, 33, 42) і отримали нелінійну однофакторну модель виду:

$$Y = 10^{\varepsilon - 1{,}4603,} X_1^{1{,}0546}.$$

Показники якості нелінійної однофакторної моделі: $R^2 = 0{,}9259$; $MMRE = 0{,}2185$; $PRED(0{,}25) = 0{,}6571$.

Також спробували використати наявну нелінійну однофакторну модель модель з [4] виду: $Y = 10^{\varepsilon - 1{,}469} X_1^{1{,}2266}$. Ця модель має незадовільну якість при

використанні для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP.

В таблиці 2.2 наведено порівняння якості однофакторної та трьохфакторної нелінійних регресійних моделей, розроблених для вказаного типу вебзастосунків.

Таблиця 2.2 – Порівняння якості моделей

Оцінки якості	Нелінійна однофакторна	Нелінійна трьохфакторна
R^2	0,9259	0,9602
$MMRE$	0,2185	0,1322
$PRED(0{,}25)$	0,6571	0,8611

Як бачимо з таблиці 2.2, за всіма показниками якості нелінійна трьохфакторна регресійна модель переважає нелінійну однофакторну регресійну модель для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP.

2.3 Висновки за розділом 2

У розділі представлено побудову нелінійної регресійної моделі для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP, із застосуванням нормалізуючого перетворення log10.

Основні результати:

– Здійснено відбір проєктів на платформі GitHub та за допомогою програмного засобу CK зібрано необхідні метрики діаграми класів для подальшого аналізу.

– Для перевірки відповідності багатовимірного набору метрик нормальному закону розподілу застосовано тест Мардіа, що ґрунтується на

оцінюванні коефіцієнтів багатовимірної асиметрії та ексцесу; результати тестування засвідчили негаусівський характер розподілу даних.

– Проведено аналіз мультиколінеарності предикторів, за результатами якого встановлено її відсутність.

– Реалізовано ітераційну процедуру побудови нелінійної регресійної моделі для оцінювання розміру вебзастосунків з відкритим кодом на мові Java, розроблених із використанням технології JSP, із застосуванням вилучення викидів та нормалізуючого перетворення на основі десяткового логарифма.

– Побудовано довірчі та прогнозні інтервали для сформованої нелінійної регресійної моделі.

– Визначено якість запропонованої нелінійної регресійної моделі для оцінювання розміру вебзастосунків із відкритим кодом на Java, що створюються із використанням технології JSP.

– Виконано порівняльний аналіз однофакторної та трьохфакторної нелінійних регресійних моделей.

3 ПРОЄКТ ПРОГРАМИ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ВЕБЗАСТОСУНКІВ З ВІДКРИТИМ КОДОМ НА JAVA, ЩО СТВОРЮЮТЬСЯ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЇ JSP

3.1 Постановка задачі на розробку програми для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP

Метою даного розділу кваліфікаційної роботи є розробка проєкту програми для оцінювання розміру вебзастосунків з відкритим кодом, що створюються з використанням технології JSP, на основі нелінійної регресійної моделі. Програма призначена для підтримки процесу прогнозування розміру програмного коду вебзастосунків на ранніх етапах їх аналізу та проєктування.

Розроблювана програма є настільним застосунком, створеним мовою програмування Java. Програма не має прямої інтеграції з репозиторіями відкритого коду; всі параметри математичної моделі та метрики вебзастосунку вводяться користувачем вручну на основі попереднього аналізу проєктів.

Програма повинна забезпечувати виконання таких основних функцій:

– введення параметрів нелінійної регресійної моделі, зокрема коефіцієнтів регресійного рівняння та матриці, необхідної для розрахунку інтервалів;

– введення значення довірчої ймовірності для подальших статистичних розрахунків;

– введення метрик вебзастосунку, для якого виконується оцінювання розміру;

– обчислення прогнозного значення розміру вебзастосунку та відповідних статистичних інтервалів;

– збереження вхідних даних у БД програми;

– збереження результатів оцінювання розміру вебзастосунків у БД програми;

– експорт результатів розрахунків у файл формату CSV.

Вимоги до організації вхідних та вихідних даних

Вхідними даними програми є:

– коефіцієнти нелінійного регресійного рівняння b_0, b_1, b_2, b_3;

– матриця, необхідна для обчислення довірчого інтервалу та інтервалу прогнозування;

– значення довірчої ймовірності α;

– значення метрик X_1 (кількість класів), X_2 (середня кількість методів у класі) та X_3 (глибина дерева успадкування).

Вхідні дані вводяться у відповідні поля вводу згідно з допустимим типом даних (числовий).

Вихідними даними програми є:

– обчислене значення залежної змінної Y, що характеризує розмір вебзастосунку та виражається у тисячах рядків коду;

– нижня та верхня межі довірчого інтервалу нелінійного регресійного рівняння;

– нижня та верхня межі інтервалу прогнозування нелінійного регресійного рівняння;

– файл формату CSV, що містить результати оцінювання розміру вебзастосунку.

Вихідні дані виводяться у відповідні поля виводу та зберігаються у БД програми.

Вимоги до складу та параметрів технічних засобів

Система повинна задовольняти вказаним вимогам на комп’ютері наступної мінімальної комплекції:

– CPU: Intel(R) i5 2,16 МГц або аналогічний за параметрами AMD;

– RAM: 4 GB;

– HDD: 10 GB вільного місця.

Вимоги до інформаційної та програмної сумісності

Оскільки розроблювана програма буде працювати у середовищі JVM, вона є платформонезалежною, тому може бути запущена на будь-якій

операційній системі, що підтримує Java: Microsoft Windows версії 10, Linux (дистрибутиви Ubuntu, Debian або аналогічні), macOS версії 10.15. Для запуску програми на комп’ютері користувача повинно бути встановлено JDK 17. JavaFX може бути підключений або як частина JDK (для відповідних збірок), або як окрема бібліотека OpenJFX. Для доступу до бази даних необхідна наявність JDBC-драйверу H2 Database. Також необхідні стандартні бібліотеки Java для роботи з файлами, серіалізацією та математичними обчисленнями, які повинні бути включені до складу застосунку.

Програмне забезпечення повинно бути реалізоване з дотриманням таких вимог: мова програмування – Java, тип застосунку – настільний, використання стандартних бібліотек Java для реалізації інтерфейсу користувача, роботи з БД та експорту даних, розділення логіки обчислень, інтерфейсу користувача та доступу до даних відповідно до принципів багатошарової архітектури.

Виконання наведених вимог забезпечує створення програми, здатної ефективно підтримувати процес оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP.

Детальна постановка задачі сформульована в технічному завданні, яке наведене у Додатку А.

3.2 Архітектура програми для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP

Для розробки програми для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP, обрано багатошарову архітектуру з використанням елементів шаблону MVC у шарі представлення. Такий підхід забезпечує чітке розділення відповідальностей між компонентами програми, підвищує зручність супроводу та дозволяє масштабувати програму без суттєвих змін її структури.

Загальна діаграма компонентів програми для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP, представлена на рисунку 3.1.

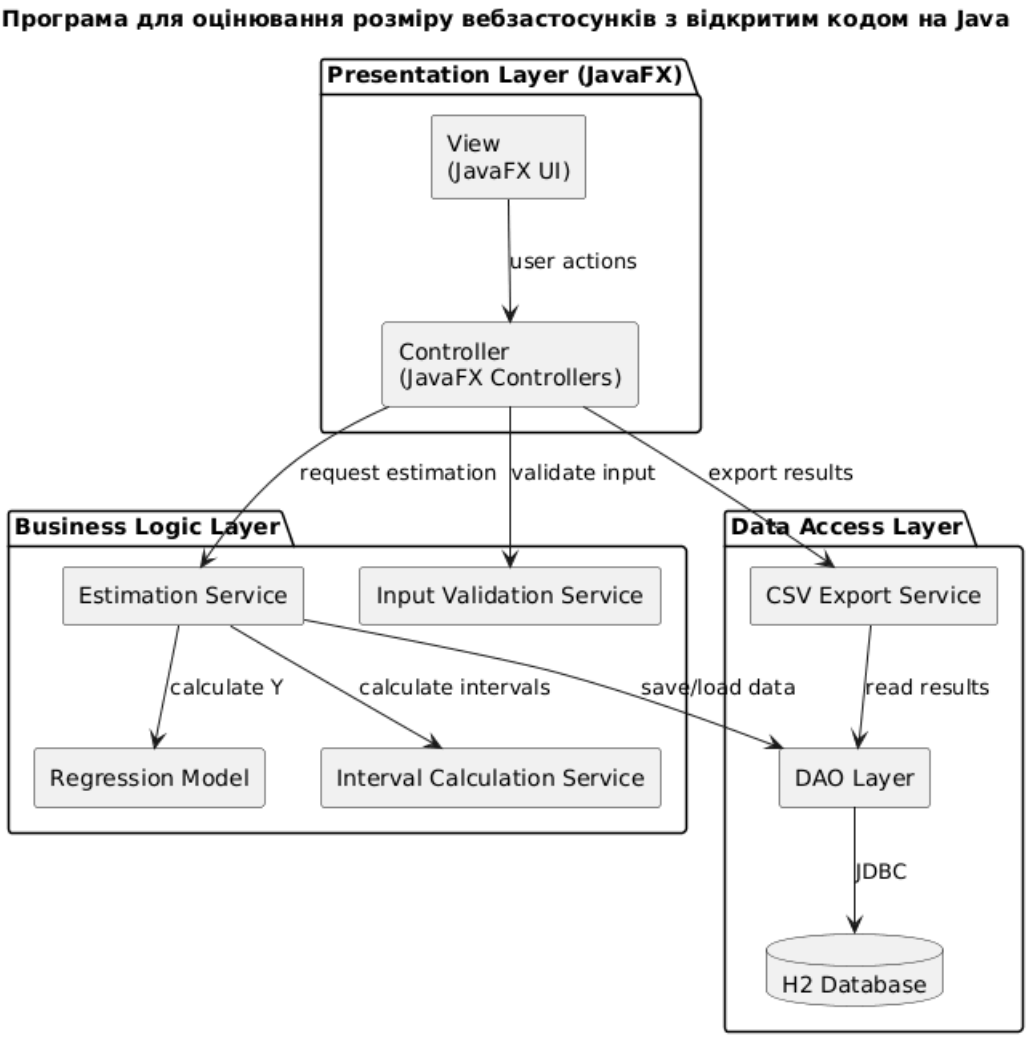

Рисунок 3.1 – Загальна діаграма компонентів програми для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP

Архітектура програми складається з трьох основних шарів: шару представлення, шару бізнес-логіки та шару доступу до даних.

Шар представлення (Presentation Layer)

Шар представлення відповідає за взаємодію користувача з програмою та реалізується з використанням технології JavaFX. У межах цього шару застосовується шаблон Model–View–Controller (MVC), де:

– View – графічні форми введення параметрів регресійної моделі, метрик вебзастосунку та елементів керування процесом обчислення, збереження і експорту результатів;

– Controller – обробляє дії користувача, виконує первинну перевірку введених даних та ініціює виклики відповідних сервісів бізнес-логіки;

– Model – об’єкти, що представляють параметри регресії, метрики вебзастосунку та результати розрахунків.

Застосування шаблону MVC дозволяє відокремити логіку інтерфейсу користувача від обчислювальної частини програми та підвищує модульність коду.

Шар бізнес-логіки (Business Logic Layer)

Шар бізнес-логіки реалізує основні алгоритми та математичну модель оцінювання розміру вебзастосунків. До його функцій належать:

– обробка параметрів нелінійної регресійної моделі;

– обчислення прогнозного значення розміру вебзастосунку;

– розрахунок меж довірчого інтервалу;

– розрахунок меж інтервалу прогнозування;

– перевірка коректності вхідних даних перед виконанням обчислень.

Компоненти цього шару не залежать від технології користувацького інтерфейсу та способу збереження даних, що забезпечує можливість повторного використання бізнес-логіки.

Шар доступу до даних (Data Access Layer)

Шар доступу до даних відповідає за збереження та отримання інформації з БД і реалізується з використанням патерну DAO (Data Access Object). Для зберігання даних застосовується вбудована реляційна СУБД H2 Database, доступ до якої здійснюється через стандартний механізм JDBC.

Основні функції шару доступу до даних:

– збереження параметрів регресійної моделі;

– збереження значень метрик вебзастосунків;

– збереження результатів оцінювання розміру;

– отримання даних для подальшого аналізу або експорту.

Використання DAO забезпечує ізоляцію бізнес-логіки від конкретної реалізації СУБД та спрощує можливу заміну бази даних у майбутньому.

Взаємодія між шарами

Взаємодія між шарами програми здійснюється лише у напрямку зверху вниз: шар представлення звертається до шару бізнес-логіки, шар бізнес-логіки взаємодіє з шаром доступу до даних.

Зворотні залежності між шарами відсутні, що відповідає принципам багатошарової архітектури та сприяє зменшенню зв’язності між компонентами програми.

Таким чином, обрана архітектура програми забезпечує:

– чітке розділення відповідальностей між компонентами;

– зручність розширення та супроводу ПЗ;

– відповідність сучасним підходам до проєктування настільних Java-застосунків.

Використання JavaFX, багатошарової архітектури з елементами MVC та вбудованої СУБД H2 дозволяє створити ефективне та надійне ПЗ для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP.

3.3 Ескізний проєкт програми для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP

Розробку ескізного проєкту програми для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP, почнемо з виявлення акторів програми та варіантів використання програми.

Виявлений актор програми Користувач – аналітик програмного забезпечення, дослідник або розробник, який виконує оцінювання розміру вебзастосунку на основі попередньо визначених метрик та параметрів регресійної моделі.

Варіанти використання програми для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP, наведено у таблиці 3.1.

Таблиця 3.1 – Варіанти використання програми для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP

ID	Назва варіанту використання	Актор	Короткий опис	Вхідні дані	Вихідні дані / результат
UC1	Введення параметрів регресійної моделі	Користувач	Введення коефіцієнтів нелінійної регресійної моделі та матриці для розрахунку інтервалів	b0,b1,b2,b3, елементи матриці	Параметри моделі прийняті та підготовлені до використання
UC2	Введення довірчої ймовірності	Користувач	Задання значення довірчої ймовірності для статистичних розрахунків	Значення α	Значення α збережено
UC3	Введення метрик вебзастосунку	Користувач	Введення метрик вебзастосунку для оцінювання його розміру	X1, X2, X3	Метрики вебзастосунку прийняті системою
UC4	Оцінювання розміру вебзастосунку	Користувач	Обчислення прогнозного значення розміру вебзастосунку та інтервалів	Параметри моделі b0,b1,b2,b3, α, метрики X1, X2, X3	Значення Y, границі інтервалів, довірчого та прогнозування
UC5	Збереження вхідних даних	Користувач	Збереження введених параметрів та метрик у БД	Вхідні параметри та метрики	Дані збережено у СУБД H2
UC6	Збереження результатів оцінювання	Користувач	Збереження результатів обчислень у БД	Результати оцінювання	Результати збережено у СУБД H2
UC7	Перегляд результатів оцінювання	Користувач	Перегляд результатів оцінювання у графічному інтерфейсі	Збережені або обчислені результати	Відображені результати оцінювання
UC8	Експорт результатів у CSV	Користувач	Формування та збереження CSV-файлу з результатами розрахунків	Результати оцінювання	CSV-файл створено

Варіанти використання UC1–UC3 відповідають за підготовку вхідних даних, варіант використання UC4 – основний варіант використання програми, варіант використання UC6 включений автоматично при оцінюванні та

перегляді результатів, варіанти використання UC5 та UC8 виконуються за ініціативою користувача.

Діаграма варіантів використання програми для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP, наведена на рисунку 3.2.

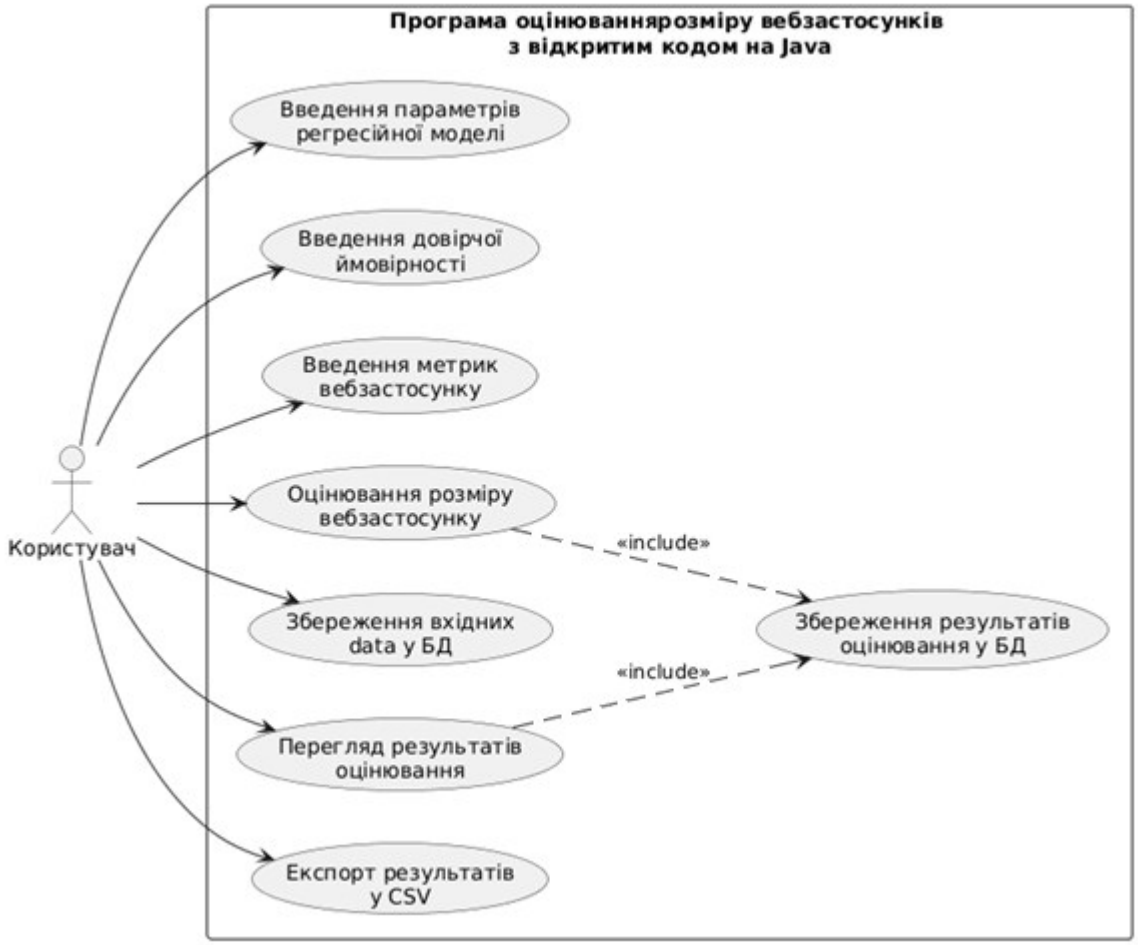

Рисунок 3.2 – Діаграма варіантів використання програми для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP

Далі розробимо специфікації варіантів використання програми для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP.

UC1 – Введення параметрів регресійної моделі

Опис прецеденту

Користувач вводить параметри нелінійної регресійної моделі, зокрема коефіцієнти регресійного рівняння та матрицю, необхідну для розрахунку статистичних інтервалів.

Передумови

Програма запущена, користувач має доступ до форми введення параметрів моделі.

Базовий потік

1. Користувач відкриває форму введення параметрів регресійної моделі.

2. Вводить значення коефіцієнтів b0,b1,b2,b3.

3. Вводить елементи матриці для розрахунку інтервалів.

4. Підтверджує введення даних.

5. Система перевіряє коректність введених значень.

Альтернативні потоки

3а. Користувач вводить некоректні або неповні дані – система відображає повідомлення про помилку та пропонує повторити введення.

Постумови

Параметри регресійної моделі прийняті системою та доступні для подальших розрахунків.

UC2 – Введення довірчої ймовірності

Опис прецеденту

Користувач задає значення довірчої ймовірності α для обчислення довірчого та прогнозного інтервалів.

Передумови

Програма запущена, форма введення параметрів доступна.

Базовий потік

1. Користувач вводить значення α.

2. Підтверджує введення.

3. Система перевіряє допустимість значення.

Альтернативні потоки

2а. Значення α виходить за допустимий діапазон – система повідомляє про помилку.

Постумови

Значення довірчої ймовірності збережено для використання в обчисленнях.

UC3 – Введення метрик вебзастосунку

Опис прецеденту

Користувач вводить структурні метрики вебзастосунку, для якого виконується оцінювання розміру.

Передумови

Програма запущена, форма введення метрик доступна.

Базовий потік

1. Користувач вводить кількість класів X1.

2. Вводить середню кількість методів у класі X2.

3. Вводить глибину дерева успадкування X3.

4. Підтверджує введення.

Альтернативні потоки

2а. Введені значення некоректні – система повідомляє про помилку.

Постумови

Метрики вебзастосунку прийняті системою.

UC4 – Оцінювання розміру вебзастосунку

Опис прецеденту

Система виконує обчислення прогнозного розміру вебзастосунку Y та статистичних інтервалів.

Передумови

Введені параметри регресійної моделі, значення α та метрики вебзастосунку.

Базовий потік

1. Користувач ініціює процес оцінювання.

2. Система обчислює значення Y.

3. Система обчислює межі довірчого інтервалу.

4. Система обчислює межі інтервалу прогнозування.

5. Результати відображаються користувачу.

Альтернативні потоки

2а. Вхідні дані відсутні або некоректні – система повідомляє про неможливість виконання обчислень.

Постумови

Отримано результати оцінювання розміру вебзастосунку.

UC5 – Збереження вхідних даних у БД

Опис прецеденту

Користувач ініціює збереження введених параметрів та метрик у БД.

Передумови

Вхідні дані введені та перевірені.

Базовий потік

1. Користувач обирає команду збереження.

2. Система передає дані до шару доступу до даних.

3. Дані зберігаються у БД H2.

Альтернативні потоки

3а. Помилка доступу до БД – система повідомляє користувача.

Постумови

Вхідні дані збережені у БД.

UC6 – Збереження результатів оцінювання у БД

Опис прецеденту

Система зберігає результати оцінювання розміру вебзастосунку у БД.

Передумови

Результати оцінювання отримані.

Базовий потік

1. Система формує структуру даних результатів.

2. Результати зберігаються у БД H2.

Альтернативні потоки

2a. Помилка збереження – система повідомляє користувача.

Постумови

Результати оцінювання збережені у БД.

UC7 – Перегляд результатів оцінювання

Опис прецеденту

Користувач переглядає результати оцінювання у графічному інтерфейсі програми.

Передумови

Результати оцінювання доступні у пам’яті програми або БД.

Базовий потік

1. Користувач відкриває форму перегляду результатів.

2. Система відображає значення Y та межі інтервалів.

Альтернативні потоки

2a. Результати відсутні – система повідомляє користувача.

Постумови

Користувач ознайомився з результатами оцінювання.

UC8 – Експорт результатів у CSV

Опис прецеденту

Користувач експортує результати оцінювання у файл формату CSV.

Передумови

Результати оцінювання доступні.

Базовий потік

1. Користувач ініціює експорт.

2. Система формує CSV-файл.

3. Файл зберігається у файловій системі.

Альтернативні потоки

3a. Помилка запису файлу – система повідомляє користувача.

Постумови

CSV-файл з результатами оцінювання створено.

Діаграма діяльності програми для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP, наведена на рисунку 3.3.

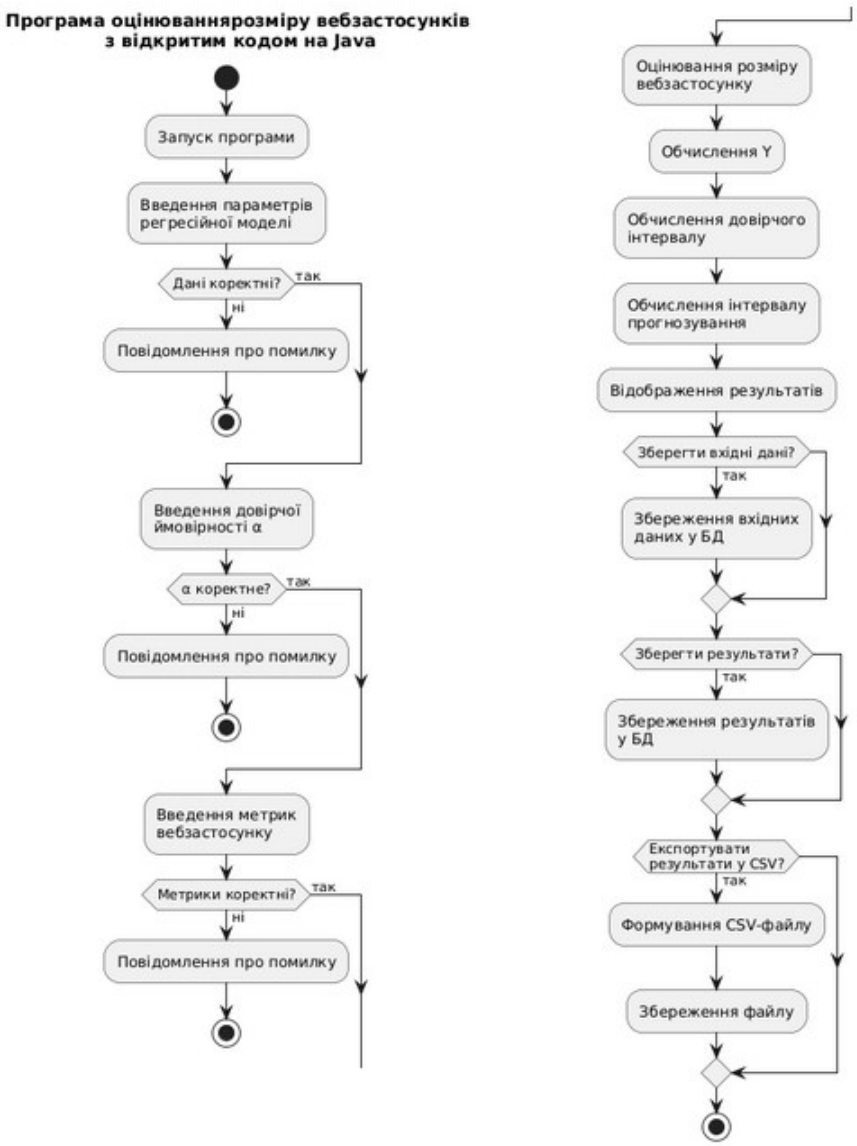

Рисунок 3.3 – Діаграма діяльності програми для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP

Діаграма діяльності програми відображає лінійний сценарій роботи користувача з умовними переходами. Перевірка коректності даних реалізована через вузли прийняття рішень. Збереження та експорт виконуються за ініціативою користувача. Діаграма діяльності програми узгоджується з діаграмою варіантів використання та багатошаровою архітектурою програми.

Розробимо ER-діаграму для програми оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP, яку наведено на рис. 3.4.

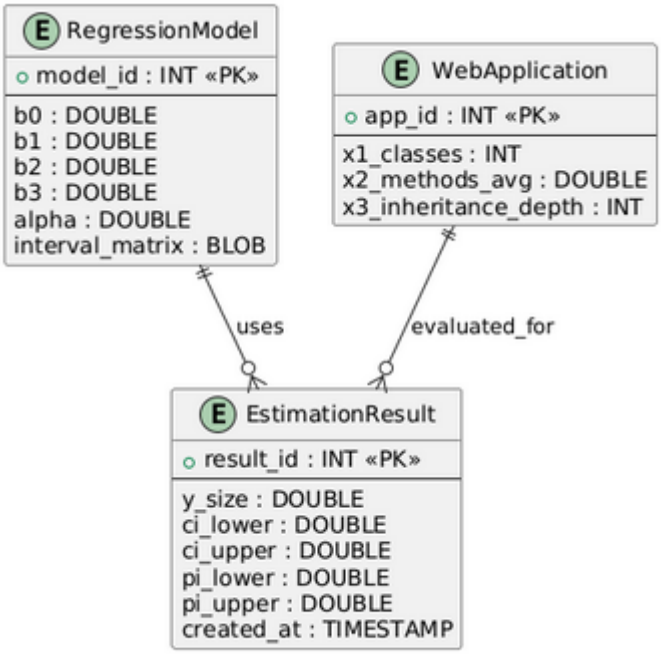

Рисунок 3.4 – ER-діаграма програми для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP

ER-діаграма відображає процес проєктування та не прив`язана до майбутнього вибору СУБД. Вона відображає зберігання параметрів регресійної моделі (RegressionModel), метрик вебзастосунку, для якого виконується оцінювання (WebApplication) та результатів оцінювання (EstimationResult). Зв’язки типу 1:N дозволяють виконувати багаторазові оцінювання з різними параметрами.

Далі розробимо прототип інтерфейсу користувача програми для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP, який наведено на рисунку 3.5.

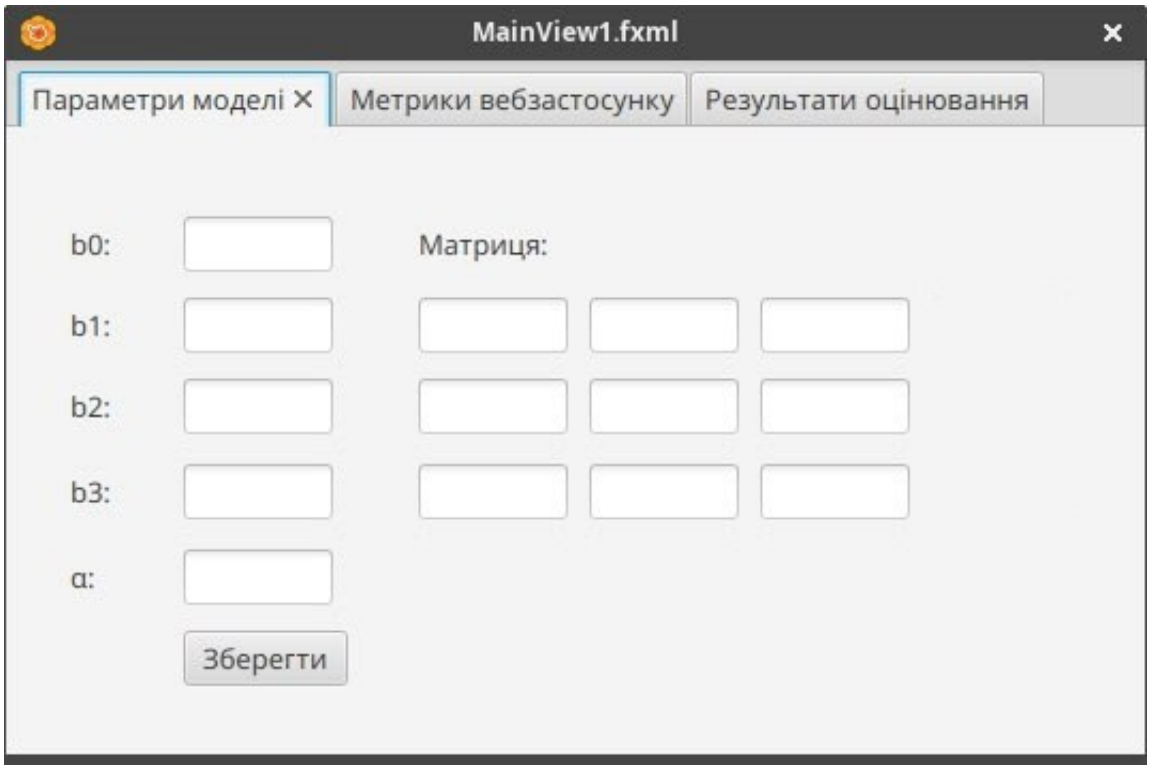

Рисунок 3.5 – Прототип інтерфейсу користувача програми для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP

Програма реалізується як настільний JavaFX-застосунок і складається з кількох логічно пов’язаних вікон. Основна взаємодія користувача відбувається в межах головного вікна з вкладками.

Головне вікно програми забезпечує доступ до всіх функцій програми оцінювання розміру вебзастосунків. Структура: верхня панель із назвою програми, вкладки: «Параметри моделі», «Метрики вебзастосунку», «Результати оцінювання».

Вкладка «Параметри моделі» призначена для введення параметрів нелінійної регресійної моделі та довірчої ймовірності.

Вкладка «Метрики вебзастосунку» призначена для введення структурних метрик вебзастосунку, для якого потрібно виконати оцінювання.

Вкладка «Результати оцінювання» призначена для відображення результатів розрахунків та виконання дій із ними.

Користувач може вільно переходити між вкладками. Результати з’являються лише після виконання оцінювання. Кнопки «Зберегти» та «Експорт» активні тільки при наявності результатів.

3.4 Технічний проєкт програми для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP

На початку технічного проєктування розробимо діаграму класів програми для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP. Розроблювана діаграма класів враховує багатошарову архітектуру, шаблон MVC, майбутню реалізацію (Controller / Service / DAO), JavaFX та вбудовану БД H2.

Діаграма класів програми для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP, наведена на рисунку 3.6.

Специфікації класів програми для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP

Розроблена діаграма класів відображає статичну структуру програми та побудована відповідно до принципів багатошарової архітектури з елементами шаблону Model–View–Controller (MVC). Класи згруповані у логічні пакети залежно від їх функціонального призначення.

Пакет app

Клас MainApp відповідає за ініціалізацію та запуск настільного застосунку на платформі JavaFX. Він містить метод start(Stage stage), у якому виконується завантаження FXML-опису інтерфейсу користувача, створення сцени та відображення головного вікна програми. Метод main(String[] args) є точкою входу в програму.

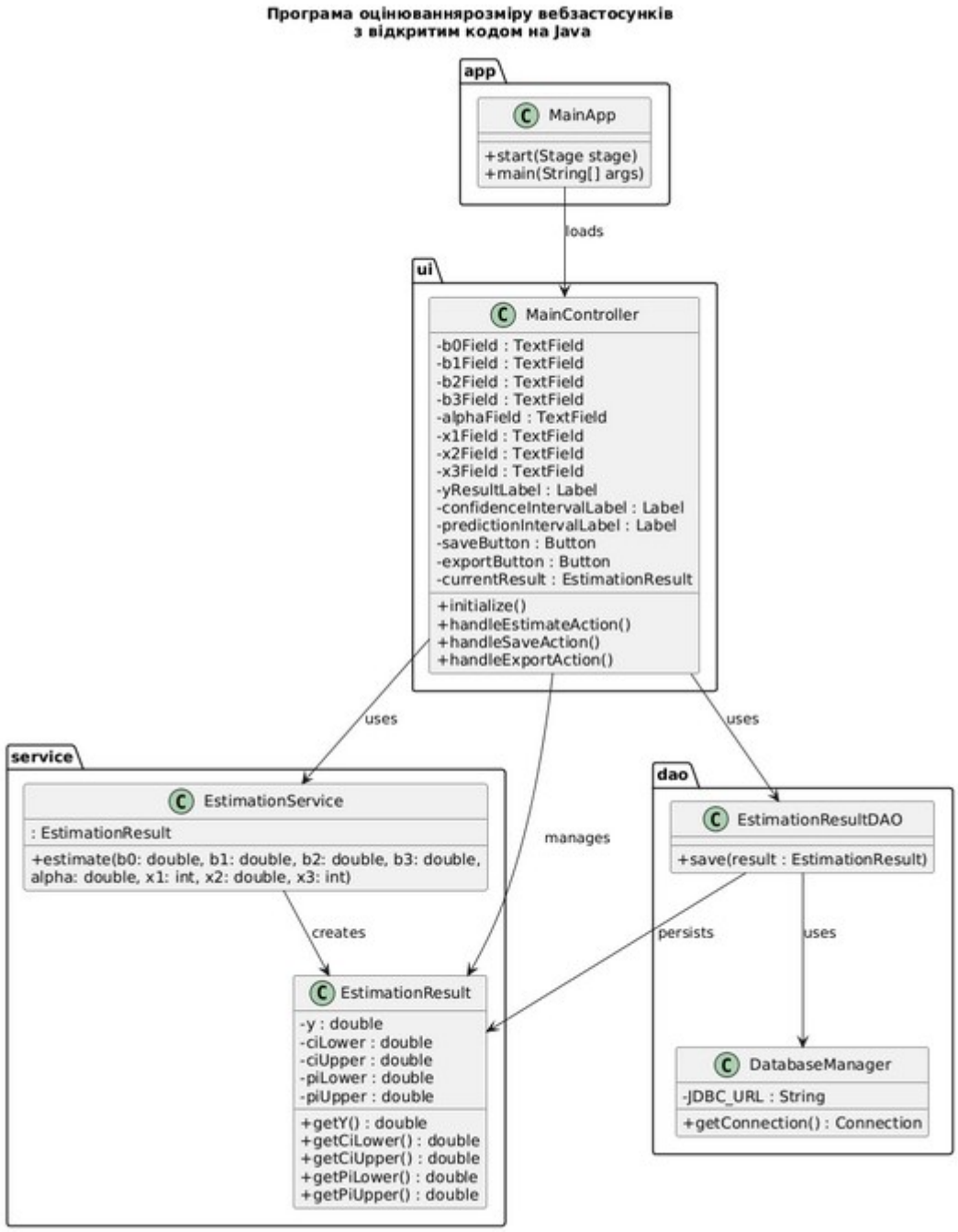

Рисунок 3.6 – Діаграма класів програми для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP

Пакет ui

Клас MainController реалізує контролер рівня представлення та відповідає за обробку подій користувача. Він містить посилання на елементи інтерфейсу користувача JavaFX (поля введення, кнопки та текстові мітки), які використовуються для введення параметрів регресійної моделі, метрик вебзастосунку та відображення результатів оцінювання.

Основними методами класу є:

initialize() – початкова ініціалізація стану інтерфейсу;

handleEstimateAction() – ініціація процесу оцінювання розміру вебзастосунку;

handleSaveAction() – збереження результатів оцінювання у базі даних;

handleExportAction() – експорт результатів у CSV-файл.

Клас MainController взаємодіє з сервісним та DAO-рівнями і керує об’єктом результатів оцінювання.

Пакет service

Клас EstimationService інкапсулює бізнес-логіку програми. Він відповідає за реалізацію нелінійної регресійної моделі, обчислення значення залежної змінної Y, а також визначення меж довірчого інтервалу та інтервалу прогнозування. Основним методом класу є estimate(...), який приймає параметри регресійної моделі та метрики вебзастосунку і повертає об’єкт результату оцінювання.

Клас EstimationResult є об’єктом передавання даних (DTO) та використовується для зберігання результатів оцінювання. Він містить значення розміру вебзастосунку, а також границі довірчого інтервалу та інтервалу прогнозування. Доступ до полів класу здійснюється через методи доступу (геттери).

Пакет dao

Клас DatabaseManager відповідає за керування підключенням до БД H2. Він містить параметри підключення та метод getConnection(), який забезпечує отримання з’єднання з БД. Також у класі реалізується ініціалізація структури БД.

Клас EstimationResultDAO реалізує шаблон доступу до даних (DAO) та відповідає за збереження результатів оцінювання у БД. Метод save(EstimationResult result) виконує запис даних у відповідну таблицю.

Взаємозв’язки між класами

Клас MainApp ініціює роботу класу MainController шляхом завантаження інтерфейсу користувача.

Клас MainController використовує EstimationService для виконання обчислень та EstimationResultDAO для збереження результатів.

Клас EstimationService створює об’єкти EstimationResult, які передаються між шарами програми.

Клас EstimationResultDAO використовує DatabaseManager для отримання з’єднання з БД та збереження об’єктів результатів оцінювання.

Таким чином, діаграма класів відображає чітке розділення відповідальностей між шарами програми, забезпечує модульність, розширюваність та підтримуваність програми, а також повністю відповідає обраній архітектурі та вимогам до системи.

Далі розробимо діаграму послідовності програми для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP. Нижче наведено діаграму послідовності для варіанту використання “Оцінювання розміру вебзастосунку” (див. рис. 3.7) та варіанту використання “Експорт результатів у CSV” (див. рис. 3.8).

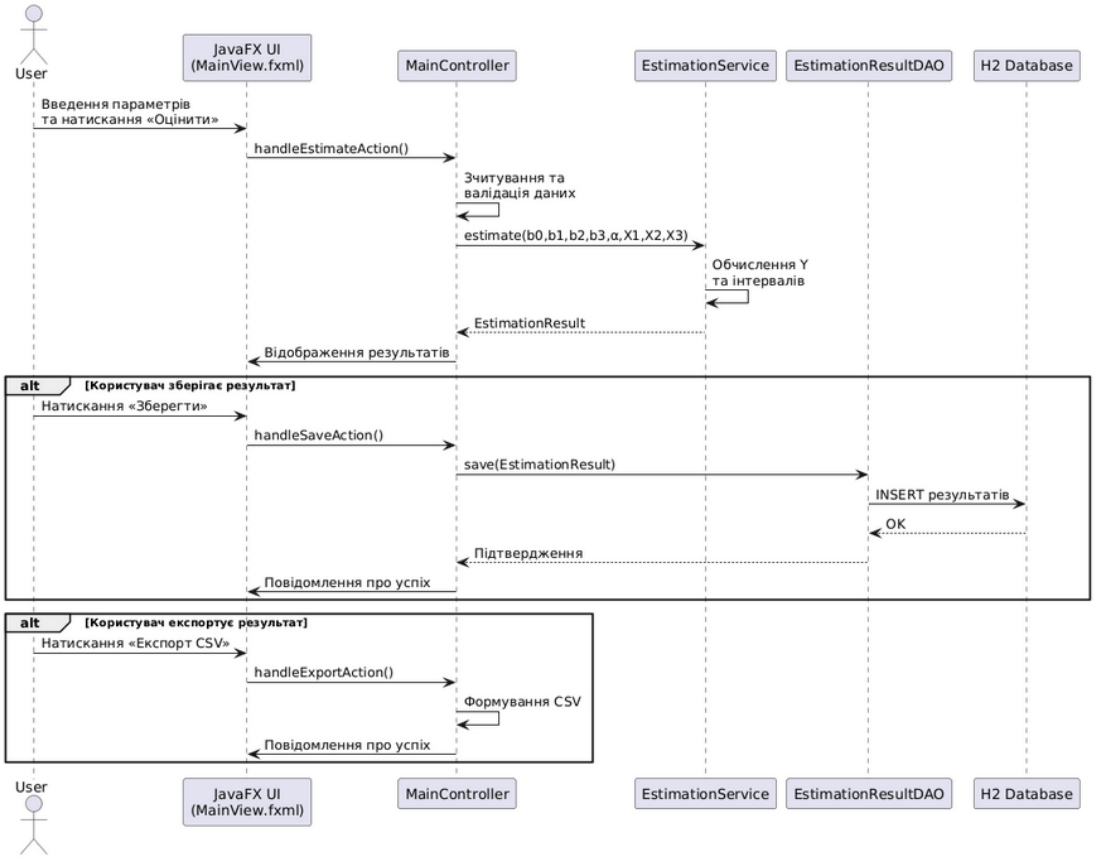

Рисунок 3.7 – Діаграма послідовності для варіанту використання “Оцінювання розміру вебзастосунку”

Діаграма послідовності для варіанту використання “Оцінювання розміру вебзастосунку” відображає ключовий функціональний сценарій, показує багатошарову архітектуру програму та узгоджена з діаграмою класів.

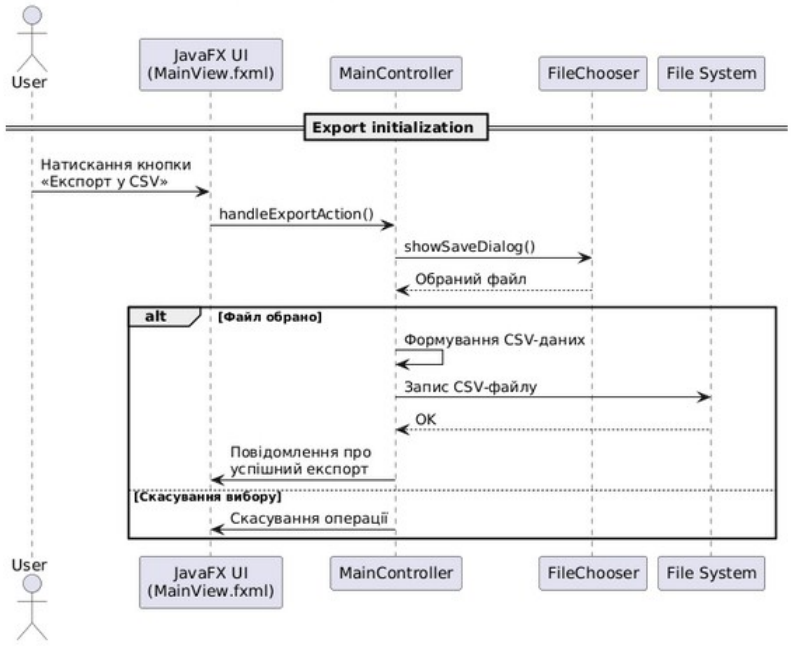

Рисунок 3.8 – Діаграма послідовності для варіанту використання “Експорт результатів у CSV”

Діаграма послідовності для варіанту використання “Експорт результатів у CSV” чітко показує роботу з файловою системою та не перевантажує основну діаграму.

3.5 Робочий проєкт програми для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP

Обґрунтування вибору технологій та стеку розробки

Для реалізації програми оцінювання розміру вебзастосунків з відкритим кодом було обрано сучасний та поширений стек розробки, який забезпечує модульність, масштабованість, надійність та зручність подальшого розширення програмного продукту.

Мова програмування Java

Мова програмування Java була обрана як основна мова розробки з огляду на такі переваги:

– платформонезалежність, що забезпечується виконанням програм у середовищі Java Virtual Machine (JVM);

– високий рівень надійності та безпеки, що є важливим для програм, які працюють з числовими розрахунками та базами даних;

– широке поширення у промисловій та академічній сферах, що робить результати роботи універсальними та зрозумілими;

– підтримка об’єктно-орієнтованої парадигми, яка добре узгоджується з використанням UML-діаграм та багатошарової архітектури;

– наявність великої кількості бібліотек для роботи з математичними моделями, базами даних та графічними інтерфейсами.

Технологія користувацького інтерфейсу JavaFX

Для реалізації графічного інтерфейсу користувача використано JavaFX, що зумовлено такими причинами:

– підтримка декларативного опису інтерфейсу за допомогою FXML, що спрощує розробку та супровід UI;

– чітке відокремлення логіки інтерфейсу від бізнес-логіки;

– підтримка сучасних компонентів, стилізації та подій;

– інтеграція з архітектурним шаблоном MVC;

– використання Scene Builder, що прискорює створення прототипів інтерфейсу.

СУБД H2 Database

Для збереження вхідних даних та результатів оцінювання було обрано H2 Database – вбудовану реляційну систему керування базами даних. Основні переваги H2:

– легковаговість та простота інтеграції у настільні застосунки;

– можливість роботи в embedded-режимі без встановлення окремого серверу;

– підтримка стандартного SQL;

– сумісність з JDBC;

– висока швидкість виконання операцій.

Технологія доступу до даних JDBC та шаблону DAO

Для взаємодії з БД використано JDBC у поєднанні з шаблоном проєктування DAO (Data Access Object). Це рішення дозволяє:

– відокремити логіку роботи з БД від бізнес-логіки;

– зменшити залежність програми від конкретної СУБД;

– полегшити зміну структури БД у майбутньому;

– забезпечити чистоту та читабельність коду.

Формат експорту даних CSV

Для експорту результатів оцінювання використано CSV-формат, оскільки він є простим та універсальним, підтримується більшістю табличних процесорів (Excel, LibreOffice), підходить для подальшого статистичного аналізу та не потребує додаткових бібліотек для обробки.

Таким чином, мова програмування Java є доцільним вибором для реалізації настільного прикладного ПЗ наукового призначення. JavaFX є сучасною та рекомендованою технологією для створення настільних застосунків мовою Java. Завдяки своїм властивостям СУБД H2 є оптимальним вибором для локальних наукових і навчальних програм, а DAO є стандартним і широко застосовуваним підходом у Java-проєктах.

Код розробленої програми для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP, наведений у додатку Б, опис програми наведений у додатку В, інструкція користувача програми, наведена у додатку Г.

Тестування програми

Для тестування розробленої програми для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP, було обрано метод еквівалентного розбиття.

Нижче приведено типи класів еквівалентності, які використовувалися при тестуванні:

– коректні значення – припустимими значення для системи згідно з її специфікацією;

– некоректні значення – неприпустимими значення для системи згідно з її специфікацією, які повинні бути відхилені в процесі роботи.

Були протестовані всі функції програми. Наведено приклад тестування функції «Введення метрик вебзастосунку». Класи еквівалентності вхідних даних для функції «Введення метрик вебзастосунку» представлені в таблиці 3.2. В таблиці 3.3 наведено тести з правильних класів еквівалентності для функції «Ввод даних про параметри розрахунку». В таблиці 3.4 наведено тести з неправильних класів еквівалентності для функції «Ввод даних про параметри розрахунку».

Таблиця 3.2 – Класи еквівалентності вхідних даних для функції «Введення метрик вебзастосунку»

Вхідні умови	Правильні класи еквівалентності	Неправильні класи еквівалентності
Метрики вебзастосунку X1, X2, X3, для якого виконується оцінювання розміру	1 Всі метрики вебзастосунку задані вірно	2 Одне чи більше значень задані невірно
		3 Відсутність одного чи декількох значень

Таблиця 3.3 – Тести з правильних класів еквівалентності для функції «Введення метрик вебзастосунку»

№ покритого класу еквівалентності	Вхідні дані	Результат
1	Всі метрики вебзастосунку задані вірно	Результат вірний

Таблиця 3.4 – Тести з неправильних класів еквівалентності для функції «Введення метрик вебзастосунку»

№ покритого класу еквівалентності	Вхідні дані	Результат
2	Одне чи більше значень задані невірно	Програма виводить повідомлення про помилку. Результат підтверджено
3	Відсутність одного чи декількох значень	Програма виводить повідомлення про помилку. Результат підтверджено

По результатам тестування можна зробити висновок про те, що програма для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP, працює вірно згідно вимог, які були сформульовані у технічному завданні.

Зовнішній вигляд програми для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP, представлено на рисунках 3.9 – 3.11: на рисунку 3.9 представлено вкладку програми "Параметри моделі", на рисунку 3.10 представлено вкладку програми "Метрики вебзастосунку", на рисунку 3.11 представлено вкладку програми "Результати оцінювання".

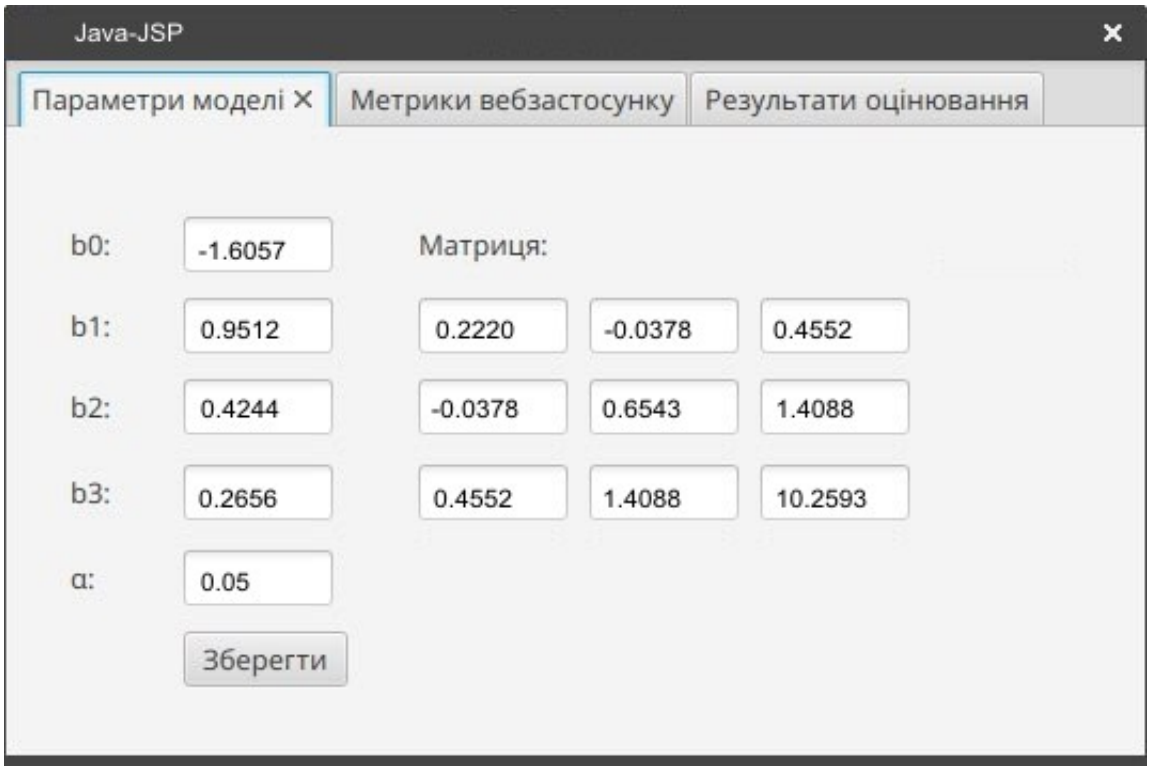

Рисунок 3.9 – Зовнішній вигляд програми, вкладка "Параметри моделі"

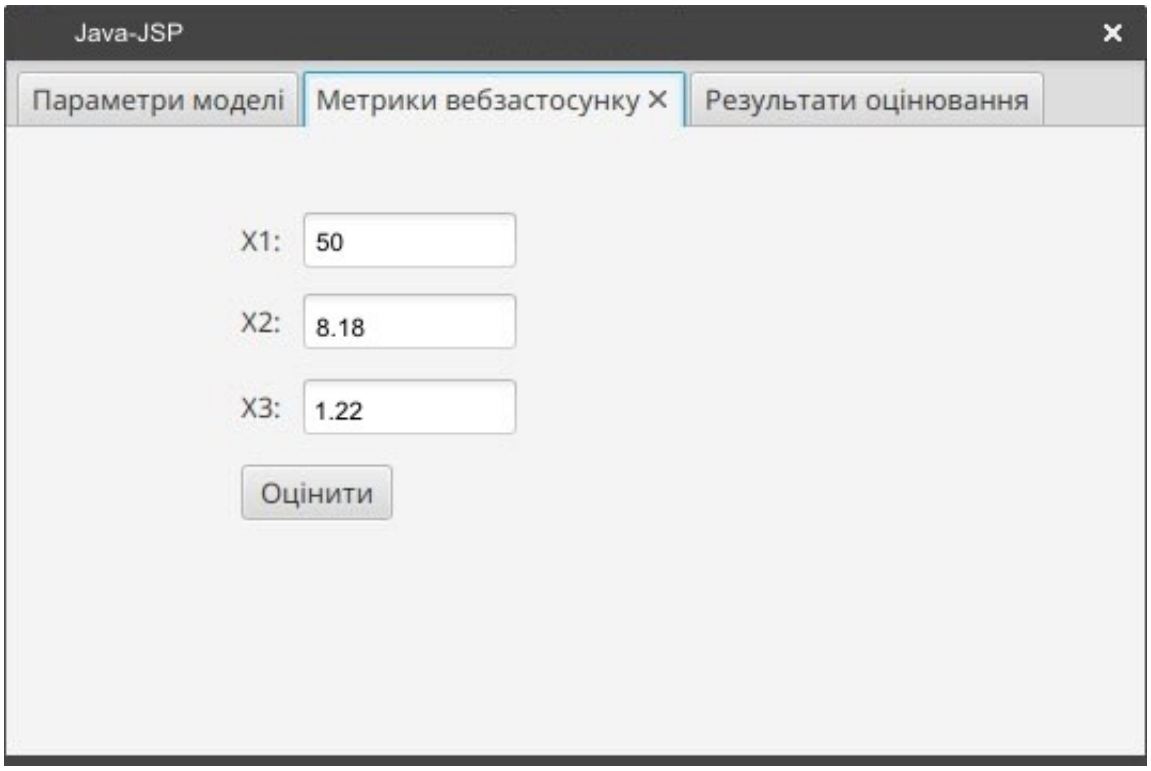

Рисунок 3.10 – Зовнішній вигляд програми, вкладка "Метрики вебзастосунку"

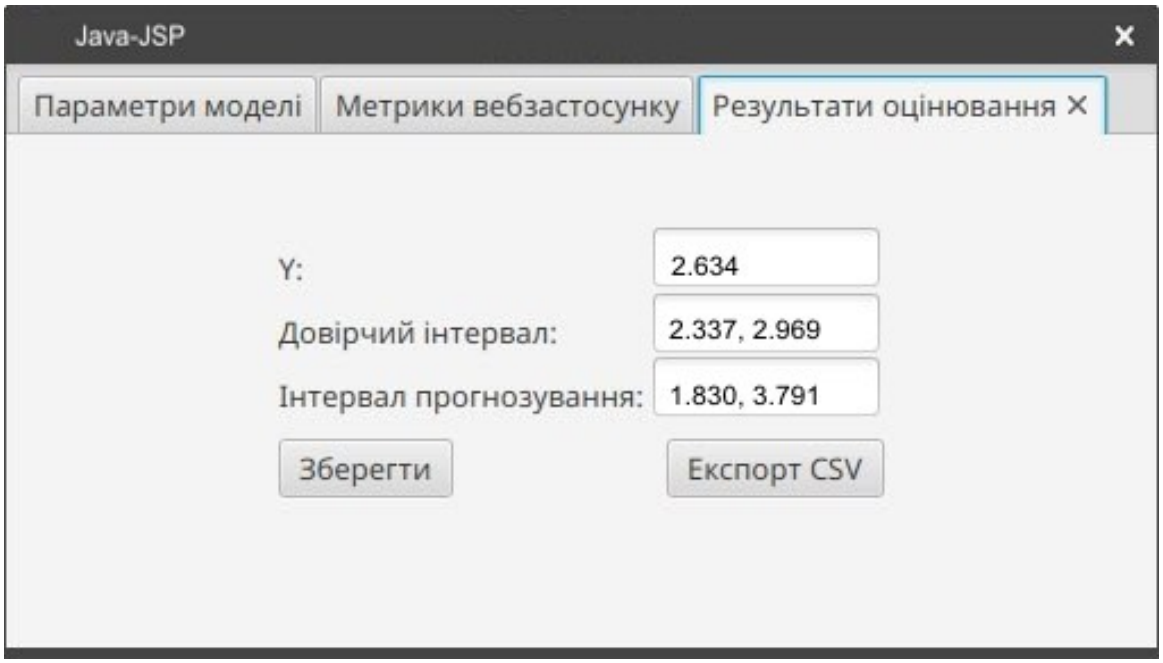

Рисунок 3.11 – Зовнішній вигляд програми, вкладка "Результати оцінювання"

Було проведено випробування програми для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP, виконане у відповідності з Програмою та методикою випробувань, яка представлена у Додатку Д.

3.6 Висновки за розділом 3

У розділі виконано проєкт програми для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP.

Основні результати:

– Виконано постановку задачі на розробку програми для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP.

- Розроблено архітектуру програми для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP.

- Виконано ескізний проєкт програми для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP, в рамках якого виявлено акторів та варіантів використання програми, розроблено діаграму та специфікації варіантів використання програми, розроблено діаграму діяльності та ER-діаграму програми та прототип інтерфейсу користувача.

- Виконано технічний проєкт програми для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP, в рамках якого розроблено діаграму та специфікації класів, та діаграму послідовності програми.

- Виконано робочий проєкт програми для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP, в рамках якого виконано обґрунтування вибору технологій та стеку розробки, проведено кодування, тестування та випробування програми, а також розроблена уся необхідна програмна документація.

4 РОЗРАХУНОК ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ ВПРОВАДЖЕННЯ

Розділ присвячений оцінюванню економічної ефективності розробки та впровадження програмного забезпечення для оцінювання розміру вебзастосунків з відкритим кодом на Java, створюваних із використанням технології JSP. Економічна ефективність розробки та впровадження програмного забезпечення є одним із ключових чинників прийняття рішень щодо інвестування в розробку.

З метою обґрунтування економічної доцільності розробки доцільно здійснити розрахунок собівартості програмного забезпечення та оцінити очікуваний прибуток від його використання. Ключовим критерієм оцінювання економічної ефективності інвестицій у створення програмного продукту є річний економічний ефект, поряд із яким важливим показником виступає термін окупності проєкту.

У цьому розділі наведено розрахунок собівартості розробки програмного забезпечення. На основі отриманих результатів визначається річний економічний ефект і прогнозований строк окупності інвестицій у розробку.

4.1 Розрахунок витрат на створення та впровадження програмного забезпечення

Витрати фінансів, які пов’язані зі створенням програмного забезпечення, охоплюють витрати на оплату праці розробників, утримання та експлуатацію комп’ютерного обладнання, необхідного для процесу розробки, використання засобів програмування, а також витрати на допоміжні матеріали й програмні компоненти.

Розробник, який бере участь у реалізації проєкту, отримує щомісячну заробітну плату в розмірі 30000 грн із передбаченим нарахуванням додаткової винагороди у розмірі 10 %. У результаті сукупний місячний дохід розробника становить 33000 грн. Середня вартість комп’ютерного обладнання складає

20000 грн. Вартість електричної енергії встановлена на рівні 4,32 грн за 1 кВт·год. Витрати на додаткові матеріали наведено у таблиці 4.1.

Таблиця 4.1 – Витрати на додаткові матеріали

Найменування витрат	Сума, грн.
Додаткові матеріали і комплектуючи вироби	470
Непередбачені витрати	200
Папір	180
Разом	850

Розрахунок витрат на розробку програмного забезпечення здійснюється на основі заданої формули:

$$С_{пр} = (З_{зп} + З_{сз} + З_{зг} + З_{е}) * Т + З_{м}, \quad (4.1)$$

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

$З_{сз}$ – відрахування на соціальні заходи (38% від $З_{зп}$), грн;

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

$З_{е}$ – витрати на електроенергію, в грн;

Т – тривалість розробки, місяців;

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

Щоб визначити місячні витрати на електроенергію ($З_{е}$), потрібно врахувати три параметри: потужність пристрою (60 Вт), час його роботи на місяць (160 годин) та вартість кіловат-години (4,32 грн).

$$З_{е} = 160 * 4{,}32 * 0{,}06 = 41{,}47 \text{ грн.}$$

Таблиця 4.2 містить перелік усіх додаткових витрат, пов'язаних із розробкою ПЗ.

Таблиця 4.2 – Додаткові витрати витрат, пов'язані із розробкою ПЗ.

Найменування витрат	Одиниця виміру	Кількість
Тривалість розробки	міс.	4
Основна та додаткова заробітні плати	грн.	33000
Відрахування на соціальні заходи	грн.	12540
Загальногосподарські витрати	грн.	3300
Витрати на додаткові матеріали	грн.	970
Витрати на електроенергію	грн.	41,47

Виходячи з формули 4.1, вартість розробки програмного забезпечення становить:

$$C_{пр} = (33000 + 12540 + 3300 + 41{,}47) * 4 + 970 = 196495{,}88 \text{ грн.}$$

Амортизаційні відрахування на устаткування складають 60% балансової вартості в рік: $A_{об} = 20000 * 0{,}6 = 12000$ гривень.

У межах підприємства річні витрати на допоміжні матеріали визначаються на рівні 5 % від вартості основного обладнання: $B_м = 20000 * 0{,}05 = 1000$ грн.

Річний обсяг часу роботи персонального комп'ютера, виражений у годинах, визначається за такою формулою: $Ф_м = 253{,}3 * T_з$, де $T_з$ – середнє місячне завантаження устаткування (приблизно 7 годин), а 253,3 – середнє значення кількості робочих днів протягом року: $Ф_м = 253{,}3 * 7 = 1773{,}1$ години.

Витрати на електроенергію $З_е$ при 1773,1 годинах роботи устаткування в рік складуть:

$$З_е = 1773{,}1 * 0{,}06 * 4{,}32 = 459{,}59 \text{ грн.}$$

Річні експлуатаційні витрати на використання персонального комп'ютера становитимуть:

$$З_{зр} = 12000 + 12540 + 459{,}59 = 24999{,}59 \text{ грн.}$$

Упродовж першого року загальний обсяг витрат, пов’язаних зі створенням та експлуатацією програмного забезпечення, становитиме:

$$З_{се} = 196495{,}88 + 24999{,}59 = 221495{,}47 \text{ грн.}$$

4.2 Розрахунок економічної ефективності розробки

Економічну ефективність упровадження програмного рішення визначимо на основі припущення, що його використання забезпечує скорочення витрат робочого часу одного працівника відповідно до експертної оцінки фахівців підприємства.

Річний фонд оплати праці одного працівника становить:

$$30000 * 12 = 360000 \text{ грн.}$$

Річний економічний ефект визначається за відповідною формулою, де:

$$Е_{рік} = \Delta С_n - Е_n * k\,, \qquad (4.2)$$

де $\Delta С_n$ – вивільнені кошти після впровадження системи, а саме 360000 гривень, без експлуатаційних витрат (24999,59гривні);
$Е_n$ – є коефіцієнтом ефективності та дорівнює коефіцієнту амортизації (0,6);
k – разові витрати на впровадження продукту, а саме 221495,47 гривні.

$$Е_{рік} = 360000 - 24999{,}59 - 221495{,}47 * 0{,}6 = 299397{,}5 \text{ грн.}$$

Термін окупності програмного забезпечення можна розрахувати за формулою:

$$\mathrm{T} = \frac{k}{\Delta \mathrm{C}} = \frac{221495{,}47}{360000 - 24999{,}59} \approx 0{,}66 \text{ року}$$

Отже термін окупності програмного забезпечення складає приблизно 8 місяців.

4.3 Висновки за розділом 4

У цьому розділі здійснено кількісну оцінку витрат, пов’язаних із розробкою та подальшою експлуатацією програмного забезпечення, а також проаналізовано його економічну ефективність і строк окупності. За результатами виконаних розрахунків встановлено, що впровадження програмного продукту забезпечує повне відшкодування понесених витрат протягом перших восьми місяців експлуатації. Отже, розробка та впровадження програмного забезпечення є економічно доцільними та фінансово обґрунтованими.

5 ОХОРОНА ПРАЦІ

5.1 Аналіз умов праці розробника програмного забезпечення

Праця розробника програмного забезпечення належить до категорії інтелектуальної діяльності з переважним використанням комп'ютерної техніки та засобів обробки інформації. Основною особливістю такої діяльності є тривала робота за персональним комп'ютером, високий рівень зорового навантаження, значна концентрація уваги та статичний характер робочої пози.

Робоче місце програміста, як правило, розташовується в офісному приміщенні та включає персональний комп'ютер або ноутбук, монітор, клавіатуру, маніпулятор типу «миша», офісний стіл та крісло. Умови праці регламентуються вимогами санітарних норм і правил, а також законодавством України у сфері охорони праці.

Відповідно до Закону України «Про охорону праці» [13], роботодавець зобов'язаний забезпечити безпечні та нешкідливі умови праці для працівників незалежно від характеру виконуваної роботи

Умови праці програміста зазвичай характеризуються:

– низьким рівнем фізичної активності;

– тривалим перебуванням у сидячому положенні;

– інтенсивною роботою з екраном дисплея;

– впливом психоемоційних навантажень, пов'язаних із дедлайнами та складністю завдань.

Згідно з Державними санітарними нормами роботи з відеодисплейними терміналами, робота програміста належить до категорії робіт з високим рівнем напруженості зорової праці.

Таким чином, умови праці розробника програмного забезпечення потребують ретельної організації робочого місця, дотримання регламентованого режиму праці та відпочинку, а також застосування профілактичних заходів щодо збереження здоров'я.

5.2 Небезпечні та шкідливі виробничі фактори при роботі з комп’ютерною технікою

Під час роботи з комп’ютерною технікою на працівника можуть впливати різноманітні небезпечні та шкідливі виробничі фактори, які умовно поділяються на фізичні, психофізіологічні та ергономічні.

Фізичні фактори

До фізичних факторів належать:

– електромагнітне випромінювання від моніторів та системних блоків;

– підвищене або недостатнє освітлення;

– несприятливі параметри мікроклімату (температура, вологість, швидкість руху повітря);

– шум від вентиляційних систем та комп’ютерного обладнання.

Норми допустимих рівнів електромагнітного випромінювання та параметрів мікроклімату регламентуються санітарними правилами.

Психофізіологічні фактори

До психофізіологічних факторів належать:

– тривале зорове напруження;

– монотонність роботи;

– висока концентрація уваги;

– емоційні перевантаження.

Тривала дія цих факторів може призводити до розвитку перевтоми, зниження працездатності, головного болю та нервового виснаження.

Ергономічні фактори

Неправильна організація робочого місця може спричиняти:

– порушення постави;

– захворювання опорно-рухового апарату;

– синдром зап’ястного каналу;

– біль у спині та шиї.

Вимоги до ергономіки робочого місця регламентуються стандартами ISO та рекомендаціями Всесвітньої організації охорони здоров’я [14].

5.3 Заходи з охорони праці та профілактики професійних захворювань

Для забезпечення безпечних умов праці розробника програмного забезпечення необхідно впроваджувати комплекс організаційних, технічних та профілактичних заходів.

Організаційні заходи

До організаційних заходів належать:

– дотримання режиму праці та відпочинку;

– проведення інструктажів з охорони праці;

– організація регламентованих перерв під час роботи за комп’ютером.

Згідно з санітарними нормами, тривалість безперервної роботи за ПК не повинна перевищувати 1–2 години залежно від характеру виконуваних завдань.

Технічні заходи

Технічні заходи включають:

– використання сучасних моніторів з низьким рівнем випромінювання;

– регулювання яскравості та контрастності екрана;

– забезпечення якісного освітлення робочої зони;

– використання ергономічних меблів.

Профілактичні заходи

Для профілактики професійних захворювань рекомендується:

– виконувати вправи для очей;

– проводити фізкультурні паузи;

– чергувати види діяльності;

– підтримувати правильну поставу.

Рекомендації щодо профілактики професійних захворювань наведені у матеріалах МОЗ України.

5.4 Висновки за розділом 5

У розділі проаналізовано умови праці розробника програмного забезпечення, визначено основні небезпечні та шкідливі виробничі фактори, а також розглянуто комплекс заходів щодо забезпечення безпеки та профілактики професійних захворювань.

Встановлено, що діяльність програміста, незважаючи на відсутність значних фізичних навантажень, пов’язана з підвищеним зоровим і психоемоційним напруженням, а також ризиками розвитку захворювань опорно-рухового апарату. Реалізація запропонованих заходів з охорони праці дозволяє створити безпечні та комфортні умови роботи, підвищити продуктивність праці та зберегти здоров’я працівників.

6 ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

6.1 Вплив інформаційних технологій на навколишнє середовище

У сучасному світі інформаційні технології є невід'ємною складовою соціально-економічного розвитку. Програмне забезпечення, комп'ютерна техніка, центри обробки даних і телекомунікаційна інфраструктура активно використовуються практично в усіх сферах діяльності людини. Водночас зростання ІТ-галузі супроводжується збільшенням навантаження на навколишнє середовище, що обумовлює необхідність аналізу екологічних аспектів розробки та експлуатації програмних систем.

Загальні правові засади охорони довкілля в Україні визначено Законом України «Про охорону навколишнього природного середовища» [15], який встановлює обов'язки суб'єктів господарювання щодо раціонального використання природних ресурсів та зменшення негативного впливу на довкілля.

Хоча діяльність програмістів і розробників програмного забезпечення не пов'язана безпосередньо з промисловим виробництвом, вона опосередковано впливає на екологічний стан через:

– споживання електричної енергії;

– використання комп'ютерної та серверної техніки;

– утворення електронних відходів;

– функціонування дата-центрів і хмарних платформ.

Особливо значний вплив мають великі центри обробки даних, які потребують значних обсягів електроенергії для живлення серверів та систем охолодження. За відсутності енергоефективних рішень це призводить до збільшення викидів парникових газів, якщо електроенергія виробляється з використанням викопного палива.

Європейські та міжнародні підходи до зменшення екологічного навантаження ІТ-сфери ґрунтуються на концепції сталого розвитку, що також

відображено в національному екологічному законодавстві України та нормативних документах Міністерства захисту довкілля та природних ресурсів.

Таким чином, інформаційні технології, попри свою нематеріальну природу, мають реальний екологічний вплив, який необхідно враховувати на всіх етапах життєвого циклу програмного забезпечення.

6.2 Екологічні ризики при експлуатації комп'ютерної техніки

Екологічні ризики, пов'язані з експлуатацією комп'ютерної техніки, проявляються насамперед через інтенсивне використання електроенергії, фізичне зношення обладнання та утворення електронних відходів. До комп'ютерної техніки належать персональні комп'ютери, ноутбуки, сервери, периферійні пристрої, мережеве обладнання та засоби безперебійного живлення.

Одним із ключових ризиків є накопичення електронних відходів (e-waste). До їх складу входять небезпечні речовини, зокрема важкі метали, пластмаси та хімічні сполуки, які у разі неналежної утилізації можуть потрапляти в ґрунт, воду та атмосферу. Питання поводження з такими відходами регулюється Законом України «Про управління відходами» [16], який визначає принципи запобігання утворенню відходів та їх екологічно безпечної обробки.

Неправильна утилізація застарілої комп'ютерної техніки є серйозною екологічною проблемою, особливо в умовах швидкого морального старіння обладнання. Постійне оновлення апаратного забезпечення призводить до зростання обсягів відходів електронного та електричного обладнання.

Ще одним екологічним ризиком є теплове забруднення, яке виникає в результаті роботи серверного обладнання та систем охолодження. Надмірне тепловиділення збільшує споживання електроенергії та негативно впливає на енергетичну ефективність об'єктів.

Крім того, експлуатація комп'ютерної техніки пов'язана з непрямими викидами в атмосферу. Ці аспекти узгоджуються з вимогами Закону України

«Про охорону атмосферного повітря» [17], який спрямований на зменшення негативного впливу господарської діяльності на стан повітряного середовища.

Таким чином, екологічні ризики експлуатації комп'ютерної техніки потребують комплексного підходу до управління життєвим циклом обладнання та впровадження екологічно відповідальних практик.

6.3 Заходи з охорони навколишнього середовища в процесі розробки програмного забезпечення

Зменшення негативного впливу IT-сфери на навколишнє середовище можливе шляхом впровадження комплексу організаційних, технічних та програмних заходів уже на етапі розробки програмного забезпечення.

До організаційних заходів належить:

– раціональне використання комп'ютерної техніки;

– продовження терміну експлуатації обладнання;

– впровадження політик екологічно відповідального використання ресурсів.

Важливу роль відіграє концепція Green Software Engineering, яка передбачає створення програмних продуктів із мінімальним споживанням обчислювальних ресурсів. Оптимізація алгоритмів, зменшення надлишкових обчислень і ефективне використання пам'яті безпосередньо знижують енергоспоживання апаратних ресурсів.

Серед технічних заходів доцільно виділити:

– використання енергоефективних серверів;

– застосування віртуалізації та контейнеризації;

– перехід на хмарні сервіси з високими стандартами енергоефективності.

Питання екологічної безпеки також пов'язані з дотриманням вимог інтегрованого запобігання забрудненню, що закріплено в Законі України «Про інтегроване запобігання та контроль забруднення» [18].

У процесі розробки програмного забезпечення важливо враховувати і вимоги щодо поводження з відходами електронного обладнання,

використовуючи сертифіковані канали утилізації та повторного використання компонентів відповідно до рекомендацій Міністерства захисту довкілля.

Отже, екологічна відповідальність у сфері розробки ПЗ є складовою сучасної інженерної культури та сприяє сталому розвитку ІТ-галузі.

6.4 Висновки за розділом 6

У розділі проаналізовано вплив інформаційних технологій на довкілля, визначено основні екологічні ризики, пов’язані з експлуатацією комп’ютерної техніки, та розглянуто заходи щодо зменшення негативного впливу ІТ-діяльності на навколишнє середовище.

Встановлено, що хоча програмна інженерія не є екологічно небезпечною галуззю у традиційному розумінні, вона має значний непрямий вплив через енергоспоживання, утворення електронних відходів і експлуатацію серверної інфраструктури. Дотримання вимог чинного екологічного законодавства України, впровадження принципів енергоефективності та використання підходів «зеленого ІТ» дозволяє зменшити негативний вплив на довкілля.

Отримані результати обґрунтовують доцільність застосування екологічно відповідальних підходів у процесі розробки програмного забезпечення та підтверджують актуальність інтеграції екологічних аспектів у сучасні ІТ-проєкти.

ВИСНОВКИ

Мета кваліфікаційної роботи полягала в підвищенні достовірності оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP і була повністю досягнута.

Для досягнення мети були розв’язані наступні завдання:

– Виконано аналіз методів та моделей оцінювання розміру вебзастосунків. Підкреслено, що в сучасних дослідженнях дедалі більшого поширення набувають регресійні підходи, у яких розмір програмного забезпечення оцінюється на основі сукупності кількох метрик. Також зазначено, що нелінійна природа залежностей між різними компонентами вебзастосунків з відкритим вихідним кодом, реалізованих мовою Java з використанням технології JSP, роблять застосування стандартних лінійних моделей прогнозування недостатньо точним. Тому розробка спеціалізованої нелінійної регресійної моделі, адаптованої до оцінювання розміру вказаного типу застосунків, є обґрунтованою та актуальною.

– Виконано збір даних з метрик обсяг коду в тисячах рядків, кількість класів, середня кількість методів на клас та глибина дерева успадкування з 42 вебзастосунків з відкритим кодом на Java, що створені з використанням технології JSP, відібраних на GitHub для побудови математичної моделі оцінювання розміру відповідного типу застосунків.

– Перевірено зібрані дані на нормальність розподілу та мультиколінеарність факторів та виявлено негаусівський розподіл та відсутність мультиколінеарності.

– Виконано нормалізацію даних нормалізуючим перетворенням десятковий логарифм.

– Побудовано трьохфакторну нелінійну регресійну модель оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP, із визначенням довірчих інтервалів і інтервалів прогнозування.

– Виконано оцінювання якості отриманої трьохфакторної нелінійної регресійної моделі з використанням метрик якості R^2, $MMRE$ та $PRED(0{,}25)$. Для порівняльного аналізу побудовано однофакторну нелінійну регресійну модель та виконано порівняння якості побудованих однофакторної та трьохфакторної моделей. Виявлено, що якість трьохфакторної моделі вагомо перевищує якість однофакторної.

– Розроблено програмне забезпечення для реалізації побудованої трьохфакторної нелінійної регресійної моделі оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Munialo S. W. A Review of Agile Software Effort Estimation Methods. International Journal of Computer Applications Technology and Research. 2016. Vol. 5. P. 612-618.

2. Tan H. B. K., Zhao Y., Zhang H.. Estimating LOC for information systems from their conceptual data models. Proceedings of the 28th International Conference on Software Engineering (ICSE '06). Shanghai, China, May 20-28, 2006. P. 321-330.

3. Приходько Н.В., Приходько С.Б. Нелінійна регресійна модель для оцінювання розміру програмного забезпечення промислових інформаційних систем на Java. Моделювання та інформаційні технології. 2018. Вип. 85. С. 81-88.

4. Макарова, Л.М., Приходько, Н.В., Кудін, О.О. Побудова нелінійної регресійної моделі для оцінювання розміру веб-додатків, реалізованих мовою Java. Вісник Херсонського національного технічного університету. 2019. № 2 (69). С.145-153.

5. Prykhodko, N.V., Prykhodko, S.B. The non-linear regression model to estimate the software size of open source java-based systems. Радіоелектроніка, інформатика, управління. 2018. № 3. С. 158-166.

6. Приходько С.Б., Приходько Н.В. Смикодуб Т.Г. Чотирьохфакорна нелінійна регресійна модель для оцінювання розміру Java-застосунків з відкритим кодом. Вчені записки ТНУ імені В.І. Вернадського. Серія: технічні науки. Київ, 2020. Том 31(70) Ч. 1 № 2. С. 157-162.

7. Орєхов О.С., Фаріонова Т.А. Математичні моделі для оцінювання розміру Java-застосунків. Вісник Херсонського національного технічного університету. Інформаційні технології. 2024. № 2. С. 196-203.

8. Петраков С.О., Базилюк О.П., Макарова Л.М., Пухалевич А.В. Математичне моделювання в задачах прогнозування розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP. Інформаційні технології: моделі, алгоритми, системи (ITMAS – 2025):

матеріали VI Міжнародної науково-практичної інтернет конференції (16-17 листопада 2025 р.). – Миколаїв: НУК імені адмірала Макарова, 2025. – С. 74-75.

9. TIOBE. URL: https://www.tiobe.com/tiobe-index/ (дата звернення: 23.11.2025).

10 Методичні вказівки та завдання до виконання лабораторних робіт з дисципліни «Емпіричні методи програмної інженерії» / С.Б. Приходько, Л.М. Макарова, Н.В. Приходько, А.В. Пухалевич. Миколаїв: НУК, 2023. 56 с.

11 GitHub. URL: https://github.com/ (дата звернення: 23.12.2025).

12. Mardia, K. V. Measures of multivariate skewness and kurtosis with applications. Biometrika. – 1970. – Vol. 57. – P. 519–530.

13. Про охорону праці: Закон України від 15.11.2024 № 2694-XII. URL: https://zakon.rada.gov.ua/laws/show/2694-12#Text (дата звернення: 10.11.2025 р.).

14. ISO 45000 family. Occupational health and safety. URL: https://www.iso.org/standards/popular/iso-45000-family (дата звернення: 10.11.2025 р.).

15. Про охорону навколишнього природного середовища: Закон України від 15.11.2024 № 1264-XII. URL: https://zakon.rada.gov.ua/laws/show/1264-12#Text (дата звернення: 13.11.2025 р.).

16. Про управління відходами: Закон України від 20.06.2022 р. № 2320-IX. URL: https://zakon.rada.gov.ua/laws/show/2320-20 (дата звернення: 13.11.2025 р.).

17. Про охорону атмосферного повітря: Закон України від 16.10.1992 № 2708-XII. URL: https://zakon.rada.gov.ua/laws/show/2707-12 (дата звернення: 13.11.2025 р.).

18. Про інтегроване запобігання та контроль промислового забруднення: Закон України від 16.07.2024 № 3855-IX. URL: https://zakon.rada.gov.ua/go/3855-20 (дата звернення: 13.11.2025 р.).

ДОДАТОК А – ТЕХНІЧНЕ ЗАВДАННЯ НА РОЗРОБКУ ПРОГРАМИ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ВЕБЗАСТОСУНКІВ З ВІДКРИТИМ КОДОМ НА JAVA, ЩО СТВОРЮЮТЬСЯ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЇ JSP

Вступ

Повна назва розроблюваного проєкту: Програма для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP.

Коротка назва: програма.

Галузь застосування: програма призначена для застосування в компаніях та командах розробників, що займаються розробкою вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP, та можуть бути використані для підтримки прийняття рішень у процесі аналізу та планування розробки вебзастосунків, а також для наукових досліджень у галузі розробки та оцінювання розміру ПЗ.

1 Підстави для розробки

Підставою для розробки є завдання на кваліфікаційну роботу, видане кафедрою ПЗАС НУК ім. адмірала Макарова.

2 Призначення розробки

2.1 Функціональне призначення програми

Дана програма призначена для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP.

2.2 Експлуатаційне призначення програми

Програма може використовуватися для підтримки прийняття рішень у процесі аналізу та планування розробки вебзастосунків, а також для наукових досліджень у галузі оцінювання програмного забезпечення.

3 Вимоги до програмного забезпечення

3.1 Вимоги до функціональних характеристик

3.1.1 Вимоги до складу виконуваних функцій

Програма повинна забезпечувати виконання таких основних функцій:

– введення параметрів нелінійної регресійної моделі, зокрема коефіцієнтів регресійного рівняння та матриці, необхідної для розрахунку інтервалів;

– введення значення довірчої ймовірності для подальших статистичних розрахунків;

– введення метрик вебзастосунку, для якого виконується оцінювання розміру;

– обчислення прогнозного значення розміру вебзастосунку та відповідних статистичних інтервалів;

– збереження вхідних даних у БД програми;

– збереження результатів оцінювання розміру вебзастосунків у БД програми;

– експорт результатів розрахунків у файл формату CSV.

3.1.2 Вимоги до організації вхідних та вихідних даних

Вхідними даними програми є:

– коефіцієнти нелінійного регресійного рівняння b_0, b_1, b_2, b_3;

– матриця, необхідна для обчислення довірчого інтервалу та інтервалу прогнозування;

– значення довірчої ймовірності α;

– значення метрик X_1 (кількість класів), X_2 (середня кількість методів у класі) та X_3 (глибина дерева успадкування).

Вхідні дані вводяться у відповідні поля вводу згідно з допустимим типом даних (числовий).

Вихідними даними програми є:

– обчислене значення залежної змінної Y, що характеризує розмір вебзастосунку та виражається у тисячах рядків коду;

– нижня та верхня межі довірчого інтервалу нелінійного регресійного рівняння;

– нижня та верхня межі інтервалу прогнозування нелінійного регресійного рівняння;

– файл формату CSV, що містить результати оцінювання розміру вебзастосунку.

Вихідні дані виводяться у відповідні поля виводу та зберігаються у БД програми.

3.2 Вимоги до надійності

Вимоги до забезпечення надійного функціонування ПЗ мають бути забезпечені виконанням сукупності організаційно-технічних заходів, перелік яких наведений нижче:

– організацією безперебійного живлення ПК;

– безперебійним каналом зв’язку.

Час відновлення після відмови

Час відновлення після відмови, викликаної проблемами з електроживленням або каналом зв’язку, не фатальним збоєм (не крахом) операційної системи, не перевищує 3 хвилин, які необхідні для перезавантаження ПК або відновлення каналу зв’язку.

3.3 Вимоги до умов експлуатації

3.3.1 Кліматичні умови експлуатації

Кліматичні умови експлуатації, при яких повинні забезпечуватися задані характеристики, повинні задовольняти вимогам, пропонованим до технічних засобів у частині умов їхньої експлуатації.

3.3.2 Вимоги до видів обслуговування

Програмний продукт не вимагає проведення будь-яких видів обслуговування.

3.3.3 Вимоги до чисельності та кваліфікації персоналу

З програмою працює один користувач, який повинен мати практичні навички роботи з комп`ютером та розумітися на предметній області.

3.4 Вимоги до складу та параметрів технічних засобів

Система повинна задовольняти вказаним вимогам на комп’ютері наступної мінімальної комплекції:

– CPU: Intel(R) i5 2,16 МГц або аналогічний за параметрами AMD;

– RAM: 4 GB;

– HDD: 10 GB вільного місця.

3.5 Вимоги до інформаційної та програмної сумісності

Для повноцінного функціонування програми можливо використовувати Microsoft Windows версії 10, Linux (дистрибутиви Ubuntu, Debian або аналогічні), macOS версії 10.15. Для запуску програми на комп’ютері користувача повинно бути встановлено JDK 17. JavaFX може бути підключений або як частина JDK (для відповідних збірок), або як окрема бібліотека OpenJFX. Для доступу до бази даних необхідна наявність JDBC-драйверу H2 Database.

3.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не пред’являються.

3.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не пред’являються.

4 Вимоги до програмної документації

Програмна документація повинна містити:

- технічне завдання;
- код програми;
- опис програми;
- інструкцію користувача;

– програму та методику випробувань ПЗ.

5 Техніко-економічні показники

Економічна ефективність впровадження ПЗ розрахована в розділі 4 кваліфікаційної роботи. Згідно з отриманими результатами впровадження даного ПЗ є доцільним.

6 Стадії та етапи розробки

Стадії та етапи розробки програми для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP, представлені в таблиці А.1.

Таблиця А.1 – Стадії та етапи розробки програмного забезпечення

Стадії розробки	Етапи розробки	Термін виконання
1. Технічне завдання	Розробка та затвердження технічного завдання	21.09.2025
2. Ескізний проєкт	Розробка ескізного проєкту	18.10.2025
	Затвердження ескізного проєкту	21.10.2025
3. Технічний проєкт	Розробка технічного проєкту	03.11.2025
	Затвердження технічного проєкту	06.11.2025
4. Робочий проєкт	Розробка програми	16.11.2025
	Розробка програмної документації	26.11.2025
	Випробування програми	01.12.2025

7 Порядок контролю та приймання

Контроль за аналізом та проєктуванням кожної окремої частини програмного забезпечення на кожному етапі з урахуванням вимог, визначених у технічному завдані. Кожна стадія розробки повинна бути представлена в зазначені строки та узгоджена із замовником.

Приймання проводяться відповідно до програми і методики випробувань і документують за допомогою протоколу проведення випробувань. У разі знаходження помилок підчас приймання програмного виробу складається акт про знайдені помилки, який підписується представниками замовника і розробника і затверджується керівниками організації – замовника та організації

– розробника. Розробник повинен протягом не більше ніж двох тижнів виправити зазначені помилки і оповістити замовника про повторне проведення перевірки.

ДОДАТОК Б – КОД ПРОГРАМИ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ВЕБЗАСТОСУНКІВ З ВІДКРИТИМ КОДОМ НА JAVA, ЩО СТВОРЮЮТЬСЯ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЇ JSP

MainApp.java

```
package app;

import dao.DatabaseInitializer;
import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Scene;
import javafx.stage.Stage;

public class MainApp extends Application {

    @Override
    public void start(Stage stage) throws Exception {

        // Ініціалізація БД при старті програми
        DatabaseInitializer.init();

        FXMLLoader loader = new
FXMLLoader(getClass().getResource("/MainView.fxml"));
        Scene scene = new Scene(loader.load());

        stage.setTitle("Software Size Estimation");
        stage.setScene(scene);
        stage.show();
    }

    public static void main(String[] args) {
        launch(args);
    }
}
```

MainController.java

```
package ui;

import dao.RegressionModelDAO;
import service.EstimationResult;
import service.EstimationService;
import service.RegressionModel;
import javafx.fxml.FXML;
import javafx.scene.control.*;

public class MainController {

    @FXML private TextField x1Field;
    @FXML private TextField x2Field;
    @FXML private TextField x3Field;

    @FXML private TextField b0Field;
    @FXML private TextField b1Field;
    @FXML private TextField b2Field;
    @FXML private TextField b3Field;
    @FXML private TextField alphaField;

    @FXML private Label yResultLabel;
    @FXML private Label confidenceIntervalLabel;
    @FXML private Label predictionIntervalLabel;

    private final EstimationService estimationService = new
EstimationService();
    private final RegressionModelDAO modelDAO = new
RegressionModelDAO();
    private EstimationResult currentResult;
    private RegressionModel currentModel;

    @FXML
    private void handleEstimateAction() {
        try {
```

```
            double b0 = Double.parseDouble(b0Field.getText());
            double b1 = Double.parseDouble(b1Field.getText());
            double b2 = Double.parseDouble(b2Field.getText());
            double b3 = Double.parseDouble(b3Field.getText());
            double alpha =
Double.parseDouble(alphaField.getText());

            int x1 = Integer.parseInt(x1Field.getText());
            double x2 = Double.parseDouble(x2Field.getText());
            int x3 = Integer.parseInt(x3Field.getText());

            // Створення RegressionModel
            double[][] intervalMatrix = new double[][];
            RegressionModel model = new RegressionModel(b0, b1,
b2, b3, alpha, intervalMatrix);

            currentResult = estimationService.estimate(model, x1,
x2, x3);
            currentModel = model;

            // Відображення результатів
            yResultLabel.setText(String.format("%.3f",
currentResult.getY()));
            confidenceIntervalLabel.setText(
                    String.format("[%.2f ; %.2f]",
currentResult.getCiLower(), currentResult.getCiUpper())
            );
            predictionIntervalLabel.setText(
                    String.format("[%.2f ; %.2f]",
currentResult.getPiLower(), currentResult.getPiUpper())
            );

        } catch (Exception e) {
            showError("Помилка", "Невірні вхідні дані");
        }
    }
```

```
    @FXML
    private void handleSaveAction() {
        try {
            modelDAO.save(currentModel);
            showInfo("Збереження", "Модель і результати збережено
в H2");
        } catch (Exception e) {
            showError("БД", "Помилка збереження");
        }
    }

    private void showError(String title, String msg) {
        Alert alert = new Alert(Alert.AlertType.ERROR);
        alert.setTitle(title);
        alert.setContentText(msg);
        alert.showAndWait();
    }

    private void showInfo(String title, String msg) {
        Alert alert = new Alert(Alert.AlertType.INFORMATION);
        alert.setTitle(title);
        alert.setContentText(msg);
        alert.showAndWait();
    }
}
```

RegressionModel.java

```
package service;

import java.io.Serializable;

public class RegressionModel implements Serializable {

    private int modelId;
    private double b0;
    private double b1;
```

```
    private double b2;
    private double b3;
    private double alpha;
    private double[][] intervalMatrix;

    public RegressionModel(double b0, double b1, double b2, double
b3, double alpha, double[][] intervalMatrix) {
        this.b0 = b0;
        this.b1 = b1;
        this.b2 = b2;
        this.b3 = b3;
        this.alpha = alpha;
        this.intervalMatrix = intervalMatrix;
    }

    public int getModelId() { return modelId; }
    public void setModelId(int modelId) { this.modelId = modelId;
}

    public double getB0() { return b0; }
    public double getB1() { return b1; }
    public double getB2() { return b2; }
    public double getB3() { return b3; }
    public double getAlpha() { return alpha; }
    public double[][] getIntervalMatrix() { return intervalMatrix;
}

    public void setIntervalMatrix(double[][] intervalMatrix) {
        this.intervalMatrix = intervalMatrix;
    }
}
```

EstimationResult.java

```
package service;

public class EstimationResult {
```

```
    private final double y;
    private final double ciLower;
    private final double ciUpper;
    private final double piLower;
    private final double piUpper;

    public EstimationResult(double y, double ciLower, double
ciUpper, double piLower, double piUpper) {
        this.y = y;
        this.ciLower = ciLower;
        this.ciUpper = ciUpper;
        this.piLower = piLower;
        this.piUpper = piUpper;
    }

    public double getY() { return y; }
    public double getCiLower() { return ciLower; }
    public double getCiUpper() { return ciUpper; }
    public double getPiLower() { return piLower; }
    public double getPiUpper() { return piUpper; }
}
```

RegressionModelDAO.java

```
package dao;

import service.RegressionModel;

import java.io.*;
import java.sql.*;

public class RegressionModelDAO {

    private static final String INSERT_SQL = """
        INSERT INTO RegressionModel (b0, b1, b2, b3, alpha,
interval_matrix)
```

```
        VALUES (?, ?, ?, ?, ?, ?)
    """;

    private static final String SELECT_SQL = """
        SELECT model_id, b0, b1, b2, b3, alpha, interval_matrix
        FROM RegressionModel
        WHERE model_id = ?
    """;

    public void save(RegressionModel model) throws SQLException,
IOException {
        try (Connection conn = DatabaseManager.getConnection();
             PreparedStatement ps =
conn.prepareStatement(INSERT_SQL,
Statement.RETURN_GENERATED_KEYS)) {

            ps.setDouble(1, model.getB0());
            ps.setDouble(2, model.getB1());
            ps.setDouble(3, model.getB2());
            ps.setDouble(4, model.getB3());
            ps.setDouble(5, model.getAlpha());

            // Серіалізація матриці у BLOB
            ByteArrayOutputStream bos = new
ByteArrayOutputStream();
            ObjectOutputStream oos = new ObjectOutputStream(bos);
            oos.writeObject(model.getIntervalMatrix());
            oos.flush();
            ps.setBytes(6, bos.toByteArray());

            ps.executeUpdate();

            try (ResultSet generatedKeys = ps.getGeneratedKeys())
{
                if (generatedKeys.next()) {
                    model.setModelId(generatedKeys.getInt(1));
                }
```

```
            }
        }
    }

    public RegressionModel findById(int modelId) throws
SQLException, IOException, ClassNotFoundException {
        try (Connection conn = DatabaseManager.getConnection();
             PreparedStatement ps =
conn.prepareStatement(SELECT_SQL)) {

            ps.setInt(1, modelId);
            try (ResultSet rs = ps.executeQuery()) {
                if (rs.next()) {
                    double b0 = rs.getDouble("b0");
                    double b1 = rs.getDouble("b1");
                    double b2 = rs.getDouble("b2");
                    double b3 = rs.getDouble("b3");
                    double alpha = rs.getDouble("alpha");

                    byte[] bytes = rs.getBytes("interval_matrix");
                    ObjectInputStream ois = new
ObjectInputStream(new ByteArrayInputStream(bytes));
                    double[][] intervalMatrix = (double[][])
ois.readObject();

                    RegressionModel model = new
RegressionModel(b0, b1, b2, b3, alpha, intervalMatrix);
                    model.setModelId(modelId);
                    return model;
                }
            }
        }
        return null;
    }
}
```

EstimationResultDAO.java

```
package dao;

import service.EstimationResult;
import java.sql.*;

public class EstimationResultDAO {

    private static final String INSERT_SQL = """
        INSERT INTO EstimationResult
        (model_id, app_id, y_size, ci_lower, ci_upper, pi_lower,
pi_upper)
        VALUES (?, ?, ?, ?, ?, ?, ?)
    """;

    public void save(int modelId, int appId, EstimationResult
result) throws SQLException {
        try (Connection conn = DatabaseManager.getConnection();
             PreparedStatement ps =
conn.prepareStatement(INSERT_SQL)) {

            ps.setInt(1, modelId);
            ps.setInt(2, appId);
            ps.setDouble(3, result.getY());
            ps.setDouble(4, result.getCiLower());
            ps.setDouble(5, result.getCiUpper());
            ps.setDouble(6, result.getPiLower());
            ps.setDouble(7, result.getPiUpper());

            ps.executeUpdate();
        }
    }
}
```

MainView.fxml

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<VBox spacing="10" xmlns="http://javafx.com/javafx/8.0.171"
      xmlns:fx="http://javafx.com/fxml/1"
      fx:controller="ui.MainController">

    <GridPane hgap="10" vgap="10">
        <Label text="b0:" GridPane.rowIndex="0"
GridPane.columnIndex="0"/>
        <TextField fx:id="b0Field" GridPane.rowIndex="0"
GridPane.columnIndex="1"/>
        <Label text="b1:" GridPane.rowIndex="1"
GridPane.columnIndex="0"/>
        <TextField fx:id="b1Field" GridPane.rowIndex="1"
GridPane.columnIndex="1"/>
        <Label text="b2:" GridPane.rowIndex="2"
GridPane.columnIndex="0"/>
        <TextField fx:id="b2Field" GridPane.rowIndex="2"
GridPane.columnIndex="1"/>
        <Label text="b3:" GridPane.rowIndex="3"
GridPane.columnIndex="0"/>
        <TextField fx:id="b3Field" GridPane.rowIndex="3"
GridPane.columnIndex="1"/>
        <Label text="α:" GridPane.rowIndex="4"
GridPane.columnIndex="0"/>
        <TextField fx:id="alphaField" GridPane.rowIndex="4"
GridPane.columnIndex="1"/>
    </GridPane>

    <Separator/>

    <GridPane hgap="10" vgap="10">
        <Label text="X1 (класів):" GridPane.rowIndex="0"
GridPane.columnIndex="0"/>
```

```
        <TextField fx:id="x1Field" GridPane.rowIndex="0"
GridPane.columnIndex="1"/>
        <Label text="X2 (методи/клас):" GridPane.rowIndex="1"
GridPane.columnIndex="0"/>
        <TextField fx:id="x2Field" GridPane.rowIndex="1"
GridPane.columnIndex="1"/>
        <Label text="X3 (глибина успадкування):"
GridPane.rowIndex="2" GridPane.columnIndex="0"/>
        <TextField fx:id="x3Field" GridPane.rowIndex="2"
GridPane.columnIndex="1"/>
    </GridPane>

    <Button text="Оцінити" onAction="#handleEstimateAction"/>
    <Button text="Зберегти" onAction="#handleSaveAction"/>
    <Button text="Експорт CSV" onAction="#handleExportAction"/>

    <Label text="Розмір Y:"/>
    <Label fx:id="yResultLabel"/>
    <Label text="Довірчий інтервал:"/>
    <Label fx:id="confidenceIntervalLabel"/>
    <Label text="Інтервал прогнозу:"/>
    <Label fx:id="predictionIntervalLabel"/>
</VBox>
```

DatabaseInitializer.java

```
package dao;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.sql.Connection;
import java.sql.Statement;
import java.util.stream.Collectors;

public class DatabaseInitializer {

    public static void init() {
```

```
        try (Connection conn = DatabaseManager.getConnection();
             Statement stmt = conn.createStatement()) {

            String sql = new BufferedReader(
                    new InputStreamReader(
                            DatabaseInitializer.class

.getResourceAsStream("/db/schema.sql")
                    )
            ).lines().collect(Collectors.joining("\n"));

            stmt.execute(sql);

        } catch (Exception e) {
            throw new RuntimeException("Помилка ініціалізації бази
даних", e);
        }
    }
}
```

```
-- Database initialization script

CREATE TABLE IF NOT EXISTS RegressionModel (
    model_id INT AUTO_INCREMENT PRIMARY KEY,
    b0 DOUBLE NOT NULL,
    b1 DOUBLE NOT NULL,
    b2 DOUBLE NOT NULL,
    b3 DOUBLE NOT NULL,
    alpha DOUBLE NOT NULL,
    interval_matrix BLOB NOT NULL
);

CREATE TABLE IF NOT EXISTS WebApplication (
    app_id INT AUTO_INCREMENT PRIMARY KEY,
    x1_classes INT NOT NULL,
    x2_methods_avg DOUBLE NOT NULL,
```

```
    x3_inheritance_depth INT NOT NULL
);

CREATE TABLE IF NOT EXISTS EstimationResult (
    result_id INT AUTO_INCREMENT PRIMARY KEY,
    model_id INT NOT NULL,
    app_id INT NOT NULL,

    y_size DOUBLE NOT NULL,
    ci_lower DOUBLE NOT NULL,
    ci_upper DOUBLE NOT NULL,
    pi_lower DOUBLE NOT NULL,
    pi_upper DOUBLE NOT NULL,

    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    CONSTRAINT fk_estimation_model
        FOREIGN KEY (model_id)
        REFERENCES RegressionModel(model_id)
        ON DELETE CASCADE,

    CONSTRAINT fk_estimation_application
        FOREIGN KEY (app_id)
        REFERENCES WebApplication(app_id)
        ON DELETE CASCADE
);
```

ДОДАТОК В – ОПИС ПРОГРАМИ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ВЕБЗАСТОСУНКІВ З ВІДКРИТИМ КОДОМ НА JAVA, ЩО СТВОРЮЮТЬСЯ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЇ JSP

1 Загальні відомості

Повна назва розроблюваного проєкту: Програма для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP.

Коротка назва: програма.

Галузь застосування: програма призначена для застосування в компаніях та командах розробників, що займаються розробкою вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP, та можуть бути використані для підтримки прийняття рішень у процесі аналізу та планування розробки вебзастосунків, а також для наукових досліджень у галузі розробки та оцінювання розміру ПЗ.

2 Функціональне призначення

Дана програма призначена для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP.

Програма повинна забезпечувати виконання таких основних функцій:

– введення параметрів нелінійної регресійної моделі, зокрема коефіцієнтів регресійного рівняння та матриці, необхідної для розрахунку інтервалів;

– введення значення довірчої ймовірності для подальших статистичних розрахунків;

– введення метрик вебзастосунку, для якого виконується оцінювання розміру;

– обчислення прогнозного значення розміру вебзастосунку та відповідних статистичних інтервалів;

– збереження вхідних даних у БД програми;

– збереження результатів оцінювання розміру вебзастосунків у БД програми;

– експорт результатів розрахунків у файл формату CSV.

3 Експлуатаційне призначення

Програма може використовуватися для підтримки прийняття рішень у процесі аналізу та планування розробки вебзастосунків, а також для наукових досліджень у галузі оцінювання програмного забезпечення.

4 Технічні засоби, що використовуються

Система повинна задовольняти вказаним вимогам на комп’ютері наступної мінімальної комплекції:

– CPU: Intel(R) i5 2,16 МГц або аналогічний за параметрами AMD;

– RAM: 4 GB;

– HDD: 10 GB вільного місця.

5 Програмні засоби, що використовуються

Для повноцінного функціонування програми можливо використовувати Microsoft Windows версії 10, Linux (дистрибутиви Ubuntu, Debian або аналогічні), macOS версії 10.15. Для запуску програми на комп’ютері користувача повинно бути встановлено JDK 17. JavaFX може бути підключений або як частина JDK (для відповідних збірок), або як окрема бібліотека OpenJFX. Для доступу до бази даних необхідна наявність JDBC-драйверу H2 Database.

6 Вхідні дані

Вхідними даними програми є:

– коефіцієнти нелінійного регресійного рівняння b_0, b_1, b_2, b_3;

– матриця, необхідна для обчислення довірчого інтервалу та інтервалу прогнозування;

– значення довірчої ймовірності α;

– значення метрик X_1 (кількість класів), X_2 (середня кількість методів у класі) та X_3 (глибина дерева успадкування).

Вхідні дані вводяться у відповідні поля вводу згідно з допустимим типом даних (числовий).

7 Вихідні дані

Вихідними даними програми є:

– обчислене значення залежної змінної Y, що характеризує розмір вебзастосунку та виражається у тисячах рядків коду;

– нижня та верхня межі довірчого інтервалу нелінійного регресійного рівняння;

– нижня та верхня межі інтервалу прогнозування нелінійного регресійного рівняння;

– файл формату CSV, що містить результати оцінювання розміру вебзастосунку.

Вихідні дані виводяться у відповідні поля виводу та зберігаються у БД програми.

ДОДАТОК Г – ІНСТРУКЦІЯ КОРИСТУВАЧА ПРОГРАМИ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ВЕБЗАСТОСУНКІВ З ВІДКРИТИМ КОДОМ НА JAVA, ЩО СТВОРЮЮТЬСЯ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЇ JSP

Програма для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP, є настільним застосунком, створеним мовою програмування Java. Програма не має прямої інтеграції з репозиторіями відкритого коду; всі параметри математичної моделі та метрики вебзастосунку вводяться користувачем вручну на основі попереднього аналізу проєктів.

Програма встановлюється безпосередньо на комп’ютер користувача. Програма запускається або за допомогою ярлика, який знаходиться на робочому столі, або за допомогою вибору відповідного пункту меню «Пуск» – «Програми». Програма має один режим використання.

Після запуску програми з’являється головне вікно програми для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP, яке представлено на рисунку Г.1.

Головне вікно програми забезпечує доступ до всіх функцій програми оцінювання розміру вебзастосунків. Структура: верхня панель із назвою програми, вкладки: «Параметри моделі», «Метрики вебзастосунку», «Результати оцінювання».

Вкладка «Параметри моделі» призначена для введення параметрів нелінійної регресійної моделі та довірчої ймовірності.

Вкладка «Метрики вебзастосунку» призначена для введення структурних метрик вебзастосунку, для якого потрібно виконати оцінювання.

Вкладка «Результати оцінювання» призначена для відображення результатів розрахунків та виконання дій із ними.

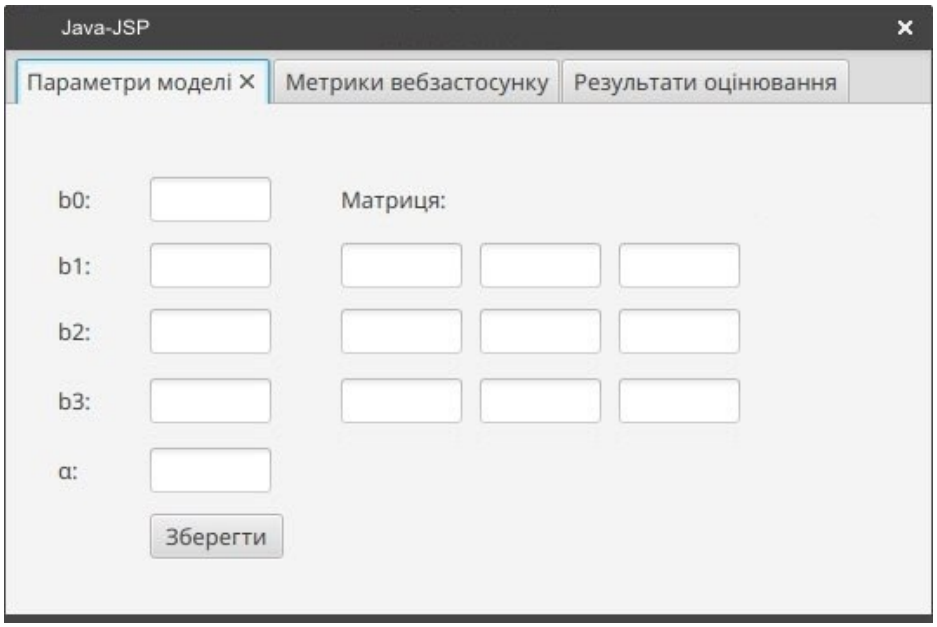

Рисунок Г.1 – Головне вікно програми для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP

Для початку роботи програми треба ввести дані про параметри регресійної моделі, значення α та матрицю для розрахунку інтервалів. Для збереження введеної інформації у БД програми потрібно натиснути кнопку "Зберегти" (див. рис. Г.2).

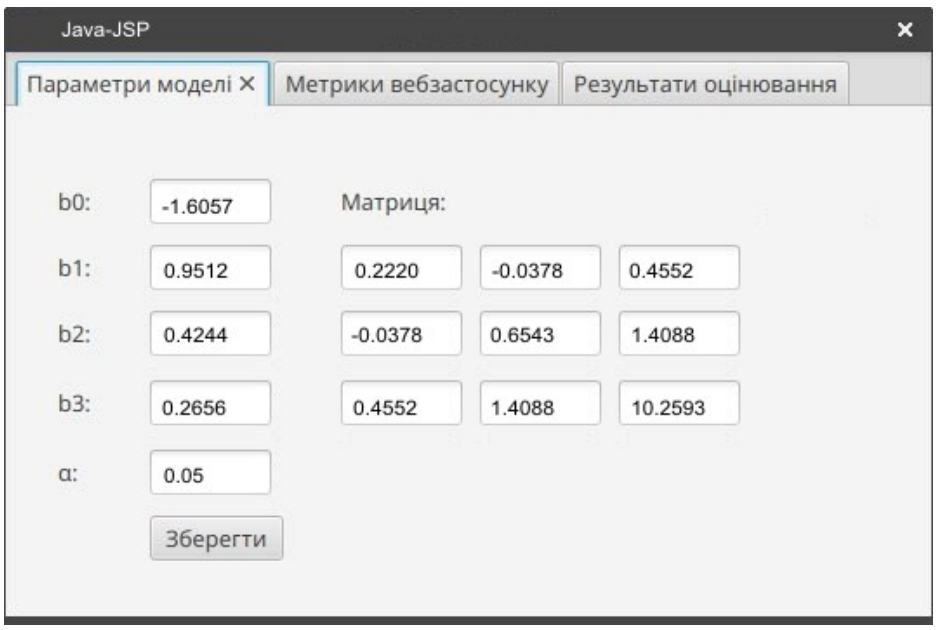

Рисунок Г.2 – Зовнішній вигляд головного вікна програми з введеними даними

Далі потрібно перейти на вкладку «Метрики вебзастосунку» та ввести значення метрик застосунку, для якого потрібно виконати оцінювання. Для виконання розрахунку потрібно натиснути кнопку "Оцінити". Зовнішній вигляд програми із вкладкою "Метрики вебзастосунку" з заповненими даними наведено на рис. Г.3.

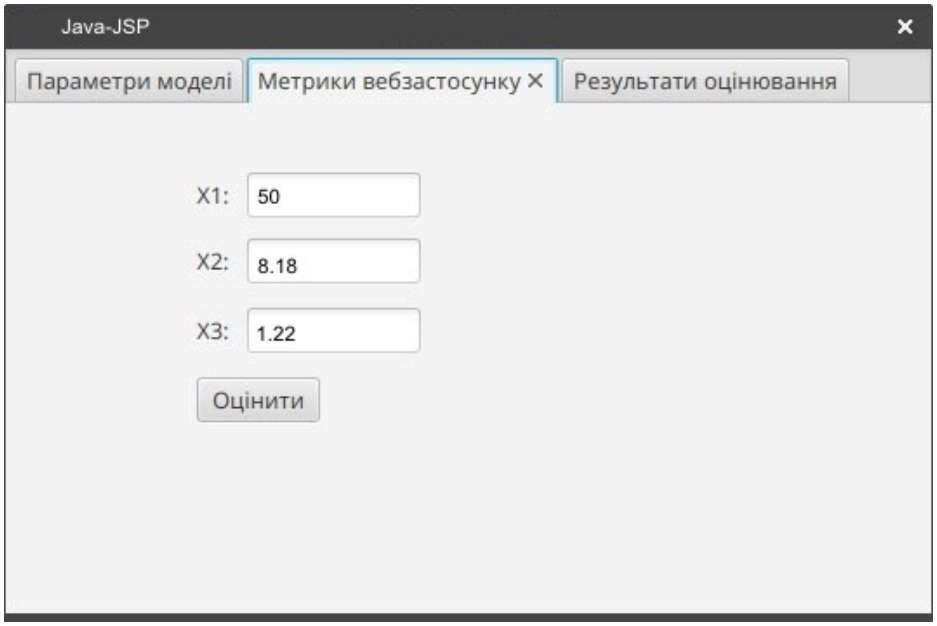

Рисунок Г.3 – Зовнішній вигляд вкладки "Метрики вебзастосунку" з введеними даними

Результати оцінювання кількості рядків коду (значення Y, KLOC) та розраховані довірчий інтервал і інтервал прогнозування нелінійної регресії відображаються на вкладці «Результати оцінювання» у відповідних полях виводу. Зовнішній вигляд програми із вкладкою "Результати оцінювання" з розрахованими результатами наведено на рис. Г.4.

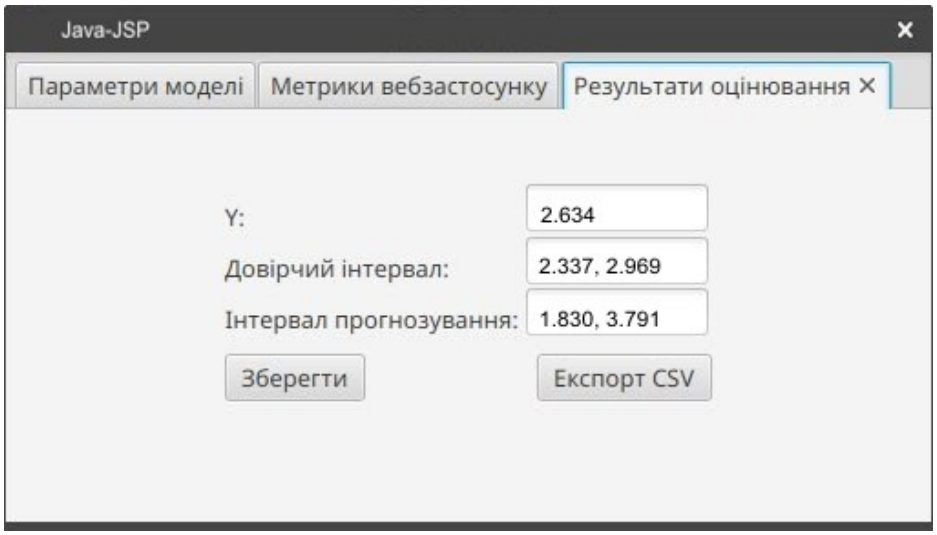

Рисунок Г.4 – Зовнішній вигляд вкладки "Результати оцінювання" з розрахованими результатами

Для збереження результатів розрахунку у БД програми потрібно натиснути кнопку "Зберегти". Для експорту результатів розрахунку у файл формату .CSV потрібно натиснути кнопку "Експорт CSV".

ДОДАТОК Д – ПРОГРАМА ТА МЕТОДИКА ВИПРОБУВАННЯ ПРОГРАМИ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ВЕБЗАСТОСУНКІВ З ВІДКРИТИМ КОДОМ НА JAVA, ЩО СТВОРЮЮТЬСЯ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЇ JSP

1 Об’єкт випробовування

Об’єкт випробувань: Програма для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP.

2 Мета випробовування

Перевірка функціональних можливостей ПЗ для оцінювання розміру вебзастосунків з відкритим кодом на Java, що створюються з використанням технології JSP, на відповідність вимогам технічного завдання.

3 Вимоги до програми

При проведенні випробувань, функціональні характеристики ПЗ підлягають перевірці на відповідність вимогам, викладеним у пункті «Вимоги до функціональних характеристик» технічного завдання.

Для проведення випробувань підлягають перевірці наступні функції:

– введення параметрів нелінійної регресійної моделі, зокрема коефіцієнтів регресійного рівняння та матриці, необхідної для розрахунку інтервалів;

– введення значення довірчої ймовірності для подальших статистичних розрахунків;

– введення метрик вебзастосунку, для якого виконується оцінювання розміру;

– обчислення прогнозного значення розміру вебзастосунку та відповідних статистичних інтервалів;

– збереження вхідних даних у БД програми;

– збереження результатів оцінювання розміру вебзастосунків у БД програми;

– експорт результатів розрахунків у файл формату CSV.

4 Вимоги до програмної документації

Програмна документація повинна містити:

– технічне завдання;

– текст програми;

– опис програми;

– інструкцію користувача;

– програму та методику випробування ПЗ.

5 Засоби та порядок випробувань

Система повинна задовольняти вказаним вимогам на комп’ютері наступної мінімальної комплекції:

– CPU: Intel(R) i5 2,16 МГц або аналогічний за параметрами AMD;

– RAM: 4 GB;

– HDD: 10 GB вільного місця.

Програмні засоби, що повинні використовуватися під час випробувань

Для повноцінного функціонування програми можливо використовувати Microsoft Windows версії 10, Linux (дистрибутиви Ubuntu, Debian або аналогічні), macOS версії 10.15. Для запуску програми на комп’ютері користувача повинно бути встановлено JDK 17. JavaFX може бути підключений або як частина JDK (для відповідних збірок), або як окрема бібліотека OpenJFX. Для доступу до бази даних необхідна наявність JDBC-драйверу H2 Database.

Випробування ПЗ повинні виконуватися в наступному порядку:

– виконуються інструкції до кожної функції;

– робиться аналіз отриманих результатів і встановлюється відповідність програмного продукту вимогам і системним документам.

При виявлені помилок у роботі програмного продукту складається їх перелік і обговорюється термін їх виправлення розробником. Після цього

замовник проводить повторне тестування в повному обсязі (можливе використання нових чи додаткових тестів).

6 Методи випробувань

Випробування проводиться за стратегією «чорного ящика». Всі функції програми будуть проходити відповідні випробовування. Перелік випробувань, що пропонуються до виконання, наведено в таблиці Д.1.

Таблиця Д.1 – Перелік випробувань, що пропонуються до виконання

№	Дія	Очікуваний результат
1	Введення параметрів нелінійної регресійної моделі, зокрема коефіцієнтів регресійного рівняння та матриці, необхідної для розрахунку інтервалів	Відображення результату введення значень коефіцієнтів регресійного рівняння b_0, b_1, b_2, b_3 та матриці для розрахунку інтервалів
2	Введення значення довірчої ймовірності для подальших статистичних розрахунків	Відображення результату введення значення довірчої ймовірності α
3	Введення метрик вебзастосунку, для якого виконується оцінювання розміру	Відображення результату введення значень метрик X_1, X_2 та X_3
4	Збереження вхідних даних у БД програми	Запис у БД програми значень вхідних даних b_0, b_1, b_2, b_3, α, та матриці для розрахунку інтервалів
5	Обчислення прогнозного значення розміру вебзастосунку та відповідних статистичних інтервалів	Відображення результату обчислення кількості рядків коду (значення Y, KLOC) та границь довірчого інтервалу і інтервалу прогнозування
6	Збереження результатів оцінювання розміру вебзастосунку у БД програми	Запис у БД програми значень метрик X_1, X_2 та X_3 та вихідних даних Y і границь довірчого інтервалу та інтервалу прогнозування
7	Експорт результатів розрахунків у файл формату CSV	Експорт результатів розрахунків (значень метрик X_1, X_2 та X_3 та вихідних даних Y і границь довірчого інтервалу та інтервалу прогнозування) у файл формату .CSV