Національний університет кораблебудування

імені адмірала Макарова

Інститут комп'ютерних наук та управління проектами
(повне найменування інституту, назва факультету)

Кафедра інформаційних управляючих систем та технологій
(повна назва кафедри)

Пояснювальна записка

до дипломного проекту (роботи)
бакалавр
(освітньо-кваліфікаційний рівень)

на тему Розробка інструментального засобу прогнозування обсягу програмного забезпечення, створюваного мовою Java на основі UML метрик.

Виконав: студент 4 курсу, групи 4142
напряму підготовки (спеціальності)
122 «Комп'ютерні науки»
(шифр і назва напряму підготовки, спеціальності)

Каратай М.В.
(прізвище та ініціали)
Керівник Беркунський Є.Ю.
(прізвище та ініціали)

Рецензент
(прізвище та ініціали)

м. Миколаїв – 2020 року

Національний університет кораблебудування імені адмірала Макарова

Інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій
Освітньо-кваліфікаційний рівень бакалавр
Спеціальність 122 «Комп'ютерні науки»
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри ІУСТ
К.т.н., доц. Михелєв І.Л. _____
" " _____ 2020 року

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТУ

Каратаю Максиму Віталійовичу
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Розробка інструментального засобу прогнозування обсягу програмного забезпечення, створюваного мовою Java на основі UML метрик.
керівник проекту (роботи) Беркунський Є.Ю.,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "19" травня 2020 р. № 288-уч
2. Строк подання студентом проекту (роботи) 18.06.2020.
3. Вихідні дані до проекту (роботи) ДСТУ щодо обробки інформації, літературні джерела, технічна документація системи, матеріали практики
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Вступ. 1 Розробка концепції системи. 1.1 Аналіз предметної області. 1.2 Аналіз існуючих методів прогнозування. 1.3 Формування вимог до проєкту. 2 Розробка проектних рішень. 2.1 Загальносистемні проєктні рішення. 2.2 Рішення по математичному забезпеченню. 2.3. Рішення по інформаційному забезпеченню. 3 Реалізація проекту автоматизованої системи. 4 Охорона праці. Додатки
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Методи прогнозування за ступенем формалізації. Схема функціональної структури. Логічна модель даних. Діаграма варіантів використання системи. Діаграма декомпозиції варіанту використання «Керування проєктами». Діаграма декомпозиції варіанту використання «Керування наборами проєктів». Діаграма діяльності користувача. Діаграма основних класів системи. Діаграма розгортання системи. Фізична модель бази даних. Приклади інтерфейсу користувача.

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Беркунський Є.Ю.		
2	Беркунський Є.Ю.		
3	Беркунський Є.Ю.		
4	Гурець Н.В.		

7. Дата видачі завдання _____.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Розробка концепції системи		
1.1	Аналіз предметної області		
1.2	Аналіз існуючих методів прогнозування		
1.3	Формування вимог до проєкту		
2	Розробка проектних рішень		
2.1	Загальносистемні проєктні рішення		
2.2	Рішення по математичному забезпеченню		
2.3	Рішення по інформаційному забезпеченню		
3	Реалізація проекту автоматизованої системи		
4	Охорона праці		

Студент _____ **Каратай М.В.**
(підпис) (прізвище та ініціали)

Керівник проекту (роботи) _____ **Беркунський Є.Ю.**
(підпис) (прізвище та ініціали)

Анотація

Представлена робота присвячена розробці інструментального засобу для прогнозування обсягу ПЗ створюваного мовою Java на основі UML метрик.

Призначення системи - надати можливість користувачам прогнозувати обсяг програмного забезпечення яке планується створювати за допомогою мови програмування Java ще на етапі проєктування UML діаграм, що надає змогу більш точно розрахувати необхідні ресурси для створення кінцевого продукту користуючись моделями розрахунків виду «COCOMO II».

В роботі аналізуються існуючі рішення та методи створення моделей прогнозування, досліджуються архітектурні підходи і засоби створення таких систем, формуються вимоги до системи, розробляються концепції системи і проводиться вибір найкращої, розробляється технічне завдання. Проводиться розробка основних проектних рішень за технічним, інформаційним та програмним забезпеченням, формується робоча документація для реалізованого програмного забезпечення.

Дипломна робота виконана на 118 аркушах машинописного тексту, містить 34 рисунки, 3 таблиці, 3 додатки, список використаних джерел із 20 посилань.

Ключові слова: обсяг програмного забезпечення, методи прогнозування, множинна регресія, Java, UML, Spring framework, JavaScript.

Abstract

This work is dedicated to the development of a tool for predicting the size of software, that will be created via the Java language, based on the UML metrics.

The main goal of the system is to allows users to forecast the size of the software that planned to be created via Java language as early as the UML diagrams building stage, which enable them to more accurately estimate the resource required to create end product using estimation models like “COCOMO II”.

In the work analyzes the existing solutions and methods for the forecasting model creation, investigates the architectural approaches, and means of creating such a system. Forms the requirements for the system, develops the concepts of the system, and selects the best one, develops the technical task. Main design solutions for technical, information, and software being developed, working documentation for the implemented software is being formed.

The work is done on 118 sheets of typewritten text, contains 34 drawings, 3 tables, 3 appendices, 20 references.

Keywords: software size, forecasting methods, multiple regression, Java, UML, Spring framework, JavaScript.

Зміст

Перелік скорочень та умовних позначень

API (Application Programming Interface) — набір визначень підпрограм, протоколів взаємодії та засобів для створення програмного забезпечення [5].

WWW (World Wide Web,) — найбільше всесвітнє багатомовне сховище інформації в електронному вигляді: десятки мільйонів пов'язаних між собою документів, що розташовані на комп'ютерах, розміщених на всій земній кулі [5].

HTTP (Hyper Text Transfer Protocol) — протокол передачі гіпер-текстових документів, що використовується в комп'ютерних мережах [5].

URI (Uniform Resource Identifier) — компактний рядок літер, який однозначно ідентифікує окремий абстрактний чи фізичний ресурс [5].

URL (Uniform Resource Locator — єдиний вказівник на ресурс) — стандартизована адреса певного ресурсу (такого як документ, чи зображення) в інтернеті (чи деінде) [5].

SOLID — це абревіатура складена з перших літер п'яти базових принципів об'єктно-орієнтованого програмування та дизайну запропонована Робертом Мартіном [5].

DI (Dependency Inversion) — один з п'яти SOLID-принципів об'єктно-орієнтованого проектування програм, суть якого полягає у розриві зв'язності між програмними модулями вищого та нижчого рівнів за допомогою спільних абстракцій [5].

REST (Representational State Transfer, «передача репрезентативного стану») — підхід до архітектури мережевих протоколів, які забезпечують доступ до інформаційних ресурсів [5].

MVC (Model-view-controller) — архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення [5].

MVVM (Model-View-ViewModel) — це шаблон проєктування, що застосовується під час проєктування архітектури застосунків (додатків) [5].

CRUD — (create read update delete) 4 базові функції управління даними «створення, зчитування, зміна і видалення» [5].

JVM (Java Virtual Machine;) — набір комп'ютерних програм та структур даних, що використовують модель віртуальної машини для виконання інших комп'ютерних програм чи скриптів [5].

JIT (Just-in-time compilation, також відома як dynamic translation) — компіляція «на льоту» — це технологія збільшення продуктивності програмних систем, що використовують байт-код, шляхом трансляції байт-коду в машинний код безпосередньо під час роботи програми [5].

JSON (JavaScript Object Notation) — це текстовий формат обміну даними між комп'ютерами [5].

CSV (comma-separated values ‘значення, розділені комою’, іноді character-separated values ‘значення, розділені символом’) — файловий формат, котрий є відмежовувальним форматом для представлення табличних даних, у якому поля відокремлюються символом коми та переходу на новий рядок [5].

HTML (Hypertext Markup Language) — стандартна мова розмітки для створення веб-сторінок і веб-додатків [5].

CSS (Cascading Style Sheets) — це спеціальна мова стилю сторінок, що використовується для опису їхнього зовнішнього вигляду. Самі ж сторінки написані мовами розмітки даних [5].

DOM (Document Object Model,) — специфікація прикладного програмного інтерфейсу для роботи зі структурованими документами (як правило, документами XML). Визначається ця специфікація консорціумом W3C [5].

Вступ

У сучасних реаліях, коли розробка програмних продуктів ведеться в умовах значно обмеженого часу та фінансових ресурсів, максимально точне оцінювання необхідних трудових витрат на написання коду програмного забезпечення є однією з основних проблем при управлінні проектами.

Розмір програмного забезпечення є одним з найвагоміших факторів в управлінні процесом розробки програмного забезпечення. Доведено, що розмір корелює з витратами, зусиллями та ресурсами, необхідними на його розробку.

Сьогодні, в період домінування гнучких методологій розробки ПЗ, завчасно відомий обсяг необхідних ресурсів для створення програмного продукту допомагає пришвидшити його випуск для кінцевого споживача, а отже швидше отримати перші відгуки, поліпшити продукт та отримати перевагу над конкурентами.

Актуальність дипломної роботи - на даний момент існує мало інструментів для визначення необхідних ресурсів для створення ПЗ. Один із них це «COCOMO II», який має три рівні точності розрахунку вартості ПЗ. Незалежно від обраного рівня, для функціонування, система потребує числове значення обсягу ПЗ, яке зазначається у тисячах рядків коду.

Нажаль не існує доступних інструментів для визначення обсягу ПЗ створеного за допомогою мови Java яке б прогнозувало обсяг продукту у кількості рядків коду ще на етапі проектування ІС.

Мета дипломної роботи – розробка інструментального засобу для прогнозування обсягу ПЗ створюваного мовою Java на основі UML метрик. Для досягнення мети булі поставлені наступні завдання:

- Проаналізувати існуючі технології та методології для прогнозування обсягу ПЗ;
- Провести огляд характеристик ПЗ які можна отримати на етапі проектування;

- Провести огляд математичного апарату для прогнозування параметру залежного від змінної кількості характеристик;
- Проаналізувати основні інструменти, технології, бібліотеки та фреймворки для розробки ІС;
- Розробити інструментальний засіб для прогнозування обсягу ПЗ створюваного мовою Java на основі UML метрик.

					ДР.122.4142.05.ПЗ.Р01		
Зм.	Аркуш	№ документа	Підпис	Дата			
						Літ.	Аркуш / Аркушів
Студент		Каратай М.В.			Розробка концепції системи		12
Керівник		Беркунський Є.Ю.				НУК ім. адм. Макарова	
Консульт.							
Зав. Каф.		Михелєв І.Л.					

Розділ 1. Розробка концепції інструментального засобу прогнозування обсягу програмного забезпечення, створюваного мовою Java на основі UML метрик.

1.1. Аналіз предметної області

Оскільки доступних аналогів для створюваної системи не існує, почнемо розгляд предметної області із метрик які ми можемо отримати після створення проектної UML-моделі.

UML (Unified Modeling Language) — уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, яка називається UML-моделлю [5].

UML може бути застосовано на всіх етапах життєвого циклу аналізу бізнес-систем і розробки прикладних програм. Різні види діаграм які підтримуються UML, і найбагатший набір можливостей представлення певних аспектів системи робить UML універсальним засобом опису як програмних, так і ділових систем.

UML чудово зарекомендувала себе в багатьох успішних програмних проектах. Засоби автоматичної генерації кодів дозволяють перетворювати моделі мовою UML у вихідний код об’єктно-орієнтованих мов програмування, що ще більш прискорює процес розробки. UML використовується 14 видів діаграм:

Структурні діаграми:

- Класів
- Компонент
- Композитної/складеної структури

- Кооперації (UML2.0)
- Розгортування
- Об'єктів
- Пакетів

Діаграми поведінки:

- Діяльності
- Станів
- Прецедентів

Діаграми взаємодії:

- Кооперації (UML1.x)
- Комунікації (UML2.0)
- Огляду взаємодії (UML2.0)
- Послідовності
- Синхронізації (UML2.0)

Діаграма класів (class diagram) — статичне представлення структури моделі. Відображає статичні (декларативні) елементи, такі як: класи, типи даних, їх зміст та відношення. Діаграма класів може містити позначення для пакетів та може містити позначення для вкладених пакетів. Також, діаграма класів може містити позначення деяких елементів поведінки, однак їх динаміка розкривається в інших типах діаграм [5].

Діаграма класів служить для представлення статичної структури моделі системи в термінології класів об'єктно-орієнтованого програмування. На цій діаграмі показують класи, інтерфейси, об'єкти й кооперації, а також їхні відносини.

Користуючись діаграмою класів ми можемо отримати достатній обсяг метрик який буде характеризувати досліджуваний нами проект ПЗ мовою Java, а саме:

CBO (Coupling between objects) - це підрахунок кількості класів, які з'єднані з певним класом, тобто, коли методи одного класу викликають методи

або отримують доступ до змінних іншого. Ці виклики потрібно рахувати в обох напрямках, тому CBO класу A - це розмір набору класів, на який посилається клас A, і тих класів, які посилаються на клас A. Оскільки це набір - кожен клас рахується лише один раз, навіть якщо посилання працює в обох напрямках, тобто якщо клас B посилається на клас A, в свою чергу A має посилання на B, то клас B враховується лише один раз. З'єднання між об'єктами, характеризує кількість залежностей класу [6].

DIT (Depth Inheritance Tree): Глибина спадкового дерева, характеризує кількість "батьків" класу. Усі класи мають DIT принаймні 1 (кожен успадковує java.lang.Object). Чим глибше певний клас знаходиться в ієрархії, тим більше можливе повторне використання успадкованих методів [6].

Classes - загальна кількість класів у проекті.

Interfaces - загальна кількість інтерфейсів у проекті.

Enums - загальна кількість перечислень у проекті.

All classes - сума характеристик: Classes, Interfaces та Enums.

Number of fields - загальна кількість атрибутів. Є сумою статичних, публічних, приватних, захищених, та не змінних атрибутів.

Number of static fields - загальна кількість статичних атрибутів.

Number of public fields - загальна кількість публічних атрибутів.

Number of private fields - загальна кількість приватних атрибутів.

Number of protected fields - загальна кількість захищених атрибутів.

Number of final fields - загальна кількість не змінних атрибутів.

Number of methods - загальна кількість методів. Є сумою статичних, публічних, абстрактних, приватних, захищених та не змінних методів.

Number of static methods - загальна кількість статичних методів.

Number of public methods - загальна кількість публічних методів.

Number of private methods - загальна кількість приватних методів.

Number of protected methods - загальна кількість захищених методів.

Number of final methods - загальна кількість не змінних методів.

Quantity of returns - кількість повернень, характеризує кількість методів які в результаті виконання повертають певний результат своєї роботи.

Таким чином ми отримуємо заздалегідь відомі характеристики проекту для якого нам необхідно визначити кількість тисяч рядків коду(KLOC). Що також можна представити у вигляді математичного виразу виду:

$$Y = f(X_1, X_2 \ldots, X_n)$$

- Де ***Y*** це прогнозована кількість тисяч рядків коду;
- $X_1, X_2 \ldots, X_n$ це кількісні характеристики проекту отримані із його UML діаграми класів;
- $f(X_1, X_2 \ldots, X_n)$ це функція від характеристик проекту яка повертає прогнозовану кількість тисяч рядків коду.

Отриману у результаті характеристику *(KLOC)* можна використовувати для визначення необхідних ресурсів для створення ПЗ, використовуючи «COCOMO II». COnstructive COst MOdel (COCOMO - модель витрат розробки) - це алгоритмічна модель оцінки трудомісткості, собівартості і плану-графіка для проектів з розробки програмного забезпечення, розроблена Баррі Боем. Модель використовує просту формулу регресії з параметрами, визначеними з даних, зібраних по ряду проектів.

«COCOMO II» має три рівні точності розрахунку вартості ПЗ. Незалежно від обраного рівня, для функціонування, система потребує числове значення обсягу ПЗ, яке зазначається у тисячах рядків коду [5].

1.2. Аналіз існуючих методів прогнозування

Методи прогнозування - це сукупність способів і прийомів мислення, які дають змогу на основі ретроспективного аналізу тенденцій та закономірностей розвитку ендогенних (внутрішніх) та екзогенних (зовнішніх) даних об´єкта прогнозування зробити висновок про його розвиток у майбутньому за певних умов. У наш час, за оцінками фахівців, нараховується понад 200 різних методів прогнозування. З них на практиці використовується не більше 15-20. Методи прогнозування, як і прогнози, класифікуються за різними ознаками [4].

Однією з найбільш важливих класифікаційних ознак методів прогнозування є ступінь формалізації. За цією ознакою виділяють інтуїтивні та формалізовані методи прогнозування.

Інтуїтивні методи використовуються в тих випадках, коли неможливо врахувати вплив багатьох факторів через складність об´єкта прогнозування. У цьому випадку використовуються думки експертів щодо поведінки об´єкта прогнозування. Вони можуть бути індивідуальні та колективні.

До **групи формалізованих** відносяться методи екстраполяції та моделювання.

Методи прогнозування за ступенем формалізації представлені на рисунку 1.2.1.

Рисунок 1.2.1 – Методи прогнозування за ступенем формалізації

Прогнозна екстраполяція може здійснюватися з використанням методів *найменших квадратів, експонентного згладжування, ковзних середніх, адаптивного згладжування.* Екстраполяційні методи є одними з найпоширеніших серед усієї сукупності методів прогнозування [4].

Роздивимось деякі з методів прогнозування засновані на найменших квадратах, згладжуванні, експонентному згладжуванні і ковзному середньому:

1.2.1. Середні і ковзні середні

Найпростішою моделлю, заснованою на простому усереднюванні є

$$Y_{t+1}=1_t[Y_t+Y_{t-1}+...+Y_1]$$

і у відмінності від найпростішої "наївної" моделі, якій відповідає принцип "завтра буде як сьогодні", цій моделі відповідає принцип "завтра буде як було в середньому за останній час". Така модель, звичайно стійкіша до коливань, оскільки в ній згладжуються випадкові викиди щодо середнього. Не дивлячись на це, цей метод ідеологічно настільки ж примітивний як і "наївні" моделі і йому властиві майже ті ж самі недоліки [7].

У приведеній вище формулі передбачалося, що ряд осереднюється по достатньо тривалому інтервалу часу. Проте як правило, значення тимчасового ряду з недалекого минулого краще описують прогноз, ніж усі попередні значення цього ж ряду. Тоді можна використовувати для прогнозування ковзне середнє:

$$Y_{t+1}=1_{T+1}[Y_t+Y_{t-1}+...+Y_{t-T}]$$

Сенс його полягає в тому, що модель бачить тільки найближче минуле (на Т відліків за часом в глибину) і ґрунтуючись тільки на цих даних будує прогноз.
При прогнозуванні досить часто використовується метод експоненціальних середніх, який постійно адаптується до даних за рахунок нових значень. Формула, що описує цю модель записується як

$$Y_{t+1}=\alpha Y_t+(1-a)Y\hat{}_t,$$

де Y_{t+1}– прогноз на наступний період часу

Y_t– реальне значення у момент часу t

$Y\hat{}$– минулий прогноз на момент часу t

a – постійна згладжування ($0<= a <=1$)

У цьому методі є внутрішній параметр a, який визначає залежність прогнозу від усіх розглянутих даних, причому вплив даних на прогноз

експоненціально зменшується із "віком" даних. Залежність впливу даних на прогноз при різних коефіцієнтах **a** приведена на рисунку 1.2.2.

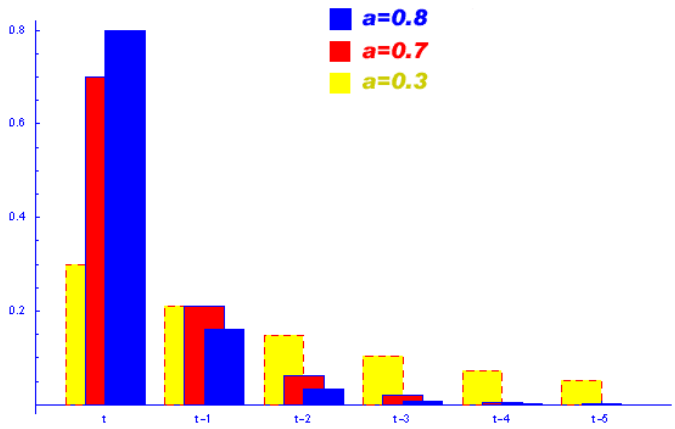


Рисунок 1.2.2 - Залежність впливу даних на прогноз при різних коефіцієнтах *a*

Видно, що при **a→1**, експоненціальна модель прагне до найпростішої "наївної" моделі. При **a→0**, прогнозована величина стає рівною попередньому прогнозу [7].

Якщо проводиться прогнозування з використанням моделі експоненціального згладжування, зазвичай на деякому тестовому наборі будуються прогнози при **a**=[0.01, 0.02 ..., 0.98, 0.99] і відстежується, при якому а точність прогнозування вища. Це значення **a** потім використовується при прогнозуванні надалі.

Хоча описані вище моделі ("наївні" алгоритми, методи, засновані на середніх, ковзних середніх і експоненціальному згладжуванні) використовуються при бізнес-прогнозуванні в не дуже складних ситуаціях, наприклад, при прогнозуванні продажу на спокійних і сталих західних ринках, не рекомендовано використовувати ці методи в завданнях прогнозування з причини явної примітивності і неадекватності моделей.

Разом з цим хотілося б відзначити, що описані алгоритми цілком успішно можна використовувати як супутні і допоміжні для перед обробки даних в завданнях прогнозування. Наприклад, для прогнозування продажу в більшості випадків необхідно проводити декомпозицію тимчасових рядів (тобто виділяти окремо тенденційну, сезонну і нерегулярну складові). Одним з методів виділення тенденційних складових є використання експоненціального згладжування [7]. На рисунках 1.2.3-4 представлені результати використання методу ковзного середнього:

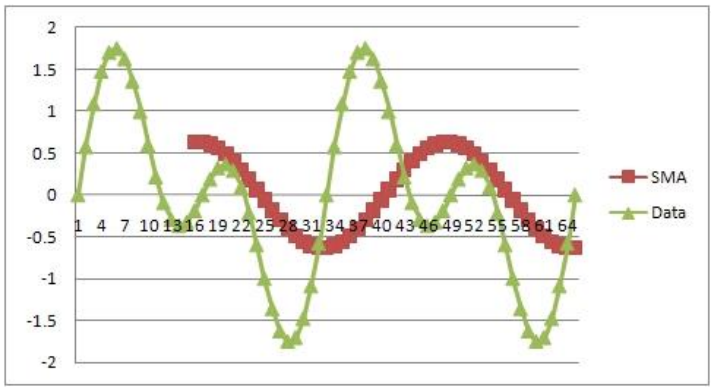


Рисунок 1.2.3 - Прогнозування ковзним середнім.

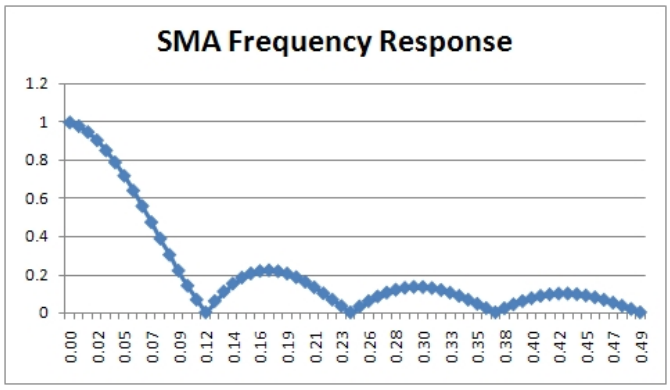


Рисунок 1.2.4 - Спад адекватності при ковзному середньому.

1.2.2. Методи Бокса-Дженкінса (ARIMA)

В середині 90-х років минулого століття був розроблений принципово новий і достатньо могутній клас алгоритмів для прогнозування тимчасових рядів. Велику частину роботи по дослідженню методології і перевірці моделей була проведена двома статистиками, Г.Е.П. Боксом і Г.М. Дженкінсом. З тих пір побудова подібних моделей і отримання на їх основі прогнозів іноді називатися методами Бокса-Дженкінса. В це сімейство входить декілька алгоритмів, найвідомішим і використовуваним з них є алгоритм ARIMA. Він вбудований практично в будь-який спеціалізований пакет для прогнозування. У класичному варіанті ARIMA не використовуються незалежні змінні. Моделі спираються тільки на інформацію, що міститься в передісторії прогнозованих рядів, що обмежує можливості алгоритму [7].

На відміну від розглянутих раніше методик прогнозування тимчасових рядів, в методології ARIMA не передбачається якої-небудь чіткої моделі для прогнозування даної тимчасової серії. Задається лише загальний клас моделей,

що описують часовий ряд і що дозволяють якось виражати поточне значення змінної через її попередні значення. Потім алгоритм, підстроюючи внутрішні параметри, сам вибирає найбільш відповідну модель прогнозування. Як вже наголошувалося вище, існує ціла ієрархія моделей Бокса-Дженкінса. Логічно її можна визначити так:

AR(p)+MA(q)→ARMA(p,q)→ARMA(p,q)(P,Q)→ARIMA(p,q,r)(P,Q,R)→...

1.2.2.1. AR(p) -авторегресивна модель порядку p

Модель має вигляд:

$$Y(t)=f_0+f_1\cdot Y(t-1)+f_2\cdot Y(t-2)+...+f_p\cdot Y(t-p)+E(t)$$

де **Y(t)** –залежна змінна у момент часу **t**. $f_0,f_1,f_2...,f_p$ - оцінювані параметри. **E(t)** - помилка від впливу змінних, які не враховуються в даній моделі. Завдання полягає в тому, щоб визначити $f_0,f_1,f_2...,f_p$. Їх можна оцінити різними способами. Найправильніше шукати їх через систему рівнянь Юла-Уолкера, для складання цієї системи буде потрібно розрахунок значень автокореляційної функції. Можна скористатися простішим способом - порахувати їх методом найменших квадратів [7].

1.2.2.2. MA(q) -модель з ковзаючим середнім порядку q

Модель має вигляд:

$$Y(t)=m+e(t)-w_1\cdot e(t-1)-w_2\cdot e(t-2)-...-w_p\cdot e(t-p)$$

Де **Y(t)** -залежна змінна у момент часу **t**. $w_0,w_1,w_2...,w_p$ - оцінювані параметри [7].

1.2.2.3. Авторегресійне ковзне середнє ARMA(p,q)

Під позначенням **ARMA(p,q)** розуміється модель, **p** авторегресійних складових, що містить **q**, ковзаючих середніх.

Точніше модель ARMA(p,q) включає моделі AR(p) і MA(q):

$$X_t = c + e_t + \sum_{i=1}^{q} \theta_i e_{t-i} + \sum_{i=1}^{p} \phi_i X_{t-i},$$

Зазвичай значення помилки e_t вважають незалежними однаково розподіленими випадковими величинами, узятими з нормального розподілу з нульовим середнім: $e_t \sim N(0, \sigma^2)$, де σ^2 — дисперсія. Припущення можна ослабити, але це може привести до зміни властивостей моделі. Наприклад, якщо не припускати незалежності і однакового розподілу помилок, поведінка моделі суттєво змінюється [7].

1.2.2.4. ARIMA (p,d,q)

У завданні аналізу тимчасового ряду з складною структурою часто використовуються моделі класу ARIMA(p,d,q) (авторегресійне інтегрування ковзаючого середнього - Autoregressive Integrated Moving Average) порядку (p,d,q), які моделюють різні ситуації, що зустрічаються при аналізі стаціонарних і нестаціонарних рядів. Залежно від аналізованого ряду модель ARIMA (p,d,q) може трансформуватися до авторегресійної моделі AR(p), моделі ковзного середнього MA(q) або змішаній моделі ARMA (p,q). При переході від нестаціонарного ряду до стаціонарного значення параметра **d**, що визначає порядок різниці, приймається рівним 0 або 1, тобто цей параметр має тільки цілочисельні значення. Зазвичай обмежуються вибором між d = 0 і d = 1. Проте з поля зору дослідників випадає ситуація, коли параметр **d** може приймати дробові значення [7].

1.2.3. Регресійні методи прогнозування

Разом з описаними вище методами, заснованими на експоненціальному згладжуванні, вже достатньо довгий час для прогнозування використовуються регресійні алгоритми. Коротко суть алгоритмів такого класу можна описати так. Існує прогнозована змінна **Y** (залежна змінна) і відібраний заздалегідь комплект змінних, від яких вона залежить, - $\mathbf{X_1, X_2 ..., X_n}$ (незалежні змінні) [7].

Природа незалежних змінних може бути різною. Наприклад, якщо припустити, що **Y** - рівень попиту на деякий продукт в наступному місяці, то

незалежними змінними можуть бути рівень попиту на цей же продукт в минулий і позаминулий місяці, витрати на рекламу, рівень платоспроможності населення, економічна обстановка, діяльність конкурентів і багато що інше. Головне - уміти формалізувати всі зовнішні чинники, від яких може залежати рівень попиту в числовій формі. Модель множинної регресії в загальному випадку описується виразом:

$$Y = f(x_1, x_2, ..., x_n)$$

У простішому варіанті лінійної регресійної моделі залежність залежної змінної від незалежних має вигляд:

$$\boldsymbol{Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_n X_n + \varepsilon}$$

Тут $\boldsymbol{\beta_1, \beta_2, ..., \beta_n}$ — підібрані коефіцієнти регресії

$\boldsymbol{\varepsilon}$ — компонента помилки.

В економічному аналізі використовують рівняння більш складного криволінійного типу, котрі краще описують багатофакторні економічні взаємозв'язки. Одним із таких рівнянь є степенева функція:

$$\boldsymbol{Y = \beta_0 \times X_1^{\beta_1} \times X_2^{\beta_2} \times ... \times X_n^{\beta_n} + \varepsilon}$$

Передбачається, що всі помилки незалежні і нормально розподілені. Для побудови регресійних моделей необхідно мати базу даних спостережень приблизно наступного вигляду(Таблиця 1.2.1):

	Змінні				
	Незалежні				**Залежна**
№	X_1	X_2	**...**	X_N	Y
1	x_{11}	x_{12}	...	x_{1N}	Y_1
2	x_{21}	x_{22}	...	x_{2N}	Y_2
...	...	...	...	...	...
m	x_{M1}	x_{M2}	...	x_{MN}	Y_m

Таблиця 1.2.1 – База спостережень

За допомогою таблиці значень минулих спостережень можна підібрати (наприклад, методом найменших квадратів) коефіцієнти регресії, побудувавши тим самим модель. При роботі з регресією треба дотримуватися певної обережності і обов'язково перевірити на адекватність знайдені моделі [7].

Аналіз адекватності моделі є важливим етапом моделювання. Для перевірки адекватності моделей множинної регресії, також як і парної лінійної регресії використовують коефіцієнт детермінації і його модифікації, що відображають особливості множинної моделі, а також процедури перевірки статистичних гіпотез і побудови довірчих інтервалів для оцінок параметрів і прогнозів залежною змінною.

Виходячи з опису усіх вище зазначених методів можна зробити висновок що регресійний метод прогнозування найбільше підходить для досліджуваного випадку, адже при знаходженні невідомої змінної він опирається на відібраний заздалегідь комплект змінних, від яких вона залежить. В той час як інші моделі у більшості випадків користуються станом системи на певному проміжку часу. Окрім того необхідну для методу базу знань можна з легкістю сформувати на основі Open-Source проектів створених на мові Java, які можна знайти в публічних репозиторіях таких веб-сервісів як: GitHub, GitLab, BitBucket тощо.

1.3. Формування вимог до проєкту

Необхідно створити інструмент (Автоматизовану систему), який буде надавати змогу користувачам на основі введених кількісних характеристик проекту, отриманих з UML діаграми класів, які були описані, отримувати прогнозовану кількість рядків коду.

Для розрахунку KLOC система повинна використовувати метод прогнозування множинної регресії. Користувачеві повинен надаватися вибір яку саме модель множинної регресії необхідно використовувати при розрахунку – *звичайної лінійної* або *складної степеневої.*

Система повинна надавати змогу створювати необхідні для розрахунку регресії набори проєктів - «бази знань», формувати нові бази знань необхідні для користувача та використовувати і редагувати вже існуючі. Самі бази знань будуть складатися з характеристик проектів для яких вже відома кількість

KLOC. Проекти які будуть додаватися до баз знань мають бути створені за допомогою мови програмування Java.

Під час створення аналізу система повинна надавати змогу вибору як «бази знань» на основі якої буде створений розрахунок, так і які саме характеристики проектів будуть брати участь у процесі прогнозування KLOC.

АС повинна мати змогу представити розгорнутий результат аналізу на основі якого був отримані прогнозований результат, як за допомогою графічного інтерфейсу так і за допомогою окремих файлів CSV(*comma-separated values*) формату. Результати повинні бути представлені у вигляді таблиці, яка буде описувати кожен крок розрахунку алгоритму, міститиме вхідні данні та результат розрахунків. *Розрахунок обов'язково необхідно перевірити на адекватність*.

Окрім того система повинна:

- Повертати список кількісних характеристик які можуть брати участь у аналізі;
- Надавати змогу виконувати CRUD операції над сутностями проектів;
- Надавати змогу виконувати CRUD операції над наборами проектів які використовуються для аналізу;
- Надавати зручну форму для введення усіх зазначених характеристик проекту;
- Окрім відображення результатів аналізу у форматі таблиці, надавати графік регресії, якій буде відображати отриману модель та довірчий інтервал для неї;
- Надавати картку з результатами прогнозування KLOC, яка буде описувати кількість рядків коду що були отримані у результаті, та для яких характеристик. Для кожної характеристики повинен бути наданий її коефіцієнт регресії. Також необхідно надавати інформацію що до середньої похибки аналізу.

- Надавати інтерфейс для завантаження CSV файлу з результатами аналізу.
- Перегляд наборів повинен бути реалізований як таблиця та стовпчикова діаграма з характеристиками усіх проектів які належать до цих наборів;

1.4 Вибір технологій та середовища розробки

Дана веб-орієнтована система є безкоштовною, отже всі технології та середовище розробки повинні бути вільними для користування, тобто не потребувати фінансових витрат. Для цього було вирішено використовувати мову програмування Java, адже вона не потребує придбання ліцензій і надає потужний механізм для розробки веб-додатків.

Java - об'єктно-орієнтована мова програмування, випущена 1995 року компанією «Sun Microsystems» як основний компонент платформи Java. З 2009 року мовою займається компанія «Oracle», яка того року придбала “Sun Microsystems”. В офіційній реалізації Java-програми компілюються у байт-код, який при виконанні інтерпретується віртуальною машиною для конкретної платформи [5].

Головним мотивом створення Java була потреба в мові програмування, яка б не залежала від платформи (тобто від архітектури) і яку можна було б використовувати для створення програмного забезпечення, що вбудовується в різноманітні побутові електронні прилади, такі як мобільні засоби зв'язку, пристрої дистанційного керування тощо.

Період становлення Java збігся у часі з розквітом міжнародної інформаційної служби World Wide Web. Ця обставина відіграла вирішальну роль у майбутньому Java, оскільки Web теж вимагала платформо-незалежних програм. Як наслідок, були зміщені акценти в розробці Sun з побутової електроніки на програмування для Інтернету.

Програми написані на мові Java можуть працювати на будь-якій підтримуваній апаратній чи системній платформі без змін у початковому коді та перекомпіляції.

Стандартні бібліотеки забезпечують загальний спосіб доступу до таких платформо залежних особливостей, як обробка графіки, багато потоковість та роботу з мережами. У деяких версіях задля збільшення продуктивності JVM байт-код можна компілювати у машинний код до або під час виконання програми.

Основна перевага використання байт-коду — це портативність. Тим не менш, додаткові витрати на інтерпретацію означають, що інтерпретовані програми будуть майже завжди працювати повільніше, ніж скомпільовані у машинний код. Проте, цей розрив суттєво скоротився після введення декількох методів оптимізації у сучасних реалізаціях JVM.

Одним із таких методів є just-in-time компіляція (JIT, що перетворює байт-код Java у машинний під час першого запуску програми, а потім кешує його). Як результаті, така програма запускається і виконується швидше, ніж простий інтерпретований код, але ціною додаткових витрат на компіляцію під час виконання [1].

Складніші віртуальні машини також використовують динамічну перекомпіляцію, яка полягає в тому, що віртуальна машина аналізує поведінку запущеної програми й вибірково перекомпілює та оптимізує певні її частини. З використанням динамічної перекомпіляції можна досягти більшого рівня оптимізації, ніж за статичної компіляції, оскільки динамічний компілятор може робити оптимізації на базі знань про оточення виконання та про завантажені класи. До того ж, він може виявляти так звані гарячі точки (англ. hot spots) — частини програми, найчастіше внутрішні цикли, які займають найбільше часу при виконанні. JIT-компіляція та динамічна перекомпіляція збільшує швидкість Java-програм, не втрачаючи при цьому портативності.

Швидкість офіційної віртуальної машини Java значно покращилася з моменту випуску ранніх версій, до того ж, деякі випробування показали, що продуктивність JIT-компіляторів у порівнянні зі звичайними компіляторами у машинний код майже однакова. Проте ефективність компіляторів не завжди свідчить про швидкість виконання скомпільованого коду, тільки ретельне тестування може виявити справжню ефективність у даній системі.

Варто зазначити що починаючи з 16 квітня 2019 року Oracle змінив політику ліцензування для Java SE, відтепер використовуючи цю версією у комерційних цілях необхідно оформити платну підписку від Oracle. Але є альтернативний шлях – використання безкоштовних версій Java (OpenJDK) з меншим терміном підтримки, або версій які підтримуються open-source community (AdoptOpenJDK, Liberica JDK тощо)

Apache Commons – це проект фонду Apache Software Foundation (далі ASF), що має на меті розробку і підтримку відкритого програмного забезпечення повторного використання на мові Java, тобто бібліотек Java. У більш вузькому сенсі Apache Commons - це «велика колекція маленьких Java-утиліт». Apache Commons містить набір бібліотек утиліт Java самого різного призначення, доступних за ліцензією Apache License, і використовуються в багатьох інших проектах з відкритим вихідним кодом. Утиліти проекту Apache Commons лежать в основі таких проектів як Apache Tomcat, Struts, Hibernate та ін.

Apache Commons Math - це найбільша бібліотека математичних функцій і утиліт для Java з відкритим вихідним кодом. Складається з математичних функцій, структур, які представляють математичні поняття (таких як комплексні числа, поліноми, вектори і т. д.), а також алгоритмів, які ми можемо застосувати до цих структур (пошук коренів, оптимізація, підбір кривої, обчислення перетину геометричних фігур і т. д.) [10].

Spring Framework — це програмний каркас (фреймворк) з відкритим кодом та контейнери з підтримкою інверсії управління для платформи Java. Основні особливості Spring Framework можуть бути використані будь-яким додатком Java, але є розширення для створення веб-додатків на платформі Java EE. Незважаючи на це, Spring Framework не нав'язує якоїсь конкретної моделі програмування, Spring Framework став популярним в спільноті Java як альтернатива, або навіть доповнення моделі Enterprise JavaBean (EJB) [5].

Apache Tomcat - контейнер сервлетів з відкритим вихідним кодом, що розробляється Apache Software Foundation. Реалізує специфікацію сервлетів, специфікацію JavaServer Pages (JSP) і JavaServer Faces (JSF). Написаний на мові

Java. Tomcat дозволяє запускати веб-додатки і містить ряд програм для само-конфігурації. Tomcat використовується в якості самостійного веб-сервера, як сервер контенту в поєднанні з веб-сервером Apache HTTP Server [5].

IntelliJ IDEA – комерційне інтегроване середовище розробки для різних мов програмування (Java, Python, Scala, PHP та ін.) від компанії JetBrains. Система поставляється у вигляді урізаної по функціональності безкоштовної версії «Community Edition» і повнофункціональної комерційної версії «Ultimate Edition», для якої активні розробники відкритих проектів мають можливість отримати безкоштовну ліцензію. Сирцеві тексти Community-версії поширюються рамках ліцензії Apache 2.0. Бінарні збірки підготовлені для Linux, Mac OS X і Windows [5].

1.5. Вибір засобів розробки серверної частини

Розглянемо одні із найпопулярніших сучасних фреймворків для створення веб-додатків мовою Java:

Play — фреймворк з відкритим кодом, написаний на Scala і Java, використовує шаблон проектування Модель-Представлення-Контроллер (MVC). Націлений на підвищення продуктивності використовуючи домовленості перед конфігурацією, гаряче перевантаження коду і відображення помилок в браузері. Як повноцінний фреймворк, Play включає всі компоненти, необхідні для створення веб-додатків та REST-сервісів, такі як інтегрований сервер HTTP, обробка форм, потужний механізм маршрутизації, підтримка I18n, і більше. Play економить дорогоцінний час розвитку, безпосередньо підтримуючи повсякденні завдання, щоб ви могли негайно переглянути результати своєї роботи.

Micronaut - це сучасний, повноцінний фреймворк для створення мікро сервісів на базі JVM, призначений для побудови модульних програм, що легко піддаються тестуванню. Micronaut має на меті забезпечити всі інструменти, необхідні для створення повнофункціональних додатків, включаючи: впровадження залежностей та інверсію управління, автоматичну конфігурацію,

сервіс діскавері, маршрутизація HTTP, HTTP-клієнт з балансуванням навантаження на стороні клієнта.

Apache Struts — веб-фреймворк з відкритим кодом для розробки Java EE веб-застосунків. Використовує і розширює Java Servlet API, надаючи архітектуру MVC. Struts був одним із перших веб-фреймворків для Java і став одим із найпопулярніших і найвідоміших. Проте його архітектура мала ряд недоліків, а проект довгий час не розвивався, тому Struts 2 було створено на базі зовсім іншого фреймворка — Webwork. Команди Webwork і Struts об'єднали свої проекти у Struts 2, узявши реалізацію Webwork і відому у корпоративному середовищі назву Struts.

Spring Framework — це фреймворк з відкритим кодом та контейнер з підтримкою інверсії управління для платформи Java. Основні особливості Spring Framework можуть бути використані будь-яким додатком Java, але є розширення для створення веб-додатків на платформі Java EE. Незважаючи на це, Spring Framework не нав'язує якоїсь конкретної моделі програмування, Spring Framework став популярним в спільноті Java як альтернатива, або навіть доповнення моделі Enterprise JavaBean (EJB).

Spring Framework складається з кількох модулів, які надають широкий спектр послуг [11]:

- Контейнер Інверсії управління: Конфігурація компонентів додатків і управління життєвим циклом об'єктів Java, здійснюється головним чином через Інверсію управління;
- Аспектно-орієнтоване програмування: дозволяє реалізувати наскрізні процедури;
- Доступ до даних: робота з реляційною системою управління базами даних на платформі Java з використанням JDBC і об'єктно-реляційні відображення та інструментів з NoSQL баз даних;
- Управління транзакціями: об'єднує кілька API, управління транзакціями та координує операції для Java-об'єктів;

- Модель-Вигляд-Управління (Model-View-Controller): програмний каркас на основі HTTP сервлета, що забезпечує створення веб-додатків і веб-служб RESTful;
- Аутентифікація і авторизація: налаштовувані процеси безпеки, які підтримують цілий ряд стандартів, протоколів, інструментів і практик за допомогою підпроекту Spring Security (колишня система безпеки AcerI для Spring);
- Тестування: підтримка класів для написання юніт-тестів та інтеграційних тестів.

Для розробки був обраний Spring Framework, легкий у використанні, він надає великий спектр готових рішень та технологій що поліпшує та пришвидшує створення надійних RESTful веб-додатків. Його використання є безкоштовним, а активна спільнота користувачів завжди готова допомогти з можливими питаннями та проблемами.

1.6. Вибір бази даних

Існують реляційні бази даних і так звані NoSQL (Not Only SQL) бази даних. NoSQL - база даних, що забезпечує інший механізм зберігання та видобування даних, ніж звичний підхід таблиць-відношень в реляційних базах даних.

NoSQL бази даних все більше і більше використовуються в задачах із застосуванням big data та real-time веб-застосунках. NoSQL системи також називають "Not only SQL" для підкреслення того, що вони можуть підтримувати SQL-подібну структуру та мову запитів.

Деякі нереляційні бази даних:

MongoDB - це документо-орієнтована система керування базами даних (СКБД) з відкритим кодом, яка не потребує опису схеми таблиць. MongoDB займає нішу між швидкими і масштабованими системами, що оперують даними у форматі ключ/значення, і реляційними СКБД, функціональними і зручними у формуванні запитів [5].

MongoDB підтримує зберігання документів в JSON-подібному форматі, має досить гнучку мову для формування запитів, може створювати індекси для різних збережених атрибутів, ефективно забезпечує зберігання великих бінарних об'єктів, підтримує журналювання операцій зі зміни і додавання даних в БД, може працювати відповідно до парадигми Map/Reduce, підтримує реплікацію і побудову відмово стійких конфігурацій [5].

Apache Cassandra — розподілена система керування базами даних, що відноситься до класу noSQL-систем і розрахована на створення високо масштабованих і надійних сховищ величезних масивів даних, представлених у вигляді хешу.

Спочатку проект був розроблений в надрах Facebook і в 2009 році переданий під оруду фонду Apache Software Foundation. Промислові рішення на базі Cassandra розгорнуті для забезпечення сервісів таких компаній, як Cisco, IBM, Cloudkick, Reddit, Digg, Rackspace і Twitter.

Реляційна база даних — база даних, заснована на реляційній моделі даних. Слово "реляційний" походить від англ. relation. Для роботи з реляційними БД застосовують реляційні СКБД. Інакше кажучи, реляційна база даних — це база даних, яка сприймається користувачем як набір нормалізованих відношень різного ступеня.

Види реляційних баз даних:

MySQL – була розроблений компанією «ТсХ» для підвищення швидкодії обробки великих баз даних. Ця система керування базами даних з відкритим кодом була створена як альтернатива комерційним системам. Зараз MySQL — одна з найпоширеніших систем керування базами даних. Вона використовується, в першу чергу, для створення динамічних веб-сторінок, оскільки має чудову підтримку з боку різноманітних мов програмування [5].

PostgreSQL – об'єктно-реляційна система керування базами даних. Є альтернативою як комерційним СКБД (Oracle Database, Microsoft SQL Server, IBM DB2 та інші), так і СКБД з відкритим кодом (MySQL, Firebird, SQLite).

Порівняно з іншими проектами з відкритим кодом, такими як Apache, FreeBSD або MySQL, PostgreSQL не контролюється якоюсь однією компанією,

її розробка можлива завдяки співпраці багатьох людей та компаній, які хочуть використовувати цю систему керування базами даних та впроваджувати у неї найновіші досягнення.

Сервер PostgreSQL написаний мовою C. Звичайно розповсюджується у вигляді набору текстових файлів із початковий кодом. Для інсталяції необхідно відкомпілювати файли на своєму комп'ютері і скопіювати в деякий каталог. Весь процес детально описаний в документації.

Оскільки присутня чітко визначена схема структури даних, якій будуть відповідати всі дані, крім того відсутня необхідність в обробці великих масивів даних та великій кількості запитів – буде обрана реляційна база даних, а саме MySQL. Остання має можливість безкоштовного користування, може бути з легкістю встановлена та налаштована.

1.7. Вибір засобів розробки клієнтської частини

Розглянемо технології які можна використовувати для побудови клієнтської частини АС:

Angular (зазвичай так називають фреймворк Angular 2 або Angular 2+, тобто вищі версії) — написаний на TypeScript front-end фреймворк з відкритим кодом, який розробляється під керівництвом Angular Team у компанії Google, а також спільнотою приватних розробників та корпорацій. Angular — це AngularJS, який переосмислили та який був повністю переписаний тією ж командою розробників [5]. Серед його можливостей:

- Зв’язування;
- Відкритий код (розповсюджується за MIT ліцензією);
- Шаблони(прототипування);
- Компонентна архітектура додатка;
- Впровадження залежностей;
- Клієнтська маршрутизація.

Angular один з найбільших за можливостями фреймворків, він не тільки допомагає в розробці односторінкового додатку а й пропонує цілу

інфраструктуру з використанням багатьох інших нових додаткових технологій таких як TypeScript й ECMAScript9.

React — відкрита JavaScript бібліотека для створення інтерфейсів користувача, яка покликана вирішувати проблеми часткового оновлення вмісту веб-сторінки, з якими стикаються в розробці односторінкових застосунків. Розробляється Facebook, Instagram і спільнотою індивідуальних розробників [5].

React дозволяє розробникам створювати великі веб-застосунки, які використовують дані, котрі змінюються з часом, без перезавантаження сторінки. Його мета полягає в тому, щоб бути швидким, простим, масштабованим. React обробляє тільки користувацький інтерфейс у застосунках. Це відповідає видові у шаблоні модель-вид-контролер (MVC), і може бути використане у поєднанні з іншими JavaScript бібліотеками або в великих фреймворках MVC, таких як Angular. Серед його можливостей:

- Одностороння передача даних;
- Віртуальний DOM;
- Не лише рендеринг HTML в браузері, а і на стороні серверу;
- Відкритий код (розповсюджується за MIT ліцензією);
- Методи життєвого циклу;
- JSX компоненти.

Vue.js — JavaScript-бібліотека що використовує шаблон MVVM для створення інтерфейсів користувача на основі моделей даних, через реактивне зв'язування даних. Був створений Еваном Ю, вихідцем з Google Creative Labs для швидкої побудови прототипів складного інтерфейсу і уникнення написання повторюваного HTML. Зараз Vue.js еволюціонував, і дозволяє швидко писати не тільки прототипи, а й складні і надійні веб-застосунки для яких характерна масштабованість [5].

Одна із найвиразніших особливостей Vue — це ненав'язлива реактивна система. Моделі це просто плоскі JavaScript об'єкти. Це робить керування

станами дуже простим та інтуїтивним. Vue надає оптимізований ре-рендеринг з коробки без потреби робити що-небудь додатково. Кожен компонент слідкує за своїми реактивними залежностями під час рендерингу, тому система знає точно коли має відбуватись ре-рендеринг і які компоненти потрібно ре-рендерити. Серед його можливостей:

- Шаблони
- Віртуальній DOM
- Реактивність
- Переходи і контроль анімацій
- Роутинг
- Відкритий код (розповсюджується за MIT ліцензією);

Для створення клієнтської частини буде використовуватися Vue.js, мала за розмірами ця бібліотека має великий спектр можливостей та інструментів для швидкої побудови сучасного, зручного та надійного інтерфейсу користувача. Велика гнучкість, яка полегшує взаємодію з різними бібліотеками та відповідає різним стратегіям та оптимальна продуктивність, яка доступна завдяки мінімалізму інструменту. Насправді Vue.js навіть менший за 90 Кб.

					ДР.122.4142.05.ПЗ.Р02		
Зм.	Аркуш	№ документа	Підпис	Дата			
					Розробка проєктних рішень по системі	Літ.	Аркуш 36 / Аркушів
Студент		Каратай М.В.					
Керівник		Беркунський Є.Ю.				НУК ім. адм. Макарова	
Консульт.							
Зав. Каф.		Михелєв І.Л.					

Розділ 2. Розробка проєктних рішень по системі

2.1. Загальносистемні проєктні рішення

2.1.1. Організаційна структура

Призначення системи прогнозування обсягу програмного забезпечення полягає у наданні змоги користувачам за допомогою інформації яку вони отримали на етапі проєктування UML діаграми класів будувати моделі прогнозування обсягу програмного забезпечення. Моделі будуть створюватися на основі наборів даних які складаються з проєктів які були реалізовані за допомогою мови програмування Java і користувач матиме змогу обирати та маніпулювати цими наборами даних для отримання найбільш точного результату.

Для реалізації цієї системи необхідно, щоб вона складалася зі:

- сховища даних яке буде містити усі необхідні набори проєктів для створення моделей прогнозування;
- підсистеми побудови та розрахунку моделей прогнозування;
- засобів вводу/виводу даних із системи користувачами.

Окрім того необхідно надати можливість одночасної роботи визначеної кількості користувачів із системою в цілому.

Варіантами архітектури системи, що задовольняють поставленим вимогам, є *клієнт-серверна* та *сервіс-орієнтована* архітектури.

За трирівневою клієнт-серверною архітектурою передбачається наявність наступних компонентів програми: клієнтський застосунок, підключений до сервера застосунків, який в свою чергу підключений до серверу бази даних. Така архітектура передбачає ізольованість рівнів один від одного що дозволяє швидко і простими засобами переконфігурувати систему при виникненні збоїв або при плановому обслуговуванні на одному з рівнів. Крім того є масштабованою, надійною та має високий рівень безпеки.

Сервіс-орієнтована архітектура є модульним підходом до розробки систем, заснованим на використанні сервісів (служб) зі стандартизованими інтерфейсами. В її основі лежать принципи багатократного використання

функціональних елементів ІТ, ліквідації дублювання функціональності в ПЗ, уніфікації типових операційних процесів тощо. Компоненти такої системи можуть бути розподілені по різних вузлах мережі, і пропонуються як незалежні, слабо пов'язані, замінні сервіси-додатки. Інтерфейс компонентів такої системи надає інкапсуляцію деталей реалізації конкретного компонента (ОС, платформи, мови програмування, розробника тощо).

Клієнт-серверна архітектура є більш зручною з боку використання, адже для доступу до системи (сервер) не потрібно специфічного програмного забезпечення, достатньо використання вбудованих можливостей операційних систем – браузерів (клієнт). Тому обираємо цей підхід як підхід до вирішення завдання розробки системи прогнозування обсягу програмного забезпечення.

Окрім того серверна частина системи буде реалізована як RESTfull веб-сервіс.

REST - (Representational State Transfer - «передача репрезентативного стану») архітектурний стиль взаємодії компонентів розподіленого додатку в мережі за моделлю клієнт-сервер. Був розроблений в дисертації Роя Філдінга в 2000 році, як альтернатива SOAP, коли запит клієнта несе в собі вичерпну інформацій про бажану відповідь серверу і сервер не зобов'язаний зберігати сесію взаємодії з клієнтом [5].

Особливості архітектурного стилю:

- Кожна сутність повинна мати унікальний ідентифікатор - URI.
- Сутності повинні бути пов'язані між собою.
- Для читання і зміни даних повинні використовуватися стандартні методи HTTP протоколу.
- Повинна бути підтримка декількох типів ресурсів.
- Взаємодія має здійснюватися без стану.

Стандартні методи такі:

- GET - отримання даних без їх зміни. Це найбільш популярний і легкий метод. Він тільки повертає дані, а не змінює їх, тому на

клієнті вам не потрібно піклуватися про те, що ви можете пошкодити дані.

- POST - метод, що припускає вставку нових записів.
- PUT - метод, що припускає зміну існуючих записів.
- PATCH - метод, що припускає зміну ідентифікатора існуючих записів.
- DELETE - метод, що припускає видалення записів.

2.1.2. Функціональна структура

У процесі свого функціонування система виконує певні функції, які представлені у схемі функціональної структури на рисунку 2.1.

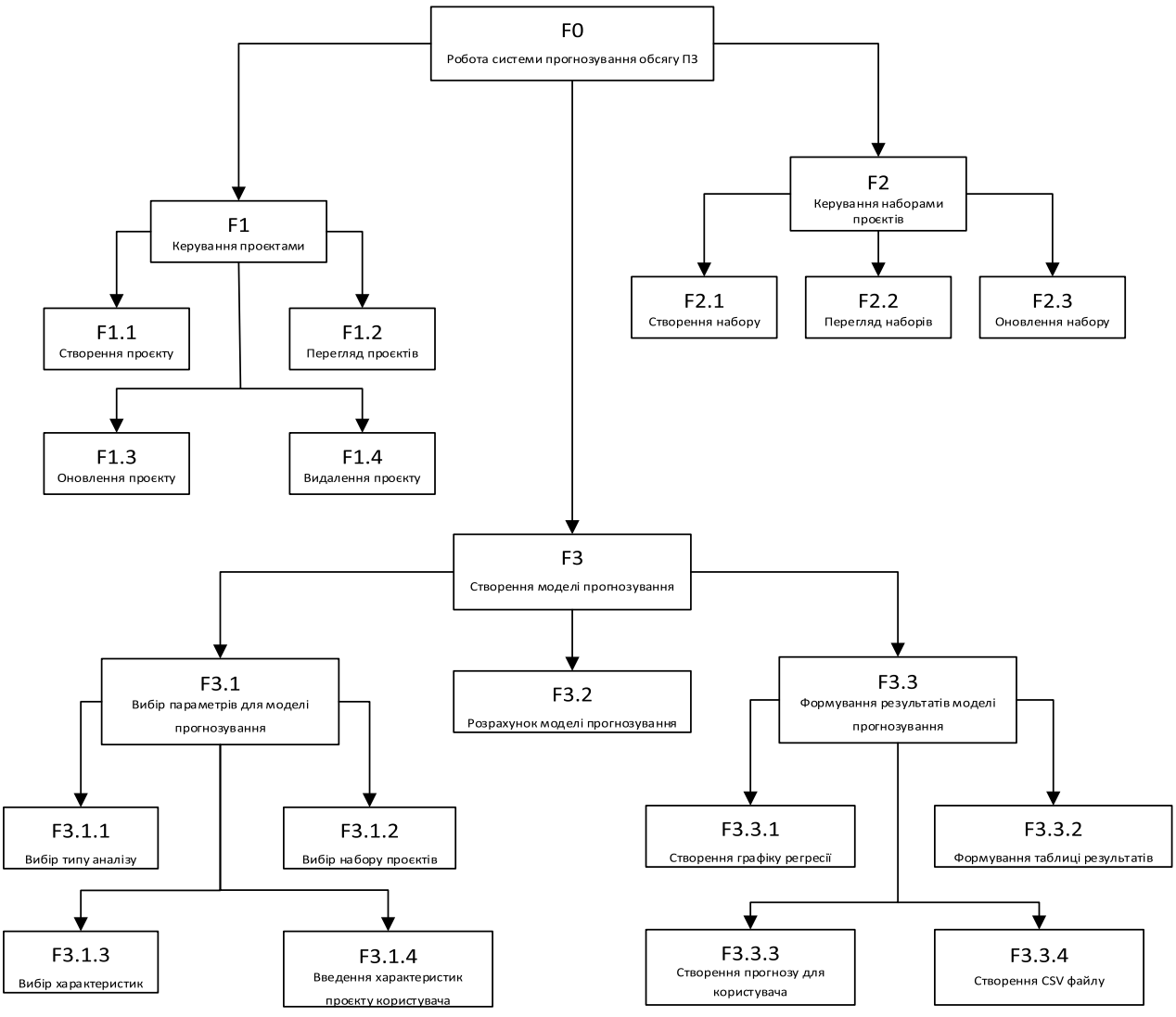


Рисунок 2.1 - Схема функціональної структури

Опис функцій що автоматизуються:

- F1.1) Надає користувачу можливість створювати нові проєкти у системі, кожен такий проєкт міститиме значення характеристик описаних у розділі 1 та власну назву;

  Процеси: реалізація інтерфейсу користувача для створення нових проєктів.

- F1.2) Надає користувачу змогу переглядати всі проєкти які наявні у системі, як у вигляді таблиці так і у вигляді графіку;

  Процеси: реалізація інтерфейсу користувача для перегляду проєктів, який матиме дві опції перегляду, у вигляді таблиці та у вигляді стовпчикового графіку.

- F1.3) Надає змогу користувачу оновлювати характеристики вже існуючих проєктів;

  Процеси: реалізація інтерфейсу користувача для оновлення характеристик проєктів.

- F1.4) Надає змогу користувачу видаляти проєкти із системи;

  Процеси: реалізація інтерфейсу користувача для видалення проєкту.

- F2.1) Надає змогу користувачу створювати та зберігати у системі набори для створення моделей прогнозування. Набор складається із змінної кількості проєктів та власної назви;

  Процеси: реалізувати інтерфейс користувача для створення наборів проєктів.

- F2.2) Надає користувачу змогу переглядати вже існуючі у системі набори проєктів;

  Процеси: реалізувати інтерфейс користувача для перегляду наборів проєктів.

- F2.3) Надає змогу користувачу редагувати вже існуючі набори проєктів у системі, шляхом включення/виключення проєктів у наборі;

Процеси: реалізувати користувацький інтерфейс для редагування наборів.

- F3.1.1) Надає користувачу можливість вибору типу аналізу для побудови моделі прогнозування, а саме лінійної або степеневої;
  Процеси: реалізувати інтерфейс користувача для вибору типу аналізу для створення моделі прогнозування.
- F3.1.2) Надає користувачу змогу обирати на основі якого набору проєктів буде створена модель прогнозування;
  Процеси: реалізувати інтерфейс користувача для вибору набору проєктів для створення моделі прогнозування.
- F3.1.3) Надає змогу користувачу обирати на основі яких характеристик проєкту буде створена моделі прогнозування;
  Процеси: реалізувати інтерфейс користувача для вибору характеристик проєкту.
- F3.1.4) Надає змогу користувачу ввести кількісні характеристики власного проєкту для якого буде створена модель прогнозування;
  Процеси: реалізувати інтерфейс для введення характеристик проєкту користувача.
- F3.2) Надає змогу користувачу ініціювати розрахунок моделі прогнозування для обраних ним параметрів;
  Процеси: реалізувати розрахунок моделі прогнозування на основі вхідних даних віл користувача, а саме типу моделі, набору проєктів та характеристик проєкту. Реалізувати інтерфейс користувача для ініціації розрахунку моделі на основі введених параметрів.
- F3.3.1) Надає змогу користувачу переглянути та проаналізувати графік регресії створений на основі отриманої моделі прогнозування;
  Процеси: реалізувати інтерфейс користувача який надаватиме змогу перегляду результатів моделі прогнозування у вигляді лінійного графіку, який описуватиме регресію, а саме реальні кількісні

характеристики обсягу проєктів(KLOC) та прогнозовані, а також довірчий інтервал для них.

- F3.3.2) Надає змогу користувачу переглянути розгорнутий результат аналізу для моделі прогнозування;

  Процеси: реалізувати користувацький інтерфейс який надаватиме змогу переглядати розгорнутий результат аналізу моделі прогнозування, що включає у себе кожен крок розрахунків алгоритму.
- F3.3.3) Надає змогу перегляду результатів прогнозування для проєкту користувача;

  Процеси: реалізувати інтерфейс який представлятиме результати прогнозування для проєкту користувача та буде включати у себе інформацію про кількість рядків коду для проєкту, які характеристики були використані та з якою похибкою був отриманий результат.
- F3.3.4) Надає змогу користувачу завантажити CSV файл який містить розгорнутий результат аналізу для моделі прогнозування;

  Процеси: реалізувати функціонал для створення CSV файлу з результатами аналізу та користувацький інтерфейс для його завантаження.

2.1.3. Опис постановки задачі

На підставі аналізу вимог і обмежень, а також попередніх концепцій реалізації системи, сформулюємо опис постановки задачі. Необхідно розробити систему управління та обробки даних, що буде задовольняти перерахованим вище функціональним вимогам. Необхідно також забезпечити надійність збереження і безпеку доступу до інформації системи. Система повинна мати змогу обслуговувати одночасну роботу багатьох користувачів.

Програмне забезпечення повинне бути розроблене на основі трирівневої клієнт-серверної архітектури. Клієнтський рівень повинен бути побудований за допомогою фреймворку Vue.js, серверний рівень за допомогою мови

програмування Java, додаткових бібліотек Apache Commons та Spring Framework, як сервер баз даних був обраний MySQL. Розробка системи повинна проходити за допомогою засобів IntelliJ IDEA.

2.1.4. Загальній опис системи

Система реалізує інструмент для створення і дослідження моделей прогнозування обсягу програмного забезпечення створеного за допомогою мови програмування Java. Система створює моделі прогнозування на основі UML метрик отриманих на етапі проектування ПЗ.

Для побудови моделей прогнозування використовуються «бази знань» які створюються на основі характеристик open-source проєктів створених мовою програмування Java. Основними функціональними характеристиками системи є:

- Формування унікальних наборів проєктів для розрахунку моделі прогнозування;
- Створення та редагування характеристик проєктів які входять у набори;
- Можливість перегляду наборів проєктів у вигляді графіків та таблиць;
- Можливість налаштування параметрів для створення моделі прогнозування;
- Можливість перегляду результатів прогнозування у вигляді графіків та таблиць;
- Збереження результатів прогнозування у вигляді CSV файлів.

Система має зручний, динамічній та інтуїтивно зрозумілий інтерфейс користувача, що полегшує та пришвидшує її використання.

2.2. Рішення по математичному забезпеченню

2.2.1 Опис алгоритмів вирішення завдань

Нижче приведений опис обраного у першому розділі для побудови моделей прогнозування алгоритму множинної регресії.

2.2.1.1. Модель множинної лінійної регресії

Модель множинної регресії призначена для перевірки і вивчення зв'язку (пояснення поведінки) між однією залежною змінною (ендогенною) і декількома незалежними (екзогенними) змінними. Передбачається, що такий зв'язок теоретично може бути описаний (специфікований) лінійною залежністю (функцією) виду:

$$Y = \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k + U,$$

де ***Y*** - залежна (яка пояснюється, ендогенна) змінна - регресант, ***U*** - випадкова складова моделі, x_j - незалежні (пояснюючі, екзогенні) змінні – регресорів, β_j – коефіцієнти регресії [9].

Для статистичної оцінки параметрів регресійної моделі необхідний набір даних спостережень залежної і незалежних змінних. Ці дані можуть бути просторовими - в цьому випадку спостерігаються різні об'єкти, але спостереження відносяться до одного і того ж моменту або періоду часу. Дані можуть бути представлені у вигляді часових рядів спостережень залежної (що пояснюється) і незалежних (що пояснюють) змінних - в цьому випадку спостерігається поведінка одного об'єкта в різні (але однакові для всіх змінних) моменти часу. Для кожного з цих спостережень згідно лінійної моделі можна записати рівняння:

$$\left.\begin{aligned}
y_1 &= \beta_1 x_{11} + \beta_2 x_{12} + \ldots + \beta_k x_{1k} + u_1,\\
y_2 &= \beta_1 x_{21} + \beta_2 x_{22} + \ldots + \beta_k x_{2k} + u_2,\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
y_i &= \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_k x_{ik} + u_i,\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
y_n &= \beta_1 x_{n1} + \beta_2 x_{n2} + \ldots + \beta_k x_{nk} + u_n
\end{aligned}\right\}$$

Дану систему рівнянь зручно записати у векторно-матричному вигляді. Для цього введемо такі позначення:

вектор - стовпець спостережень залежної змінної (регресанду)

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ \cdot \\ y_n \end{bmatrix}$$

матриця спостережень незалежних змінних (регресорів)

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdot & \cdot & \cdot & x_{1k} \\ x_{21} & x_{22} & \cdot & \cdot & \cdot & x_{2k} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ x_{i1} & x_{i2} & \cdot & \cdot & \cdot & x_{ik} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ x_{n1} & x_{n2} & \cdot & \cdot & \cdot & x_{nk} \end{bmatrix}$$

вектор - стовпець коефіцієнтів (параметрів)

$$\beta = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \cdot \\ \cdot \\ \cdot \\ \beta_k \end{bmatrix}$$

вектор - стовпець реалізацій випадкової складової в окремих спостереженнях

$$u = \begin{bmatrix} u_1 \\ u_2 \\ \cdot \\ \cdot \\ \cdot \\ u_n \end{bmatrix}$$

Таким чином, тут **y** - вектор - стовпець розмірності **n**, елементи якого - спостереження залежної змінної (регресанду); **X** - матриця розмірності (**n×k**), її стовпці містять спостереження незалежних змінних (регресорів), **i**-стовпець матриці **X** (крім першого) містить n спостережень **i**-регресору, перший стовпець цієї матриці складається з одиниць; **β** - вектор - стовпець розмірності **k**, його елементи - коефіцієнти рівняння регресії; **u** - вектор - стовпець розмірності **n**, його елементи - реалізації випадкової складової моделі в кожному спостереженні. З використанням введених позначень, систему з **n** рівнянь можна записати наступним чином:

$$y = X\beta + u$$

Отримана матрична форма запису рівнянь лінійної регресії істотно простіша, а її використання дозволяє отримати і записати багато результатів для оцінок моделі в компактному вигляді [9].

Емпірична лінійна функція регресії

Систематична частина рівняння регресії, в якій замість теоретичних значень параметрів стоять деякі їх оцінки, називається емпіричною лінійною функцією регресії, і записується у наступному вигляді:

$$\hat{y}_i = b_1 x_{i1} + b_2 x_{i2} + \ldots + b_k x_{ik}$$

де коефіцієнти $\mathbf{b_j}$, **(j = 1,2, ..., k)** - оцінки теоретичних значень параметрів моделі. Змінна $\widehat{\boldsymbol{y}}_{\boldsymbol{i}}$ є *точковим прогнозом* залежною змінною **Y** при деяких заданих значеннях незалежних змінних (регресорів). Емпірична лінійна регресійна функція визначає регресійну гіперплоскість в лінійному **k**-вимірному просторі [9].

Помилки (залишки) регресійного рівняння

Різниця:

$$\boldsymbol{e_i} = \boldsymbol{y_i} - \widehat{\boldsymbol{y}}_{\boldsymbol{i}}$$

називається помилкою (залишком) рівняння в **i**-му спостереженні.

Помилки є обчислюваними величинами і тому їх можна вважати відомими при кожному фіксованому наборі спостережень і заданих значеннях параметрів моделі. Очевидно, величини помилок (при фіксованих спостереженнях) залежать тільки від вибору коефіцієнтів **β**, які є оцінками справжніх (теоретичних) значень коефіцієнтів. Чим точніше оцінки, тим, взагалі кажучи, менше залишки моделі (за умови її правильної специфікації), тим краще дана модель відповідає спостерігаємій вибірці, і тим краще модель "налаштована" саме на цю вибірку. Оцінки параметрів класичної багатовимірної лінійної моделі найчастіше будуються з використанням звичайного (класичного) ***багатовимірного методу найменших квадратів*** [9].

2.2.1.2. Проблема оцінювання параметрів моделі. Багатовимірний метод найменших квадратів

При оцінці параметрів моделі за методом найменших квадратів мірою якості (критерієм якості) підгонки емпіричної регресійної функції до спостерігаємої виборці служить сума квадратів помилок (залишків). У застосуванні до класичної багатовимірної лінійної регресії цей метод називається *звичайним (класичним) або одно-кроковим багатовимірним методом найменших квадратів* [9].

Критерій найменших квадратів

Оскільки помилку лінійного рівняння регресії в **i**-му спостереженні можна представити наступним чином:

$$e_i = y_i - \hat{y}_i = y_i - b_1 x_{i1} - b_2 x_{i2} - \ldots - b_k x_{ik}$$

і відповідно, квадрат помилки дорівнює:

$$e_i^2 = (y_i - b_1 x_{i1} - b_2 x_{i2} - \ldots - b_k x_{ik})^2$$

Використовуючи вираз квадрата помилки, представимо критерій (цільову функцію) найменших квадратів в багатовимірному випадку:

$$S(b) = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_i - b_1 x_{i1} - b_2 x_{i2} - \ldots - b_k x_{ik})^2$$

Або, використовуючи векторно-матричну форму:

$$S(b) = e^T e = (y - Xb)^T (y - Xb)$$

де вектор $\mathbf{e} = (e_1, e_2, \ldots, e_n)^T$, вектор $\hat{\boldsymbol{y}} = (\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n)^T$, $\hat{\boldsymbol{y}} = Xb$.

Виведення системи нормальних рівнянь

Для виведення системи нормальних рівнянь, перетворимо вираз критерію найменших квадратів з використанням правил дій з векторами і матрицями, отримаємо:

$$S(b) = (y - Xb)^T (y - Xb) = (y^T - b^T X^T)(y - Xb) =$$

$$= y^T y - y^T Xb - b^T X^T y + b^T X^T Xb =$$

$$= y^T y - 2b^T X^T y + b^T X^T Xb.$$

При виведенні виразу використовувалась рівність:

$$y^T Xb = b^T X^T y,$$

яка має місце, оскільки величини в правій і лівій частині його - скаляри (**S(b)** - скалярна функція).

Функція **S(b)** являє собою квадратичну форму щодо вектору оцінок **b**. Її мінімум за параметрами **b** існує і визначається єдиним чином за умови рівності нулю часткових похідних функції **S(b)** по змінним $\mathbf{b_i, i = 1,2, ..., k}$.

Використовуючи правила диференціювання скалярної функції по векторному аргументу, отримаємо вираз для похідної критерію найменших квадратів:

$$\frac{\partial S(b)}{\partial b} = \frac{\partial (y^T y)}{\partial b} - 2\frac{\partial (b^T X^T y)}{\partial b} + \frac{\partial (b^T X^T Xb)}{\partial b} = \\ = -2X^T y + 2X^T Xb$$

Тут, $\frac{\partial S(b)}{\partial b} = \left(\frac{\partial S(b_1)}{\partial b_1}, \frac{\partial S(b_2)}{\partial b_2}, \ldots, \frac{\partial S(b_k)}{\partial b_k} \right)^T$ - вектор-стовпець розмірності **k** приватних похідних цільової функції [9].

Мінімум цільової функції досягається в точці **b**, що задовольняє наступній системі лінійних рівнянь, записаної в векторно-матричній формі:

$$\frac{\partial S(b)}{\partial b} = 2X^T Xb - 2X^T y = 0,$$

або у наступному вигляді:

$$X^T Xb = X^T y$$

Остання називається системою нормальних рівнянь і містить **k** лінійних рівнянь щодо **k** невідомих $\mathbf{b_i, i = 1,2, ..., k}$. У розгорнутому вигляді цю систему можна представити так:

$$\begin{bmatrix} 1 & 1 & \dots & 1 \\ x_{12} & x_{22} & \dots & x_{n2} \\ \dots & \dots & \dots & \dots \\ x_{1k} & x_{2k} & \dots & x_{nk} \end{bmatrix} \begin{bmatrix} 1 & x_{12} & \dots & x_{1k} \\ 1 & x_{22} & \dots & x_{2k} \\ \dots & \dots & \dots & \dots \\ 1 & x_{n2} & \dots & x_{nk} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_k \end{bmatrix} =$$

$$= \begin{bmatrix} 1 & 1 & \dots & 1 \\ x_{12} & x_{22} & \dots & x_{n2} \\ \dots & \dots & \dots & \dots \\ x_{1k} & x_{2k} & \dots & x_{nk} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{bmatrix}$$

Перемножуючи матриці і вектори в вираженні, отримаємо розгорнуту форму запису системи нормальних рівнянь:

$$\left.\begin{aligned} & b_1 n + b_2 \sum x_{i2} + \dots + b_k \sum x_{ik} = \sum y_i \\ & b_1 \sum x_{i2} + b_2 \sum x_{i2}^2 + \dots + b_k \sum x_{i2} x_{ik} = \sum x_{i2} y_i \\ & \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ & b_1 \sum x_{ik} + b_2 \sum x_{ik} x_{i2} + \dots + b_k \sum x_{ik}^2 = \sum x_{ik} y_i \end{aligned}\right\}$$

де для простоти запису опущені межі підсумовування за індексом **i = 1,2, ..., n**.

Вирішення системи нормальних рівнянь в явному вигляді (тобто у вигляді розрахункової формули) можна отримати тільки в векторно-матричної формі. Розглянемо векторно-матричну запис нормальних рівнянь

$$X^T X b = X^T y$$

Матриця спостережень регресорів **X** має повний ранг і квадратна матриця (**$X^T X$**) розмірності (**k×k**) також має повний ранг. Отже, існує зворотна матриця**$(X^T X)^{-1}$**. Помножимо зліва обидві частини рівняння на цю матрицю, отримаємо:

$$(X^T X)^{-1} (X^T X) b = (X^T X)^{-1} X^T y$$

Далі, враховуючи, що **$(X^T X)^{-1}(X^T X) = I_k$**, де **$I_k$** - одинична матриця розмірності **k**, отримуємо вираз для оцінок коефіцієнтів у вигляді:

$$b = (X^T X)^{-1} X^T y$$

Саме ця формула ***визначає оцінку*** за методом найменших квадратів коефіцієнтів багатовимірної лінійної регресії. Оцінену за допомогою методу найменших квадратів емпіричну лінійну регресійну функцію можна записати у вигляді:

$$\hat{y}_i = b^T x_i = b_1 x_{i1} + b_2 x_{i2} + \ldots + b_k x_{ik}$$

де вектор **b** - оптимальна за методом найменших квадратів оцінка вектору коефіцієнтів регресії що визначається за останнім виразом, $\mathbf{x_i = (x_{i1}, x_{i2},\ldots,x_{ik})^T}$ - вектор-стовпець розмірності **k** [9].

Інтерпретація оцінок

Отриманим регресійний коефіцієнтами можна дати наступну інтерпретацію. Оцінений (емпіричний) регресійний коефіцієнт $\mathbf{b_j}$ **(j=1,2,…,k)** є приватною похідної емпіричної регресійної функції по **j**-му регресору (незалежній змінній). Він показує, на скільки зміниться оцінене значення при зміні **j**-ого регресору на одиницю *при фіксованих значеннях інших регресорів* [9].

Властивості помилок (залишків) моделі

Вираз для емпіричної регресійної функції можна записати у вигляді:

$$\hat{y} = Xb$$

Вектор залишків (помилок) моделі:

$$e = y - \hat{y} = y - Xb$$

де **b**-оптимальна в сенсі критерію найменших квадратів оцінка параметрів моделі, є оцінкою вектору випадкових складових **u** регресії (нагадаємо, що під векторами розуміємо вектори-стовпці). З умови оптимальності оцінок випливає, що

$$X^T y - X^T Xb = X^T (y - Xb) = X^T e = 0$$

Таким чином, була представлена перша властивість помилок, що помилки моделі в кожному спостереженні при оптимальних МНК оцінках параметрів, ортогональні спостереженнями незалежних змінних.

Далі, згадуючи, що перший стовпець матриці спостережень **X** складається з одиниць і, отже, перший рядок матриці $\mathbf{X^T}$ також складається з одиниць, отримуємо другу властивість помилок:

$$\sum_{i=1}^{n} e_i = 0$$

тобто сума помилок регресійній моделі за умови, що її коефіцієнти оцінені за методом найменших квадратів, дорівнює нулю.

Дані властивості помилок багатовимірної регресії є узагальненням відповідних властивостей помилок парної лінійної регресії [9].

2.2.1.2. Перевірка адекватності моделей множинної лінійної регресії

Аналіз адекватності моделі є важливим етапом моделювання. Для перевірки адекватності моделей множинної регресії, також як і парної лінійної регресії використовують коефіцієнт детермінації і його модифікації, що відображають особливості множинної моделі, а також процедури перевірки статистичних гіпотез і побудови довірчих інтервалів для оцінок параметрів і прогнозів залежною змінною.

Коефіцієнт детермінації

Важливим показником, що характеризує якість емпіричної регресійної функції (її відповідності спостережуваним даними), є коефіцієнт детермінації. Повну суму квадратів відхилень залежної змінної від її вибіркового середнього в моделі множинної регресії можна представити у вигляді:

$$\sum_{i=1}^{n}(y_i - \bar{y})^2 = \sum_{i=1}^{n}[(y_i - \hat{y}_i) + (\hat{y}_i - \bar{y})]^2 =$$
$$= \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2 + 2\sum_{i=1}^{n}(y_i - \hat{y}_i)(\hat{y}_i - \bar{y})$$

Або у векторній формі:

$$(y - \bar{y}E)^T(y - \bar{y}E) =$$
$$= e^T e + (\hat{y} - \bar{y}E)^T(\hat{y} - \bar{y}E) + 2e^T(\hat{y} - \bar{y}E)$$

де **E** - вектор-стовпець з одиничними елементами, $\mathbf{E=(1,1,....,1)^T}$, розмірності **n**. Згадуючи властивості залишків моделі, маємо:

$$e^T(\hat{y} - \bar{y}E) = e^T(Xb - \bar{y}E) = e^T Xb - \bar{y}e^T E = 0$$

З урахуванням останнього, вираз приймає наступну форму:

$$(y - \bar{y}E)^T(y - \bar{y}E) = (\hat{y} - \bar{y}E)^T(\hat{y} - \bar{y}E) + e^T e$$

Очевидно, перший доданок у формулі - це пояснена сума квадратів відхилень, друге - сума квадратів відхилень яку не можна пояснити, а отже, використовуючи раніше введені позначення цих величин, можна записати

$$TSS = ESS + RSS.$$

Представимо одну з форм запису коефіцієнта детермінації $\mathbf{R^2}$:

$$R^2 = 1 - \frac{RSS}{TSS} = 1 - \frac{e^T e}{(y - \bar{y}E)^T(y - \bar{y}E)}$$

Значення коефіцієнта детермінації у випадку множинної регресії, також як і парної, належать інтервалу **[0, 1]**. Покажемо це. Визначимо верхню межу $\mathbf{R^2}$. Якщо

$$RSS = e^T e = \sum_{i=1}^{n} e_i^2 = 0,$$

то, очевидно, **TSS = ESS**, і з цього випливає, що $\mathbf{R^2} = 1$. Так буде, якщо все $\mathbf{e_i=0}$, і тоді $y_i = \hat{y}_i$, (i = 1,2, ..., n), тобто функція регресії повністю (на всі сто відсотків) пояснює поведінку залежною змінною.

Визначимо нижню межу $\mathbf{R^2}$. Якщо **TSS = RSS**, то **ESS=0** і з цього випливає, що тоді $\mathbf{R^2} = 0$. Так буде, якщо

$$b_1 = \bar{y}, \qquad b_2 = b_3 = \ldots = b_k = 0$$

і в цьому випадку, очевидно $\hat{y}_i = \bar{y}$, для всіх значень **i=1,2,…,n**. Це означає, що поведінка залежною змінною повністю визначається незалежними випадковими помилками моделі, і функція регресії не пояснює поведінку залежною змінною [9].

Як і в випадку парної лінійної регресії, коефіцієнт детермінації багатовимірної (множинної) регресії слід розуміти (інтерпретувати) як частку (частину) дисперсії (вибіркової) змінної **y**, пояснену рівнянням регресії.

Коефіцієнт детермінації служить ***мірою адекватності*** моделі: чим він більший, тим краще (за інших рівних умов) оцінено рівняння регресії.

Вище, говорячи про коефіцієнт детермінації як міру адекватності моделі, було відзначено, що про краще оцінювання моделі можна говорити тільки "при інших рівних умовах". Це означає в тому числі, що застосовувати коефіцієнт $\mathbf{R^2}$ коректно порівнюючи тільки моделей *з рівним числом змі*нних - регресорів [9].

Скоригований коефіцієнт детермінації

Оскільки коефіцієнт детермінації збільшується (точніше, майже завжди збільшується) при збільшенні кількості регресорів в моделі. Це призводить до того, що якщо при оцінці якості моделі орієнтуватися на середній показник R2, то рівняння з великим числом регресорів даватимуть кращий результат, ніж з меншим. Це може привести до невиправданого включенню в модель великого числа малозначущих регресорів.

Включення кожного додаткового регресорів призводить до втрати одного ступеня свободи (нагадаємо, що кількість ступенів свободи дорівнює кількості спостережень мінус кількість оцінюваних коефіцієнтів регресії). При додаванні одного додаткового регресорів додається один коефіцієнт і втрачається одна ступінь свободи. Оскільки кількість спостережень при побудові моделей, як правило, обмежена, а при застосуванні **t-** і **F-** тестів для побудови довірчих і прогнозних інтервалів, а також перевірці гіпотез щодо коефіцієнтів, бажано мати якомога більше ступенів свободи, так як ці інтервали будуть тим менше, ніж більше ступенів свободи, то невиправдане включення додаткового регресорів в статистичному відношенні не бажано. У зв'язку з цим, при аналізі адекватності моделей множинної регресії поряд зі звичайним коефіцієнтом детермінації використовують так звані скориговані коефіцієнти детермінації [9].

Скоригований коефіцієнт детермінації Тейла

Вираз

$$R^2 = 1 - \frac{RSS}{TSS} = 1 - \frac{\sum_{i=1}^{n} e_i^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2},$$

можна записати в еквівалентному вигляді:

$$R^2 = 1 - \frac{\hat{s}^2}{\hat{s}_y^2},$$

де $\hat{s}^2 = \frac{1}{n}\sum_{i=1}^{n} e_i^2$,

а $\hat{s}_y^2 = \frac{1}{n}\sum_{i=1}^{n} (y_i - \bar{y})^2$

це зміщені оцінки дисперсії випадкової складової моделі σ^2 і залежної змінної σ_y^2. Якщо тепер у виразі **R^2** зміщені оцінки дисперсії замінити незміщеними, то отримаємо *скоригований коефіцієнт детермінації Тейла:*

$$\overline{R}^2 = 1 - \frac{RSS/(n-k)}{TSS/(n-1)} = 1 - (1 - R^2)\frac{(n-1)}{(n-k)}$$

Скоригований коефіцієнт детермінації завжди менше звичайного, тобто має місце співвідношення $\overline{R}^2 < R^2$.

Раніше було відзначено, що додавання додаткового регресору, як правило, збільшує значення звичайного коефіцієнта детермінації. Цього не відбувається, якщо використовувати скоригований коефіцієнт детермінації. Його зміна, викликане додаванням регресорів, може бути як позитивним, так і негативним і тому, орієнтуючись на значення скоригованого коефіцієнта, можна більш об'єктивно оцінити, чи доцільно введення додаткового регресорів при зменшенні ступенів свободи (чи призводить це до більш адекватної моделі). Кращою визнається модель, для якої скоригований коефіцієнт $\overline{R}^2$ більше [9].

2.2.1.3. Побудова довірчих інтервалів

Побудова довірчих інтервалів як для окремих коефіцієнтів регресії так і для прогнозу залежної змінної є найважливішим етапом аналізу регресійній моделі. Для побудови довірчих інтервалів і перевірки гіпотез використовуються властивості t-статистики Стьюдента, яка має вигляд:

$$t = \frac{b - \beta_i}{s_{b_i}}$$

де s_{b_i} - оцінка стандартного відхилення σ_{b_i} **i**-го коефіцієнту регресії. У припущенні, що випадкова складова моделі має нормальний розподіл, випадкова змінна **t** підпорядковується центральному **t**-розподілу Стьюдента з **n - k** ступенями свободи. Для розрахунку **t**-статистики необхідно знати оцінки стандартних відхилень або дисперсії оцінок параметрів моделі, які є діагональними елементами оціненої матриці коваріацій вектору оцінок. Отримаємо вираз для цих величин [9].

Емпірична оцінка коваріаційної матриці вектору оцінок параметрів

Для коваріаційної матриці є дійсним наступний вираз:

$$\Sigma_b = \sigma^2 (X^T X)^{-1}$$

У цьому виразі невідомо теоретичне значення дисперсії випадкової складової моделі σ^2. Оцінена по методу найменших квадратів коваріаційна матриця вектору **b** виходить, якщо в вираженні для теоретичної коваріаційної матриці справжнє значення дисперсії σ^2 замінити його незміщеною оцінкою. Отримаємо вираз для такої оцінки:

$$\begin{aligned} e = y - \hat{y} = y - Xb = y - X(X^T X)^{-1} X^T y = \\ = [I - X(X^T X)^{-1} X^T] y = \\ = [I - X(X^T X)^{-1} X^T](X\beta + u) = Gu \end{aligned}$$

де матриця:

$$\mathbf{G = I - X(X^T X)^{-1} X^T}$$

це ідемпотентна матриця. Коваріаційна матриця вектору помилок дорівнює

$$M\{ee^T\} = GM\{uu^T\}G^T = G\sigma^2 IG^T = \sigma^2 G$$

Використовуючи цей вираз, а також такі властивості ідемпотентних матриць: **G= G^T**(ідемпотентна матриця симетрична), **G=GG**, обчислимо величину

$$M\{e^T e\} = tr\{\sigma^2 G\} = \sigma^2 tr\{G\} = \sigma^2 (n - k)$$

де **tr {}** - означає слід матриці (слід матриці дорівнює сумі її діагональних елементів, слід ідемпотентної матриці дорівнює її рангу, ранг одиничної матриці дорівнює її розмірності, ранг ідемпотентної матриці **G** дорівнює **n - k**. З виразу легко бачити, що незміщена емпіричної оцінкою дисперсія випадкової складової буде оцінка виду

$$s^2 = \frac{e^T e}{n-k}$$

Таким чином, для оціненої коваріаційної матриці отримуємо вираз:

$$S_b = s^2 (X^T X)^{-1} = \begin{bmatrix} s_{b_1}^2 & s_{b_1 b_2} & \cdots & s_{b_1 b_k} \\ s_{b_2 b_1} & s_{b_2}^2 & \cdots & s_{b_k b_2} \\ \cdots & \cdots & \cdots & \cdots \\ s_{b_k b_1} & s_{b_k b_2} & \cdots & s_{b_k}^2 \end{bmatrix}.$$

Елементи цієї матриці, що стоять на головній діагоналі, є емпіричними оцінками дисперсії відповідних коефіцієнтів моделі, а елементи, розташовані поза головною діагоналі - оцінками коваріацій оцінок **i-го** і **j-го** коефіцієнтів, для всіх $i \neq j, i, j = 1,2,\ldots,n$ [9].

2.2.1.4. Нелінійні моделі регресії: методи лінеаризації

При моделюванні реальних соціально-економічних процесів далеко не завжди можна описати процес за допомогою лінійної залежності. Однак, можна спробувати так перетворити нелінійну модель, щоб звести її до лінійної. В багатьох випадках це вдається зробити, і досить простим способом.

Багато економічних процесів можна описати нелінійними функціями виду:

$$y_i = \beta_1 x_{i2}^{\beta_2} x_{i3}^{\beta_3} \ldots x_{ik}^{\beta_k} u_i$$

Права частина рівняння є нелінійною функцією як щодо параметрів (степеневу), так і випадкової складової. У цьому випадку говорять, що модель містить мультиплікативні обурення. Як приклад подібного виду залежностей можна привести добре відому виробничу функцію Кобба-Дугласа (її економетричну версію). Незважаючи на примарну складність нелінійної залежності, її також можна звести до лінійної шляхом логарифмічного

перетворення. Дійсно, логарифмуючи праву і ліву частини рівняння, отримаємо:

$$\ln y_i = \ln \beta_1 + \beta_2 \ln x_{i2} + \beta_3 \ln x_{i3} + ... + \beta_k \ln x_{ik} + \ln u_i$$

Рівняння лінійно щодо логарифмів змінних моделі. Таким чином, вводячи перетворення змінних виду:

$$\tilde{y} = \ln y,\ \tilde{x}_j = \ln x_j,\ \tilde{u} = \ln u,\ \tilde{\beta}_1 = \ln \beta_1,$$

нелінійну модель можна звести до лінійної [9].

Важливі зауваження

1) Слід пам'ятати, що логарифмічна перетворення можна застосовувати тільки в разі, якщо змінні вихідної нелінійної моделі приймають позитивні значення. В іншому випадку (при негативних значеннях змінних) логарифмічна функція не визначена.

2) Якщо у вихідній моделі обурення нормально розподілені, то перетворені обурення $\tilde{u}_i = \ln u_i$ цією властивістю володіти не будуть, що призводить до проблем із застосуванням тестів, які засновані на припущенні про нормальний розподіл збурень.

3) При нелінійних перетвореннях, подібних логарифмічному, не можна стверджувати, що властивості оцінок перетвореної моделі після зворотного перетворення збережуться і для вихідної моделі (матимуть місце і для вихідної моделі). Зокрема, оцінка виду $b_1 = \exp(\tilde{b}_1)$ параметра β_1 (тут $\tilde{b}_1$ - оцінка параметра $\tilde{\beta}_1$ в перетвореній моделі) не володітиме тими ж властивостями, що і оцінка $\tilde{b}_1$ [9].

2.2.2 Опис прийнятого методу моделювання

Для полегшення реалізації розрахунку моделі прогнозування методом множинної регресії буде використовуватись математична бібліотека Apache Commons Math.

Commons Math - це бібліотека легких, автономних компонентів з математики та статистики, які вирішують найпоширеніші проблеми, недоступні в мові програмування Java або пакеті Commons Lang [10].

Керівні принципи:

- Випадки використання додатків у реальному світі визначають пріоритет розвитку.
- Цей пакет підкреслює невеликі, легко інтегровані компоненти, а не великі бібліотеки зі складними залежностями та конфігураціями.
- Усі алгоритми повністю задокументовані та відповідають загальноприйнятим найкращим практикам.
- У ситуаціях, коли існує декілька стандартних алгоритмів, використовується стратегія для підтримки декількох реалізацій.
- Обмежені залежності. Ніяких зовнішніх залежностей, крім компонентів Commons та основної платформи Java. Розповсюджується за ліцензією Apache 2.0 .

Окрім того, щоб бути впевненими у адекватності створених моделей прогнозування, будуть побудовані довірчі інтервали та скореговані коефіцієнти детермінації.

2.3. Рішення по інформаційному забезпеченню

2.3.1. Опис інформаційного забезпечення системи

Одну з основ інформаційного забезпечення системи складає визначення її внутрішнього стану, що буде відображено в моделі класів. На даному етапі нас цікавлять постійні сутності бізнесів-процесів. Бізнес-процес – це будь-яка діяльність, що має вхідний продукт, додає вартість до нього, та забезпечує вихідний продукт для внутрішнього або зовнішнього споживача. Вид класів, що забезпечують опис бізнес-процесів назвемо класами-сутностями (чи класами предметної області).

Для створення моделі класів складемо словник даних, ґрунтуючись на сутностях, виявлених на етапі бізнесу-аналізу:

Project Class Statistic – сутність яка описує характеристики проєкту на основі UML діаграми класів.

Project Set – сутність яка описує набор проєктів на основі яких буде створена модель прогнозування.

Forecasting model – сутність яка описує модель прогнозування обсягу програмного забезпечення для проєкту користувача.

Одним із способів змалювання внутрішнього стану системи є створення логічної моделі бази даних. Також логічну модель іноді називають концептуальною моделлю даних. *Концептуальна модель* — це модель предметної області, що складається з переліку взаємопов'язаних понять, що використовуються для опису цієї області, разом з властивостями й характеристиками, класифікацією цих понять, за типами, ситуацій, ознаками в даній області і законів протікання процесів в ній.

На даному етапі проектування сутності словника даних стають кандидатами для класів проектованої системи. Логічна модель даних представлена на рисунку 2.2.

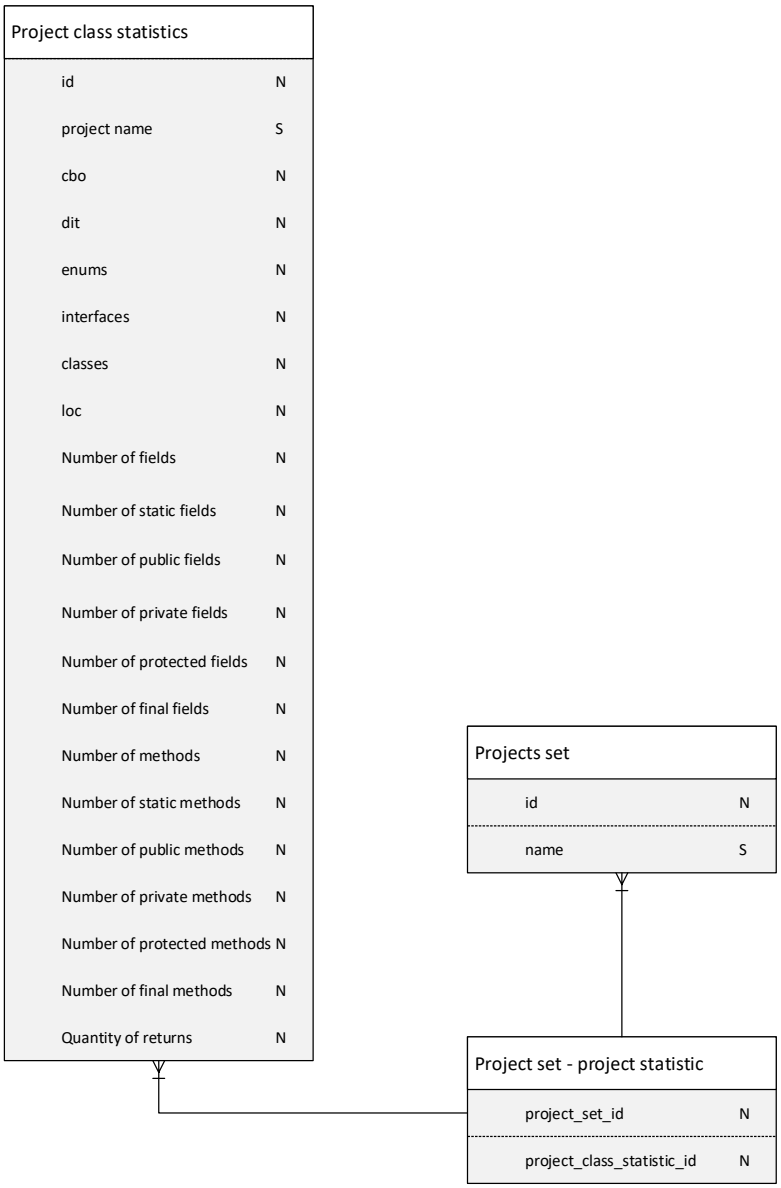


Рисунок 2.2 - Логічна модель даних

На цій діаграмі зображено наступні сутності:

Project Class Statistic – інформація про проєкт.

Project Set – інформація про набор проєктів.

Для відображення відношення many-to-many між проєктом та набором проєктів використовується таблиця project set – project statistic.

До логічної моделі даних не була включена сутність Forecasting model оскільки дані що входять до неї є досить великими, окрім того система повинна реалізовувати збереження моделі прогнозування у окремий CSV файл, що робить не доречним процес їх зберігання у базі даних. Система буде зберігати результат у файл та відправляти його у JSON форматі з відповіддю клієнтській частині.

2.3.2. Опис процесу роботи із системою.

Для опису процесу використання системи використаємо діаграми UML. Канонічні діаграми UML:

- варіантів використання (use case diagram);
- класів (class diagram);
- діяльності (activity diagram);
- розгортання (deployment diagram).

Кожна з цих діаграм деталізує і конкретизує різні уявлення про модель складної системи в термінах мови UML. При цьому діаграма варіантів використання є найбільш загальною концептуальною моделлю складної системи, яка є базовою для побудови решти всіх діаграм. Діаграма класів, за своєю суттю, логічна модель, що відображає статичні аспекти структурної побудови складної системи. Діаграми кооперації і послідовностей є різновиди логічної моделі, які відображають динамічні аспекти функціонування складної системи. Діаграми станів і діяльності призначені для моделювання поведінки системи. І, нарешті, діаграми компонентів і розгортання служать для представлення фізичних компонентів складної системи і тому відносяться до її фізичної моделі.

Діаграма варіантів використання зображена на рис. 2.3.

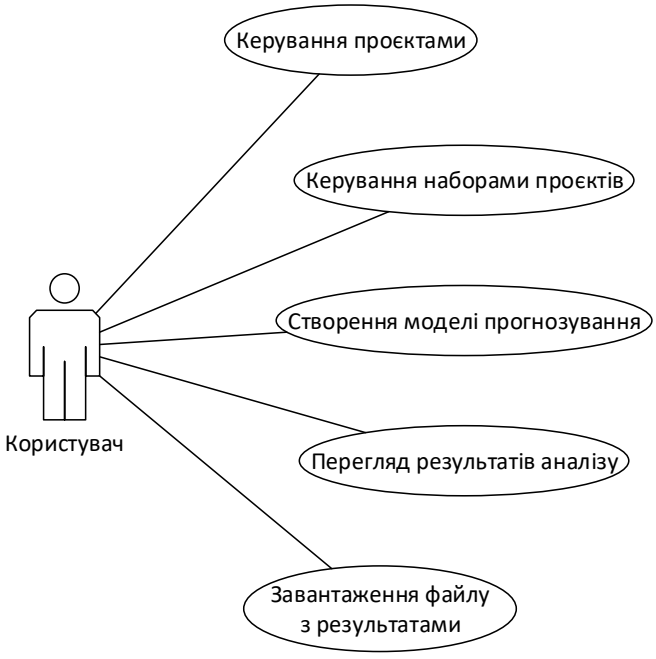


Рисунок 2.3 - Діаграма варіантів використання системи

На рисунку 2.4 представлена декомпозиція варіанту використання «Керування проєктами».

Рисунок 2.4 - Діаграма декомпозиції варіанту використання «Керування проєктами».

На рисунку 2.5 представлена декомпозиція варіанту використання «Керування наборами проєктів».

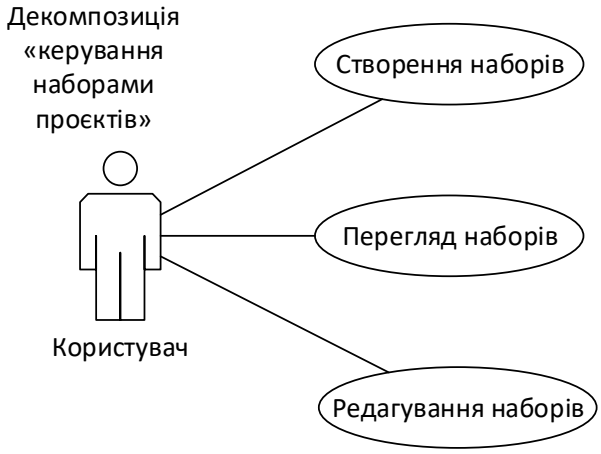


Рисунок 2.5 - Діаграма декомпозиції варіанту використання «Керування наборами проєктів».

На рисунку 2.6 представлена діаграма діяльності користувача при роботі з системою.

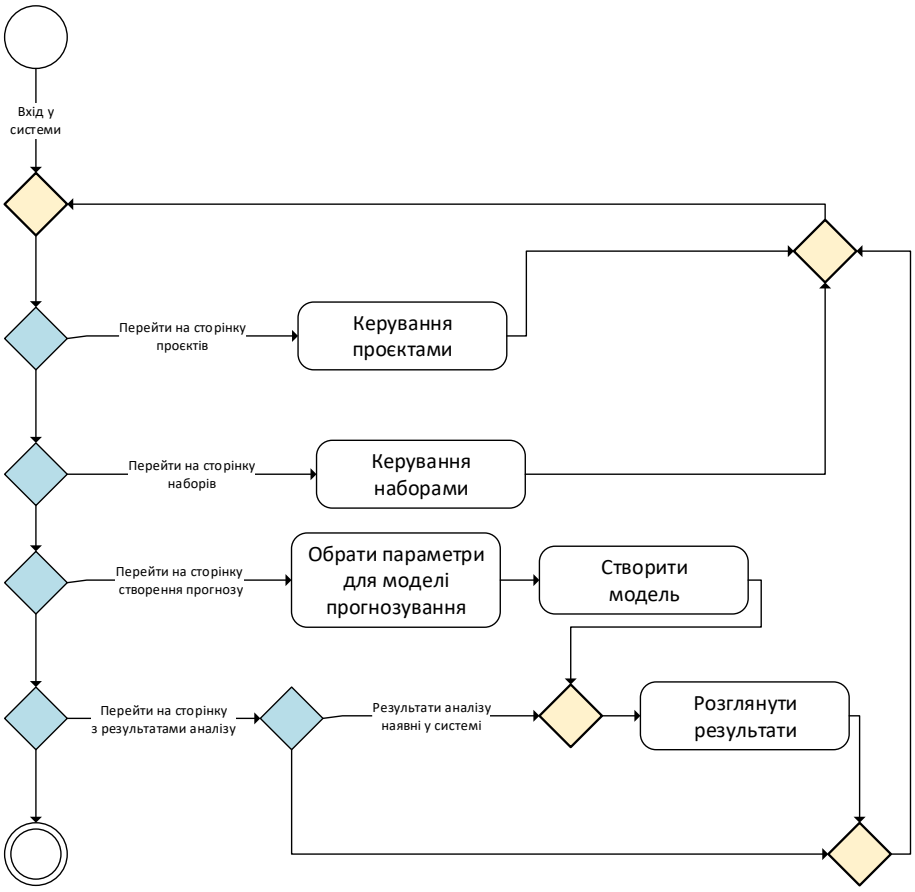
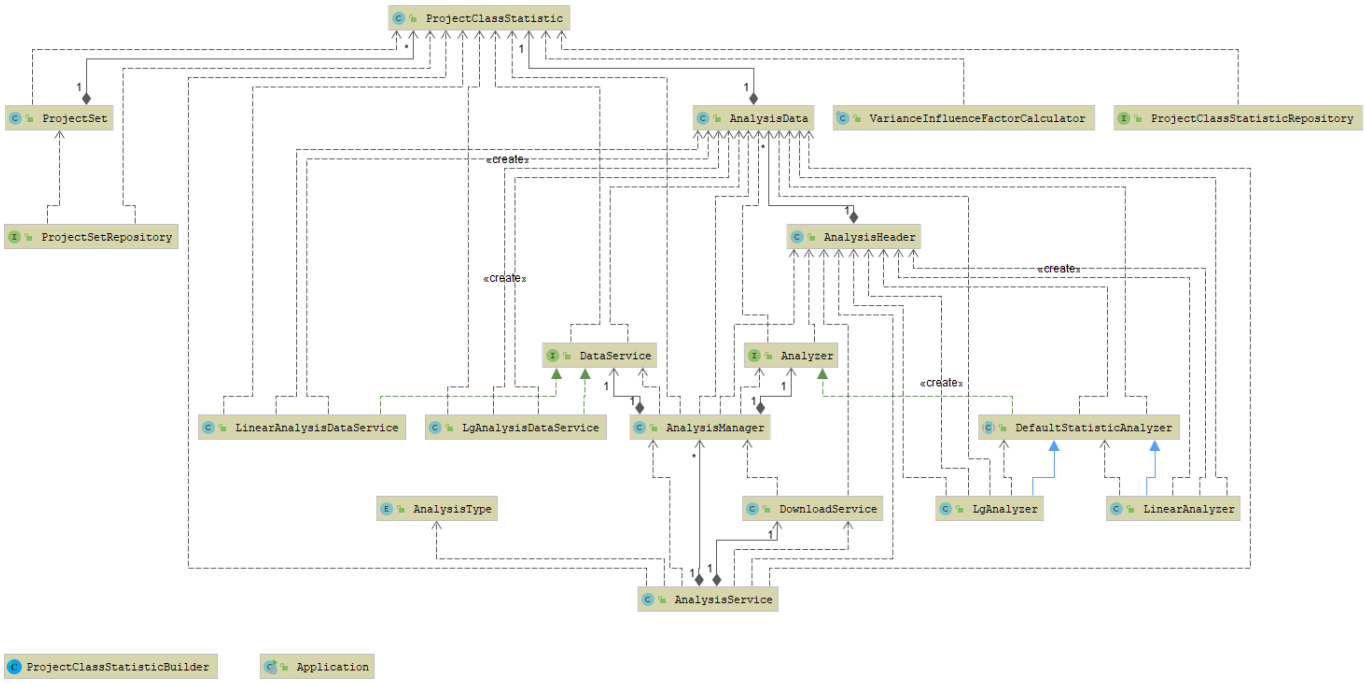


Рисунок 2.6- Діаграма діяльності користувача

Діаграма класів для системи представлена на рисунку 2.7



Рисунку 2.7 – Діаграма основних класів системи

На цій діаграмі представлені наступні класи:

- *ProjectClassStatistic* – описує сутність проєкту, його назву та характеристики згідно з попередніми концепціями;
- *ProjectClassStatisticBuilder* – реалізація шаблону проєктування «Builder» для сутності ProjectClassStatistic;
- *ProjectSet* – описує сутність набору проєктів згідно з попередніми концепціями;
- *ProjectClassStatisticRepository* – інтерфейс який описує шаблон проєктування «Repository» для сутності ProjectClassStatistic, надає змогу користуватися CRUD операціями для бази даних;
- *ProjectSetRepository* - інтерфейс який описує шаблон проєктування «Repository» для сутності ProjectSet, надає змогу користуватися CRUD операціями для бази даних;
- *AnalysisHeader* – сутність яка представляє основні характеристики та метрики для розрахованої моделі прогнозування;
- *AnalysisData* – сутність яка зберігає результати кожного етапу розрахунку моделі прогнозування для кожного окремого проєкту на основі якого створюється модель;
- *LinearAnalysisDataService* – реалізація інтерфейсу DataService, виконує підготовку даних з набору проєктів перед аналізом лінійного типу;
- *LgAnalysisDataService* – реалізація інтерфейсу DataService, виконує підготовку даних з набору проєктів перед аналізом степеневого типу;
- *DefaultStatisticAnalyzer* – абстрактний клас що втілює інтерфейс *Analyzer* та надає реалізації розрахунків моделі прогнозування ідентичні для всіх типів аналізу;
- *LgAnalyzer* – наслідник класу DefaultStatisticAnalyzer містить конкретні реалізації для розрахунку моделі прогнозування на основі множинної степеневої регресії;

- *LinearAnalyzer* – наслідник класу DefaultStatisticAnalyzer містить конкретні реалізації для розрахунку моделі прогнозування на основі множинної лінійної регресії;
- *AnalysisManager* – клас зберігає у собі всі реалізації інтерфейсів *Analyzer та DataService*. Надає необхідний набор реалізацій в залежності від обраного типу аналізу;
- *DownloadService* – надає можливість завантаження CSV файлу з результатами моделі прогнозування;
- *AnalysisService* – надає інтерфейс високого рівня для створення моделей прогнозування;
- *AnalisisType* – перечислення що характеризує типи аналізу;
- *Application* – головний клас системи який є стартовим пунктом запуску додатку.

Згідно з трирівневою клієнт-серверною архітектурною моделлю, система повинна бути поділена на три рівні. Для розгортання кожного з рівнів будуть використовуватися наступні компоненти:

Клієнтський рівень – NGSNX, вільний веб-сервер і проксі-сервер, розповсюджується під BSD-подібною ліцензією.

Серверний рівень – Embedded Apache Tomcat простий і надійний вбудований сервер який поставляється разом із Spring Framework. Є вільним програмним забезпеченням що постачається під ліцензією Apache License 2.0 .

MySQL – як сервер баз даних.

Діаграма розгортання системи зображена на рисунку 2.8

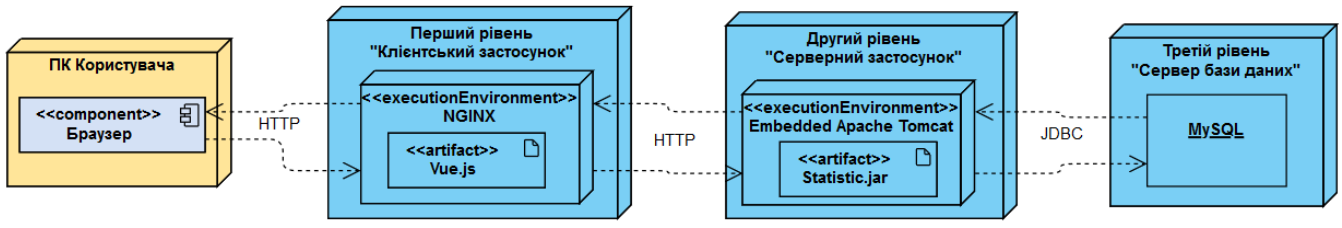


Рисунок 2.8 - Діаграма розгортання системи

2.4. Рішення по программному забезпеченню

2.4.1. Структура і функції частин програмного забезпечення

Сформульовані для системи вимоги приймемо за обмеження. Поряд з моделлю сутностей для проектування класів будемо користатися побудованими раніше схемами функціональної структури, у відповідності до рекомендацій методики Rational Unified Process. При цьому модель класів буде відповідати словнику даних, що був отриманий на підставі аналізу прецедентів, і який є текстовим аналогом моделі сутностей.

Програмне забезпечення розроблюваної системи складатиметься з програмного комплексу, що розміщується на серверах системи. Для роботи користувача з системою у загальному випадку встановлення додаткового ПЗ не передбачається – взаємодія відбувається за допомогою веб-браузера.

Модуль управління проєктами використовується для введення даних про проєкти у систему, дозволяє змінювати характеристики проєктів у системі, переглядати та видаляти їх.

Модуль управління наборами проєктів надає змогу користувачам створювати унікальні набори проєктів, переглядати та модифікувати вже існуючі набори, порівнювати змісти наборів.

Модуль статистики надає функціонал для створення моделей прогнозування на основі наборів проєктів за допомогою методу множинної регресії.

Модуль завантажень надає змогу користувачам завантажити результати моделі прогнозування у вигляді CSV файлу.

Модуль генерації таблиць надає змогу створювати відображення у вигляді таблиць для наборів проєктів або моделі прогнозування.

Модуль генерації графіків надає змогу створювати відображення у вигляді графіків для наборів проєктів та графік регресії для моделі прогнозування.

Модуль збереження стану надає змогу клієнтській частині додатку зберігати та обмінюватися даними із користувачами та серверною частиною системи.

2.4.2. Фізична модель бази даних

Для забезпечення можливості зберігати дані в реляційній базі даних, з моделі класів була отримана фізична модель даних. Для цього серед усіх класів були визначені постійні. Це такі класи, об'єкти яких повинні зберігати свої значення не тільки підчас роботи програми. Далі було проведено відображення постійних класів на таблиці моделі даних. При цьому атрибути класів відобразилися в поля реляційних таблиць. Асоціації між класами змінилися в зв'язки між реляційними таблицями з використанням механізму зовнішніх ключів. Для забезпечення можливості підтримки цілісності даних по сутностям були використані первинні ключі. Обрана СКБД не підтримує зберігання ієрархії спадкування класів, тому при будуванні фізичної моделі даних ієрархії відображалися на таблиці за принципом одна таблиця на клас. Розроблена фізична модель наведена на рисунку 2.9.

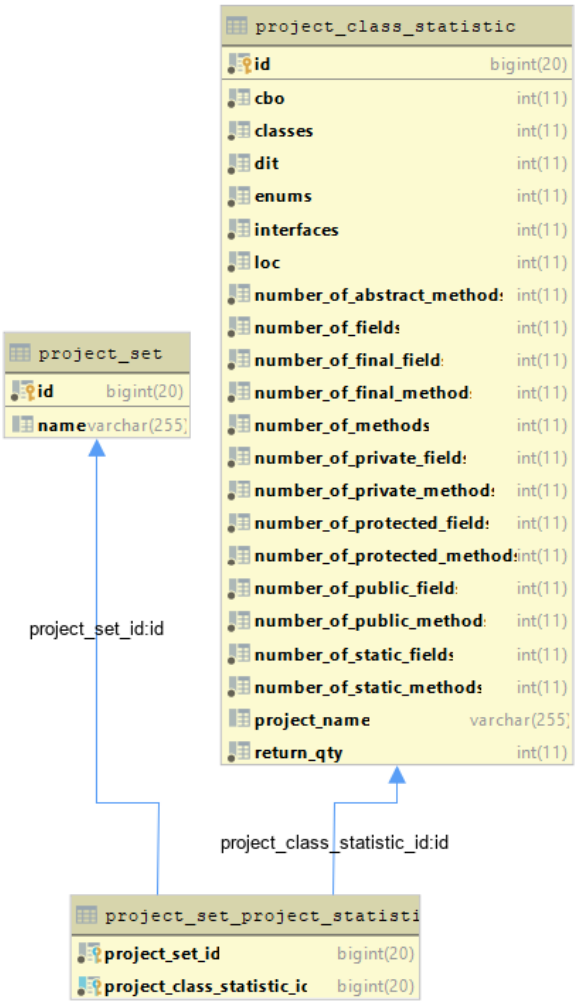


Рисунок 2.9 - Фізична модель бази даних

2.4.3. Методи і засоби розробки програмного забезпечення

При аналізі і проектуванні Системи застосовувався об’єктно-орієнтований підхід (ООП), тому програмне забезпечення системи повинне бути розроблене з його використанням.

Використання ООП у розробці дозволяє:

- використовувати отримані в результаті аналізу і проектування логічні моделі для первинної генерації коду;
- скоротити час розробки за рахунок повторного використання коду програмних компонентів;
- проводити налаштування і тестування ґрунтуючись на поводженні об'єктів, без знання їхньої внутрішньої структури;
- цілком використовувати потенціал обраних засобів розробки ПО.

Засобом розробки програмного забезпечення клієнтського рівня системи є JavaScript — динамічна, об'єктно-орієнтована прототипна мова програмування, використовується для створення сценаріїв веб-сторінок, що надає можливість на стороні клієнта (пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб-сторінки. Окрім того використовуються такі технології: Vue.js, Vue-resources, Vue-router, Vuex, Chart.js, Chartjs plugin zoom, Popper.js, JQuery, Bootstrap 4. У якості середовища розробки був використаний продукт IntelliJ IDEA 2019.

Засобом розробки програмного забезпечення серверного рівня системи є платформо-незалежна мова програмування високого рівня Java, використовуються такі технології: Spring Framework, Spring Boot, Spring Data JPA, Spring MVC, Hibernate 5, Lombok, Apache Commons Math. У якості середовища розробки був використаний продукт IntelliJ IDEA 2019.

На рівні серверу баз даних використовується вільна система керування реляційними базами даних MySQL(MariaDB), це сучасна серверна реляційна СКБД, що має високу продуктивність, здатна працювати з великими обсягами інформації і підтримує стандарт SQL-92.

					ДР.122.4142.05.ПЗ.Р03		
Зм.	Аркуш	№ документа	Підпис	Дата			
					Реалізація проекту автоматизованої системи	Літ.	Аркуш 69 / Аркушів
Студент		Каратай М.В.					
Керівник		Беркунський Є.Ю.				НУК ім. адм. Макарова	
Консульт.							
Зав. Каф.		Михелєв І.Л.					

Розділ 3. Реалізація проєкту прогнозування обсягу програмного забезпечення, створюваного мовою Java на основі UML метрик.

3.1. Технічне завдання

Технічне завдання для реалізованого проєкту, розроблене і представлене у додатку А.

3.2. Опис розробки системи

3.2.1. Структура бази знань

За основу бази знань яка необхідна для створення моделей прогнозування були взяті кількісні характеристики відомих open-source проєктів з веб-сервісу GitHub. Відібрані проєкти представлені у додатку Б.

Для підрахунку метрик проєктів використовувався open-source проєкт CK версії 0.3.2-SNAPSHOT(від 27.04.2019) [8]. Який розповсюджується за ліцензією Apache 2.0 License.

3.2.2. Інструкція з використання системи

Одразу після переходу на домашню сторінку користувач має змогу створити модель прогнозування користуючись формою створення або перейти на будь-яку доступну сторінку додатку користуючись навігаційним баром. Домашня сторінка та форма створення представлені на рисунках 3.1- 3.2

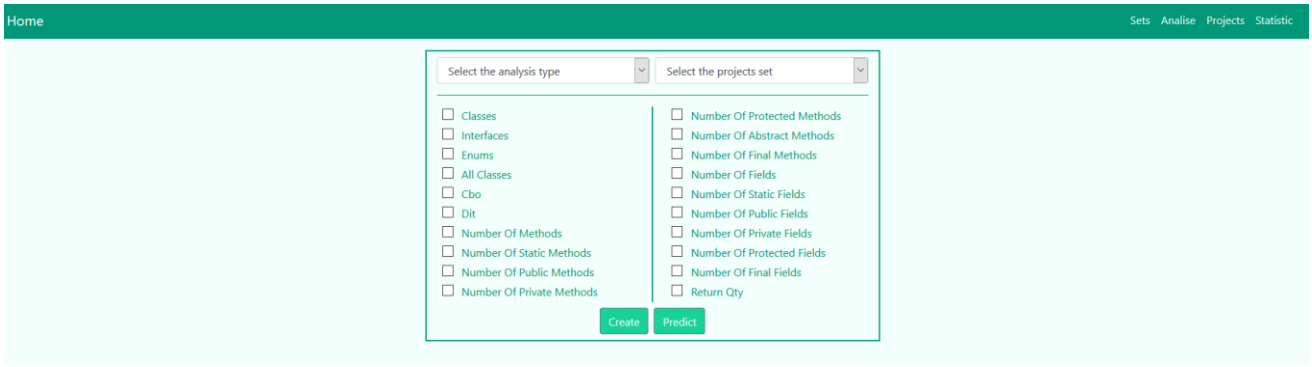


Рисунок 3.1 – Домашня сторінка

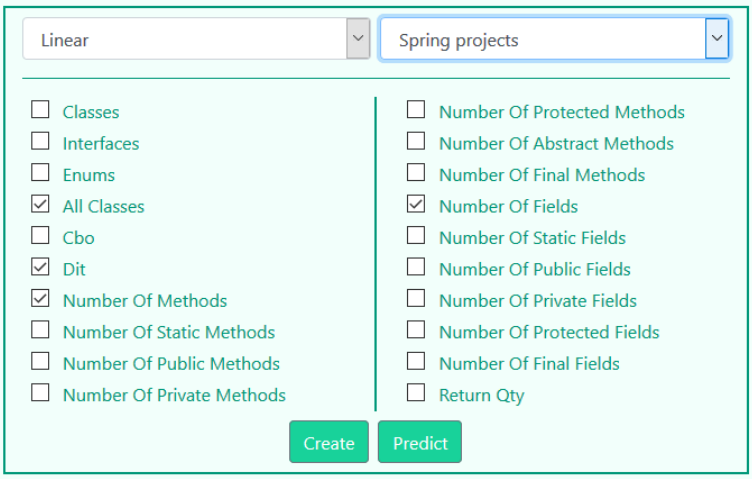


Рисунок 3.2 – Форма створення моделі прогнозування

Користуючись формою створення користувач має змогу обрати тип аналізу, набор проєктів на основі яких буде створена модель прогнозування та параметри які будуть використовуватися для розрахунку. Налаштувавши необхідні для моделі параметри користувач має 2 опції для створення моделі:

- Натиснувши на кнопку *"Create"* буде створена модель прогнозування та користувач буде автоматично пере направлений на сторінку з результатами.
- Натиснувши на кнопку *"Predict"* користувач матиме змогу ввести характеристики свого проєкту для якого буде виконаний прогноз кількості рядків коду на основі моделі прогнозування створеної за допомогою обраних користувачем параметрів. Секція користувацьких параметрів представлена на рисунку 3.3

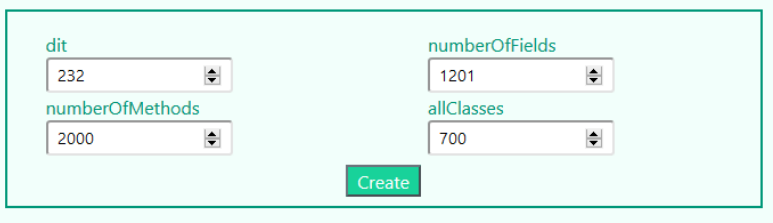


Рисунок 3.3 – Секція користувацьких параметрів

Після введення необхідних даних користувач має натиснути на кнопку *"Create"* в результаті чого він буде пере направлений на сторінку з результатами розрахунку.

У випадку якщо аналіз був створений із використанням користувацьких даних для прогнозування сторінка з результатами буде мати додаткову секцію під назвою *“Prediction”* . Приклад сторінки з результатами на якій присутня секція *“Prediction”* представлений на рисунку 3.4

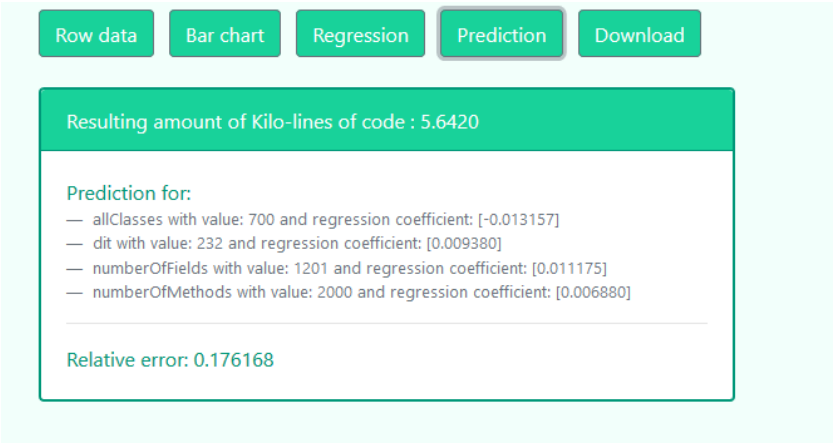
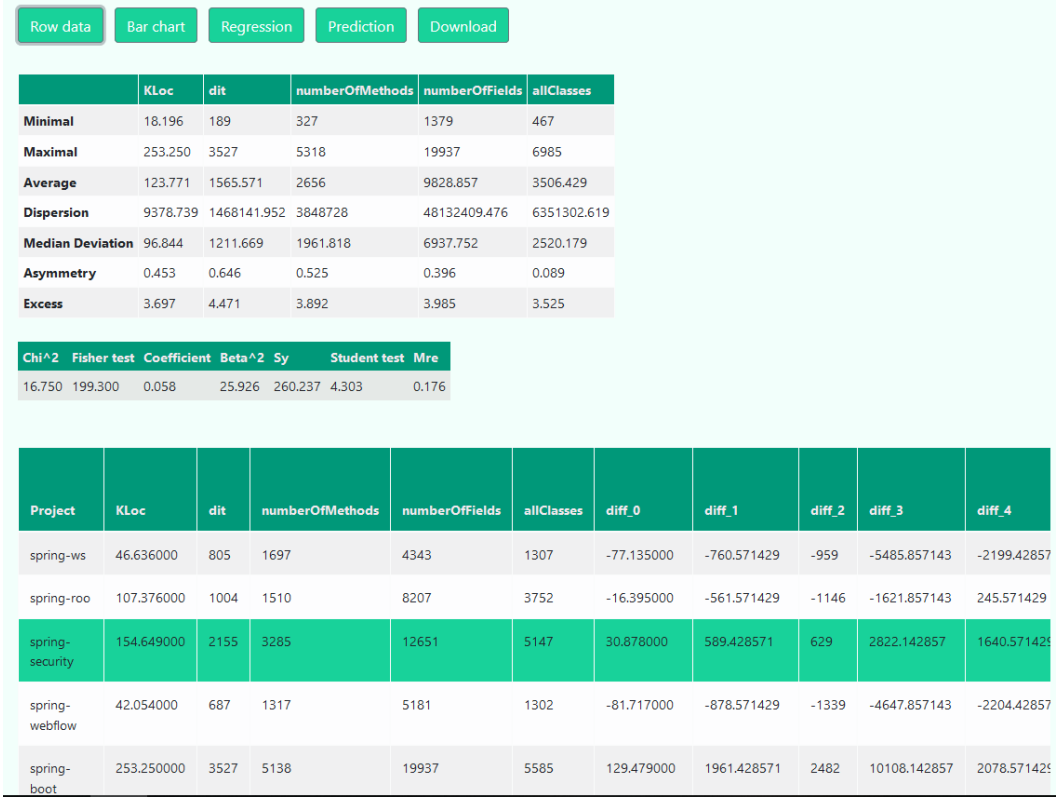


Рисунок 3.4 – Приклад сторінки з результатами

Виходячи з результатів прогнозування на рисунку 3.4 можна сказати що обсяг програмного забезпечення для параметрів: “All classes” кількістю 700, “DIT” кількістю 232, “number of fields” кількістю 1201 та “number of methods” кількістю 2000 може становити *5.642* тисяч рядків коду. Відносна похибка розрахунку становить *0.176168* .

Окрім того на сторінці присутні наступні секції:

- “Row data” – надає змогу переглянути детальний результат розрахунку моделі прогнозування. Розрахунок поділений на декілька таблиць: перша містить основну інформацію що до характеристик проєктів які брали участь у розрахунку, друга значення різних коефіцієнтів для перевірки надійності моделі, третя безпосередньо містить кожний крок розрахунку моделі прогнозування. Вигляд секції “Row data” представлений на рисунку 3.5



	KLoc	dit	numberOfMethods	numberOfFields	allClasses
Minimal	18.196	189	327	1379	467
Maximal	253.250	3527	5318	19937	6985
Average	123.771	1565.571	2656	9828.857	3506.429
Dispersion	9378.739	1468141.952	3848728	48132409.476	6351302.619
Median Deviation	96.844	1211.669	1961.818	6937.752	2520.179
Asymmetry	0.453	0.646	0.525	0.396	0.089
Excess	3.697	4.471	3.892	3.985	3.525

Chi^2	Fisher test	Coefficient	Beta^2	Sy	Student test	Mre
16.750	199.300	0.058	25.926	260.237	4.303	0.176

Project	KLoc	dit	numberOfMethods	numberOfFields	allClasses	diff_0	diff_1	diff_2	diff_3	diff_4
spring-ws	46.636000	805	1697	4343	1307	-77.135000	-760.571429	-959	-5485.857143	-2199.42857
spring-roo	107.376000	1004	1510	8207	3752	-16.395000	-561.571429	-1146	-1621.857143	245.571429
spring-security	154.649000	2155	3285	12651	5147	30.878000	589.428571	629	2822.142857	1640.57142
spring-webflow	42.054000	687	1317	5181	1302	-81.717000	-878.571429	-1339	-4647.857143	-2204.42857
spring-boot	253.250000	3527	5138	19937	5585	129.479000	1961.428571	2482	10108.142857	2078.57142

Рисунок 3.5 - Секція “Row data”

- *“Bar chart”* – секція містить стовпчиковій графік який відображає реальний обсяг проєктів які брали участь у аналізі, а також прогнозовані для них величини обсягу. Крім того там присутні деякі аналізовані величини. Секція *“Bar chart”* представлена на рисунку 3.6

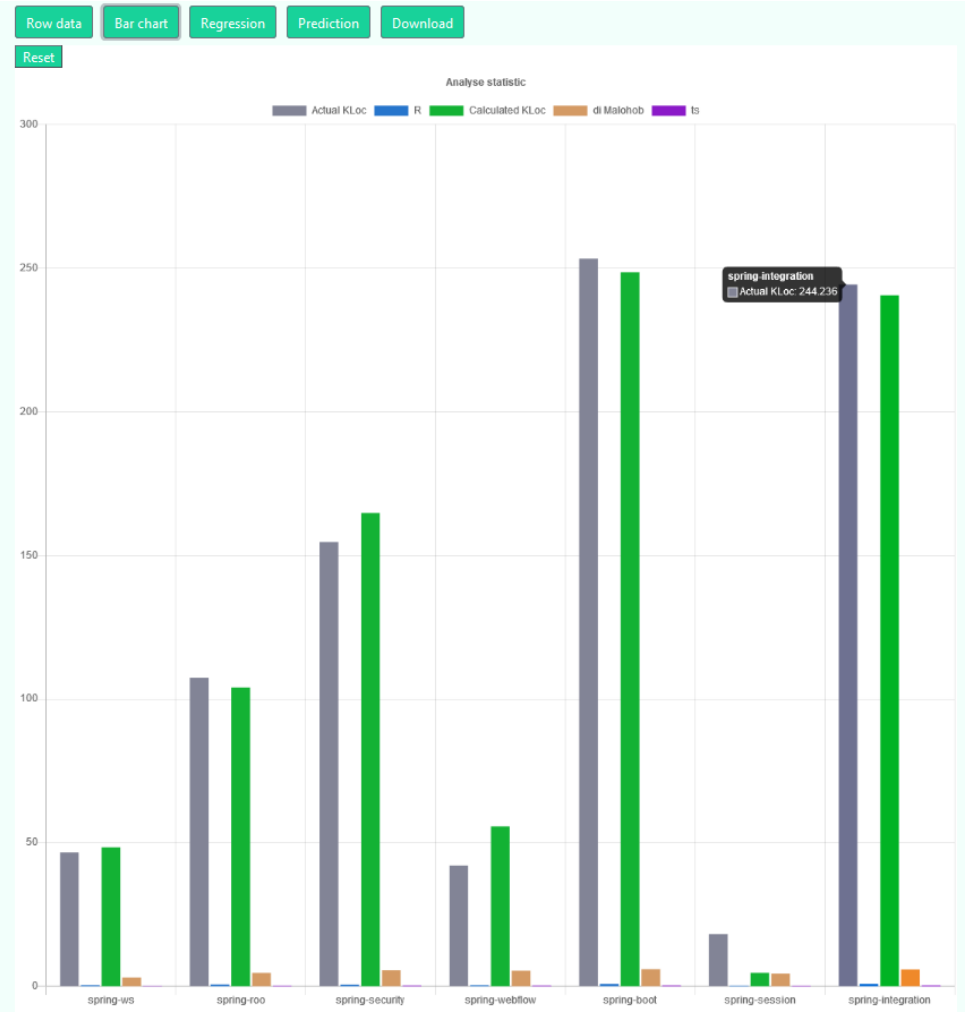


Рисунок 3.6 – секція *"Bar chart"*

- *"Regression"* – секція містить графік регресії для створеної моделі прогнозування. На ньому присутні точки які відповідають кожному проєкту, а саме прогнозована та реальна кількість рядків коду. Окрім того на графіку присутній довірчий інтервал побудований для кожного проєкту. Секція *"Regression"* представлена на рисунку *3.7*

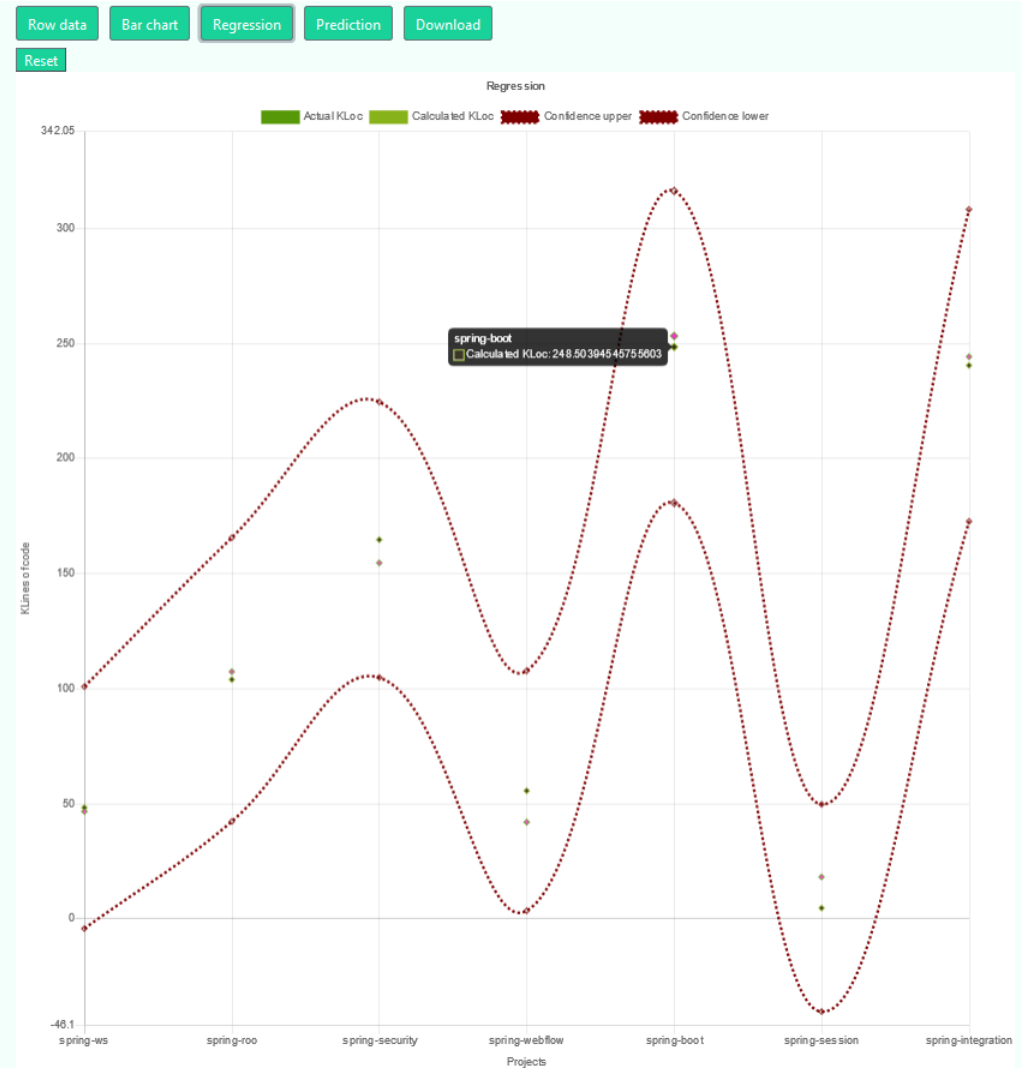


Рисунок 3.7 – секція *"Regression"*

- *"Download"* – надає змогу розпочати діалог для завантаження csv файлу з результатами аналізу. Діалог завантаження результатів аналізу представлений на рисунку 3.8

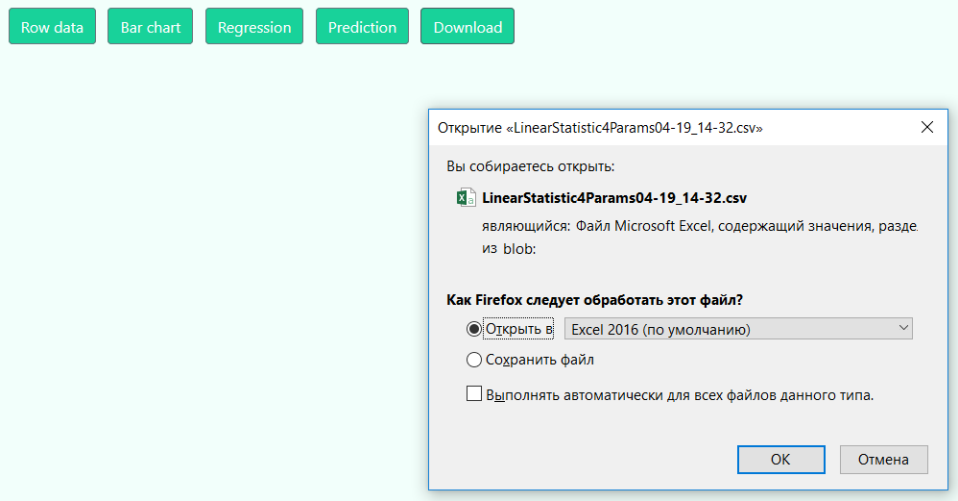


Рисунок 3.8 – Діалог завантаження результатів аналізу

Сторінка Projects

Розглянемо сторінку *"Projects"*. На ній користувач має змогу додавати до системи будь-які проєкти за його бажанням. Модифікувати вже існуючі проєкти та видаляти проєкти із системи. Сторінка *"Projects"* представлена на рисунку 3.9

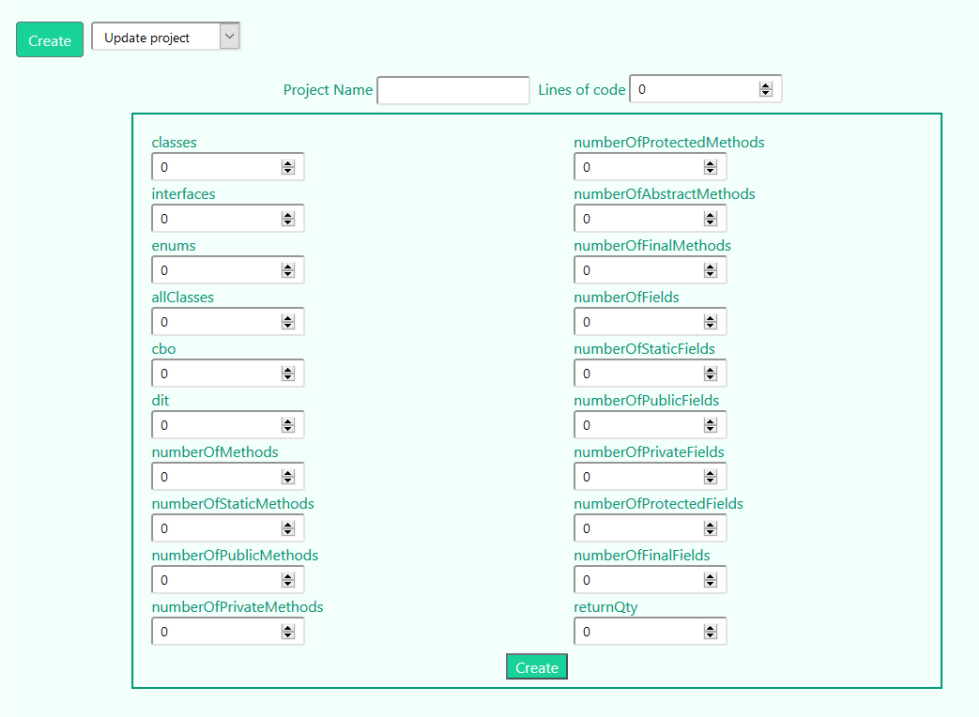


Рисунок 3.9 – Сторінка *"Projects"*

Заповнивши всі необхідні поля з даними користувач матиме змогу створити проєкт натиснувши на кнопку "Create".

Для оновлення вже існуючого проєкту необхідно натиснути на кнопку … та обрати необхідній проєкт, після внесення необхідних змін натиснути на кнопку … , після чого проєкт буде оновлено. Процес оновлення проєкту представлений на рисунках 3.10 – 3.11

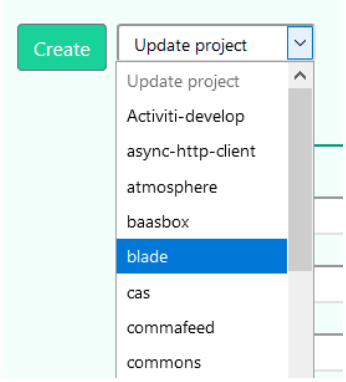


Рисунок 3.10 – Вибір проєкту для оновлення

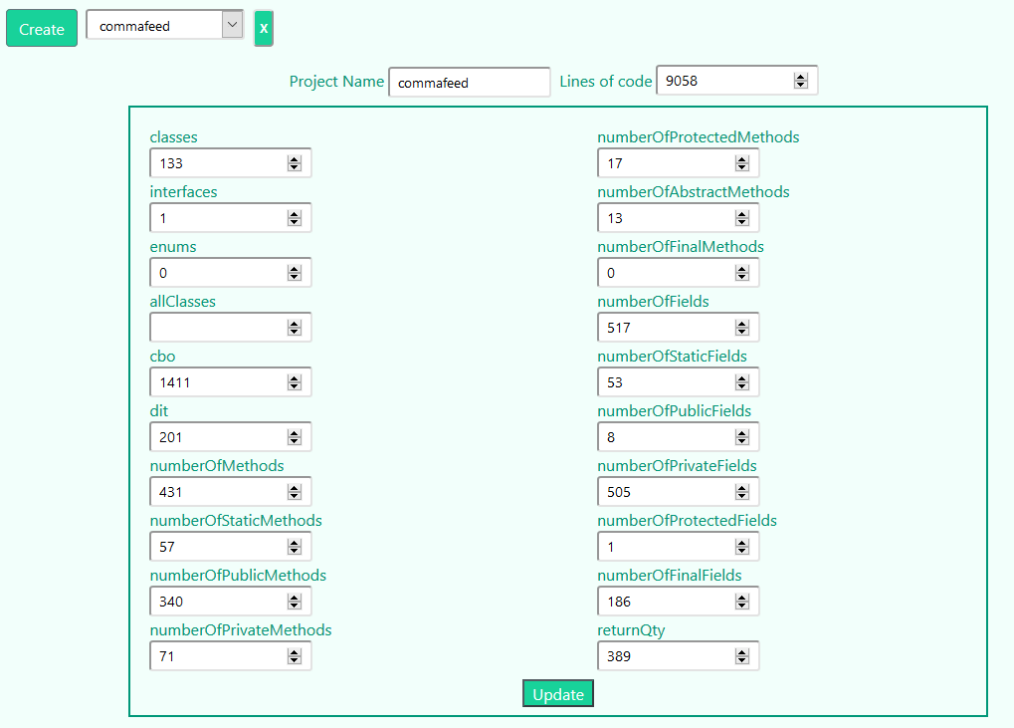


Рисунок 3.11 – Форма оновлення проєкту

Окрім того натиснувши на кнопку *"X"* користувач має змогу видалити проєкт із системи, після натискання з'явиться діалог для підтвердження видалення. Цей процес представлено на рисунках 3.12 -3.13

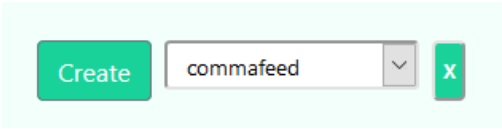


Рисунок 3.12 – Кнопка для видалення

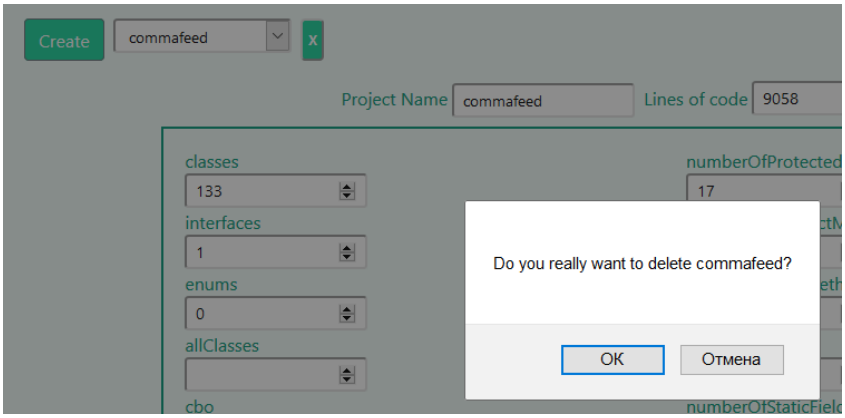


Рисунок 3.13 – Діалог підтвердження

Сторінка Sets

Розглянемо сторінку *"Sets"*. На ній користувач має змогу переглядати вже існуючі набори проєктів, характеристики проєктів які входять до набору як за допомогою таблиці так і у вигляді стовпчикового графіку. Окрім того користувач має змогу створювати нові набори та оновлювати вже існуючі. Сторінка з наборами представлена на рисунку 3.14

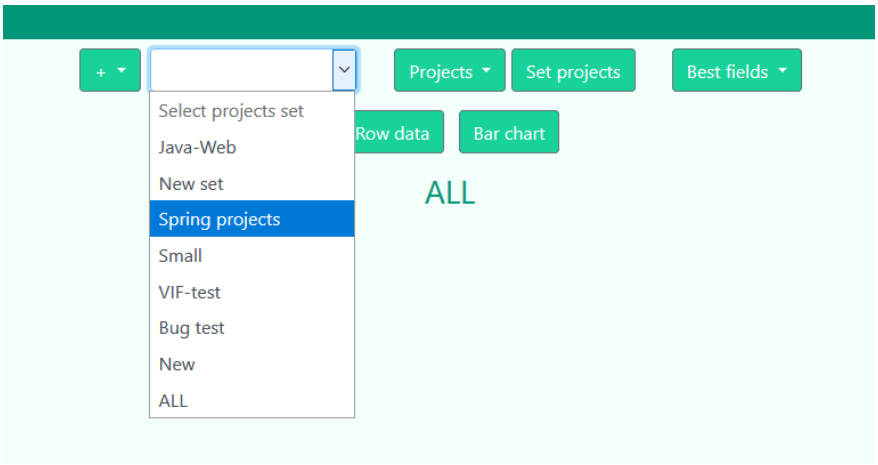
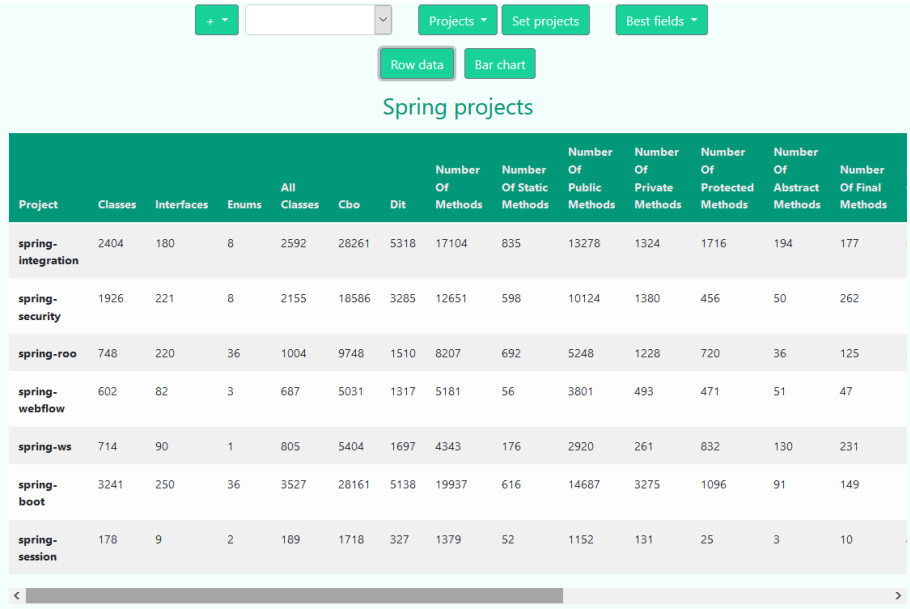


Рисунок 3.14 – Сторінка з наборами

Після вибору набору користувач може натиснути на кнопку "Row data" для того щоб розглянути вміст набору у вигляді таблиці, або на кнопку "Bar char" для того щоб отримати ті ж самі дані у вигляді стовпчикового графіку. Ці опції представлені на рисунку 3.15 – 3.16



Project	Classes	Interfaces	Enums	All Classes	Cbo	Dit	Number Of Methods	Number Of Static Methods	Number Of Public Methods	Number Of Private Methods	Number Of Protected Methods	Number Of Abstract Methods	Number Of Final Methods
spring-integration	2404	180	8	2592	28261	5318	17104	835	13278	1324	1716	194	177
spring-security	1926	221	8	2155	18586	3285	12651	598	10124	1380	456	50	262
spring-roo	748	220	36	1004	9748	1510	8207	692	5248	1228	720	36	125
spring-webflow	602	82	3	687	5031	1317	5181	56	3801	493	471	51	47
spring-ws	714	90	1	805	5404	1697	4343	176	2920	261	832	130	231
spring-boot	3241	250	36	3527	28161	5138	19937	616	14687	3275	1096	91	149
spring-session	178	9	2	189	1718	327	1379	52	1152	131	25	3	10

Рисунок 3.15 – Вміст набору у вигляді таблиці

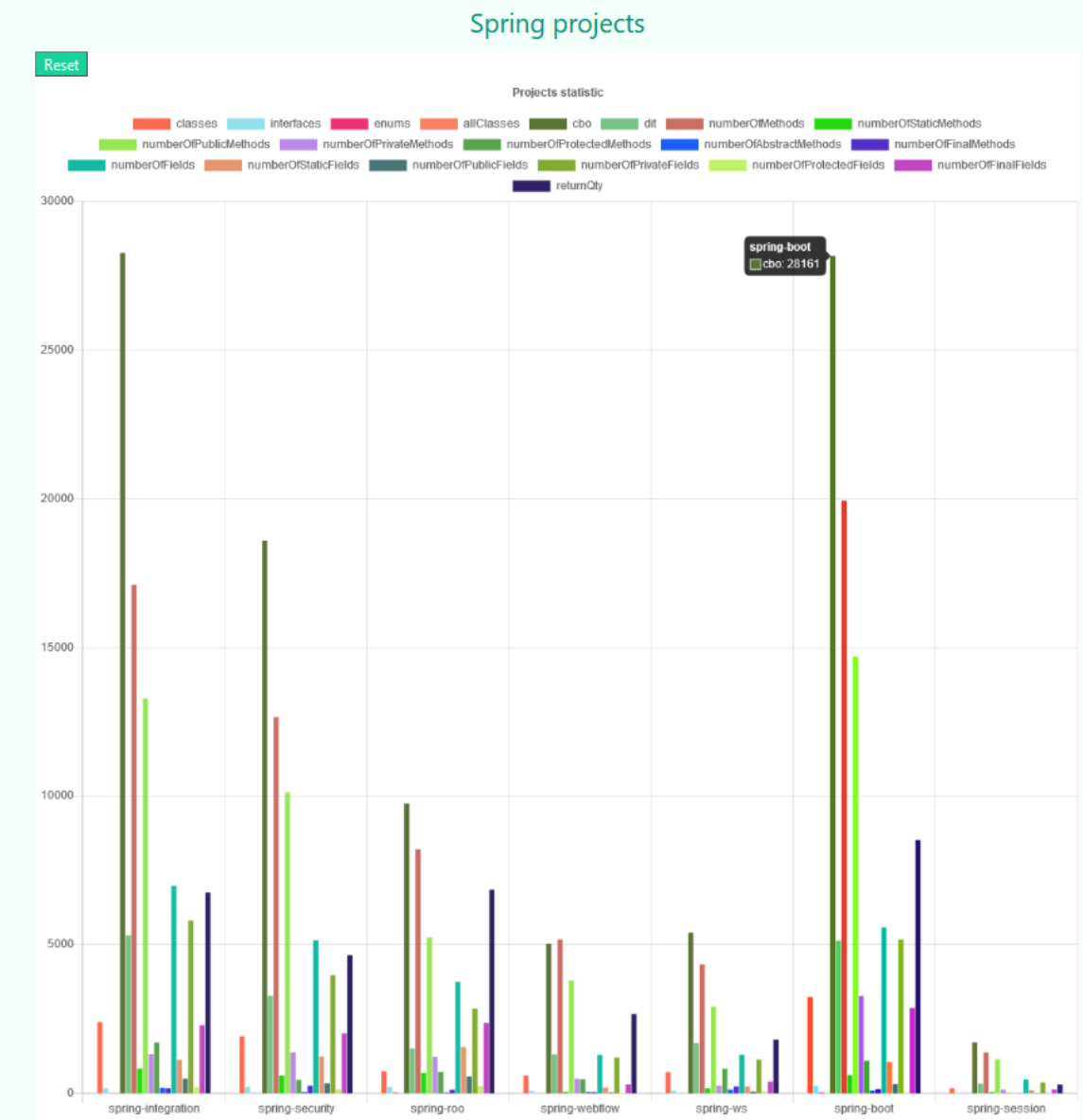


Рисунок 3.16 – Вміст набору у вигляді стовпчикового графіку

Для зміни складу набору необхідно натиснути на вкладку “Projects” та обрати необхідні проєкти, після цього натиснути кнопку “Set projects” яка оновить склад набору згідно з обраними проєктами. Вкладка “Projects” представлена на рисунку 3.17

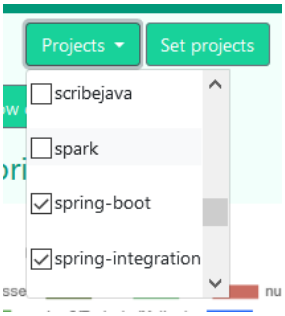


Рисунок 3.17 - Вкладка “Projects”

Для створення нового набору необхідно натиснути на вкладку “+” після чого ввести назву нового набору та натиснути на кнопку “Create”. Після чого новій набор буде доступній для вибору у системі. Процес створення набору представлений на рисунку 3.18 – 3.19

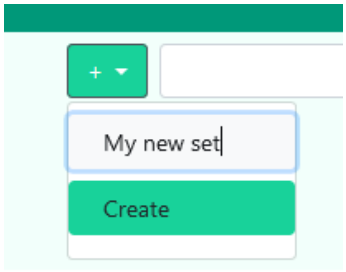


Рисунок 3.18 – Вкладка для створення набору

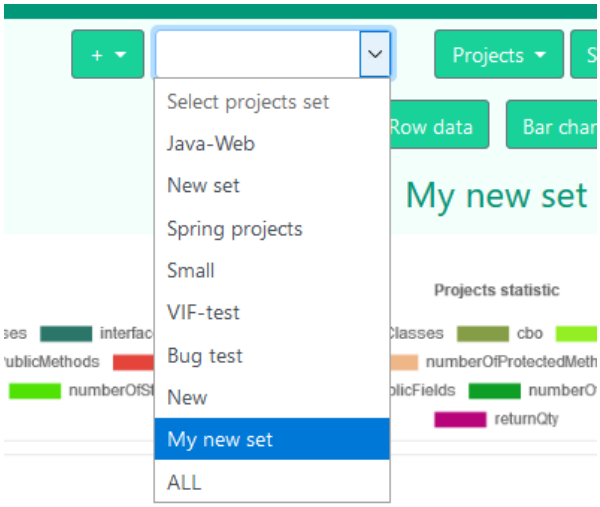


Рисунок 3.19 – Результат створення набору

					ДР.122.4142.05.ПЗ.Р04		
Зм.	Аркуш	№ документа	Підпис	Дата			
					Охорона праці	Літ.	Аркуш 81 / Аркушів
Студент		Каратай М.В.					
Керівник		Беркунський Є.Ю.				НУК ім. адм. Макарова	
Консульт.		Гурець Н.В.					
Зав. Каф.		Михелєв І.Л.					

Розділ 4. Охорона праці

4.1. Вступ

Визначення поняття охорони праці дається в ст. 1 Закону України від 14 жовтня 1992 р. «Про охорону праці». *Охорона праці* — це система правових, соціально-економічних, організаційно-технічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження здоров'я і працездатності людини в процесі праці. В поняття охорони праці входять і всі ті заходи, що спеціально призначені для створення особливих полегшених умов праці для жінок і неповнолітніх, а також працівників зі зниженою працездатністю.

Охорону праці і здоров'я громадян віднесено до пріоритетних напрямків соціальної політики України. Так, Конституція України одним з основних соціальних прав громадян визначає право кожного на належні, безпечні й здорові умови праці, встановлює, що використання праці жінок і неповнолітніх на небезпечних для їхнього здоров'я роботах забороняється.

Завдання охорони праці:

- проектування підприємств, технологічних процесів і конструювання обладнання з обов'язковим виконанням вимог охорони праці;

- знаходження оптимальних співвідношень між різними факторами виробничого середовища, що дозволяє забезпечити мінімум несприятливого впливу їх на здоров'я працівників;

- встановлення, законодавче оформлення визначених норм кожного з несприятливих або небезпечних факторів, систематичний король за їх застосуванням;

- розробка конкретних заходів щодо покращення умов праці та забезпечення її безпеки на основі застосування у виробництві новітніх досягнень науки і техніки;

- застосування раціональних засобів захисту працівників від впливу несприятливих факторів виробничого середовища, а також втілення

організаційних заходів, які нейтралізують або послаблюють ступінь їх впливу на організм людини;

- розробка та застосування методів і засобів оцінки ефективності заходів з охорони праці, що плануються і здійснюються.

Стан охорони праці в Україні

Критична ситуація в Україні у сфері безпеки праці проявляється високим рівнем виробничого травматизму і професійної захворюваності, незадовільними умовами праці та санітарним станом підприємств, внаслідок чого держава втрачає кваліфікованих працівників, а натомість отримує десятки тисяч осіб, які потребують повноцінного соціального захисту.

За останні 5 років на виробництві загинуло майже 5 тисяч працівників і понад 85 тисяч – травмовано.

Не дивлячись на те, що з року в рік кількість травмованих на виробництві в Україні зменшується, рівень виробничого травматизму у порівнянні з країнами Західної і навіть Східної Європи залишається високим. Аналіз ситуації показує, що зазначене зниження за останні 10 років відбувається у більшій мірі в результаті зменшення числа робочих місць, значних обсягів неврахованих, або переведених у категорію не пов’язаних з виробництвом нещасних випадків.

Домінуючими причинами формування несприятливих умов праці залишаються недосконалі технології, машини і механізми, їхня несправність, невикористання засобів захисту, порушення правил техніки безпеки, режимів праці і відпочинку.

Крім цього, роботодавці масово порушують вимоги статті 8 Закону України «Про охорону праці» щодо забезпеченості працівників спеціальним одягом, спеціальним взуттям та іншими засобами індивідуального захисту згідно з Порядком та типовими нормами.

4.2. Аналіз шкідливих та небезпечних факторів на робочому місці програміста

На робочому місці користувача ПК виникають небезпечні та шкідливі фактори: підвищений рівень шуму, несприятливі мікрокліматичні умови, недостатній рівень освітленості, шкідливі речовини, підвищений рівень електромагнітних випромінювань радіочастот, висока напруга електричної мережі, статична електрика та інші. Робота з ПК супроводжується також підвищеним ступенем напруженості трудового процесу. До хімічно небезпечних факторів, що постійно діють на користувача ПК, відноситься виникнення активних часток у результаті іонізації повітря при роботі комп'ютера. Біологічні шкідливі виробничі фактори в даному приміщенні відсутні. Неправильна організація робочого місця сприяє загальній і локальній напрузі м'язів шиї, тулуба, верхніх кінцівок, скривленню хребта й розвитку остеохондрозу.

Робота програміста пов’язана з значним зоровим навантаження, що вимагає забезпечення належного освітлення. Інженер-програміст працює з ЕОМ та іншим офісним обладнанням, що є джерелом небезпеки ураження електричним струмом. Трудова діяльність програміста пов’язана з постійним перебуванням в приміщенні, тому для комфортних умов праці необхідно створити належний мікроклімат в комп’ютерній лабораторії. Згідно нормативним документам [16] та [17] можна виділити такі шкідливі виробничі чинники, що діють на програміста:

- Недостатній рівень штучного освітлення.
- Мікроклімат робочої зони: температура, відносна вологості, швидкість руху повітря.
- Підвищений рівень шуму на робочому місці.
- Небезпечна напруга в електричному ланцюзі.
- Підвищений рівень вібрації.

4.2.1. Рівень штучного освітлення

Основним документом, який регламентує норми освітленості є [18]. Категорія виконуваних робіт програміста відноситься до робіт високої точності з присвоєнням розряду III в. Тому нормативне значення загального освітлення робочого приміщення повинно бути Е = 300-500 Лк.

Освітлення на робочому місці програміста повинно бути таким, щоб працівник міг без напруги зору виконувати свою роботу. Розрахунок освітленості робочого місця зводиться до вибору системи освітлення, визначенню необхідного числа світильників, їхнього типу і розміщення. Відповідно до вибраного розрядом зорових робіт допустиме значення освітленості робочої поверхні приймається Е = 400 лк.

Для місцевого освітлення робочих місць слід використовувати світильники з відбивачами, що не просвічуються. Світильники повинні розташовуватися так, щоб їх елементи, які світяться, не потрапляли в поле зору працюючих на освітленому робочому місці і на інших робочих місцях.

Місцеве освітлення робочих місць повинно бути обладнане регуляторами освітлення.

4.2.2. Мікроклімат робочої зони: температура, відносна вологості, швидкість руху повітря

Відповідно до документу [17] праця програміста за важкістю відноситься до легкої фізичної роботи категорії Іа. Основним документом, який регламентує норми мікроклімату робочої зони є [19].

Комп’ютери і офісна техніка є джерелом істотних тепловиділень, що може привести до підвищення температури і зниження відносної вогкості в приміщенні. В приміщеннях, де встановлені комп'ютери, повинні дотримуватися певні параметри мікроклімату. В санітарних нормах встановлені величини параметрів мікроклімату, що створюють комфортні умови. Значення параметрів оптимальних та допустимих параметрів

мікроклімату згідно з [19] для приміщень, та фактичних параметрів представленні в таблиці 4.1.

Таблиця 4.1 – Оптимальні та допустимі параметри мікроклімату

Період року	Параметр клімату	Значення	
		Оптимальне	Допустиме
Холодний	Температура повітря в приміщенні	21,0-23,4°С	23,5-25,4°С
	Відносна вологість	40-60%	75%
	Швидкість руху повітря	0,1м/с	до 0,1м/с
Теплий	Температура повітря в приміщенні	21,0-23,4°С	23,5-25,4°С
	Відносна вологість	40-60%	75%
	Швидкість руху повітря	0,1 м/с	0,2-0,1м/с

Для забезпечення комфортних умов використовуються як організаційні методи (раціональна організація проведення робіт залежно від пори року і доби, чергування праці і відпочинку), так і технічні засоби (вентиляція, кондиціонування повітря, опалювальна система).

4.2.3. Рівень шуму на робочому місці

Методи вимірювання шуму та допустимі рівні звукового тиску у октавних смугах частот, еквівалентні рівні звуку на робочому місці регламентовані документом [20].

Шум погіршує умови праці здійснюючи шкідливу дію на організм людини. Працюючі в умовах тривалої шумової дії випробовують дратівливість, головні болі, запаморочення, зниження пам'яті, підвищену стомлюваність, пониження апетиту, болі у вухах тощо. Такі порушення в роботі ряду органів і систем організму людини можуть викликати негативні зміни в емоційному стані людини аж до стресових ситуацій. Під впливом шуму знижується концентрація уваги, порушуються фізіологічні функції, з'являється стомленість у зв'язку з підвищеними енергетичними витратами і нервово-психічною напругою,

погіршується мовна комутація. Все це знижує працездатність людини і її продуктивність, якість і безпеку праці.

Для пониження рівня шуму необхідна додаткова звукоізоляція. У якості звукоізолюючих матеріалів, які застосовують у конструкціях перекриттів для зниження передачі структурного (ударного) звуку переважно використовують мати та плити із скляного та мінерального волокна, м'які плити з деревних стружок, картон, гуму, утеплений лінолеум, а також заміна вікон на звукоізолюючі.

4.3. Ергономіка та техніка безпеки на робочому місці

Проектування робочих місць, забезпечених відео терміналами, відноситься до числа важливих проблем ергономічного проектування в області обчислювальної техніки.

Робоче місце і взаємне розташовує всіх його елементів повинне відповідати антропометричним, фізичним і психологічним вимогам. Велике значення має також характер роботи. Зокрема, при організації робочого місця програміста повинні бути дотримані наступні основні умови: оптимальне розміщення устаткування, що входить до складу робочого місця і достатній робочий простір, що дозволяє здійснювати всі необхідні рухи і переміщення.

Ергономічними аспектами проектування відео термінальних робочих місць, зокрема, є: висота робочої поверхні, розміри простору для ніг, вимоги до того, що розташовує документів на робочому місці (наявність і розміри підставки для документів, можливість різного розміщення документів, відстань від очей користувача до екрану, документа, клавіатури тощо), характеристики робочого крісла, вимоги до поверхні робочого столу, можливість регулювання елементів робочого місця . Головними елементами робочого місця програміста є стіл і крісло. Основним робочим положенням є положення сидячи.

Робоча поза сидячи викликає мінімальне стомлення програміста. Раціональне планування робочого місця передбачає чіткий порядок і постійність розміщення предметів, засобів праці і документації. Те, що потрібне для виконання робіт частіше, розташоване в зоні легкої досяжності робочого простору.

Моторне поле – простір робочого місця, в якому можуть здійснюватися рухові дії людини. Максимальна зона досяжності рук - це частина моторного поля робочого місця, обмеженого дугами, описуваними максимально витягнутими руками при русі їх в плечовому суглобі.

Оптимальна зона - частина моторного поля робочого місця, обмеженого дугами, описуваними передпліччями при русі в ліктьових суглобах з опорою в точці ліктя і з відносно нерухомим плечем(рисунок 4.1).

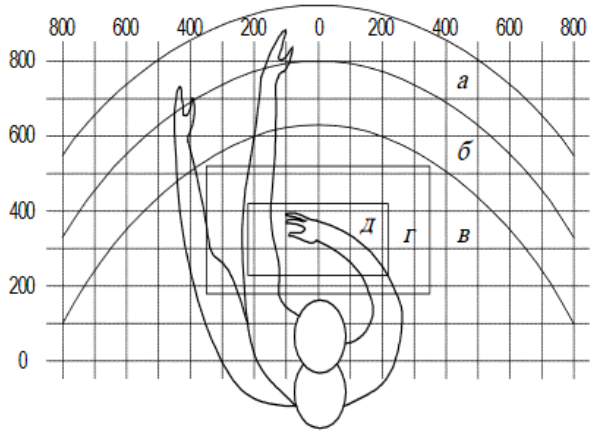


Рисунок 4.1 –Зони досяжності рук в горизонтальній площині:

а –зона максимальної досяжності;

б – зона досяжності пальців при витягнутій руці;

в – зона легкої досяжності долоні;

г – оптимальний простір для грубої ручної роботи;

д – оптимальний простір для тонкої ручної роботи.

Оптимальне розміщення предметів праці і документації в зонах досяжності:

1.Дисплей розміщується в зоні **а** (в центрі).

2.Системний блок розміщується в передбаченій ніші столу.

3.Клавіатура – в зоні **г/д.**

4.«миша» – в зоні **в** справа.

5.Сканер в зоні **а/б**(зліва).

6.Принтер знаходиться в зоні **а**(справа).

7.Документація: необхідна при роботі – в зоні легкої досяжності долоні – **в**, а у висувних ящиках столу – література, невживана постійно.

На рисунку 4.2 показаний приклад розміщення основних і периферійних складових ПК на робочому столі програміста.

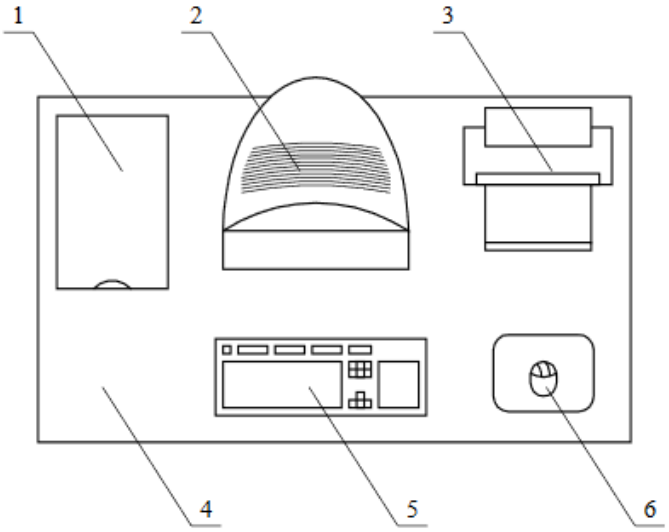


Рисунок 4.2 –Розміщення основних і периферійних складових ПК: 1 – сканер, 2 – монітор, 3 – принтер, 4 – поверхня робочого столу, 5 – клавіатура, 6 – маніпулятор типу «миша»

Для комфортної роботи стіл повинен задовольняти наступним умовам :

1. висота столу повинна бути вибрана з урахуванням можливості сидіти вільно, в зручній позі, при необхідності спираючись на підлокітники;
2. нижня частина столу повинна бути сконструйована так, щоб програміст міг зручно сидіти, не був вимушений підтискати ноги;
3. поверхня столу повинна володіти властивостями, що виключають появу відблисків в полі зору програміста;
4. конструкція столу повинна передбачати наявність висувних ящиків (не менше 3 для зберігання документації, лістингів, канцелярських обладнань);
5. висота робочої поверхні рекомендується в межах 680-760 мм. Висота поверхні, на яку встановлюється клавіатура, повинна бути біля 650 мм.

Велике значення надається характеристикам робочого крісла. Так, висота сидіння над рівнем підлоги, що рекомендується, знаходиться в межах

420-550 мм. Поверхня сидіння м'яка, передній край закруглює, а кут нахилу спинки - регульований.

Необхідно передбачати при проектуванні можливість різного розміщення документів: збоку від відео-терміналу, між монітором і клавіатурою тощо. Крім того, у випадках, коли відео-термінал має низьку якість зображення, наприклад помітні мигтіння, відстань від очей до екрану роблять більше (біля 700мм), ніж відстань від ока до документа (300-450мм). Взагалі при високій якості зображення на відео-терміналі відстань від очей користувача до екрану, документа і клавіатури може бути рівним.

Положення екрану визначається:

1. Відстанню прочитування (0,6 - 0,7 м).

2. Кутом прочитування, напрямом погляду на 20°нижче горизонталі до центру екрану, причому екран перпендикулярний цьому напряму.

Повинна також передбачатися можливість регулювання екрану:

1.По висоті +3 см.

2.По нахилу від -10°до +20°щодо вертикалі.

3.Влівому і правом напрямах.

Велике значення також надається правильній робочій позі користувача. При незручній робочій позі можуть з'явитися болі в м'язах, суглобах і сухожиллях.

Вимоги до робочої пози користувача відео терміналу наступні:

1.Голова не повинна бути нахилена більш ніж на 20°.

2.Плечі повинні бути розслаблені.

3.Лікті – під кутом 80° - 100°.

4.Передпліччя і долоні рук – в горизонтальному положенні.

Причина неправильної пози користувачів обумовлена наступними чинниками: немає хорошої підставки для документів, клавіатура знаходиться

дуже високо, а документи - низько, нікуди покласти руки і кисті, недостатній простір для ніг.

В цілях подолання вказаних недоліків даються загальні рекомендації: краще пересувна клавіатура; повинні бути передбачені спеціальні пристосування для регулювання висоти столу, клавіатури і екрану, а також підставка для рук.

Істотне значення для продуктивної і якісної роботи на комп'ютері мають розміри знаків, густину їх розміщення, контраст і співвідношення яскравості символів і фону екрану. Якщо відстань від очей оператора до екрану дисплея складає 60 - 80 см, то висота знаку повинна бути не менше 3мм , оптимальне співвідношення ширини і висоти знаку складає 3:4, а відстань між знаками – 15.20% їх висоти. Співвідношення яскравості фону екрану і символів – від 1:2 до 1:15.

Під час користування комп'ютером медики радять встановлювати монітор на відстані 50-60 см від очей. Фахівці також вважають, що верхня частина відео-дисплея повинна бути на рівні очей або трохи нижче. Коли людина дивиться прямо перед собою, його очі відкриваються ширше, ніж коли він дивиться вниз. За рахунок цього площа огляду значно збільшується, викликаючи обезводнення очей. До того ж якщо екран встановлений високо, а очі широко відкриті, порушується функція моргання. Це значить, що очі не закриваються повністю, не слізною рідиною, не одержують достатнього зволоження, що приводить до їх швидкої стомлюваності.

Створення сприятливих умов праці і правильне естетичне оформлення робочих місць на виробництві має велике значення як для полегшення праці, так і для підвищення її привабливості, що позитивно впливає на продуктивність праці.

Висновок

В роботі була проведена розробка інструментального засобу для прогнозування обсягу ПЗ створюваного мовою Java на основі UML метрик.

В роботі були виконані:

- Аналіз існуючих технологій та методологій для прогнозування обсягу ПЗ;
- Огляд характеристик ПЗ які можна отримати на етапі проектування;
- Огляд математичного апарату для прогнозування залежного параметру від змінної кількості характеристик;
- Аналіз основних інструментів, технологій, бібліотек та фреймворків для розробки ІС;
- Розроблено інструментальний засіб для прогнозування обсягу ПЗ створеного за допомогою мови Java на основі UML метрик. Для чого була проаналізована предметна область, сформовано вимоги до системи, розроблено концепцію системи, розроблено технічне завдання на створення системи, розроблено загальносистемні рішення, рішення з інформаційного, технічного та програмного забезпечення системи та розроблено робочу документацію.

Під час прийняття загальносистемних рішень була визначена архітектура системи, її дерево функцій, а також вирішені питання її взаємодії з іншими системами. В питанні інформаційного забезпечення була визначена модель даних системи та прийнято рішення про використання реляційної бази даних. Рішення з технічного забезпечення склалися зі схеми автоматизації системи та проробки необхідного комплексу технічних засобів.

На етапі прийняття рішень, що стосувалися програмного забезпечення, було обрано методологію проведення розробки, визначено структуру ПЗ та функції його частин.

Список використаних джерел

1. Cay S. Horstmann . «Core Java, Volume I- Fundamentals (10th Edition)» / Cay S. Horstmann. – К. : «Prentice Hall», 2016.
2. Cay S. Horstmann . «Core Java, Volume II-Advanced Features (10th Edition)» / Cay S. Horstmann. – К. : «Prentice Hall», 2016.
3. Cosmina, I., Harrop, R., Schaefer, C., Ho, C. «Pro Spring 5» / Cosmina, I., Harrop, R., Schaefer, C., Ho, C – «Apress», 2017.
4. «Методи економічного прогнозування» www.info-library.com.ua [Електронній ресурс]: www.info-library.com.ua. Режим доступу: http://www.info-library.com.ua/books-text-5951.html - Дата доступу: 01.04.19.
5. «Вікіпедія, вільна енциклопедія» uk.wikipedia.org/wiki [Електронній ресурс]: uk.wikipedia.org/wiki. Режим доступу: https://uk.wikipedia.org/wiki - Дата доступу: 01.04.19.
6. «Assessing UML Model Quality by Utilizing Metrics» db.informatik.uni-bremen.de [Електронній ресурс]: db.informatik.uni-bremen.de. Режим доступу: http://www.db.informatik.uni-bremen.de/publications/Hoang_2018_QUATIC.pdf – Дата доступу: 03.04.19.
7. «Методи прогнозування» wiki.tntu.edu.ua [Електронній ресурс]: wiki.tntu.edu.ua. Режим доступу: https://wiki.tntu.edu.ua/Методи_прогнозування – Дата доступу: 03.04.19.
8. «CK - Code metrics for Java code» github.com [Електронній ресурс]: github.com. Режим доступу: https://github.com/mauricioaniche/ck – Дата доступу: 06.04.19.
9. «В.В. Домбровский – Эконометрика, Множественная линейная регрессия» sun.tsu.ru [Електронній ресурс]: sun.tsu.ru. Режим доступу: http://sun.tsu.ru/mminfo/2016/Dombrovski/book/chapter-3/chapter-3.htm – Дата доступу: 02.06.19.

10. «The Apache Commons Mathematics Library» Commons Math. [Електронній ресурс]: Commons Math. Режим доступу: http://commons.apache.org/proper/commons-math/ – Дата доступу: 17.06.19.
11. «Spring Core Technologies» Spring.io. [Електронній ресурс]: Spring.io. Режим доступу: https://docs.spring.io/spring/docs/5.2.5.RELEASE/spring-framework-reference/core.html – Дата доступу: 10.06.19.
12. «Spring Boot Reference Guide» Spring.io. [Електронній ресурс]: Spring.io. Режим доступу: https://docs.spring.io/spring-boot/docs/2.2.6.RELEASE/reference/html/using-spring-boot.html – Дата доступу: 10.07.19.
13. «Spring Web MVC» Spring.io. [Електронній ресурс]: Spring.io. Режим доступу: https://docs.spring.io/spring/docs/5.2.5.RELEASE/spring-framework-reference/web.html. – Дата доступу: 02.09.19.
14. «Vue.js getting started» vuejs.org [Електронній ресурс]: vuejs.org. Режим доступу: https://vuejs.org/v2/guide/ – Дата доступу: 14.09.19.
15. «Chart.js getting started» chartjs.org [Електронній ресурс]: chartjs.org Режим доступу: https://www.chartjs.org/docs/latest/ – Дата доступу: 20.11.19.
16. ССБТ «Небезпечні і шкідливі виробничі фактори. Класифікація». ДСТУ 12.0.003-74*. [Електронний ресурс]: ДСТУ 12.0.003-74*. – М., 1980 – Режим доступу: http://online.budstandart.com/ua/catalog/doc-page.html?id_doc=48127. – Дата доступу: 21.05.2020.
17. «Гігієнічна класифікація праці за показниками шкідливості та небезпечності факторів виробничого середовища». [Електронний ресурс] zakon.rada.gov.ua/laws : –Режим доступу: https://zakon.rada.gov.ua/laws/show/z0472-14. –Дата доступу : 21.05.2020.
18. «Державні будівельні норми України. Природне та штучне освітлення». ДБН В .2.5-28:2018. [Електронний ресурс]: ДБН В .2.5-28:2018. – Режим доступу: https://ledeffect.com.ua/images/__branding/dbn2018.pdf . – Дата доступу: 22.05.2020.

19.«Державні санітарні норми мікроклімату виробничих приміщень». ДСН 3.3.6.042-99. [Електронний ресурс]: ДСН 3.3.6.042-99. – Режим доступу: http://mozdocs.kiev.ua/view.php?id=1972 . – Дата доступу: 22.05.2020.

20.«Санітарні норми виробничого шуму, ультразвуку та інфразвуку». ДСН3.3.6.037-99. [Електронний ресурс]: ДСН 3.3.6.037-99. – Режим доступу: http://normativ.com.ua/types/tdoc4878.php – Дата доступу : 23.05.20

					ДР.122.4142.05.ПЗ		
Зм.	Аркуш	№ документа	Підпис	Дата			
					Додатки	Літ.	Аркуш 1 / Аркушів
Студент		Каратай М.В.					
Керівник		Беркунський Є.Ю.				НУК ім. адм. Макарова	
Консульт.							
Зав. Каф.		Михелєв І.Л.					

Додаток А – Технічне завдання на розробку

1. Вступ

1.1. Назва - Проєкт прогнозування обсягу програмного забезпечення, створюваного мовою Java на основі UML метрик.

1.2. Призначення програми – надати можливість користувачам прогнозувати обсяг програмного забезпечення яке планується створювати за допомогою мови програмування Java ще на етапі проєктування UML діаграм.

1.3. Область застосування – Сегмент розробки програмного забезпечення з використанням мови програмування Java.

2. Призначення розробки

Система повинна надавати змогу прогнозувати обсяг програмного забезпечення у тисячах рядків коду. Це важливий показник який надає змогу у подальшому більш точно розрахувати необхідні ресурси для створення кінцевого продукту, наприклад за допомогою моделі розрахунку «COCOMO II», який має три рівні точності розрахунку вартості ПЗ. Незалежно від обраного рівня, для функціонування, система потребує числове значення обсягу ПЗ, яке зазначається у тисячах рядків коду.

3. Вимоги до системи

3.1 Вимоги до функціональних характеристик

Система буде містити наступні функціональні характеристики:

- Створення моделі прогнозування, вибір параметрів для моделі, таких як: тип аналізу, набору проєктів на основі яких буде створена модель та параметри які будуть брати участь у розрахунку.
- Створення прогнозу обсягу програмного забезпечення для користувача, відображення результатів для моделі прогнозування, як у вигляді таблиць так і за допомогою графіків.
- Можливість завантажити результат розрахунку у вигляді CSV файлів.
- Формування унікальних наборів проєктів для розрахунку моделі прогнозування;

- Створення та редагування характеристик проєктів які входять у набори;
- Можливість перегляду наборів проєктів у вигляді графіків та таблиць;

3.2. Вимоги до надійності

3.2.1. Вимоги до забезпечення надійності функціонування програми

Надійне (стійке) функціонування системи має бути забезпечено надійним (стійким) функціонуванням бездротової мережі і мережі електропостачання в місці, де буде розміщений сервер.

3.2.2. Відмова виконання програмного продукту через некоректні дії користувача.

- Перевірка формату вводу даних для коректної роботи.
- Функція автоматичного backup кожен тиждень.

3.3. Умови експлуатації

Данною програмою будуть користуватися працівники, які є впевненими користувачами, та не потребують додаткового навчання по даному ПО, але можуть вимагати додаткову допомогу у роботі з вихідними даними. Окрім того буде надана інструкція з використання програми.

3.4. Вимоги до інформаційної та програмної сумісності

3.4.1. Вимоги до інформаційних структур і методів розв'язання

Вимоги до інформаційних структур і методів розв'язання відсутні.

3.4.2. Вимоги до вихідних кодів і мов програмування

Додаткові вимоги не пред'являються.

3.4.3. Вимоги до програмних засобів , які використовуються програмою

Програмні засоби що використовуються мають бути вільно поширювані

3.5. Вимоги до маркування та упаковки

3.5.1. Вимоги до маркування :

- Читабельність тексту, чіткість ілюстрацій;

- Достовірність інформації про систему

3.5.2. Вимоги до упаковки:

Вимоги до упаковки відсутні.

3.6. Вимоги до транспортування та зберігання

Вимоги до транспортування програми відсутня.

3.7. Спеціальні вимоги

Спеціальні вимоги відсутні.

4. Вимоги до програмної документації

4.1. Попередній склад програмної документації

Попередній склад програмної документації повинен включати в себе:

- технічне завдання;

- устав проекту.

4.2. Склад програмної документації

Програмна документація складається з наступних програмних документів:

- Устав проекту;

- Технічне завдання ;

- Опис системи ;

- Код програми ;

- Керівництво користувача.

Сертифікація містить список всіх програмних документів .

Технічне завдання містить опис завдання роботи, склад функціональних характеристик програми, опис вимог для розробки.

Опис програми містить опис логічної структури програми, форматів вхідних та вихідних даних, використовуваних технічних засобів.

Код програми містить опис блоків програми і запис блоків в програмному коді.

Керівництво користувача містить інформацію , необхідну для ознайомлення користувача з програмним продуктом.

5. Техніко-економічні показники

5.1. Економічні переваги розробки

Орієнтовна економічна ефективність не розраховується.

6. Стадії та етапи розробки програмного забезпечення

6.1. Стадії розробки

Розробка повинна бути проведена в 5 стадій:

1. Технічне завдання
2. Ескізний проект
3. Технічний проект
4. Робочий проект
5. Впровадження.

Додаток Б – Проєкти які входять до бази знань

Таблиця 1 – проєкти для бази знань.

Name	Summary	Version/Access date	Link
Activiti	Activiti is a light-weight workflow and Business Process Management (BPM) Platform targeted at business people, developers and system admins. Activiti runs in any Java application, on a server, on a cluster or in the cloud.	7.1.0-SNAPSHOT / 26.03.2019	https://github.com/Activiti/Activiti
Apache Dubbo	Apache Dubbo is a high-performance, java based open source RPC framework.	2.7.1-SNAPSHOT/ 24.03.2019	https://github.com/apache/incubator-dubbo
AsyncHttpClient	The AsyncHttpClient (AHC) library allows Java applications to easily execute HTTP requests and asynchronously process HTTP responses. The library also supports the WebSocket Protocol.	2.8.2-SNAPSHOT/ 24.03.2019	https://github.com/AsyncHttpClient/async-http-client
Athou commafeed	Google Reader inspired self-hosted RSS reader, based on Dropwizard and AngularJS. CommaFeed is now considered feature-complete and is in maintenance mode.	2.5.0-SNAPSHOT / 25.03.2019	https://github.com/Athou/commafeed
Atmosphere	Atmosphere - Realtime Client Server Framework for the JVM, supporting WebSockets with Cross-Browser Fallbacks	3.0.0-SNAPSHOT / 25.03.2019	https://github.com/Atmosphere/atmosphere

BaasBox	BaasBox is an Open Source project that aims to provide a backend for mobile and web apps.	0.9.5-snapshot / 25.03.2019	https://github.com/baasbox/baasbox
Blade	Blade a simple, elegant java web framework!	2.0.14.RELEASE / 25.03.2019	https://github.com/lets-blade/blade
Comsat	Integrates standard Java web-related APIs with Quasar fibers and actors.	0.7.1-SNAPSHOT / 25.03.2019	https://github.com/puniverse/comsat
Crawler4j	Simple and lightweight web crawler	4.5.0-SNAPSHOT/ 25.03.2019	https://github.com/yasserg/crawler4j
CSSEmbed	CSSEmbed is a small program/library to automate embedding of data URIs in CSS files.	0.4.5/ 25.03.2019	https://github.com/nzakas/cssembed
CUBA Platform	High-level framework for developing enterprise applications with a rich web interface, based on Spring, EclipseLink and Vaadin	7.1-SNAPSHOT/ 25.03.2019	https://github.com/cuba-platform/cuba
Dropwizard	Opinionated framework for setting up modern web applications with Jetty, Jackson, Jersey and Metrics.	2.0.0-rc1-SNAPSHOT/ 25.03.2019	https://github.com/dropwizard/dropwizard
Eureka	REST-based service registry for resilient load balancing and failover.	1.9.9/ 25.03.2019	https://github.com/Netflix/eureka
Finagle	Extensible RPC system for constructing high-concurrency servers. It implements uniform client and server APIs for several protocols, and is protocol-agnostic to simplify	19.2.0/ 25.03.2019	https://github.com/twitter/finagle

	implementation of new protocols.		
Gargl	Record web requests as they happen and turn them into reusable code in any programming language.	1.0-SNAPSHOT/ 25.03.2019	https://github.com/jodoglevy/gargl
Generator jhipster	Open Source application generator for creating Spring Boot + Angular/React projects in seconds!	5.8.2/ 25.03.2019	https://github.com/jhipster/generator-jhipster
Gwt-bootstrap	A GWT Library that provides the widgets of Bootstrap, from Twitter	2.3.2.1-SNAPSHOT/ 25.03.2019	https://github.com/gwtbootstrap/gwt-bootstrap
Handlebars.java	Handlebars provides the power necessary to let you build semantic templates effectively with no frustration.	4.1.3-SNAPSHOT/ 25.03.2019	https://github.com/jknack/handlebars.java
HBC	A Java HTTP client for consuming Twitter's realtime Streaming API	2.2.1-SNAPSHOT/ 25.03.2019	https://github.com/twitter/hbc
Http kit	http-kit is a minimalist, event-driven, high-performance Clojure HTTP server/client library with WebSocket and asynchronous support	2.4.0-alpha3/ 25.03.2019	https://github.com/http-kit/http-kit
Http request	A simple convenience library for using a HttpURLConnection to make requests and access the response.	1.0/ 25.03.2019	https://github.com/kevinsawicki/http-request
Ice	Ice provides a birds-eye view of our large and complex cloud landscape from a usage and cost perspective.	1.1.0/ 25.03.2019	https://github.com/Teevity/ice

Ja-micro	Lightweight Java framework for building microservices	3.2.42/ 25.03.2019	https://github.com/Sixt/ja-micro
Jasig CAS (Central Authentication Service)	CAS is an enterprise multilingual single sign-on solution for the web and attempts to be a comprehensive platform for your authentication and authorization needs.	6.1.0-RC3-SNAPSHOT/ 26.03.2019	https://github.com/apereo/cas
Jclouds	Apache jclouds is an open source multi-cloud toolkit for the Java platform that gives you the freedom to create applications that are portable across clouds while giving you full control to use cloud-specific features.	2.2.0-SNAPSHOT/ 26.03.2019	https://github.com/jclouds/jclouds
KryoNet	Provides a clean and simple API for efficient TCP and UDP client/server network communication using NIO and Kryo.	2.22.0-RC1/ 26.03.2019	https://github.com/EsotericSoftware/kryonet
Mashape Unirest java	Unirest for Java	v3.3.00/ 26.03.2019	https://github.com/Kong/unirest-java
Micronaut	Modern full-stack framework with focus on modularity, minimal memory footprint and startup time.	1.1.0.BUILD-SNAPSHOT/ 26.03.2019	https://github.com/micronaut-projects/micronaut-core
Moco	Moco is an easy setup stub framework.	0.12.1-SNAPSHOT/ 26.03.2019	https://github.com/dreamhead/moco
MSF4J	High throughput & low memory footprint Java microservices framework.	2.6.5-SNAPSHOT/ 26.03.2019	https://github.com/wso2/msf4j

Mustache.java	Implementation of mustache.js for Java	0.9.7-SNAPSHOT/ 26.03.2019	https://github.com/spullara/mustache.java
Nanohttpd	*NanoHTTPD* is a light-weight HTTP server designed for embedding in other applications, released under a Modified BSD licence.	2.3.2-SNAPSHOT/ 26.03.2019	https://github.com/NanoHttpd/nanohttpd
Netflix Ribbon	Ribbon is a client side IPC library that is battle-tested in cloud. It provides the following features	6.4.3-SNAPSHOT/ 26.03.2019	https://github.com/Netflix/ribbon
Ninja	Ninja is a full stack web framework for Java. Rock solid, fast and super productive.	4.0.0-SNAPSHOT/ 26.03.2019	https://github.com/ninjaframework/ninja
Orbit	Orbit allows developers to write highly distributed and scalable applications while greatly simplifying clustering, discovery, networking, state management, actor lifetime and more.	1.12.1-SNAPSHOT/ 26.03.2019	https://github.com/orbit/orbit
Parboiled	Elegant parsing in Java and Scala - lightweight, easy-to-use, powerful.	1.3.0/ 26.03.2019	https://github.com/sirthias/parboiled
Pippo	It's an open source micro web framework in Java, with minimal dependencies and a quick learning curve.The goal of this project is to create a micro web framework in Java	1.13.0-SNAPSHOT/ 26.03.2019	https://github.com/pippo-java/pippo
Play Framework	The Play Framework combines productivity and performance making it easy to build scalable web applications with Java and Scala	2.8.0-SNAPSHOT/ 26.03.2019	https://github.com/playframework/playframework

QBit	Reactive programming library for building microservices.	2.0.1-SNAPSHOT/ 26.03.2019	https://github.com/advantageous/qbit
Ratpack	Ratpack is a simple, capable, toolkit for creating high performance web applications. Ratpack is built on Java and the Netty event-driven networking engine.	1.7.0-SNAPSHOT/ 26.03.2019	https://github.com/ratpack/ratpack
RXjava-http-tail	A RxJava-based port of Net::HTTP::FollowTail to the JVM-land.	0.1.2/ 26.03.2019	https://github.com/myfreeweb/rxjava-http-tail
ScribeJava	ScribeJava, the simple OAuth client Java lib	6.5.2-SNAPSHOT/ 26.03.2019	https://github.com/scribejava/scribejava
Spark	Spark - a Sinatra inspired web framework.	2.8.0/ 26.03.2019	https://github.com/perwendel/spark
Spring Boot	Spring Boot makes it easy to create Spring-powered, production-grade applications and services with absolute minimum fuss.	2.2.0.BUILD-SNAPSHOT/ 26.03.2019	https://github.com/spring-projects/spring-boot
Spring Integration	Extends the Spring programming model to support the well-known Enterprise Integration Patterns.	5.2.0.BUILD-SNAPSHOT/ 26.03.2019	https://github.com/spring-projects/spring-integration
Spring Roo	Spring Roo is an easy-to-use development tool for quickly building Spring-powered applications	2.0.0.RELEASE/ 26.03.2019	https://github.com/spring-projects/spring-roo
Spring security	Spring Security provides security services for the Spring IO Platform. Spring Security 5.0 requires Spring 5.0 as a minimum and also requires Java 8.	5.2.0.BUILD-SNAPSHOT/ 26.03.2019	https://github.com/spring-projects/spring-security

Spring Session	Spring Session provides an API and implementations for managing a user's session information, while also making it trivial to support clustered sessions without being tied to an application container specific solution.	2.2.0.BUILD-SNAPSHOT/ 26.03.2019	https://github.com/spring-projects/spring-session
Spring Web Flow	Spring Web Flow facilitates building web applications that require guided navigation -- e.g. a shopping cart, flight check-in, a loan application, and many others.	2.5.2.BUILD-SNAPSHOT/ 26.03.2019	https://github.com/spring-projects/spring-webflow
Spring Web Services	Spring Web Services is a product of the Spring community focused on creating document-driven Web services.	3.0.8.BUILD-SNAPSHOT/ 26.03.2019	https://github.com/spring-projects/spring-ws
Square Okhttp	An HTTP & HTTP/2 client for Android and Java applications.	4.0.0-SNAPSHOT/ 26.03.2019	https://github.com/square/okhttp
Sshj	ssh, scp and sftp for java	0.27.0/ 26.03.2019	https://github.com/hierynomus/sshj
Swagger	Swagger is a specification and complete framework implementation for describing, producing, consuming, and visualizing RESTful web services.	2.0.8-SNAPSHOT/ 26.03.2019	https://github.com/swagger-api/swagger-core
Takes	Opinionated web framework which is built around the concepts of True Object-Oriented Programming and immutability.	2.0-SNAPSHOT/ 26.03.2019	https://github.com/yegor256/takes
Twitter common	Twitter common libraries for python and the jvm	1.3.0rc1/ 26.03.2019	https://github.com/twitter/commons

Twitter4J	Twitter4J is a Twitter API binding library for the Java language licensed under Apache License 2.0.	4.0.8-SNAPSHOT/ 26.03.2019	https://github.com/Twitter4J/Twitter4J
Vertx-Web	Vert.x-Web is a set of building blocks for building web applications with Vert.x, a toolkit for building reactive applications on the JVM.	4.0.0-SNAPSHOT/ 26.03.2019	https://github.com/vert-x3/vertx-web
VRaptor	VRaptor 4 delivers high productivity to your Java Web applications on top of CDI. VRaptor is an opensource MVC framework with a large developers and users community	4.3.0-beta-4-SNAPSHOT/ 26.03.2019	https://github.com/caelum/vraptor4
Webbit	A Java event based WebSocket and HTTP server	0.4.16-SNAPSHOT/ 26.03.2019	https://github.com/webbit/webbit
Webmagic	A scalable web crawler framework.	0.7.3/ 26.03.2019	https://github.com/code4craft/webmagic
zipkin	Zipkin is a distributed tracing system. It helps gather timing data needed to troubleshoot latency problems in microservice architectures.	2.12.9-SNAPSHOT/ 26.03.2019	https://github.com/openzipkin/zipkin
ZK	ZK is a highly productive Java framework for building amazing enterprise web and mobile applications.	8.6.2.1/ 26.03.2019	https://github.com/zkoss/zk

Додаток В - Тексти програмних модулів системи

Оскільки система складається з багатьох модулів та підсистем, нижче наведено тексти деяких (не усіх) програмних модулів.

Лістинг 1 – DefaultStatisticAnalyzer.java

```
package com.statistic.logic;

import com.statistic.domain.AnalysisData;
import com.statistic.domain.AnalysisHeader;
import org.apache.commons.math3.distribution.ChiSquaredDistribution;
import org.apache.commons.math3.distribution.FDistribution;
import org.apache.commons.math3.distribution.TDistribution;
import org.apache.commons.math3.linear.LUDecomposition;
import org.apache.commons.math3.linear.MatrixUtils;
import org.apache.commons.math3.linear.RealMatrix;
import
org.apache.commons.math3.stat.regression.OLSMultipleLinearRegression;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collection;
import java.util.Collections;
import java.util.List;
import java.util.stream.Collectors;

public abstract class DefaultStatisticAnalyzer implements Analyzer {

    public    double    calculateCoefficientOfDetermination(AnalysisHeader
header) {
        List<AnalysisData> dataList = header.getDataList();
        double sSRes = calculateSSRes(dataList);
        double  sSTot  =  calculateSSTot(dataList,  header.getAverage()[0]);
// due to first el. is y
        int size = dataList.size();
        sSRes /= (size - header.getN());
        sSTot /= (size - 1);
        return 1.0 - (sSRes / sSTot);
    }

    /**
     * calculate total sum of squares
     */
    protected  double  calculateSSTot(Collection<AnalysisData>  projects,
double meanKLOC) {
        double sSTot = 0;
        for (AnalysisData project : projects) {
            double y = project.getY();
            sSTot += Math.pow(y - meanKLOC, 2);
        }
        return sSTot;
    }

    /**
     * calculate rests sum of square
     */
    protected   abstract   double   calculateSSRes(Collection<AnalysisData>
projects);

    protected    double    calculateAnalysisMre(Collection<AnalysisData>
analysisData) {
```

```
            return
analysisData.stream().mapToDouble(AnalysisData::getMre).average().getAsDouble()
;
        }

        @Override
        public void regresionCalculation(AnalysisHeader header) {
            List<AnalysisData> data = header.getDataList();
            double student = header.gettStud();
            double Sy = header.getSy();
            for (AnalysisData ad : data) {
                ad.setConfidenceLB(ad.getyLinear() - student * Math.sqrt(Sy) *
Math.sqrt((double) 1 / data.size() + ad.getR()));
                ad.setConfidenceUB(ad.getyLinear() + student * Math.sqrt(Sy) *
Math.sqrt((double) 1 / data.size() + ad.getR()));
                ad.setPredictionLB(ad.getyLinear() - student * Math.sqrt(Sy) *
Math.sqrt(1 + (double) 1 / data.size() + ad.getR()));
                ad.setPredictionUB(ad.getyLinear() + student * Math.sqrt(Sy) *
Math.sqrt(1 + (double) 1 / data.size() + ad.getR()));
            }
        }

        @Override
        public void kovarCalculation(double[][] m, AnalysisHeader header) {
            int n = header.getN();
            List<AnalysisData> dataList = header.getDataList();

            for (AnalysisData ad : dataList) {
                double[] diff = ad.getDiff();
                double[] regElements = new double[n];
                double reg = 0;

                for (int i = 0; i < n; i++) {
                    for (int j = 0; j < n; j++) {
                        regElements[i] += m[j][i] * diff[j+1] ;
                    }
                    reg += regElements[i] * diff[i+1];
                }

                ad.setRegressionElements(regElements);
                ad.setR(reg);
            }
        }

        @Override
        public double[][] kovarMatrix(AnalysisHeader header) {
            int n = header.getN();
            double[][] m = new double[n][n];
            List<Double> tmp = new ArrayList<>();
            List<double[]>                          diff2                          =
header.getDataList().stream().map(AnalysisData::getDiff2).collect(Collectors.to
List());
            List<double[]>                       elements                       =
header.getDataList().stream().map(AnalysisData::getMatrixElements).collect(Coll
ectors.toList());

            for (int i = 0; i < n; i++) {
                for (double[] d : diff2) {
                    tmp.add(d[i+1]); // we need only x members
                }
                m[i][i] = tmp.stream().mapToDouble(e -> e).sum();
                tmp.clear();
            }

            int k = n;
```

```
                for (int i = 0; i < n; i++) {
                    for (int j = i+1; j < n; j++) {
                        for (double[] d : elements) {
                            tmp.add(d[k]);
                        }
                        m[i][j]   =   m[j][i]   =   tmp.stream().mapToDouble(e   ->
e).sum();
                        tmp.clear();
                        k++;
                    }
                }
                return m;
            }

            @Override
            public void additionalStatistic(double kvantil, AnalysisHeader header)
{
                List<AnalysisData> dataList = header.getDataList();

header.setSy(dataList.stream().mapToDouble(AnalysisData::getyLinearDiff2)
                        .sum() / (dataList.size() - header.getN() - 1));
                header.settStud(new TDistribution(dataList.size() - (header.getN()
+ 1))
                        .inverseCumulativeProbability(1 - 0.05 / 2));          //
two-tailed inverse of the Student's t-distribution
            }

            @Override
            public void functionCalculation(double[] b, AnalysisHeader header) {
                int n = header.getN();
                List<AnalysisData> dataList = header.getDataList();
                for (AnalysisData ad : dataList) {
                    double[] x = ad.getX();
                    double yFun = b[0];
                    for (int i = 0; i < n; i++) {
                        yFun += x[i] * b[i+1];
                    }
                    ad.setyLinear(yFun);
                    ad.setyLinearDiff2((ad.getY() - yFun) * (ad.getY() - yFun));
                }
            }

            @Override
            public double[] mnk(AnalysisHeader header) {
                int n = header.getN();
                List<AnalysisData> dataList = header.getDataList();
                double[]                                y                                =
dataList.stream().mapToDouble(AnalysisData::getY).toArray();
                double[][] x = new double[dataList.size()][];
                for (int i = 0; i < dataList.size(); i++) {
                    x[i] = Arrays.copyOf(dataList.get(i).getX(), n);
                }

                OLSMultipleLinearRegression      linearRegression      =      new
OLSMultipleLinearRegression();
                linearRegression.newSampleData(y, x);
                return linearRegression.estimateRegressionParameters();
            }

            @Override
            public void calculateDistancesByStatisticTests(AnalysisHeader header)
{
                List<AnalysisData> dataList = header.getDataList();
                for (AnalysisData ad : dataList) {
                    double diMalohob = ad.getDiMalohob();
```

```
                ad.setChi2Test( diMalohob > header.getChi2() );
                ad.setDi2(diMalohob * diMalohob);
                ad.setTS(diMalohob * header.getCoefficient());
                ad.setFisherTest(ad.getTS() > header.getFisher());
            }

header.setBeta2(dataList.stream().mapToDouble(AnalysisData::getDi2).sum()     /
dataList.size());
        }

        @Override
        public void statisticByChi2AndFisherTest(double alpha, AnalysisHeader
header) {
            int n = header.getN();
            double size = header.getDataList().size();
            header.setChi2(new           ChiSquaredDistribution(n           +
1).inverseCumulativeProbability(1 - alpha));
            header.setFisher(new FDistribution(n + 1, size - (n+1)) // TODO
check degree of freedom(e.g. -1, 0)
                    .inverseCumulativeProbability(1 - alpha));
            header.setCoefficient((size - (n + 1)) * size / (size * size - 1)
/ (n+1));
        }

        @Override
        public void distanceCalculation(double[][] m, AnalysisHeader header) {
            int n = header.getN();
            List<AnalysisData> data = header.getDataList();
            for (AnalysisData ad : data) {
                double[] diff = ad.getDiff();
                double[] disElements = new double[n+1];
                double disMal = 0;

                for (int i = 0; i <= n; i++) {   // calculate distance elements
                    for (int j = 0; j <= n; j++) {
                        disElements[i] += m[j][i] * diff[j];
                    }
                    disMal += disElements[i] * diff[i];
                }
                ad.setDistanceCheckElements(disElements);
                ad.setDiMalohob(disMal);
            }
        }

        @Override
        public double[][] getInverseMatrix(double[][] m) {
            RealMatrix               inverse               =               new
LUDecomposition(MatrixUtils.createRealMatrix(m)).getSolver().getInverse();
return inverse.getData();
        }

        @Override
        public double[][] throwingMatrix(AnalysisHeader header) {
            int n = header.getN();
            int size = header.getDataList().size();
            List<double[]>                       diff2                       =
header.getDataList().stream().map(AnalysisData::getDiff2).collect(Collectors.to
List());
            List<double[]>                     mElements                     =
header.getDataList().stream().map(AnalysisData::getMatrixElements).collect(Coll
ectors.toList());
            List<Double> tmp = new ArrayList<>();
            double[][] m = new double[n+1][n+1];

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

```
                for (double[] d : diff2) {
                    tmp.add(d[i]);
                }
                m[i][i] = tmp.stream().mapToDouble(e -> e).sum() / size;
                tmp.clear();
            }

            int k = 0;
            for (int i = 0; i <= n; i++) {
                for (int j = i+1; j <= n; j++) {
                    for (double[] d : mElements) {
                        tmp.add(d[k]);
                    }
                    m[i][j] = m[j][i] = tmp.stream().mapToDouble(e -> e).sum()
/ size ;
                    k++;
                    tmp.clear();
                }
            }
            return m;
        }

        @Override
        public void calculateMatrixElements(AnalysisHeader header) {
            int n = header.getN();
            int size = ( (n + 1) * (n + 1) - (n + 1) ) / 2;     // number of
elements that need to be calculated
            for (AnalysisData ad : header.getDataList()) {
                double[] diff = ad.getDiff();
                double[] matrixElements = new double[size];
                int k = 0;
                for (int i = 0; i <= n; i++) {
                    for (int j = i+1; j <= n; j++) {
                        matrixElements[k] = diff[i] * diff[j];
                        k++;
                    }
                }
                ad.setMatrixElements(matrixElements);
            }
        }

        @Override
        public void baseHeaderCalculation(AnalysisHeader analysisHeader) {
            List<AnalysisData> dataList = analysisHeader.getDataList();
            List<double[]>                          x                          =
dataList.stream().map(AnalysisData::getX).collect(Collectors.toList());
            List<Double> tmp = new ArrayList<>();
            int n = analysisHeader.getN();
            double[] min = analysisHeader.getMin();
            double[] max = analysisHeader.getMax();
            double[] average = analysisHeader.getAverage();

            min[0]                                                              =
dataList.stream().mapToDouble(AnalysisData::getY).min().getAsDouble();
            for (int i = 1; i <= n; i++) {
                for (double[] d : x) {
                    tmp.add(d[i - 1]);
                }
                min[i] = Collections.min(tmp);
                tmp.clear();
            }

            max[0]                                                              =
dataList.stream().mapToDouble(AnalysisData::getY).max().getAsDouble();
            for (int i = 1; i <= n; i++) {
```

```
                for (double[] d : x) {
                    tmp.add(d[i - 1]);
                }
                max[i] = Collections.max(tmp);
                tmp.clear();
            }

            average[0]                                                    =
dataList.stream().mapToDouble(AnalysisData::getY).average().getAsDouble();
            for (int i = 1; i <= n; i++) {
                for (double[] d : x) {
                    tmp.add(d[i - 1]);
                }
                average[i]       =       tmp.stream().mapToDouble(e       ->
e).average().getAsDouble();
                tmp.clear();
            }

        }

        @Override
        public void statisticHeaderCalculation(AnalysisHeader header) {
            int n = header.getN();
            double         num         =         header.getDataList().size();
// less casts in calculation
            double[] dispersion = header.getDispersion();
            double[] medianDeviation = header.getMedianDeviation();
            double[] asymmetry = header.getAsymmetry();
            double[] excess = header.getExcess();

            List<double[]>                          diff2                     =
header.getDataList().stream().map(AnalysisData::getDiff2).collect(Collectors.to
List());
            List<double[]>                          diff3                     =
header.getDataList().stream().map(AnalysisData::getDiff3).collect(Collectors.to
List());
            List<double[]>                          diff4                     =
header.getDataList().stream().map(AnalysisData::getDiff4).collect(Collectors.to
List());
            List<Double> tmp = new ArrayList<>();

            for (int i = 0; i <= n; i++) {
                for (double[] d : diff2) {
                    tmp.add(d[i]);
                }
                dispersion[i] = tmp.stream().mapToDouble(e -> e).sum() / (num
- 1);
                medianDeviation[i] = Math.sqrt(dispersion[i]);
                tmp.clear();
            }

            double nA = num / (num - 1) / (num - 2);
            for (int i = 0; i <= n; i++) {
                for (double[] d : diff3) {
                    tmp.add(d[i]);
                }
                asymmetry[i] = nA * tmp.stream().mapToDouble(e -> e).sum() /
Math.pow(medianDeviation[i], 3);
                tmp.clear();
            }

            double nE = num * (num + 1) / (num - 1) / (num - 2) / (num - 3);
            for (int i = 0; i <= n; i++) {
                for (double[] d : diff4) {
                    tmp.add(d[i]);
```

```
                }
                excess[i] = nE * tmp.stream().mapToDouble(e -> e).sum() /
Math.pow(medianDeviation[i], 4);
                tmp.clear();
            }
        }

        @Override
        public void calculateDifferences(AnalysisHeader header, AnalysisData
ad) {
            int n = header.getN();
            double[] headerAverage = header.getAverage();
            double[] x = ad.getX();

            double[] diff = new double[n + 1];
            double[] diff2 = new double[n + 1];
            double[] diff3 = new double[n + 1];
            double[] diff4 = new double[n + 1];

            diff[0] = ad.getY() - headerAverage[0];
            diff2[0] = diff[0] * diff[0];
            diff3[0] = diff[0] * diff[0] * diff[0];
            diff4[0] = diff[0] * diff[0] * diff[0] * diff[0];
            for (int i = 1; i <= n; i++) {
                diff[i] = x[i-1] - headerAverage[i];
                diff2[i] = diff[i] * diff[i];
                diff3[i] = diff[i] * diff[i] * diff[i];
                diff4[i] = diff[i] * diff[i] * diff[i] * diff[i];
            }

            ad.setDiff(diff);
            ad.setDiff2(diff2);
            ad.setDiff3(diff3);
            ad.setDiff4(diff4);
        }
    }
```

Лістинг 2 – LgAnalysisDataService.java

```
    package com.statistic.service;

    import com.statistic.domain.AnalysisData;
    import com.statistic.domain.Field;
    import com.statistic.domain.ProjectClassStatistic;
    import com.statistic.utils.Normalizer;
    import org.apache.commons.math3.util.FastMath;
    import org.springframework.stereotype.Service;

    import java.util.*;

    @Service
    public class LgAnalysisDataService implements DataService {
        @Override
        public                                          Collection<AnalysisData>
prepareData(Collection<ProjectClassStatistic> statistics, Set<Field> fields) {
            List<AnalysisData> list = new ArrayList<>(statistics.size());
            for (ProjectClassStatistic pcs : statistics) {
                AnalysisData data = new AnalysisData(Math.log10((double)
pcs.getLoc() / 1000),
                        fields.stream().mapToDouble(e                    ->
Math.log10(e.getVal(pcs))).toArray());
                data.setProject(pcs);
                list.add(data);
            }
            return list;
```

```
        }

        @Override
        public                                          Collection<AnalysisData>
prepareData(Collection<ProjectClassStatistic> statistics, Set<Field> fields,
                                                        List<String[]>
normalization) {
            List<AnalysisData> list = new ArrayList<>(statistics.size());
            for (ProjectClassStatistic pcs : statistics) {
                double[] x = new double[fields.size()];
                int i = 0;
                for (Field f : fields) {
                    x[i] = f.getVal(pcs);
                    for (int j = 0; j < normalization.size(); j++) {
                        String[] cur = normalization.get(j);
                        if (cur[0].equals(f.name())) {
                            Normalizer             normalizer             =
Normalizer.getNormalizerByOperation(cur[1]);
                            x[i]         =         normalizer.normalize(x[i],
Field.valueOf(cur[2]).getVal(pcs));
                        }
                    }
                    x[i] = Math.log10(x[i]);
                    i++;
                }

                AnalysisData  data  =  new  AnalysisData(Math.log10((double)
pcs.getLoc() /1000), x);
                data.setProject(pcs);
                list.add(data);
            }
            return list;
        }
    }
```

Лістинг 3 – StatisticResult.vue

```
    <template>
        <div>
            <template v-if="getStatisticData.dataList">
                <div class="d-inline-block my-2">
                    <button    class="btn    btn-secondary    mr-2"    data-
target="#rowData" data-toggle="collapse"
                            aria-expanded="false"  aria-controls="rowData">Row
data
                    </button>
                    <button    class="btn    btn-secondary    mr-2"    data-
target="#barChart" data-toggle="collapse" aria-expanded="false"
                            aria-controls="barChart">Bar chart
                    </button>
                    <button    class="btn    btn-secondary    mr-2"    data-
target="#regressionChart" data-toggle="collapse"
                            aria-expanded="false"
                            aria-controls="regressionChart">Regression
                    </button>
                    <button v-if="getUserData.length" class="btn btn-secondary
mr-2" data-target="#predictionCard" data-toggle="collapse"
                            aria-expanded="false"
                            aria-controls="regressionChart">Prediction
                    </button>
                    <button             class="btn             btn-secondary"
@click="downloadAnalyse(getStatisticData.fileName)">
                            Download
                    </button>
                </div>
```

```
                    <div class="collapse" id="rowData">
                        <generic-table
                                :fields="getStatisticFields"
                                :analyse-head="getStatisticData"
                        />
                        <div class="d-flex align-content-center statisticResult">
                            <projects-statistic-table
                                    :projects-data="getStatisticData.dataList"
                                    :statistic-fields="getStatisticFields"
                                    :statistic-type="getStatisticType"
                            />
                        </div>
                    </div>
                    <div class="collapse" id="barChart">
                        <button     class="btn-secondary"     @click="reset     =
!reset">Reset</button>
                        <bar-chart-statistic
                                v-if="getStatisticData.dataList"
                                :chart-data="getDataForBar(getStatisticData)"
                                :reset="reset"
                        />
                    </div>
                    <div class="collapse" id="regressionChart">
                        <button  class="btn-secondary"  @click="regressionReset  =
!regressionReset">Reset</button>
                        <line-regression-chart
                                v-if="getStatisticData.dataList"
                                :chart-
data="getDataForRegressionChart(getStatisticData.dataList)"
                                :reset="regressionReset"
                        />
                    </div>
                    <div    v-if="getUserData.length"    class="collapse    mt-3"
id="predictionCard">
                        <PredictionCard
                                :user-data="getSortedUserData"
                                :mnk="getStatisticData.mnkResult"
                                :mre="getStatisticData.mre"
                                :type="getStatisticType"
                        />
                    </div>
                </template>
                <template v-else>
                    <h2 class="row justify-content-center h-100 mt-5">
                        Select parameters for the analise !
                    </h2>
                </template>
            </div>
        </template>

        <script>
            import {mapActions, mapGetters} from "vuex"
            import fieldsMap from '../fieldsNameToAnalyseTypeMap.js'
            import GenericTable from './table/GenericTable.vue'
            import                    ProjectsStatisticTable                    from
'./table/ProjectsStatisticTable.vue'
            import BarChartStatistic from './chart/BarChartStatistic.vue'
            import LineRegressionChart from "./chart/LineRegressionChart.vue";
            import PredictionCard from "./PredictionCard.vue";

            export default {
                components: {
                    GenericTable,    ProjectsStatisticTable,    BarChartStatistic,
LineRegressionChart, PredictionCard
                },
```

```
            data() {
                return {
                    reset: false,
                    regressionReset: false
                }
            },
            computed: {
                ...mapGetters(["getStatisticData",        "getStatisticType",
"getStatisticFields", "getUserData", "getSortedUserData"])
            },
            methods: {
                ...mapActions(["createStatisticByParams"]),
                getDataForBar(statistic) {
                    let dList = statistic.dataList;
                    return {
                        labels: dList.map(el => el.project.projectName),
                        datasets: [
                            {
                                label: 'Actual KLoc',
                                backgroundColor: this.randomColor(),
                                data:    dList.map(project    =>    project[
fieldsMap[this.getStatisticType].actualKLoc ])
                            },
                            {
                                label: 'R',
                                backgroundColor: this.randomColor(),
                                data: dList.map(el => el.r)
                            }, {
                                label: 'Calculated KLoc',
                                backgroundColor: this.randomColor(),
                                data:    dList.map(project    =>    project[
fieldsMap[this.getStatisticType].predictedKLoc ])
                            }, {
                                label: 'di Malohob',
                                backgroundColor: this.randomColor(),
                                data: dList.map(el => el.diMalohob)
                            }, {
                                label: 'ts',
                                backgroundColor: this.randomColor(),
                                data: dList.map(el => el.ts)
                            }
                        ]
                    }
                },
                randomColor() {
                    return 'rgb(' + Math.round(Math.random() * 255) + ',' +
Math.round(Math.random() * 255) + ',' + Math.round(Math.random() * 255) + ')';
                },
                getDataForRegressionChart(dataList) {
                    return {
                        labels: dataList.map(el => el.project.projectName),
                        datasets: [{
                            label: 'Actual KLoc',
                            backgroundColor: this.randomColor(),
                            borderColor: this.randomColor(),
                            // borderDash: [5, 5],
                            showLine: false,
                            fill: false,
                            data:    dataList.map(project    =>    project[
fieldsMap[this.getStatisticType].actualKLoc ])
                        }, {
                            label: 'Calculated KLoc',
                            backgroundColor: this.randomColor(),
                            borderColor: this.randomColor(),
                            // borderDash: [5, 5],
```

```
                                showLine: false,
                                fill: false,
                                data:     dataList.map(project     =>     project[
fieldsMap[this.getStatisticType].predictedKLoc ])
                            }, {
                                label: "Confidence upper",
                                borderColor: 'maroon',
                                borderDash: [3,3],
                                fill: false,
                                data:     dataList.map(project     =>     project[
fieldsMap[this.getStatisticType].confidenceUB ])
                            }, {
                                label: "Confidence lower",
                                borderColor: 'maroon',
                                borderDash: [3,3],
                                fill: false,
                                data:     dataList.map(project     =>     project[
fieldsMap[this.getStatisticType].confidenceLB ])
                            }]
                        }
                    },
                    downloadAnalyse(fileName) {
                        let url = this.apiAddress + '/download';
                        let config = {
                            params: {
                                fileName: fileName
                            },
                            responseType: 'blob'
                        };
                        this.$http.get(url, config).then(response => {
                            let   blob   =   new   Blob([response.data],   {type:
'text/csv;charset=utf-8;'});
                            let   filename   =   (response.headers.map['content-
disposition'] || '')[0].split('attachment;filename=')[1];
                            let result = document.createElement('a');
                            result.href = window.URL.createObjectURL(blob);
                            result.download = filename;
                            result.click();
                        })
                    }
                },
                mounted() {
                    this.createStatisticByParams()
                }
            }
        </script>

        <style>
            #barChart {
                background-color: white;
            }
        </style>
```