МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

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

«Допущений до захисту»
Завідувач кафедри
_____
«___» _____ 2021 р.

КВАЛІФІКАЦІЙНАРОБОТА

на здобуття ступеня вищої освіти «магістр»

на тему: ***_Регресійна модель для оцінювання розміру застосунків, що створюються з використанням фреймворку .Net, та розробка програми для її реалізації***

Виконав:
студент 6151м групи Мисько Ю.М.
(підпис, ПІБ)

Керівник роботи:
доцент, к.т.н. Пономаренко Т.В.
(посада, науковий ступень вчене звання)

_____
(підпис, ПІБ)

Миколаїв – 2021 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

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

«ЗАТВЕРДЖУЮ»
Гарант освітньої програми
_____проф. С.Б.Приходько
(підпис)
«___» _____ 2021 р.

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

Студенту Мисько Юрію Михайловичу
(Прізвище, ім'я, по батькові)

1. Тема роботи:Регресійна модель для оцінювання розміру застосунків, що створюються з використанням фреймворку .Net, та розробка програми для її реалізації
Керівник роботи: Пономаренко Тетяна Вікторівна
Затверджені наказом ректора № 1239уч від «13» _____10_____ 2021 р.
2. Термін подання роботи: 06.12.2021 р._____
3. Вихідні дані по роботі:завдання на розробку видане керівником Пономаренко Т.В.
4. Перелік питань, що належать до розробки (найменування розділів) _____
Вступ (Актуальність теми. Зв'язок роботи з науковими програмами, планами, темами. Мета і завдання дослідження. Об'єкт дослідження. Предмет дослідження. Методи дослідження. Наукова новизна одержаних результатів. Практичне значення одержаних результатів. Особистий внесок здобувача. Апробація результатів досліджень**.** Публікації.); Огляд літератури та обґрунтування необхідності проведення досліджень за обраною темою; Викладення результатів власних досліджень з висвітленням того нового, що пропонується; Проект програмного забезпечення; Організаційно-економічний розділ; Розділи з охорони праці та охорони навколишнього середовища; Висновки; Список використаних джерел; Додатки (технічне завдання, текст програми, опис програми, інструкція користувача, програма і методика випробувань програмного забезпечення)
5. Перелік презентаційних матеріалів _____
_____
_____
_____
_____
_____
_____
_____
_____
6. Консультанти розділів роботи

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

7. Дата видачі завдання 13.10.2021 р. _____

К А Л Е Н Д А Р Н И Й П Л А Н

Номер	Назва етапів роботи	Терміни виконання	Примітка
1.	Підготовка розділу Вступ	18.10.2021	НС*
2.	Підготовка розділу з огляду літератури та обґрунтування необхідності проведення досліджень за обраною темою	19.10.2021	НС*
3.	Підготовка розділу (ів) з результатів власних досліджень	22.10.2021	НС*
4.	Підготовка розділу з проекту програмного забезпечення	16.11.2021	
5.	Підготовка організаційно-економічного розділу	19.11.2021	
6.	Підготовка розділу з охорони праці	22.11.2021	
7.	Підготовка розділу з охорони навколишнього середовища	24.11.2021	
8.	Підготовка розділу Висновки	25.11.2021	
9.	Оформлення списку використаних джерел та додатків	29.11.2021	
10.	Подання на кафедру ПЗАС тексту остаточного варіанту роботи, підписаного її керівником, у роздрукованому та електронному форматі разом із заявами щодо самостійності виконання роботи та ідентичності друкованої та електронної версії роботи (Додатки 1 і 2 «Порядку здійснення заходів з перевірки робіт на наявність текстових збігів/ідентичності/схожості із використанням програмно-технічних засобів», який введений в дію наказом ректора НУК за №20 від 20.01.2020 р.)	06.12.2021	не пізніше, ніж за 14 діб до захисту (згідно п.4.1 зазначеного Порядку
11.	Підготовка презентації та доповіді	09.12.2021	
12.	Попередній захист роботи на засіданні кафедри ПЗАС	10.12.2021	
13.	Подання на кафедру ПЗАС електронних версії наступних документів у форматі pdf: кваліфікаційної роботи; файлу-опису кваліфікаційної роботи (згідно Додатку до наказу ректора НУК за №287-уч від 19.05.2020 р.); презентації доповіді	20.12.2021	

* - за результатами наукового стажування (НС), яке було з 01.09.2021 до 10.10.2021 р.

Студент _____ _____
(підпис) (ПІБ)

Керівник роботи _____ _____
(підпис) (ПІБ)

РЕФЕРАТ

Мисько Юрій Михайлович

«Регресійна модель для оцінювання розміру застосунків, що створюються з використанням фреймворку .Net, та розробка програми для її реалізації»

Кваліфікаційна робота на здобуття освітнього рівня магістра зі спеціальності 121 – «Інженерія програмного забезпечення».

Національний університет кораблебудування імені адмірала Макарова.

Миколаїв,

2021 р.

Обсяг роботи: 95 стор., 9 табл., 26 рис., 29 використаних джерел, 5 додатків.

Актуальність теми роботи: Актуальність обраної теми для кваліфікаційної роботи полягає в тому, що підвищення достовірності оцінювання розміру програмного забезпечення (ПЗ), оцінювання розміру застосунків, що створюються з використанням фреймворку .Net, дозволить зменшити кількість невдало завершених проєктів з розробки ПЗ таких застосунків.

Метою роботи є підвищення достовірності оцінювання розміру застосунків, що створюються з використанням фреймворку .Net за допомогою регресійної моделі.

Об’єктом дослідження є процес оцінювання розміру застосунків, що створюються з використанням фреймворку .Net.

Предметом дослідження є нелінійна багатофакторна регресійна модель оцінювання розміру застосунків, що створюються з використанням фреймворку .Net.

Методи дослідження. Для вирішення поставлених задач були застосовані методи теорії ймовірностей та математичної статистики,

регресійного аналізу, побудови нелінійних регресійних моделей на основі нормалізуючих перетворень.

Наукова новизна одержаних результатів полягає в удосконаленні множинної нелінійної регресійної моделі для оцінювання розміру застосунків, що створюються з використанням фреймворку .Net за рахунок використання одновимірного нормалізуючого перетворення у вигляді десяткового логарифму, що дозволило підвищити достовірність оцінювання розміру веб застосунків що створюються з використанням фреймворку .Net в порівнянні з існуючою моделлю.

Практичне значення отриманих результатів полягає в розробці програми для оцінювання розміру застосунків, що створюються з використанням фреймворку .Net на основі нелінійної регресійної моделі та за рахунок використання багатофакторної нелінійної регресійної моделі.

Публікації. Основні результати магістерської роботи викладено у Матеріалах IV Всеукраїнської науково-практичної інтернет-конференції студентів, аспірантів та молодих вчених за тематикою «Сучасні комп'ютерні системи та мережі в управлінні» та підготовлено статтю для публікації у фаховому виданні.

Ключові слова: .NET, ОЦІНЮВАННЯ РОЗМІРУ ЗАСТОСУНКІВ, БАГАТОФАКТОРНА РЕГРЕСІЙНА МОДЕЛЬ.

ABSTRACT

Mysko Yuriy Mykhailovych

" A regression model for estimating the size of applications created using the .Net framework and developing the software for its implementation"

Qualification work in obtaining a master's degree in specialty 121 - "Software Engineering".

Admiral Makarov National University of Shipbuilding. Mykolayiv, 2021

The work contains: 95 pages, 9 tables, 26 figures, 29 used sources, 5 appendices.

Relevance of the topic: The relevance of the chosen topic for the qualification work is that increasing the reliability of estimating the size of software (software), estimating the size of applications created using the .Net framework, will reduce the number of unsuccessful projects for software development of such applications.

The purpose and objectives of the study is that increasing the reliability of estimating the size of software (software), estimating the size of applications created using the .Net framework, will reduce the number of failed software development projects for such applications.

The object of research is the process of estimating the size of applications created using the .Net framework.

The subject of research is a nonlinear multifactor regression model for estimating the size of applications created using the .Net framework.

Research methods. Methods of probability theory and mathematical statistics, regression analysis, construction of nonlinear regression models based on normalizing transformations were used to solve the problems.

The scientific novelty of the obtained results is to improve the multiple nonlinear regression model for estimating the size of applications created using the .Net framework by using one-dimensional normalizing transformation in the form of a decimal logarithm, which increased the reliability of estimating the size of web applications created using. compared to the existing model.

The practical value of the obtained results is to develop a program for estimating the size of applications created using the .Net framework based on a nonlinear regression model and by using a multifactor nonlinear regression model.

Approbation of research results. The main results of the master's work are presented in the Proceedings of the IV All-Ukrainian scientific-practical online conference of students, graduate students and young scientists on "Modern computer systems and networks in management" and prepared an article for publication in a professional publication.

Keywords: .NET, EVALUATION OF APPLICATION SIZE, MULTIFACTOR REGRESSION MODEL.

ЗМІСТ

ВСТУП

Актуальність теми.

В епоху великих веб-фреймворків, мобільних додатків, інтернету речей web API як ніколи актуальні, і .NET вирішує цю проблему якнайкраще завдяки сумісності, надійності та якісній підтримці від It-гіганта Microsoft. .NET 5 вже зараз надає бібліотеки, фреймворки, інструменти та API для створення, тестування, запуску та розгортання програмного забезпечення, призначеного для всіх платформ, включаючи Windows, Linux, IoT, macOS, iOS, Android, tvOS, watchOS та WebAssembly. Також всі пристрої, включаючи настільні комп'ютери, веб-браузери, пристрої IoT, планшети, мобільні телефони та багато іншого підтримують роботу з .NET.

Виходячи з широкого розповсюдження актуальність прогнозування розміру та підвищення достовірності оцінювання розміру майбутніх .NET доданків на початковому етапі проектів з розробки програмного забезпечення є актуальною задачею.

Нормализуючі перетворення дуже часто є надійним способом побудови нелінійних регресійних моделей та рівнянь [1-2]. Однак добре відомі методи їх побудови, що засновані на одновимірних нормалізуючих перетвореннях (таких як логарифмічне і Бокса-Кокса, Джонсона), не враховують кореляції між випадковими змінними у разі нормалізації багатовимірних негаусових даних. Це призводить до необхідності використання багатовимірних нормализуючих перетворень [3-4], які враховують цю кореляцію, для побудови нелінійних регресійних моделей та рівнянь для оцінювання розміру ПЗ інформаційних систем, в тому числі з відкритим кодом на PHP. В [5-6] було побудовано нелінійне регресійне рівняння для оцінювання розміру програмного забезпечення ІС з відкритим кодом на PHP на основі багатовимірного перетворення Джонсона для сімейства SB. Виникає потреба порівнянні моделей та у побудові відповідного рівняння для інших видів ПЗ

інформаційних систем з відкритим кодом, наприклад для застосунків, що створюються за допомогою .Net.

Підвищення надійності оцінки часу розробки програмного забезпечення відіграє важливу роль в практичних завданнях. Як правило, час отриманий при реальних розробках є негаусівською випадковою величиною, яка залежить від ряду факторів. Для оцінки часу розробки ПЗ необхідно побудувати відповідну регресійну модель [7-10], яка буде нелінійною [11]. Підвищити надійність її оцінки можна за рахунок побудови довірчого інтервалу нелінійної регресії [12,13].

У разі негауссовских випадкових величин побудова довірчого інтервалу нелінійної регресії без припущення про нормальність ВВ утруднено. Застосування такого припущення може істотно спотворити результати, тому до існуючих даних необхідно застосовувати нормалізуючі перетворення.

При нормальному законі розподілу випадкової величини довірчий інтервал лінійного рівняння регресії можливо побудувати традиційним методом з використанням t-розподілу Стьюдента [14]. Однак для нелінійної регресії даний метод не враховує ряд особливостей емпіричного розподілу даних, наприклад його асиметрію.

Використання нормалізуючих перетворень зводиться до отримання лінійної регресійної моделі з вихідної нелінійної шляхом заміни змінних і коефіцієнтів. Однак така заміна приводить до спрощення регресійній моделі і деякої втрати інформації, пов'язаної з нелінійністю [15-16].

Застосування нормалізуючих перетворень дозволяє перейти до лінійної регресії для нормалізованих даних, для неї побудувати довірчий інтервал традиційним способом з використанням t-розподілу Стьюдента, а потім шляхом застосування відповідного перетворення перейти до нелінійної регресії і її довірчого інтервалу [17-20]. Даний підхід позбавлений недоліків, зазначених у попередніх методах. В якості нормалізуючого перетворення використовується логарифмічне перетворення за натуральною основою [22].

Метою роботи є підвищення достовірності оцінювання розміру застосунків, що створюються з використанням фреймворку .Net

Для досягнення поставленої мети необхідно вирішити такі завдання: проаналізувати існуючі моделі та методи оцінювання розміру застосунків, що створюються з використанням фреймворку .Net, сформулювати постановку задачі; побудувати математичну модель для оцінювання розміру застосунків, що створюються з використанням фреймворку .Net; розробити програмне забезпечення для оцінювання розміру .Net доданків, використовуючи розроблену математичну модель.

Об'єктом дослідження є процес оцінювання розміру застосунків, що створюються з використанням фреймворку .Net.

Предметом дослідження є нелінійна регресійна модель оцінювання розміру застосунків, що створюються з використанням фреймворку .Net.

Методи дослідження. Для вирішення поставлених задач були застосовані методи теорії ймовірностей та математичної статистики, регресійного аналізу, побудови нелінійних регресійних моделей на основі нормалізуючих перетворень.

Наукова новизна одержаних результатів полягає в удосконаленні множинної нелінійної регресійної моделі для оцінювання розміру застосунків, що створюються з використанням фреймворку .Net за рахунок використання одновимірного нормалізуючого перетворення у вигляді десяткового логарифму, що дозволило підвищити достовірність оцінювання розміру веб застосунків що створюються з використанням фреймворку .Net в порівнянні з існуючою моделлю.

Практичне значення отриманих результатів полягає в розробці програми для оцінювання розміру застосунків, що створюються з використанням фреймворку .Net на основі нелінійної регресійної моделі та за рахунок використання багатофакторної нелінійної регресійної моделі.

Особистий внесок здобувача. Усі основні результати кваліфікаційної роботи отримані здобувачем самостійно. Кваліфікаційна робота є самостійно виконаною працею. У тезах доповіді [23] які написано у співавторстві здобувачу належить розробка математичної моделі оцінювання оцінювання розміру застосунків, що створюються з використанням фреймворку .Net.

Апробація результатів досліджень. Основні результати магістерської роботи викладено у Матеріалах IV Всеукраїнської науково-практичної інтернет-конференції студентів, аспірантів та молодих вчених за тематикою «Сучасні комп'ютерні системи та мережі в управлінні» (Херсон 30.11.2021).

Публікації. За матеріалами кваліфікаційної роботи опубліковано 1 наукову працю - тези доповіді на конференції.

1 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ ТА МОДЕЛЕЙ ОЦІНЮВАННЯ РОЗМІРУ ЗАСТОСУНКІВ, ЩО СТВОРЮЮТЬСЯ З ВИКОРИСТАННЯМ ФРЕЙМВОРКУ .NET

1.1 Аналіз особливостей оцінювання розміру застосунків, що створюються з використанням фреймворку .Net

Платформа .NET Framework складається із загальномовного середовища виконання (середовища CLR) та бібліотеки класів .NET Framework. Основою платформи .NET Framework є середовище CLR. Середовище виконання можна вважати агентом, який керує кодом під час виконання та надає основні служби, такі як управління пам'яттю, управління потоками та віддалену взаємодію. При цьому середовищем накладаються умови суворої типізації та інші види перевірки точності коду, що забезпечують безпеку та надійність.

Основним завданням .NET є керування кодом. Бібліотека класів є комплексною об'єктно-орієнтованою колекцією повторно використовуваних типів, які застосовуються для розробки додатків, що використовують останні технологічні можливості ASP.NET, такі як веб-форми та веб-служби XML.

Особливостями платформи є те, що текст програми повинен бути написаний мовою, яка відповідає специфікації Common Language Specification. Особливістю всіх мов програмування, що відповідають специфікації CLS, є те, що компілятори з цих мов переводять вихідний текст програми в Microsoft Intermediate Language (MSIL). Цим досягається висока сумісність між різними мовами, а також незалежність від архітектури

комп'ютера та його операційної системи. Таким чином, хоча платформа Microsoft .NET і створювалася для Windows на IBM-сумісних комп'ютерах, вона може бути реалізована для будь-яких інших операційних систем та для комп'ютерів, що мають несумісний з x86 набір машинних команд. Так, наприклад, існує та успішно розвивається проект DotGNU для Linux. Ще є проект Mono, що паралельно розвивається і для Windows, і для Linux.

1.2 Аналіз існуючих моделей оцінювання розміру застосунків, що створюються з використанням фреймворку .Net

У ряді робіт [1-4] побудовано математичні моделі для оцінювання розміру програмного забезпечення, але мені не вдалося знайти жодної роботи з математичною моделлю для оцінювання розміру саме .Net застосунків, тому розробка математичної моделі для оцінювання розміру застосунків, що створюються з використанням фреймворку .NET є актуальним завданням.

На сайті http://softwarecost.org/ скріншот якого зображено на рисунку 1.1 є онлайн калькулятор за моделлю COCOMO та методом функціональних точок за 5 мовами, але можливості прогнозувати розмір для програмного забезпечення, що створене з використанням платформи .Net там відсутній.

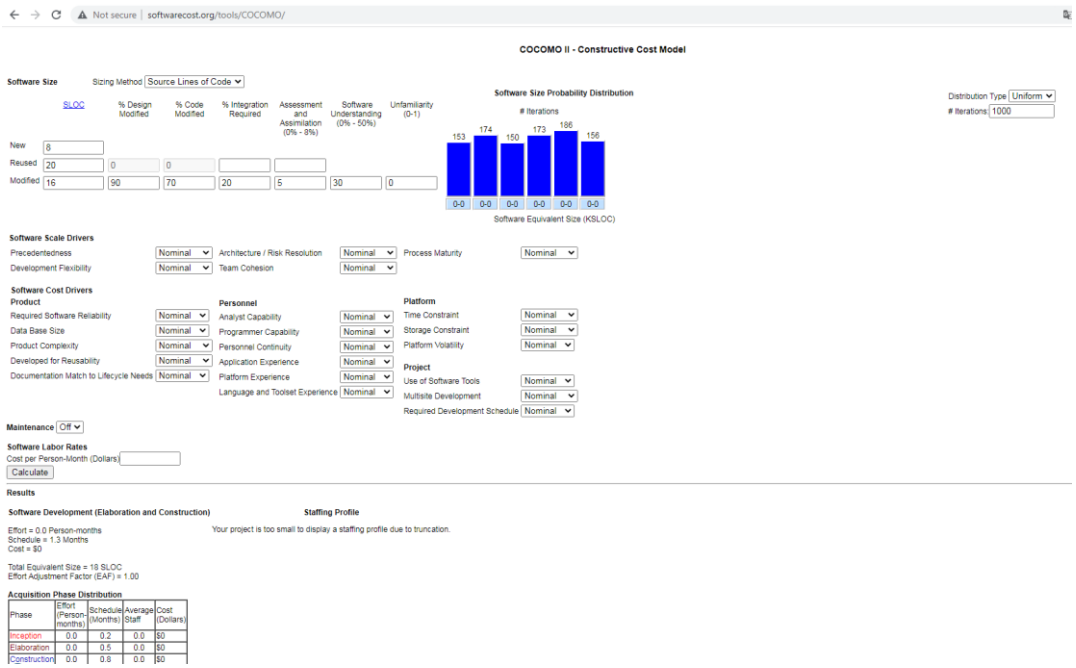


Рисунок 1.1 – Онлайн калькулятор розміру програмних застосунків

Треба відзначити, що проблема оцінювання розміру стає ще більш невизначеною завдяки можливості використання великої кількості різноманітних метрик та складності вибору саме тих, що найкраще описують найголовніші якості програмного забезпечення.

2 МАТЕМАТИЧНА МОДЕЛЬ ОЦІНЮВАННЯ РОЗМІРУ ЗАСТОСУНКІВ, ЩО СТВОРЮЮТЬСЯ З ВИКОРИСТАННЯМ ФРЕЙМВОРКУ .NET

2.1 Регресійна модель для оцінювання розміру застосунків, що створюються з використанням фреймворку .Net

При вирішенні завдань оцінювання статистичних характеристик та функцій розподілу працюють з простим випадковим вибором. Однак, якщо формувати вибірки за певним планом, то з'являється можливість оцінити кількісно багатофакторну статистичну залежність при обмеженому обсязі вибірок. Розглянемо такий підхід для оцінювання статистичних залежностей виду

$$Y=f(x_1, x_2, ..., x_k), \quad (1)$$

де x_s, где s=1, 2, ..., k, множина змінних факторів, коли в результаті емпіричних дослідів ми набуваємо деяку інформацію про x_s.

Тут потрібно скористатися прийомами, які існують у теорії планування експерименту.

При класичному підході до оцінки залежності (1) процес дослідження або експеримент (вибірка) будується за такою схемою:

$$x1=var;\ x2, x3,..., xk=const;\ Y=f(x1),$$
$$x2=var;\ x1, x3,..., xk=const;\ Y=f(x2),$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$xk=var;\ x1, x2, x3,..., xk\text{-}1=const;\ Y=f(xk),$$

тобто змінюється лише один параметр.

Неважко побачити, що в такому випадку необхідний суттєвий обсяг даних щодо кожної змінної. Методи планування експерименту дають змогу уникнути накопичення великого обсягу інформації. Планування експерименту - це постановка спостережень (взяття вибірок) за деякою заздалегідь складеною схемою, що має якісь оптимальні властивості. Розробка таких схем є складним математичним завданням, але дозволяє побудувати стратегію експерименту (спостережень) таким чином, що на виявлення залежності (1) потрібно мінімум інформації. Мінімізація кількості спостережень здійснюється за рахунок оптимального використання всього *K*-мірного факторного простору змінних $\{x\}$.

При визначенні залежності (1) методами теорії планування експерименту змінюються одночасно всі змінні фактори x_1; x_2, x_3,..., x_k=var, Причому варіювання складає обмежене число рівнів - двох, трьох. Крім того, рівні (значення) факторів x_s вибирають так, щоб вони охоплювали весь багатовимірний простір експерименту.

Порядок варіювання x_s, поєднання рівнів варіювання всіх параметрів визначаються планом експерименту.

Перши етапом для побудови моделі повинен бути збір початкових даних для формування вибірки. Зазвичай він є трудомістким і потребує багато часу, тому мною було вирішено створити власне програмне забезпечення для збору даних про проекти .NET.

Діаграму компонентів цього програмного забезпечення наведено на рисунку 2.1. Весь код програмного забезпечення наведено у додатку Б.

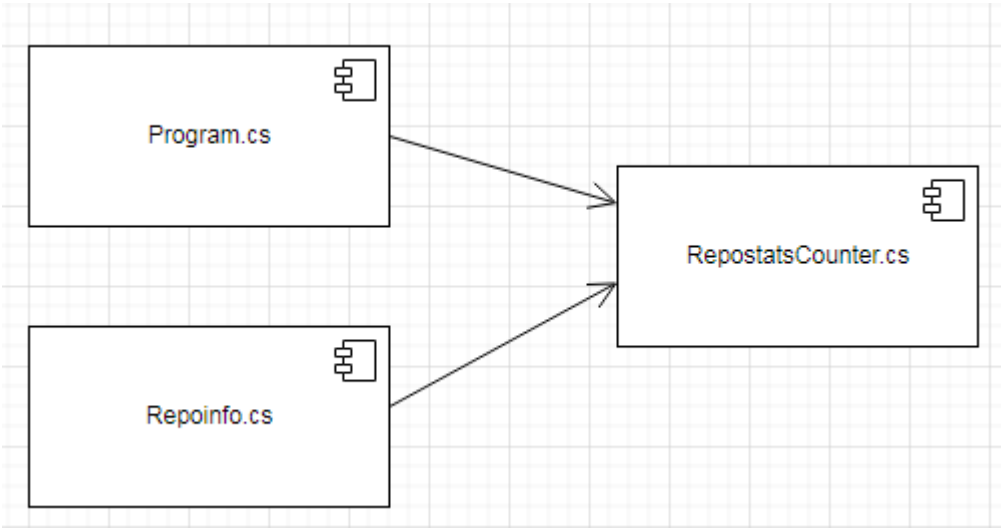


Рисунок 2.1 – Діаграма компонентів програмного забезпечення, яке створено мною для збору статистики .Net проектів

За результатами роботи цього програмного забезпечення було отримано вибірку даних, яку наведено у таблиці 2.1.

Таблиця 2.1. Початкові дані

Project Name	x1	x2	Y (Code lines count)
CryPy	10	5540	1385
ExchangeIndicatorsTester	4	5024	1256
AutoExchangeSystem	15	10016	2504
Telegram Repost Exchange (TRX)	18	18772	4693
ACA - MAIN	40	3428	857
ACA - ДЛЯ ОТЧЕТОВ	59	22320	5580
AutoCommunityAdmin	29	51216	12804
RollingOutTools	49	206940	51735
IRO	64	28176	7044
IRO.Mvc	23	11100	2775
IRO.XWebView	82	193848	48462
IROApps.PortForwarding	26	2076	519
Telegram.Bot.AspNetPipeline	61	39068	9767
UndergroundIRO.TradingViewKit	11	7032	1758
cybertron	58	32668	8167
helpified-cryptanalysis-engine	10	90712	22678
OptionAnalysis	122	54332	13583
QuasarApi	231	70580	17645
S2A_Mobile	123	209972	52493
S2A_Old_ForSale	11	175460	43865
CRM система	10	560	140
KworkTelegramBot	2	3740	935
OlxParser	5	6256	1564
TabletopHelperSite	40	476660	119165
Main	1	920	230
youtube_random_playlist_bug_fix	5	624	156
TgReminderBot	11	5636	1409
TSS.Console_v2	5	5136	1284
TSS.SharpedJs	3	127376	31844
TSS.WebVersionCompiled	4	115092	28773
TSS.WinForms	6	14884	3721
TopdogGame	214	139668	34917
BattleshipServer	68	17500	4375

Виходячи з того, що ми маємо 2 незалежні змінні, будемо будувати багатофакторну регресійну модель вигляду

$$Y = b_0 + b_1 x_1 + b_2 x_2 \ + \mathcal{E} \qquad (1)$$

де в нас буде 3 коефіцієнти регресії b_0, b_1, b_2. Коефіцієнти регресії (вектор b) можна обчислити за формулою $b = (X^T X)^{-1} (X^T Y)$ або в іншому позначенні транспонованих матриць : $b = (X' X)^{-1} (X' Y)$.

Таблиця 2.2 Матриця кореляції

1	-0,85
-0,85	1

Таблиця 2.3. Матриця X'X

34	45,503007	143,7490093
45,503007	76,097785	196,332854
143,74901	196,33285	631,4374045

Таблиця 2.4. Матриця зворотня до матриці X'X (коваріації)

0,8118608	-0,043534	-0,171287074
-0,043534	**0,0687707**	-0,011472255
-0,171287	-0,011472	**0,044144879**

Таблиця 2.5. Матриця X'Y

125,1882
175,07418
549,12557

За матричним методом розрахуємо регресійні коефіцієнти і отримаємо значення b_0 , b_1 , b_2 0 які будуть відповідно дорівнювати 0,044367, 0.2903509 та 0.7894653.

При цьому в нас кількість спостережень (розмір вибірки) буде дорівнювати 34, число оцінюваних параметрів моделі (коефіцієнтів регресії) трьом, число незалежних змінних в нас буде 2, а число ступенів свободи залишків моделі в буде 29.

На головній діагоналі зворотньої коваріаційної матриці у таблиці 2,4 ми маємо значення VIF, його значення менші за 5. Це вказує на відсутність мультиколінеарності факторів X_1, X_2.

Якщо між факторними змінними є високий ступінь кореляції, то матриця (X^TX) близька до виродженої, тобто, чим ближче до 0 визначник матриці межфакторной кореляції, тим сильніше мультиколінеарність факторів і ненадійніше результати множинної регресії. Як показує аналіз в нас мультиколінеарність відсутня.

Розрахунки прогнозних значень з використанням отриманих коефіцієнтів регресії та залишки між розрахунковими і фактичними значеннями , величини відносної похибки та відстань Махалонобіса наведено в таблиці 2.6.

Таблиця 2.6. Нормовані та прогнозні значення, залишки, величина відносної похибки та відстань Махалонобіса

log10(x1)	log10(x2)	log10(y)	Прогноз	Залишки	MRE	MD2
1	2	3	4	5	6	7
1	3,74351	3,141449773	3,2013545	0,059904769	0,0190691	0,6993171
0,60206	3,70105	3,098989639	3,0522915	0,04669813	0,0150688	1,0736572
1,1760913	4,000694	3,398634325	3,4555211	0,05688675	0,0167381	0,3310327
1,2552725	4,273511	3,671450554	3,6938904	0,022439813	0,006112	0,0318348
1,60206	3,535041	2,932980822	3,2115842	0,278603362	0,0949898	0,3647647
1,770852	4,348694	3,746634199	3,9029442	0,156310008	0,0417201	0,4703139
1,462398	4,709406	4,107345665	4,0981535	0,009192163	0,002238	0,5147327
1,6901961	5,315844	4,713784453	4,6430573	0,070727195	0,0150043	1,223834
1,80618	4,449879	3,847819347	3,9930839	0,145264529	0,0377524	0,58635
1,3617278	4,045323	3,443262987	3,5446536	0,101390611	0,0294461	0,1353072
1,9138139	5,287461	4,685401333	4,6855774	0,000176053	3,757E-05	1,3897835
1,4149733	3,317227	2,715167358	2,9853072	0,270139882	0,0994929	0,7089298

Продовження таблиці 2.6

1	2	3	4	5	6	7
1,7853298	4,591821	3,989761188	4,0990882	0,109326991	0,0274019	0,689278
1,0413927	3,847079	3,245018871	3,2951372	0,050118283	0,0154447	0,5760996
1,763428	4,514123	3,912062556	4,0313886	0,11932603	0,0305021	0,6046176
1	4,957665	4,355604751	4,1598878	0,195716977	0,0449345	0,3327146
2,0863598	4,735056	4,132995701	4,2995712	0,16657548	0,0403038	1,0669028
2,363612	4,848682	4,246621663	4,4697753	0,223153677	0,0525485	1,3991492
2,0899051	5,322161	4,720101394	4,7641001	0,043998738	0,0093216	1,5689561
1,0413927	5,244178	4,642118134	4,3980986	0,244019582	0,0525664	0,6114348
1	2,748188	2,146128036	2,4155826	0,269454526	0,1255538	1,5453406
0,60206	3,572872	2,970811611	2,8636951	0,10711653	0,0360563	1,438484
0,69897	3,796297	3,194236749	3,1556237	0,038613043	0,0120883	0,9103237
1,60206	5,678209	5,076148717	4,9035409	0,172607834	0,0340037	1,456928
0,7781513	2,963788	2,361727836	2,2954402	0,066287587	0,0280674	2,2120807
0,69897	2,795185	2,193124598	2,3652804	0,172155797	0,078498	1,761269
1,0413927	3,750971	3,148910993	3,2192633	0,070352326	0,0223418	0,6577913
0,69897	3,710625	3,108565024	3,0879889	0,020576173	0,0066192	0,9831446
0,4771213	5,105088	4,503027615	4,1244547	0,378572906	0,0840707	0,0135771
0,60206	5,061045	4,458985146	4,1259608	0,333024367	0,0746861	0,082339
0,7781513	4,17272	3,57065967	3,4757869	0,094872784	0,0265701	0,5230601
2,3304138	5,145097	4,543036923	4,6941458	0,151108852	0,0332616	1,6228837
1,8325089	4,243038	3,640978057	3,8374345	0,196456427	0,053957	0,4329145
3,2143139	2,217484	3,524655712	2,6395381	0,885117632	0,2511217	0,1142722

Для великої вибірки, де кількість значень більше 30 стандартне відхилення знаходиться за формулою:

$$\sigma = \pm\sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{X})^2}{n}}$$

За залишками знаходимо значення суми квадратів відхилень σ^2, яке дорівнює 1,689.

2.2 Побудова математичної моделі для оцінювання розміру програмного забезпечення, що створене із застосуванням фреймворку .Net

Отож, отримана модель у вигляді рівняння багатофакторної регресії має вигляд:

$$Y=-0,044+0,290*x1+0,789*x2.$$

Для перевірки адекватності лінійного рівняння регресії використаємо коефіцієнт детермінації R^2, який в цьому випадку дорівнює 0,91.

Для додаткової первірки адекватності моделі використовується величина відносної похибки MRE (Magnitude of Relative Errors) за якою знайдемо фактичне значення випадкової величини у. MMRE (Mean of Magnitude of Relative Errors – середня величина відносної похибки), яка в нашому випадку дорівнює 0,0446. Рівень прогнозування PRED(0,25) дорівнює 0,088.

Шляхом використання зворотнього перетворення побудуємо нелінійну регресійну модель оцінювання розміру програмного забезпечення, що створене із застосуванням фреймворку .Net у вигляді:

$$Y = 10^{\varepsilon+1.04} * x_1^{0.04} * x_2^{0.25}.$$

Значення параметрів нової нелінійної множинної регресійної моделі R2 , MMRE і PRED(0,25), які дорівнюють відповідно 0,94, 0,135 і 0,900, є задовільними. Причому значення цих показників для моделі є кращими за

значення R^2 , MMRE і PRED(0,25) для лінійної регресійної моделі, які дорівнюють відповідно 0,91, 0,0446 і 0,600.

2.3 Постановка задачі на розробку програмного забезпечення для реалізації математичної моделі оцінювання розміру програмного забезпечення, що створене із застосуванням фреймворку .Net.

Математична модель, яка відображає багатофакторну статистичну залежність кількості комітів, кількості розробників та розміру програмного забезпечення має добру якість і може буди реалізована у вигляді програмного інструменту для оцінювання розміру. Застосоване перетворення десятковим логарифмуванням дозволило удосконалити лінійне рівняня регресії в нелінійне (експотенціальне) регресійне рівняння, що відображає модель кращої якості для оцінювання розміру програмного забезпечення, що створене із застосуванням фреймворку .Net.

3 ПРОЕКТ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ОЦІНЮВАННЯ РОЗМІРУ ЗАСТОСУНКІВ, ЩО СТВОРЮЮТЬСЯ З ВИКОРИСТАННЯМ ФРЕЙМВОРКУ .NET

3.1 Ескізний проект

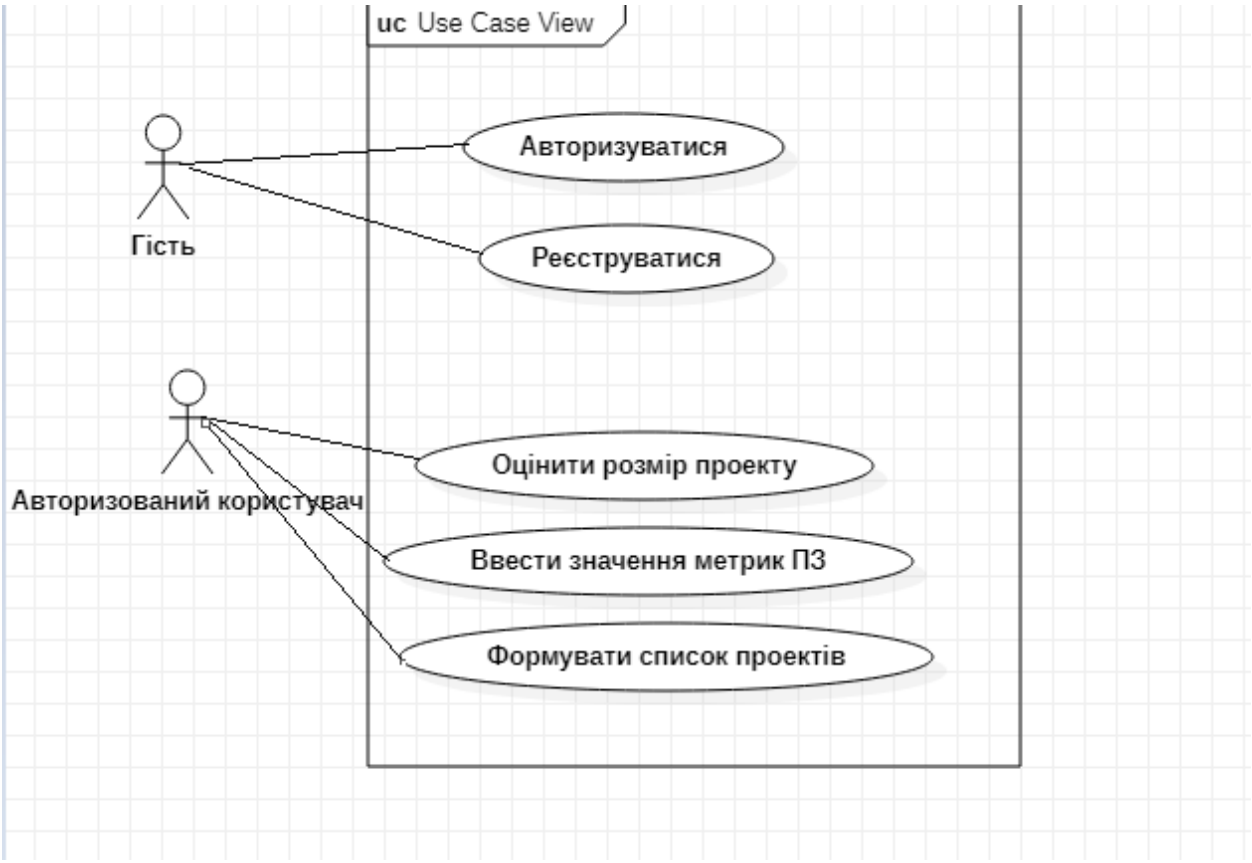


Рисунок 3.1 - Діаграма прецедентів програмного забезпечення для оцінювання розміру застосунків, що створюються з використанням фреймворку .NET

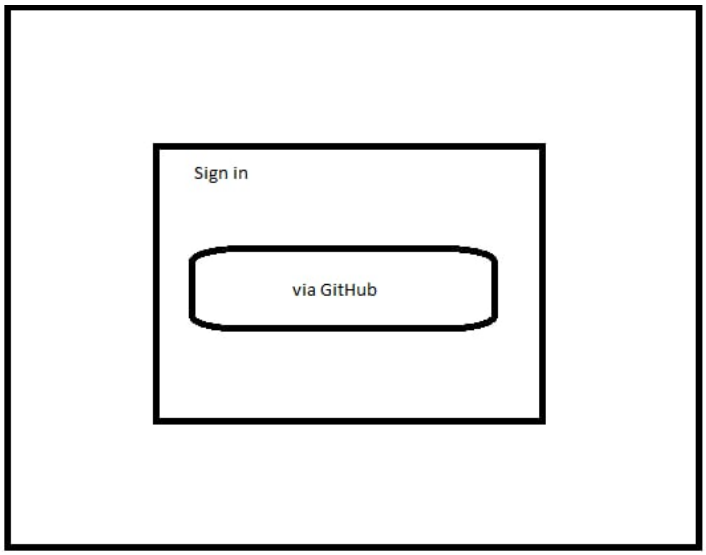


Рисунок 3.2. – Ескіз форми авторизації ПЗ для оцінювання розміру застосунків, що створюються з використанням фреймворку .NET

Після авторизації авторизований користувач повинен мати можливість перегляду сторінки зі статистикою облікового запису, ескіз якої наведено на рисунку 3.3.

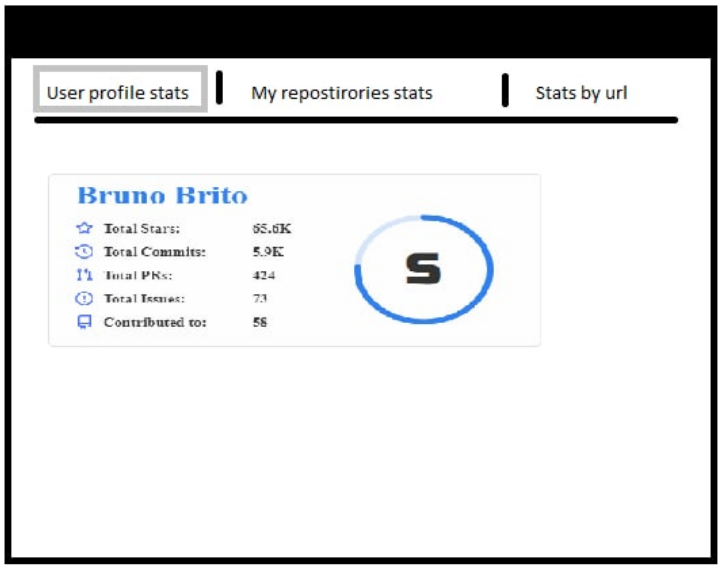


Рисунок 3.3. - Ескіз форми статистики облікового запису Github ПЗ для оцінювання розміру застосунків, що створюються з використанням фреймворку .NET

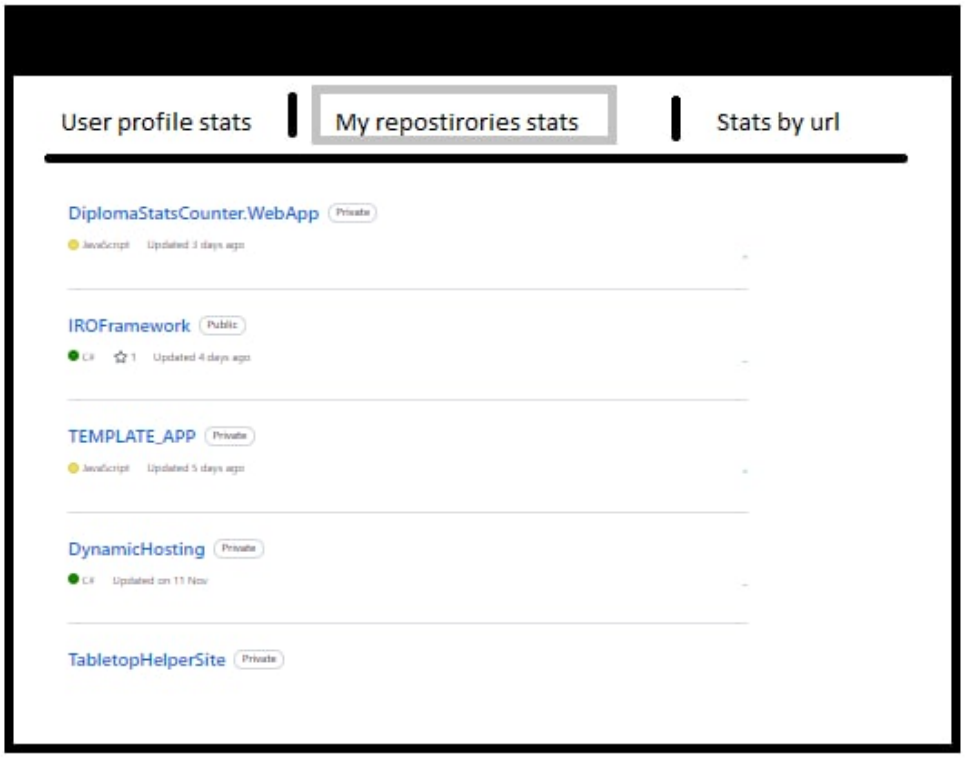


Рисунок 3.4. - Ескіз форми списку доступних репозиторіїв ПЗ для оцінювання розміру застосунків, що створюються з використанням фреймворку .NET

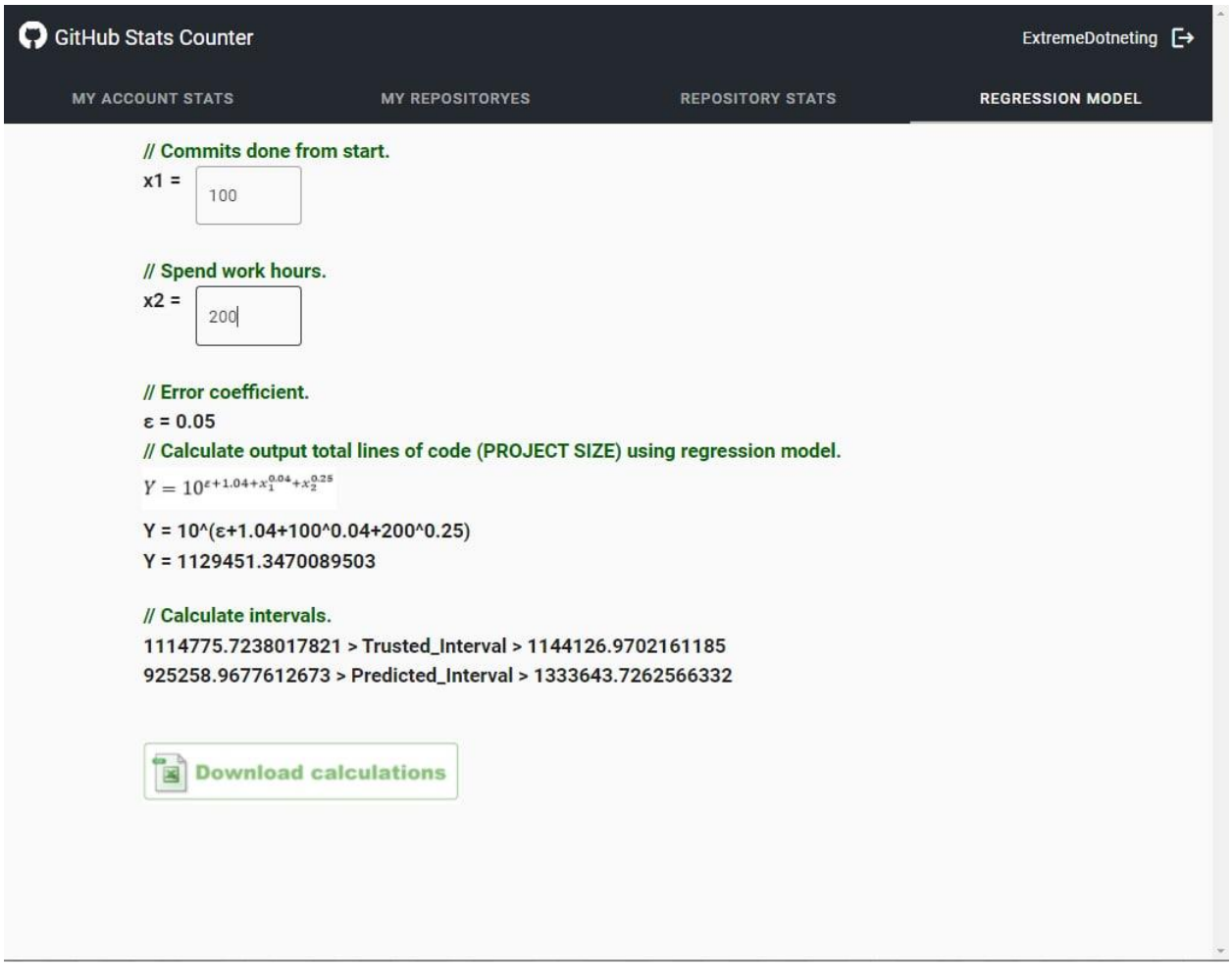


Рисунок 3.5 - Ескіз форми розрахунку розміру програмного забезпечення з використанням розробленої математичної моделі

3.2. Технічний проект

Архітектура програмного забезпечення буде визначена паттерном MVC структуру якого наведено на рисунку 3.6.

А логічна структура проекту розподілена на фронтенд та бекенд компоненти, що взаємодіють за допомогою Application Programming Interface та розташовані в репозиторії на github. Структуру наведено на рисунку 3.7.

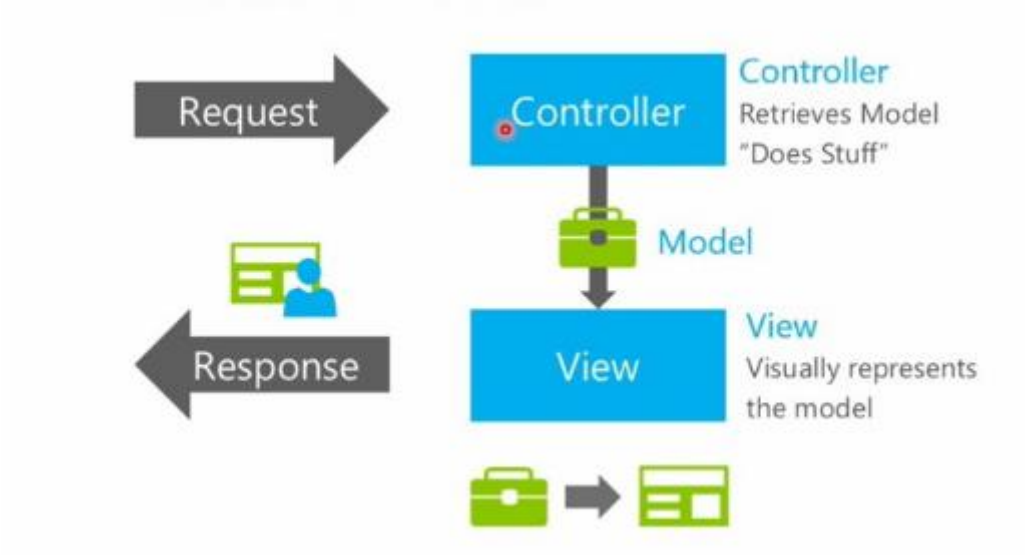


Рисунок 3.6 – Структура паттерну, що використовуэться при створенні програмного забезпечення для оцінювання розміру застосунків, що створюються з використанням фреймворку .NET

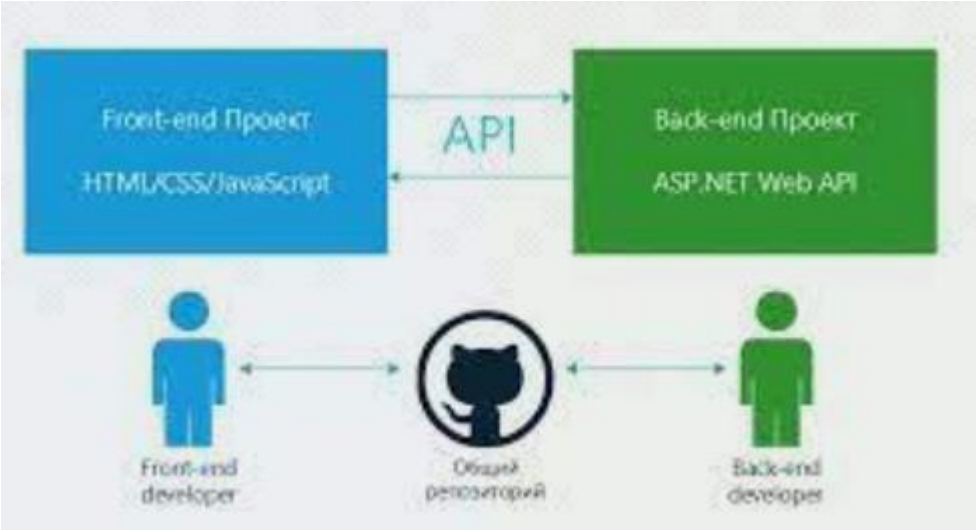


Рисунок 3.7. - Логічна структура ПЗ для оцінювання розміру застосунків, що створюються з використанням фреймворку .NET

Атрибути та методи основних класів наведено на рисунку 3.8

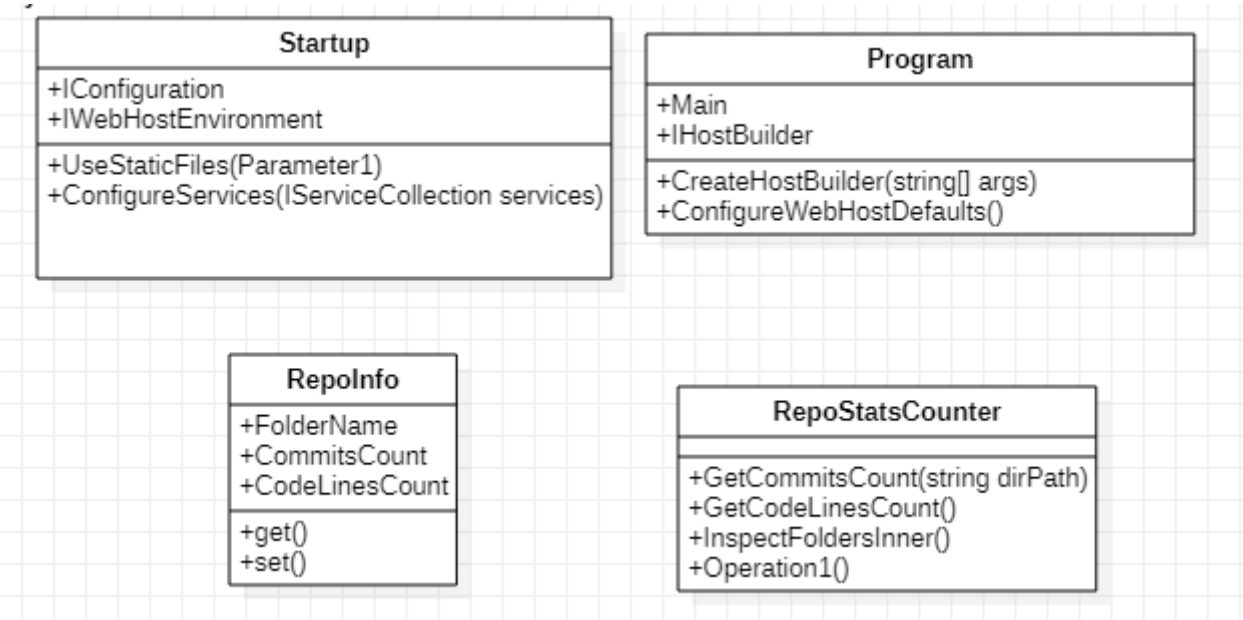


Рисунок 3.8 Базові класи застосунку

Взаємовідносини класів наведено на діаграмі класів на рисунку 3.9.

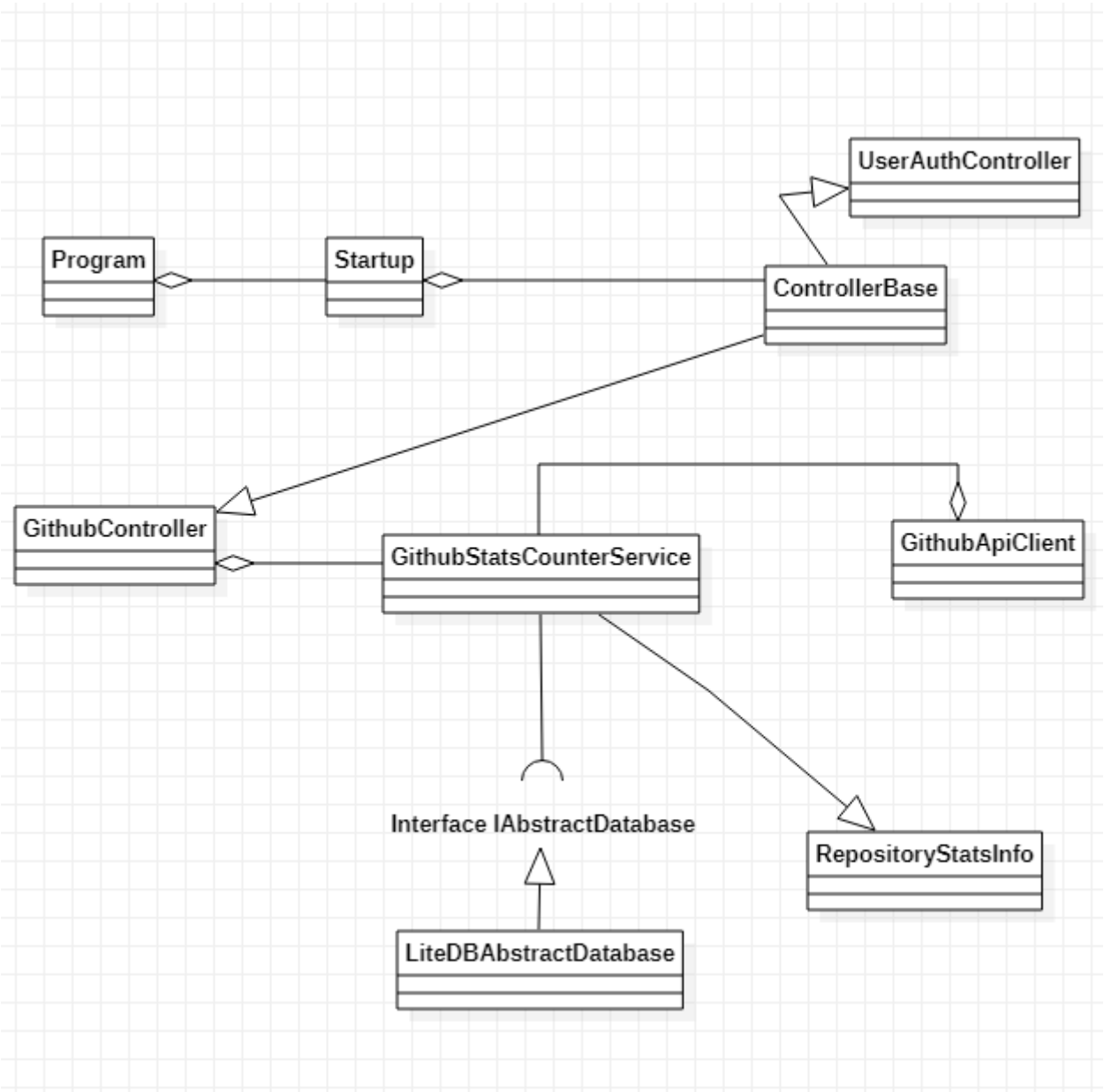


Рисунок 3.9 – Діаграма класів програмного забезпечення оцінювання розміру застосунків, що створюються з використанням фреймворку .NET

3.3. Робочий проект

3.3.1 Вибір мови та середовища розробки програмного забезпечення

Враховуючи те, що я відмінно володію навичками створення програмного забезпечення за допомогою .Net було прийнято рішення розробки програмного забезпечення саме за допомогою цього фреймворку , також важливим є те, що в ньому є наступні переваги:

- єдині засоби API для розробки програм різними мовами;

- простота стикування різномовних модулів;

- багато тисяч готових до вживання класів, що реалізують різні алгоритми, скорочують терміни розробки нових програм та підвищують надійність цих програм;

- установка програм під .NET не потребує програм-інсталяторів, робиться просте копіювання програми в потрібну папку.

Як наслідок, при установці не вносяться жодні записи до реєстру Windows, тому після видалення таких програм у реєстрі не залишається сміття. Було прийнято рішення розробки програмного забезпечення саме за допомогою цього фреймворку.

3.3.2 Кодування та випробування ПЗ

За результатами проведеного моделювання було створено програмне забезпечення зі структурою, що наведена на рисунку 3.10.

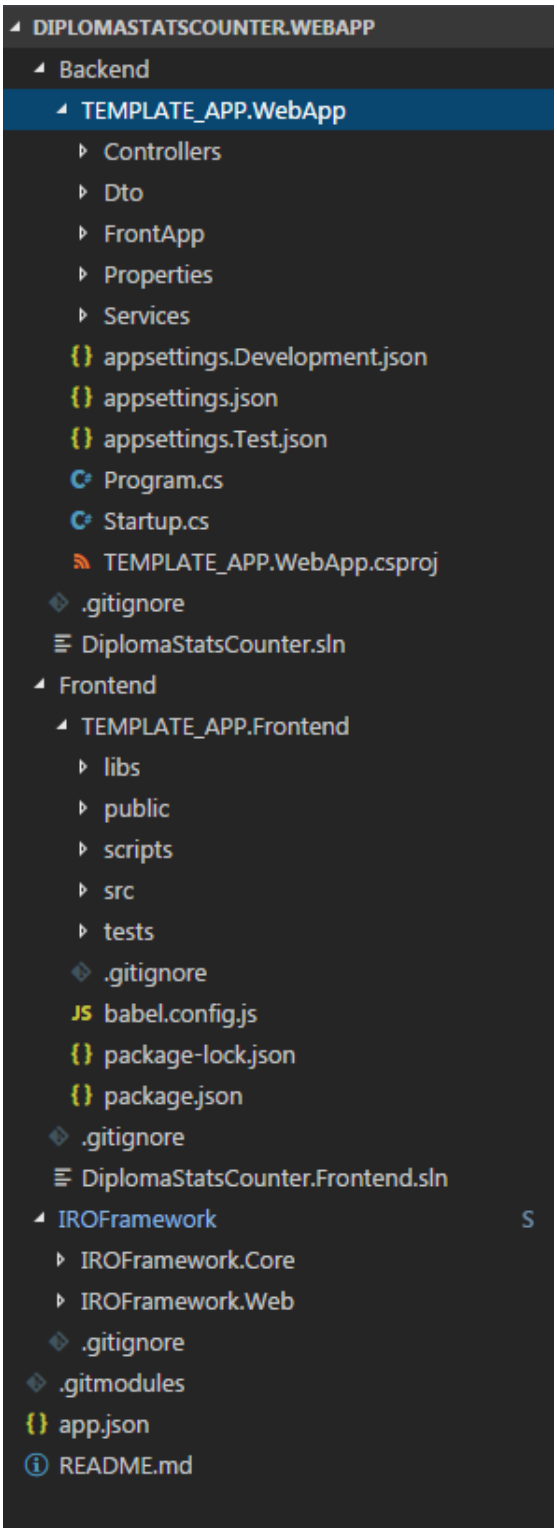


Рисунок 3.9. Структура та склад розробленого програмного забезпечення ПЗ для оцінювання розміру застосунків, що створюються з використанням фреймворку .NET

Текст основних модулів програмного забезпечення наведено у додатку Б, а повний код проекту доступний за посиланням https://github.com/ExtremeDotneting/DiplomaStatsCounter.WebApp.

3.3Тестування

Тестування розробленого програмного забезпечення було проведено за допомогою Unit тестів, текст яких наведено нижче.

```
import TestsApi from "../libs/testsApi/testsApi";
import dialogs from "../src/js/dialogs"
import SignInPage from '@/pages/SignInPage'
import Vue from 'vue'

TestsApi.registerTest("DialogTests_HtmlElementContent", async function () {
    let el = document.createElement("p");
    el.innerHTML = "HI!!!!!!!";
    await dialogs.showDialog({
        htmlElementContent: el,
        maxWidth: 600
    });
});

TestsApi.registerTest("DialogTests_CreatedComponentContent", async function () {
    let CompClass = Vue.extend(SignInPage);
    let instance = new CompClass( /* here can configure it */);
    await dialogs.showDialog({
        createdComponentContent: instance,
        maxWidth: 600
    });
});

TestsApi.registerTest("DialogTests_TypeOfComponentContent", async function () {
    await dialogs.showDialog({
        typeOfComponentContent: SignInPage,
        fullscreen: true
    });
});

TestsApi.registerTest("DialogTests_AwaitTest", async function () {
    var index = 1;
    var promiseRes = await dialogs.showDialog({
        title: "Form 1",
        text: "Form text 1",
        maxWidth: 1000
    });
    console.log("Form #" + index + ":\n" + promiseRes);
    index++;

    promiseRes = await dialogs.showDialog({
        title: "Form 2",
```

```
        text: "Form text 2",
        fullscreen: true
    });
    console.log("Form #" + index + ":\n" + promiseRes);
    index++;

    promiseRes = await dialogs.showDialog({
        title: "Form 3",
        maxWidth: 500
    });
    console.log("Form #" + index + ":\n" + promiseRes);
    index++;

});

TestsApi.registerTest("DialogTests_FS", async function () {
    var index = 1;
    var promiseRes = await dialogs.showDialog({
        title: "Fullscreen form",
        text: "Form text",
        fullscreen: true
    });
    console.log("Form #" + index + ":\n" + promiseRes);
});

TestsApi.registerTest("DialogTests_NotFS", async function () {
    var index = 1;
    var promiseRes = await dialogs.showDialog({
        title: "1000px width form",
        maxWidth: 1000
    });
    console.log("Form #" + index + ":\n" + promiseRes);

});

TestsApi.registerTest("DialogTests_Multiple", function () {
    dialogs.showDialog({
        title: "Form 1",
        text: "Open many forms.",
        maxWidth: 1000
    });
    dialogs.showDialog({
        title: "Form 2",
        text: "Open many forms.",
        maxWidth: 800
    });
    dialogs.showDialog({
        title: "Form 3",
        text: "Open many forms.",
        maxWidth: 600
```

```
    });
    dialogs.showDialog({
        title: "Form 4",
        text: "Open many forms.",
        maxWidth: 500
    });
    dialogs.showDialog({
        title: "Form 5",
        text: "Open many forms.",
        maxWidth: 400
    });

});
```

Розроблене програмне забезпечення було успішно випробовано згідно з розробленого документа «Програма і методика випробування», який представлено у додатку В.

За результатами розробки та тестування можна зробити висновок , що реалізоване програмне забезпечення повною мірою відповідає поставленому завданню та реалізує наступні функціональні характеристики:

- авторизація за протоколом OAuth з використанням GitHub форму якої наведено на рисунку 3.10.

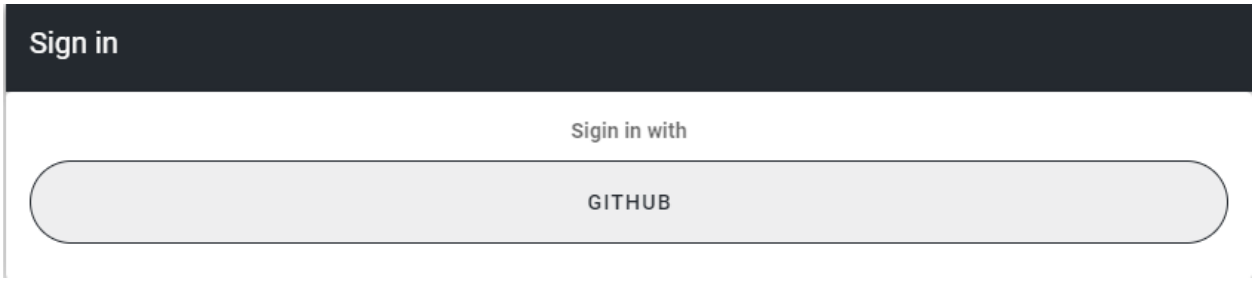


Рисунок 3.10 – Форма авторизації програмного забезпечення ПЗ для оцінювання розміру застосунків, що створюються з використанням фреймворку .NET

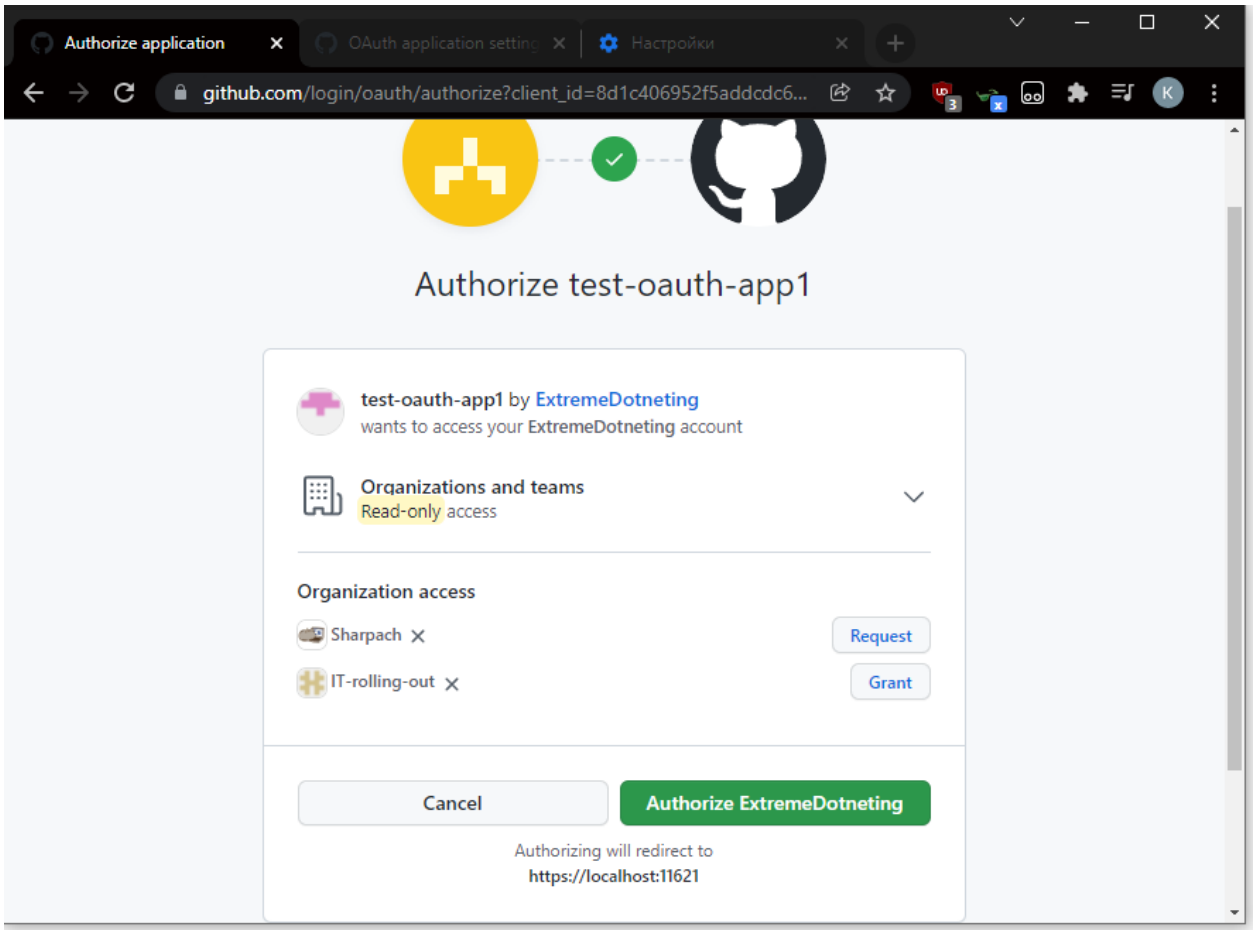


Рисунок 3.11– Сторінка статистики облікового запису користувача з Github програмного забезпечення ПЗ для оцінювання розміру застосунків, що створюються з використанням фреймворку .NET

На сторінці "Мої репозиторії", яку наведено на рисунку 3.12 можна вибрати репозиторій зі списку, клікнути та перейти на статистику по ньому, або подивитися усі репозиторії, що доступні поточному користувачу. Є можливість додати або прибрати резозиторії, що не підходять до вибірки по тим чи іншим причинам.

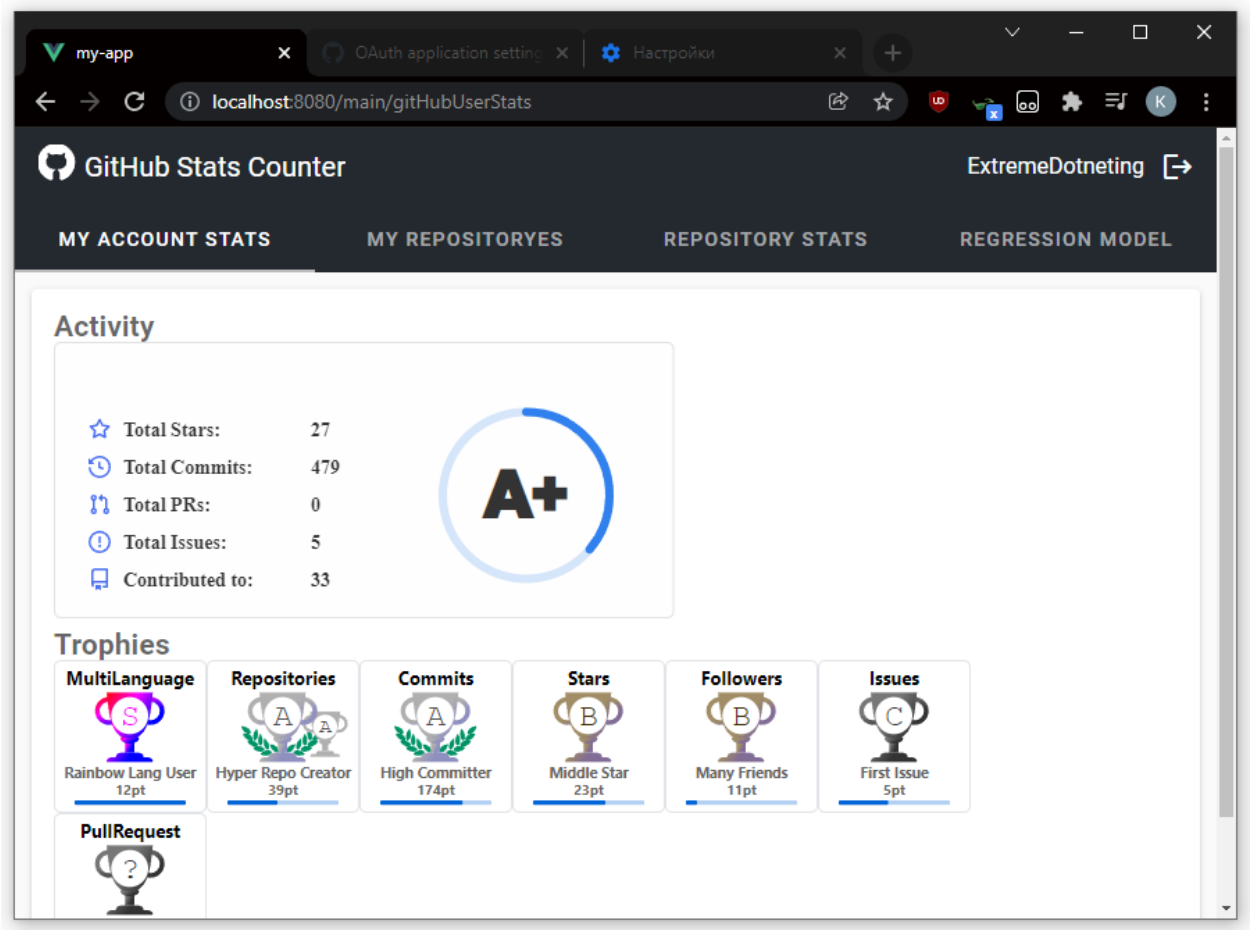


Рисунок 3.12 – Сторінка статистики репозиторіїв користувача програмного забезпечення ПЗ для оцінювання розміру застосунків, що створюються з використанням фреймворку .NET

На рисунку 3.12 наведено сторінку статистики, де зверху можна ввести метрики проекту та подивитися результати оцінювання розміру майбутнього програмного забезпечення, його прогнозні та довірчі інтервали.

.

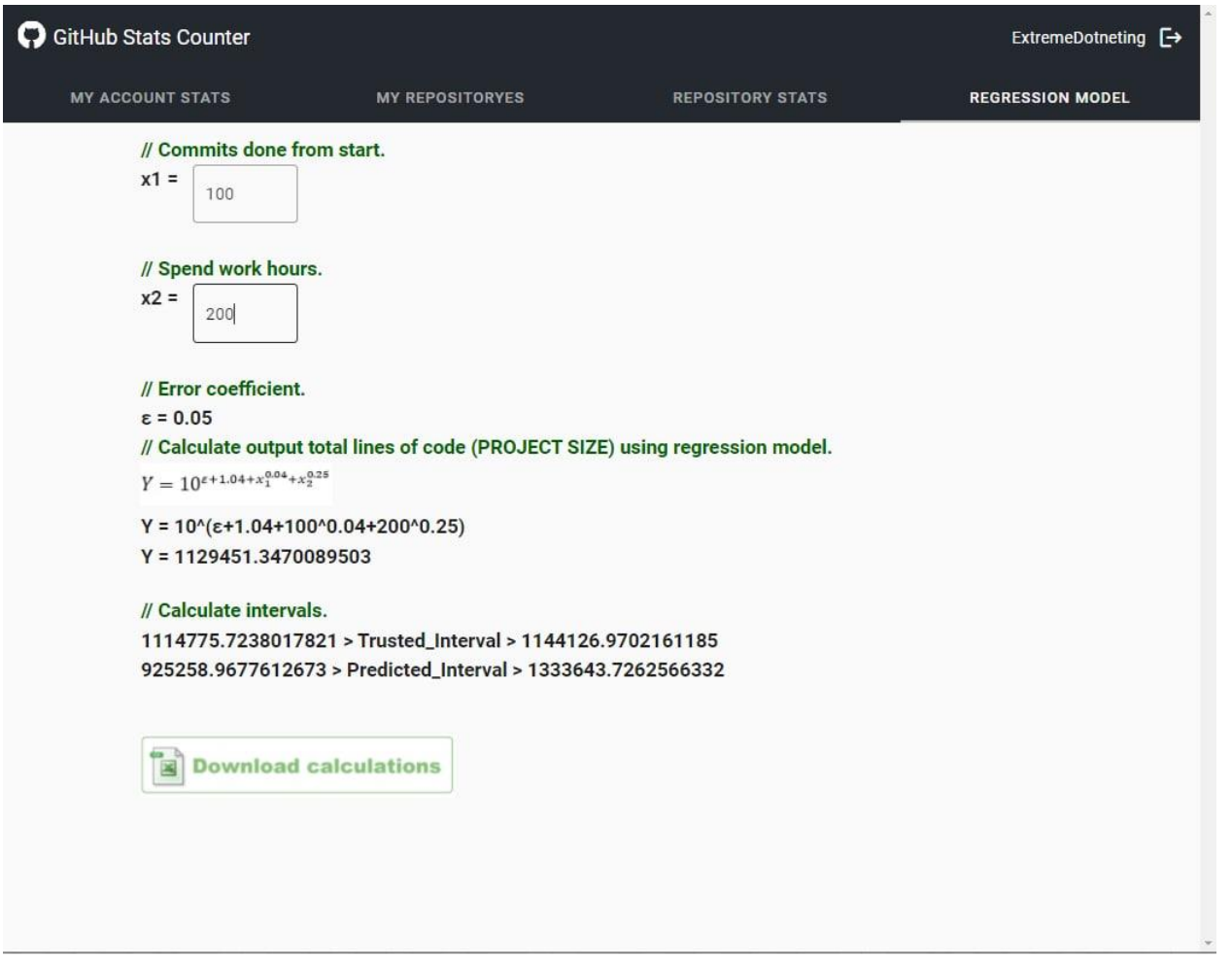


Рисунок 3.12 Розрахунок розміру з використанням математичної моделі програмного забезпечення ПЗ для оцінювання розміру застосунків, що створюються з використанням фреймворку .NET

4 РОЗРАХУНОК ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ ВПРОВАДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ОЦІНЮВАННЯ РОЗМІРУ ЗАСТОСУНКІВ, ЩО СТВОРЮЮТЬСЯ З ВИКОРИСТАННЯМ ФРЕЙМВОРКУ .NET

Актуальність проблеми вибору програмних інструментальних засобів графічного дизайну викликало значний обсяг публікацій з пропозиціями її розв’язання. Головним чином – це вербальні моделі з рекомендаціями, на які характеристики програмних продуктів треба звертати увагу при виборі та їх оцінювання. Вдалим прикладом такої роботи є стаття Ф. Тейксейра [36], де процес вибору розглянуто по кроках від обґрунтування потреби в інструментальних засобах з подальшим визначенням цілей, формулюванням вимог до прийняття власного рішення з вибору. Втім, принаймні два фактори обмежують ефективність такого підходу. По-перше, рекомендації базуються на суб’єктивному, хоча і кваліфікованому, але обмеженому досвіді застосування. По-друге, вибір має бути зроблений за результатами відповідей на велике число питань, які фахівець, що обирає, може розуміти і розв’язувати суб’єктивно через обмеженість досвіду.

В зв’язку з актуальністю даного питання для студентів, що навчаються за напрямком «Програмне забезпечення автоматизованих систем» та з причини відсутності інформаційного забезпечення студентів з даної дисципліни, в рамках даної магістерської роботи, виконується розробка програмного забезпечення для оцінювання якості ПЗ графічного дизайну.

Проаналізувавши вище зазначені проблеми можливо зробити висновок про доцільність розробки моделі якості ПЗ графічного дизайну. В цій роботі, для інформаційної підтримки і зменшення суб’єктивної складової при виборі програмних інструментальних засобів графічного дизайну, запропоновано використовувати методологію, що базується на розробленій моделі якості, специфічній для даного виду ПЗ.

Найбільш важливим моментом для розробника, з економічної точки зору, є процес формування вартості програмного продукту. Очевидно, що він являє собою дуже специфічний товар з безліччю особливостей.

Створення програмного продукту вимагає одноразових витрат на його розробку, придбання необхідних технічних засобів і поточних витрат на функціонування продукту. Економія від функціонування програмного продукту визначається з урахуванням витрат на його експлуатацію. Відношення цієї економії до витрат на створення програмного продукту характеризує економічну ефективність капітальних вкладень. Економічні показники визначаються діючим на момент розрахунку оптовим цінам, тарифам і ставкам заробітної плати.

4.1 Розрахунок витрат на створення і експлуатацію програми

Витрати на розробку продукту складаються з витрат на зарплату розробника, на зборку ЕОМ, на якій виконується розробка, на експлуатацію цієї ЕОМ, на засоби розробки та витрат на матеріали комплектуючі.

Розробка програмного забезпечення виконується програмістом, місячний оклад якого складає 5000 грн. Додаткова заробітна плата складає 20% від основної. Виходячи з цього, основна і додаткова заробітна плата розроблювана 6000 грн/міс , а вартість сучасного ПК складає 10800 грн. (приведена вартість машини на базі Intel core i3).При вартості кіловат-години електроенергії рівної 0,90 грн., розрахується вартість розробки програми. Витрати на допоміжні матеріали приведені в табл. 5.1.

Таблиця 5.1 – Витрати на допоміжні матеріали

Пункти витрат	**Сума (грн.)**
Папір	80
Заправлення картриджа до принтера	160
CD-диск	7
Непередбачені витрати	300
Разом матеріали і комплектуючі	547

Вартість програми оцінювання розміру розрахуємо по формулі:

$$Спр = (Ззп + Зсз + Ззг + Зе) * Т + Зм \text{ ,де}$$

$Т$ – тривалість розробки (міс);

$З_{зп}$ – основна і додаткова заробітна плата обслуговуючого персоналу, грн.:

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

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

$З_{е}$ – витрати на електроенергію;

$З_{м}$ – витрати на основні і допоміжні матеріали;

$З_{е}$ при споживанні потужності 0,5 кВт, тривалості роботи за місяць, рівної 22*8=176 год., вартості кіловат-години електроенергії 0,90 грн..

$$Зе = 176 * 0,90 * 0,5 = 79,2$$

Таблиця 5.2 – Витрати на розробку ПЗ.

Найменування витрат	Одиниця	Кількість
Тривалість розробки	міс.	1,5
Основна і додаткова заробітна	грн.	6000
Відрахування на соціальні	грн.	1140
Загальногосподарські витрати	грн.	250
Витрати на основні і допоміжні	грн.	547
Витрати на електроенергію	грн.	79,2

Відповідно вартість програми:

$$C_{пр} = (6000 + 1140 + 250 + 79,2) * 1,5 + 547 = 11750,80 \text{ грн.}$$

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

$$A_{об} = 10800 * 0,6 = 6480 \text{ грн.}$$

У масштабах підприємства річні витрати на основні і допоміжні матеріали (гнучкі диски, папір) визначаються в розмірі 5% вартості основного устаткування:

$$B_{м} = 10800 * 0,05 = 540 \text{ грн.}$$

Річний обсяг робіт ПК у годинах визначається в такий спосіб:

$$Ф_{м} = 264,5 * Тз$$

де $T_з$ – середнє місячне завантаження устаткування (близько 4 годин);

264,5 – середня кількість робочих днів у році.

Отже, річний обсяг роботи ПК складе:

$$Ф_{м} = 264,5 * 4 = 1058 \text{ годин}$$

Витрати електроенергію $З_е$ при 1013 годинах роботи устаткування в рік складуть

$$З_{е} = 1058*0,5*0,90 = 476,10 \text{ грн.}$$

Експлуатаційні витрати для ПК за рік складатимуть:

$$З_{зр} = 6480+540+476,10 = 7496,10 \text{ грн.}$$

Отже, у перший рік витрати на створення й експлуатацію програми складуть:

$$З_{се} = 11750,80 + 7496,10 = 19246,90 \text{ грн.}$$

7.3 Економічна ефективність розробки і впровадження програми

Основним показником економічної ефективності функціонування програмного продукту є підвищення ефективності керування інформацією у вигляді зниження витрат на керування при одночасному збільшенні швидкості і якості одержання потрібного результату.

Крім багатьох інших негативних ефектів, ручна обробка інформації спричиняє наступні негативні економічні ефекти:

— високі витрати на складування паперових документів (сейфи, шафи, папки й ін.);

— підвищені витрати на канцтовари;

— витрати, пов'язані з роботою виявлення раніше допущених помилок (людський фактор).

До числа основних факторів, що визначають приріст прибутку в зв'язку з упровадженням програми, відносяться:

— підвищення продуктивності праці;

— вивільнення робочого часу.

Крім того, не піддається прямій грошовій оцінці підвищення оперативності керування, якість одержуваних результатів, поліпшення організації праці і т.д.

Обов'язковою умовою визначення економічної ефективності програми для агенства є порівнянність усіх показників у часі, за цінами й іншими нормами, використовуваними для визначення показників, за змістом і колом елементів витрат.

Визначимо пряму економічну ефективність, ґрунтуючись на тому, що впровадження програмного продукту вивільняє 0,6 працівника (за експертною оцінкою фахівців підприємства).

Зарплата 0,6 працівника в рік складає:

$$(6000 * 12) * 0,6 = 43200,00 \text{ грн.}$$

Річний економічний ефект розраховується по формулі:

$$Э_{год} = \Delta C_n - E_n * k ,$$

де ΔC_n – вивільнені кошти після впровадження програмного продукту *43200,00 грн.)* мінус експлуатаційні витрати (*7496,10 грн.);*

E_n – коефіцієнт ефективності (дорівнює коефіцієнту амортизації - 0,6);

k – одноразові витрати на впровадження продукту (*11750,80 грн*.).

$$Э_{год} = 35703,90 – (0,6 * 11750,80) = 28653,42 \text{ грн.}$$

Строк окупності програми розраховується по формулі:

$$T = k / \Delta C_n = 11750,80 / 28653,42 = 0.41 \approx 5 \text{ (місяців)}$$

Отже строк окупності програмного продукту складає приблизно 5 місяців.

Витрати на перший рік витрати на створення й експлуатацію програми складуть: *43200,00 грн.* Строк окупності програмного продукту складає приблизно 5 місяців.

ОХОРОНА ПРАЦІ

Питання безпечної життєдіяльності людини необхідно вирішувати на всіх стадіях життєвого циклу, будь це розробка, впровадження в життя або експлуатація програми.

Забезпечення безпечної життєдіяльності людини в значній мірі залежить від правильної оцінки небезпечних, шкідливих виробничих факторів. Однакові по складності зміни в організмі людини можуть бути викликані різними причинами. Це можуть бути які-небудь фактори виробничого середовища, надмірне фізичне і розумове навантаження, нервово-емоційна напруга, а також різне сполучення цих причин.

У даному розділі вирішується питання безпечної життєдіяльності на стадіях розробки та реалізації прграми.

5.1Аналіз шкідливих та небезпечних факторів при розробці програми

Небезпечні і шкідливі виробничі фактори[11] по природі виникнення поділяються на наступні групи:

–фізичні;

–хімічні;

–психофізіологічні;

–біологічні.

У приміщенні офіса на програмістаможуть негативно діяти наступні фізичні фактори:

–підвищена і знижена температура повітря;

–надмірна запиленість і загазованість повітря;

–підвищена і знижена вологість повітря;

–недостатня освітленість робочого місця;

–перевищуючі припустимі норми шуму;

–підвищений рівень статичної електрики;

–небезпека ураження електричним струмом;

–бляклість екрана дисплея.

До хімічно небезпечних факторів, що постійно діють на програміставідносяться наступні:

–виникнення, у результаті іонізації повітря при роботі комп'ютера, активних часток.

Біологічні шкідливі виробничі фактори в даному приміщенні відсутні.

До психологічно шкідливих факторів, що впливають на оператора протягом його робочої зміни можна віднести наступні:

–нервово - емоційні перевантаження;

–розумові перевантаження;

–перевантаження зорового аналізатора.

Далі більш докладно розглянуті небезпечні і шкідливі фактори, що впливають на оператора ПЕОМ, що виникли в зв'язку з розробкою даної системи та способи їх запобігання.

5.2 Розробка заходів щодо зменшення впливу шкідливих та небезпечних факторів

Мікроклімат виробничих приміщень - це клімат внутрішнього середовища цих приміщень, що визначається діючими на організм людини сполученнями температури, вологості і швидкості руху повітря

Лабораторія є приміщенням 1 категорії (виконуються легкі фізичні роботи), тому повинні дотримуватися наступні вимоги:

- оптимальна температура повітря - 220С (припустима - 20-240С), оптимальна відносна вологість - 40 -60% (припустима - не більш 75%) , швидкість руху повітря не більш 0.1 м/с.

Для створення й автоматичної підтримки в офіса незалежно від зовнішніх умов оптимальних значень температури, вологості, чистоти і швидкості руху повітря, у холодний час року використовується водяне опалення, у теплий час року застосовується кондиціонування повітря. Кондиціонер являє собою вентиляційну установку, яка за допомогою приладів автоматичного регулювання підтримує в приміщенні задані параметри повітряного середовища.

5.3 Освітлення робочого місця

Робота, виконувана з використанням обчислювальної техніки, має наступні недоліки:

- ймовірність появи прямого блиску;

- погіршена контрастність між зображенням і фоном;

- відбивання екрана.

У зв'язку з тим, що природне освітлення слабке, на робочому місці повинне застосовуватися також штучне освітлення. Далі буде зроблений розрахунок штучного освітлення.

Розміщення світильників визначається наступними розмірами:

H = 3 м. - висота приміщення

hc = 0,25 м. - відстань світильників від перекриття

hn = *H* - *hc* = 3 - 0,25 = 2,75 м. - висота світильників над підлогою

hp = висота розрахункової поверхні = 0,7 м (для приміщень, зв'язаних з роботою ПЕОМ)

h = *hn* - *hp* = 2,75 - 0,7 = 2,05 - розрахункова висота.

Світильники типу ЛДР (2х40 Ут). Довжина 1,24 м, ширина 0,27 м, висота 0,10 м.

L - відстань між сусідніми світильниками (рядами люмінесцентних світильників), *La* (по довжині приміщення) = 1,76 м, *Lв* (по ширині приміщення) = 3 м.

l - відстань від крайніх світильників або рядів світильників до стіни, *l* = 0,3 - 0,5*L*.

la = 0,5La, lв = 0,3Lв

la = 0,88 м., *lв* = 0,73 м.

Світильники з люмінесцентними лампами в приміщеннях для роботи рекомендують установлювати рядами.

Метод коефіцієнта використання світлового потоку призначений для розрахунку загального рівномірного освітлення горизонтальних поверхонь при відсутності великих предметів, що затемнюють. Потрібний потік ламп у кожнім світильнику

$$Ф = E * r * S * z / N * \eta,$$

де *E* - задана мінімальна освітленість = 300 лк., так як розряд зорових робіт = 3

r - коефіцієнт запасу = 1,3 (для приміщень, зв'язаних з роботою ПЕОМ)

S - освітлювана площа = 30 м^2.

z - характеризує нерівномірне освітлення, *z = Ecp / Emin* - залежить від відношення $\lambda = L/h$, $\lambda a = La/h = 0,6$, $\lambda y = Lв/h = 1,5$. Так як. λ перевищує припустиме значення, то *z=1,1* (для люмінесцентних ламп).

N - число світильників, до розрахунку. Спочатку намічається число рядів *n*, що підставляється замість *N*. Тоді *Ф* - потік ламп одного ряду.

N = Ф/Ф1, де *Ф1* - потік ламп у кожному світильнику.

η - коефіцієнт використання. Для його знаходження вибирають індекс приміщення *i* і приблизно оцінюються коефіцієнти відображення поверхонь приміщення $\rho стл.$ (стелі) = 70%, $\rho ст.$ (стіни) = 50%, $\rho р.$ (підлоги) = 30%.

$$Ф = 300 * 1{,}3 * 25 * 1{,}1/2 * 0{,}3 = 21450 \text{ лм.}$$

Я пропоную установити два світильники в ряд. Світильники вміщаються в ряд, тому що довжина ряду близько 4 м. Застосовуємо світильники з лампами 2х40 Вт із загальним потоком 5700 лм.

5.4 Вплив шуму. Захист від шуму

У приміщеннях з низьким рівнем загального шуму, яким є лабораторія де працює програміст, джерелами шумових перешкод можуть стати вентиляційні установки, кондиціонери або периферійне устаткування для ЕОМ (плотери, принтери й ін). Тривалий вплив цих шумів негативно позначаються на емоційному стані персоналу.

Відповідно ДО ДЕРЖСТАНДАРТУ 12.1.003-76 ССБТ еквівалентний рівень звуку не повинний перевищувати 50 дБА. Для того, щоб домогтися цього рівня шуму рекомендується застосовувати звукопоглинаюче покриття стін.

Як міри по зниженню шуму можна запропонувати наступне:

– облицювання стелі і стін звукопоглинаючим матеріалом (знижують шум на 6-8 дб);

– екранування робочого місця (постановкою перегородок, діафрагм);

– установка в комп'ютерних приміщеннях устаткування, що робить мінімальний шум;

– раціональне планування приміщення.

Тому я пропоную для зменшення шуму в офіса використовувати замість матричного принтера, що робить багато шуму, більш тихий – лазерний принтер.

Захист від шуму варто виконувати відповідно до ДСТ 12.1.003-76, а звукоізоляція огороджуючих конструкцій повинна відповідати вимогам розділу Снип 11-12-77 «Захист від шуму. Норми проектування».

5.5 Електронебезпека. Статична електрика

Приміщення по небезпеці ураження електричним струмом можна віднести до 1 класу, тобто це приміщення без підвищеної небезпеки (сухе, безпильне, з нормальною температурою повітря, ізольованими підлогами і малим числом заземлених приладів).

На робочому місці програміста з всього устаткування металевим є лише корпус системного блоку комп'ютера, але тут використовуються системні блоки, що відповідають стандартові фірми IBM, у яких крім робочої ізоляції передбачений елемент для заземлення і провід з жилою, що заземлює, для приєднання до джерела живлення. Таким чином, устаткування обмінного пункту виконано по класу 1

Електробезпечність приміщення забезпечується відповідно до ПУЭ. Небезпечний і шкідливий вплив на людей електричного струму, електричної дуги й електромагнітних полів виявляється у виді електротравм і професійних захворювань.

Ступінь небезпечного і шкідливого впливу на людину електричного струму, електричної дуги й електромагнітних полів залежить від:

– Роду і величини напруги і струму

– Частоти електричного струму

– Шляхи струму через тіло людини

– Тривалості впливу на організм людини

Електробезпечність у приміщенні офіса забезпечується технічними способами і засобами захисту, а так само організаційними і технічними заходами.

Розглянемо основні причини ураження людини електричним струмом на робочому місці:

– Дотик до металевих неструмоведучих частин (корпусові, периферії комп'ютера), що можуть виявитися під напругою в результаті ушкодження ізоляції.

– Нерегламентоване використання електричних приладів.

Відсутність інструктажу співробітників по правилах електробезпечності.

На протязі роботи на корпусі комп'ютера накопичується статична електрика. На відстані 5-10 см від екрана напруженість електростатичного поля складає 60-280 кв/м, тобто в 10 разів перевищує норму 20 кв/м. Для зменшення напруженості застосовувати зволожувачі і нейтралізатори, антистатичне покриття підлоги.

Крім того, при несправності яких-небудь блоків комп'ютера корпус може виявитися під струмом, що може привести до електричних травм або

електричних ударів. Для усунення цього я пропоную забезпечити приєднання металевих корпусів устаткування до жили, що заземлює.

Електробезпечність забезпечується відповідно до ДСТ 12.1. 030. - 81. Небезпечний і шкідливий вплив на людей електричного струму виявляється у виді електротравм і професійних захворювань.

Електробезпечність у офіса забезпечується технічними способами і засобами захисту, а так само організаційними і технічними заходами.

Розглянемо основні причини ураження програміста електричним струмом на робочому місці:

1. Дотик до металевих неструмоведучих частин системного блоку ПЕОМ, які можуть здійснюватись під напругою в результаті ушкодження ізоляції.

2. Заборонене використання електричних приладів, таких як електричні плити, чайники, обігрівачі.

Усі струмоведучі частини ЕОМ ізольовані, тому випадковий дотик до струмоведучих частин виключено.

Для забезпечення захисту від ураження електричним струмом при дотику до металевих неструмоведучих частин, що можуть виявитися під напругою в результаті ушкодження ізоляції, я рекомендую застосовувати захисне заземлення.

Заземлення корпуса ЕОМ забезпечено підведенням жили, що заземлює, до живильних розеток. Опір заземлення 4 Ом, згідно (ПУЭ) для електроустановок з напругою до 1000 В.

5.6 Організаційні і технічні заходи щодо забезпечення електробезпеки

Основним організаційним елементом є інструктаж і навчання безпечним методам праці, а також перевірка знань правил безпеки й інструкцій відповідно до займаної посади стосовно до виконуваної роботи.

При проведенні незапланованого і планового ремонту обчислювальної техніки виконуються наступні дії:

– Відключення комп'ютера від мережі

– Перевірка відсутності напруги

Після виконання цих дій проводиться ремонт несправного устаткування.

Якщо ремонт проводиться на струмоведучих частинах, що знаходяться під напругою, то виконання роботи проводиться не менш чим двома особами з застосуванням електрозахисних засобів.

5.7 Пожежобезпека

Ступінь вогнестійкості будинків приймається в залежності від їхнього призначення, категорії по вибухопожежній і пожежній небезпеці, по поверховості, площі поверху в межах пожежного відсіку.

Будинок, у якому знаходиться лабораторія по пожежній небезпеці будівельних конструкцій відноситься до категорії К1 (малопожежонебезпечні), оскільки тут присутні займисті (книги, документи,

меблі, оргтехніка і т.д.) і тяжкогорючі речовини (сейфи, різне устаткування і т.д.), що при взаємодії з вогнем можуть горіти без вибуху.

По конструктивних характеристиках будинок можна віднести до будинків з несучими і огороджуючими конструкціями із природних або штучних кам'яних матеріалів, бетону або залізобетону, де для перекриттів допускається використання дерев'яних конструкцій, захищених штукатуркою або важкогорючими листовими, а також плитними матеріалами.

Отже, ступінь вогнестійкості будинку можна визначити як третю (III).

Приміщення офіса по функціональній пожежній небезпеці відноситься до класу Ф 4.2 – вищі навчальні заклади, установи підвищення кваліфікації.

Пожежна профілактика являє собою комплекс організаційних і технічних заходів, спрямованих на забезпечення безпеки людей, на запобіганні пожежі, обмеження його поширення, а також створення умов для успішного гасіння пожежі. Для профілактики пожежі надзвичайно важлива правильна оцінка пожежонебезпеки будинку, визначення небезпечних факторів і обґрунтування способів і засобів пожежопопередження і захисту.

Одне з умов забезпечення пожежобезпеки - ліквідація можливих джерел запалення.

У офіса джерелами запалення можуть бути:

– несправне електроустаткування, несправності в електропроводці, електричних розетках і вимикачах. Для виключення виникнення пожежі з цих причин необхідно вчасно виявляти й усувати несправності, проводити плановий огляд і вчасно усувати всі несправності;

– несправні електроприлади. Необхідні міри для виключення пожежі містять у собі своєчасний ремонт електроприладів, якісне виправлення поломок, не використання несправних електроприладів;

– обігрівання приміщення електронагрівальними приладами з відкритими нагрівальними елементами. Відкриті нагрівальні поверхні можуть спричинити пожежу, тому що в приміщенні знаходяться паперові документи і довідкова література у виді книг, посібників, а папір – легкозаймистий предмет. З метою профілактики пожежі пропоную не використовувати відкриті обігрівальні прилади в приміщенні офіса;

– коротке замикання в електропроводці. З метою зменшення імовірності виникнення пожежі внаслідок короткого замикання необхідно, щоб електропроводка була схованою.

– влучення в будинок блискавки. У літній період під час грози можливе влучення блискавки внаслідок чого можливий пожежа. Щоб уникнути цього я рекомендую установити на даху будинку блискавковідвід;

– недотримання мір пожежної безпеки і паління в приміщенні також може спричинити пожежу. Для усунення загоряння в результаті паління в приміщенні офіса пропоную категорично заборонити паління, а дозволити тільки в строго відведеному для цього місці.

З метою запобігання пожежі пропоную проводити з інженерами, що працюють у офіса, протипожежний інструктаж, на якому ознайомити працівників із правилами протипожежної безпеки, а також навчити використанню первинних засобів пожежогасіння.

У цьому розділі було досліджено питання безпечної життєдіяльності на стадіях розробки програми. Було вирішено проблему освітлення при розробці програми.

6 ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

Охорона навколишнього середовища являє собою форму відносин між суспільством і природою. Вона здійснюється різними засобами: економічними, правовими, науково-технічними, санітарно-гігієнічними, біологічними та іншими. Охорона навколишнього середовища на підприємстві характеризується комплексом вжитих заходів, які спрямовані на попередження негативного впливу діяльності підприємства на навколишнє середовище, що забезпечує сприятливі та безпечні умови праці.

В загальному випадку проблема охорони навколишнього середовища зводиться до вирішення двох завдань:

- організації раціонального природокористування;
- забезпечення чистоти природних (екологічних) систем. При здійсненні різних видів економічної діяльності суб'єкти.

6.1 Забруднення навколишнього середовища

Під забрудненням навколишнього середовища слід розуміти зміну властивостей середовища (хімічних, механічних, фізичних, біологічних і пов'язаних з ними інформаційних), що відбуваються в результаті природних, або штучних процесів і призводять до погіршення функцій середовища по відношенню до будь-якого біологічного, або технологічного об'єкту. Використовуючи різні елементи навколишнього середовища у своїй діяльності, людина змінює її якість. Часто ці зміни виражаються в не сприятливій формі забруднення.

Інформаційні та телекомунікаційні технології, включивши в себе екологію в якості гуманних підвалин розвитку, перетворились на ідею інформаційного суспільства, стали способом життя людства, запорукою нового циклу розвитку цивілізації та планети.

Інформаційні технології сьогодні є екологічнішими за більшість інших видів активної людської діяльності, проте їх ще не можна назвати справді екологічними. Скажімо, ефективність інформаційних мереж напряму залежить від кількості користувачів, тобто, від кількості комп'ютерів, включених до мережі. Але для виготовлення одного звичайного персонального комп'ютера потрібно від 15 до 19 тонн матеріалів. Це порівнювано з 25 тоннами, потрібними для виготовлення автомобіля. На кожен функціонуючий комп'ютер (використовуваний в середньому протягом 4 років) припадає 1,5 комп'ютери вироблених. А близько третини комп'ютерів ніколи не буває продано взагалі – через швидкість, з якою вони втрачають технологічну актуальність. Це означає, що затрачувані ресурси справді наближаються до рівня автомобіля.

Електронні пристрої містять дуже токсичні з'єднання, які, потрапляючи в навколишнє середовище, створюють серйозну небезпеку для життя людей. Так, наприклад, 22% ртуті, що видобувається щороку в усьому світі, йде саме на потреби електронної промисловості і, зокрема, міститься в мобільних телефонах. Кадмій, який є канцерогеном, використовується практично в усіх напівпровідникових пристроях. Свинець, особливо токсичний для нервової системи, міститься в акумуляторах і екранах моніторів. У міру розкладання захисних покриттів з електронних пристроїв у навколишнє середовище виділяється діоксин та інші високотоксичні з'єднання.

Стурбованість громадськості проблемами екології, а також нові, більш жорсткі закони по захисту навколишнього середовища змушують великих

виробників устаткування створювати мережі по збору, що вийшли з обігу техніку і заводи з її утилізації. Крім того, в конструкції обладнання максимально збільшується частка матеріалів, придатних для переробки. Розміри мережі з утилізації "електронного брухту" залежать від регіону і місцевого законодавства.

Вся оргтехніка включає в свій склад як органічні складові (пластик різних видів, матеріали на основі полівінілхлориду, фенолформальдегіда), так і майже повний набір металів.

Отже, звичайний комп'ютер містить як цінні метали, такі як золото, срібло, алюміній, мідь, так і небезпечні, такі як кадмій, свинець, цинк, нікель, а тому при списанні та утилізації обладнання керівнику необхідно керуватися і законодавством в області охорони навколишнього середовища.

6.2 Розробка заходів щодо зменшення забруднення

Персональні комп'ютери, ноутбуки та інша інформаційна техніка, як відомо широко використовується в галузі наукових досліджень, промисловості, а також у повсякденному домашньому користуванні. Але будь-яка техніка стрімко застаріває, їй на зміну приходять нові, більш потужні, більш сучасні ПК та оргтехніка. Поступово виникає проблема, що робити зі старою технікою, морально застарілою або з тих чи інших причин, що вийшла з ладу, яка захаращує підсобні приміщення та склади.

Утилізація оргтехніки та комп'ютерів це процес, який проводиться в кілька етапів. Найперше дію це списання обладнання безпосередньо з підприємства. Етап другий це розбір техніки і сортування отриманих матеріалів. Якщо деталі здатні служити вихідною сировиною, наприклад, кінескоп, деталі, в складі яких є дорогоцінні метали, то їх відправляють на очищення.

Як нам відомо до складу комп'ютера входить безліч металів таких як золото, срібло, алюміній, мідь та інших. Ще етапом по утилізації персональних комп'ютерів можна досягнути завдяки вторинній переробці.

Сутність даного процесу полягає в тому, що можна витягти з цієї сировини частку корисних та рідких матеріалів таких як іридію, міді та інших. Цей процес відтворити набагато легше аніж наприклад видобути тонну міді, яка міститься в тисячотонних гірських породах.

Одне з нововведень для утилізації друкованих плат придумали Співробітники з Національної фізичної офіса Великобританії, продемонстрували можливість спеціального розчину який розчинюють у гарячій воді. Дія якого зумовлює відшарування електронних компонентів.

Таким чином 90% компонентів нових друкованих плат можна використовувати знову, тоді як у випадку звичайним методам - тільки 2%.

Практично жодне підприємство не зможе самостійно утилізувати комп'ютери та оргтехніку, так як цей процес вимагає сучасного обладнання та специфічних знань. Тому довірити таку роботу можна тільки професіоналам, які мають великий досвід у даній сфері.

Проблема утилізації використаних комп’ютерів, периферійного обладнання, стає гострішою з кожним роком. Обсяги виробництва продуктів інформаційно-телекомунікаційних технологій та частота їх заміни на нові моделі примушують компанії замислюватись над проблемою біодеградації. Успіхи в цій галузі допоможуть, серед іншого, компаніям-виробникам зменшити податки, котрі вони сплачують зараз за утилізацію застарілих моделей. Останнє тим більше важливо, оскільки робить екологізацію економічно вигідною, тож спрямовує у цю сферу дедалі більше зусиль дослідників та довгострокових капіталовкладень. Таким чином, подальше

поширення інформаційних технологій не збільшить, а навпаки – зменшить техногенне навантаження на довкілля.

В кінцевому результаті виконаної роботи можна стверджувати, що вдосконалення сучасних інформаційних технологій, слід направляти не тільки для того щоб створювати людині максимально комфортні умови життя теперішнього часу. Але і для досягнення безвідходного процесу утилізації відпрацьованої техніки, не завдаючи шкоду навколишньому середовищу.

ВИСНОВКИ

Підсумовуючи підсумки роботи впевнено можна стверджувати що розробка математичної моделі для підвищення достовірності оцінювання розміру застосунків, що створюються з використанням фреймворку .Net та розробка відповідного програмного забезпечення є успішною.

Вирішено такі завдання:

проаналізовано існуючі моделі та методи оцінювання розміру застосунків, що створюються з використанням фреймворку .Net,

сформульовано постановку задачі;

побудувано математичну модель для оцінювання розміру застосунків, що створюються з використанням фреймворку .Net;

розроблено програмне забезпечення для оцінювання розміру .Net доданків, використовуючи розроблену математичну модель.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

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

2. Tan H.B.K. Conceptual data model-based software size estimation for information systems / H. B. K. Tan, Y. Zhao, H. Zhang // Transactions on Software Engineering and Methodology. – 2009. – Vol. 19. – Issue 2. – October 2009. – Article No. 4. DOI: 10.1145/1134285.1134331

3. Prykhodko S.B. Constructing the non-linear regression equation to estimate the software size of open source PHPbased information systems / S. B. Prykhodko, N. V. Prykhodko, T. G. Smykodub, A. V. Spinov // Проблеми інформаційних технологій. – 2018. – № 1 (023). – С.118-125. – ISSN 1998-7005

4. Prykhodko S. Application of the Squared Mahalanobis Distance for Detecting Outliers in Multivariate NonGaussian Data / S. Prykhodko, N. Prykhodko, L. Makarova, A. Pukhalevych // Proceedings of 14th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET), Lviv-Slavske, Ukraine, February 20–24, 2018. – P. 962-965. DOI: 10.1109/TCSET.2018.8336353

5. Boehm B. Software engineering economics / B. Boehm. – New Jersey: Prentice-Hall, 1981. – 42 p.

6. Boehm B. Software Cost Estimation with Cocomo II. New Jersey, Prentice-Hall. 2000. 544 p

7. Титов А. И. Выбор метрики размера проекта в модели оценки трудоемкости разработки программ / Титов А. И. // Intellectual Technologies on Transport. – 2016. – №1. –С.31-37.

8. Prykhodko N. V. The non-linear regression model to estimate the software size of open source java-based systems / N. V. Prykhodko, S. B. Prykhodko // Радіоелектроніка, інформатика, управління. - 2018. - № 3. - С. 158-166. - Режим доступу: http://nbuv.gov.ua/UJRN/riu_2018_3_19

9. Грешилов, А. А. Математические методы построения прогнозов [Текст] / А. А. Грешилов, В. А. Стакун, А. А. Стакун. – М.: Радио и связь, 1997. – 112 с.

10. Демиденко, Е.З . Линейная и нелинейная регрессии [Текст] / Е. З. Демиденко. – М.: Финансы и статистика, 1981. – 302 с.

11. Bates, Douglas M. Nonlinear Regression Analysis and Its Applications [Text] / Douglas M. Bates, Donald G. Watts. – Wiley, 1988. – 384 p.

12. Pardoe, Iain Applied regression modeling [Text] / Iain Pardoe. – Wiley, 2012. – 325 p.

13. Seber, George A. F. Nonlinear Regression [Text] / George A. F. Seber, C. J. Wild. – John Wiley & Sons, Inc., 2003. – 792 p.

14. Yan, Xin Linear regression analysis: theory and computing [Text] / Xin Yan, Xiao Gang Su. – Singapore: World Scientific Publishing Co. Pte. Ltd., 2009. – 328 p.

15. Айвазян, С. А. Прикладная статистика. Основы эконометрики: Учебник для вузов [Текст]: В 2 т. 2-е изд., испр. – Т. 1: Теория вероятностей и прикладная статистика / С. А. Айвазян, В. С. Мхитарян. – М.: ЮНИТИ-ДАНА, 2001. – 656 с.

16. Кобзарь, А. И. Прикладная математическая статистика. Для инженеров и научных работников [Текст] / А. И. Кобзарь. – М.: ФИЗМАТЛИТ, 2006. – 816 с.

17. Chatterjee, Samprit Handbook of Regression Analysis [Text] / Samprit Chatterjee, Jeffrey S. Simonoff. – Wiley, 2012. – 240 p.

18. Приходько, С. Б. Інтервальне оцінювання статистичних моментів негаусівських випадкових величин на основі нормалізуючих перетворень [Текст] / С. Б. Приходько // Математичне моделювання: науковий журнал. – Дніпродзержинськ: ДДТУ. – 2011. – № 1 (24). – С. 9–13.

19. Приходько, С. Б. Метод побудови нелінійних рівнянь регресії на основі нормалізуючих перетворень [Текст] : тези доп. міждерж. наук.-методич. конф. / С. Б. Приходько // Проблеми математичного моделювання. – Дніпродзержинськ: ДДТУ, 2012. – С. 31–33.

20. Ryan, Thomas P. Modern Regression Methods [Text] / Thomas P. Ryan – Wiley, 2008. – 672 p. 31 Математика и кибернетика - прикладные аспекты

21. Приходько, С. Б. Розробка нелінійної регресійної моделі тривалості програмних проектів на основі нормалізуючого перетворення Джонсона [Текст] / С. Б. Приходько, А. В. Пухалевич // Радіоелектронні і комп`ютерні системи. – – 2012. – № 4 (56). – С. 90–93.

22. Приходько, С. Б. Определение доверительных интервалов статистических моментов времени наработки между отказами устройств терминальной сети [Текст] / С. Б. Приходько, Л. Н. Макарова // Наукові праці: науково-методичний журнал. Комп`ютерні технології. – 2013. – Вип. 201, Т. 213. – С. 82–86.

23. Ponomarenko T.V., Mysko Y.M., Beregkov M.V. BUILDING THE MATHEMATICAL MODELS OF THE SOFWARE APPLICATION SIZE ESTIMATION BASED ON THE SMALL DATASETS // Матеріали IV Всеукраїнської науково-практичної інтернет-конференції студентів, аспірантів та молодих вчених за тематикою «Сучасні комп'ютерні системи та мережі в управлінні»: збірка наукових праць / Під редакцією Г.О. Райко. – Херсон: Видавництво ФОП Вишемирський В. С., 2021. – 287 с

24. I.J. Information Technology and Computer Science, 2021, 1, 1-17 Published Online February 2021 in MECS (http://www.mecs-press.org/) DOI: 10.5815/ijitcs.2021.01.01

24. Закони України: «Про охорону праці»; «Про загальнообов’язкове державне соціальне страхування від нещасного випадку на виробництві і професійного захворювання, що спричинили втрату працездатності»; «Про забезпечення санітарного та епідемічного благополуччя населення»; Кодекс цивільного захисту України; «Про дорожній рух»; «Про фізичний захист ядерних установок, ядерних матеріалів, радіоактивних відходів, інших джерел іонізуючого випромінювання». [Електронний ресурс]. – Режим доступу: http://zakon4.rada.gov.ua/laws

25. Жидецький В. Ц. Охорона праці користувачів комп'ютерів / В.Ц. Жидецький. – Львів: Афіша, 2000. – 176 с.

26. Безпека людини у життєвому середовищі: Навч. посібник / Голінько В.І., Шибка М.В., Безщасний О.В. За ред. В.І.Голінька.– 3-є вид., перероб. і доп. – Д.: Національний гірничий університет, 2004. – 187 с.

27. Безпека людини у надзвичайних ситуаціях: Навч. посібник / В.І.Голінько, С.О.Алексеєнко, М.Ф.Кременчуцький та ін.; За ред. В.І.Голінька. – 3-є вид., перероб. і доп. – Д.: Національний гірничий університет, 2004. – 160 с.

28. Безопасность производственных процессов: Справочник / С. В. Белов, В.Н. Бринза, Б.С. Векшин и др.; Под общ. ред. С.В. Белова. – М.: Машиностроение, 1985. - 448 с.

29. Закон України "Про охорону навколишнього природного середовища" N 1264-XII від 25 червня 1991 року.

ДОДАТОК А – ТЕХНІЧНЕ ЗАВДАННЯ

Вступ

Назва проекту: Розробка програмного забезпечення «DiplomaStatsCounter» для оцінювання розміру застосунків, що створюються з використанням фреймворку .Net

Умовне позначення програмного комплексу «DiplomaStatsCounter»

Розробник Мисько Юрій

Підстави для розробки

«DiplomaStatsCounter» розробляється на підставі завдання на магістерську роботу “Регресійна модель для оцінювання розміру застосунків, що створюються з використанням фреймворку .Net, та розробка програми для її реалізації” затвердженого наказом№ 1239-уч від 13.10.2021 р., що видане кафедрою ПЗАС НУК ім. адмірала Макарова.

2. Призначення розробки

2.1. Функціональне призначення

Функціональним призначенням розробки є спрощення процесу постановки задач кожному з розробників, відслідковування процесу виконання задач, відслідковуванні скільки часу витрачається на вирішення задач, пріоритизації задач, організації комунікації розробників.

2.2. Експлуатаційне призначення.

Сервіс повинний бути доступний як веб додаток.

3. Вимоги до програмного продукту

3.1. Вимоги до функціональних характеристик

3.1.1. Вимоги до складу виконуваних функцій

Користувачам програмного комплексу мають бути доступні функції:

- авторизація за протоколом OAuth з використанням GitHub;
- ведення інформації про метрики додатків, чий розмір буде оцінений;
- перегляд списку додатків користувачів;
- редагування списку додатків;
- перегляд звіту оцінки розміру додатку.

3.1.2. Вимоги до організації вхідних та вихідних даних:

- рендеринг сторінок сайту з використанням фреймворку Vue.js;
- перевірка типів даних;
- імпорт налаштувань оточення з сервера;
- універсальна обробка url-callback.

3.2. Вимоги до надійності

Передбачити обробку ситуацій некоректного вводу інформації з інформуванням користувача про шляхи усунення.

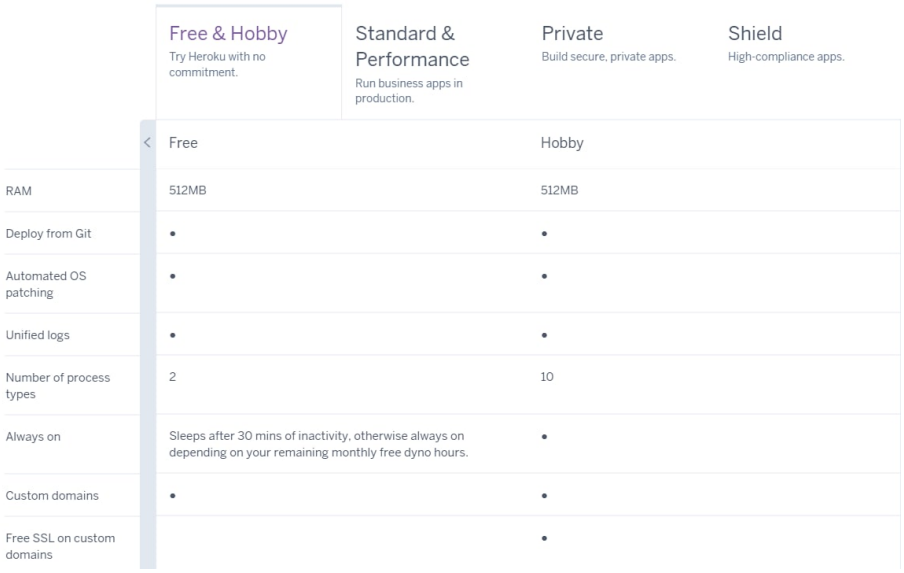
3.3. Умови експлуатації

Умови експлуатації серверної частини згідно з ASHRAE: - рекомендована температура в приміщенні 18 - 27 ° C, для цього необхідно кондиціонування повітря; - вологість повітря в серверній повинна бути в межах від 20% до 80% без конденсації вологи; швидкість зміни вологості 6% в годину; 70 - запиленість не повинна перевищувати 0,75 мг / м³; - тиск в серверній повинен перевищувати тиск в сусідніх приміщеннях. Рекомендується перевищення тиску не менше 14.7 Па.; - рівень освітлення має становити не менше 500 лк, виміряному на висоті 1 метр в горизонтальній площині; - рівень електромагнітного випромінювання не повинні перевищувати 3 В / м в усіх

діапазонах частот; - для певних видів обладнання і кросів необхідно обмежити вібрацію.

3.4. Вимоги до складу та параметрів технічних засобів.

В якості технічних засобів рекомендується використовувати серверне обладнання хмарного сервісу Heroku у мінімальній конфігурації, яку наведено на рисунку А.1.



	Free & Hobby Try Heroku with no commitment.	Standard & Performance Run business apps in production.	Private Build secure, private apps.	Shield High-compliance apps.
	Free		Hobby	
RAM	512MB		512MB	
Deploy from Git	•		•	
Automated OS patching	•		•	
Unified logs	•		•	
Number of process types	2		10	
Always on	Sleeps after 30 mins of inactivity, otherwise always on depending on your remaining monthly free dyno hours.		•	
Custom domains	•		•	
Free SSL on custom domains			•	

Рисунок А.1 – Рекомендована конфігурація серверного обладнання

3.5 Вимоги до інформаційної та програмної сумісності

Сервер: Apache, СУБД MongoDB.

Клієнт: браузер Chrome, FireFox, Opera.

4 Вимоги до програмної документації

Програмна документація повинна мати у своєму складі наступне:

- технічне завдання;
- інструкція користувача;
- опис програми;
- текст програми;
- програма та методика випробувань.

Розробка ведеться у навчальних цілях, техніко-економічні показники не розраховуються.

5 Стадії та етапи розробки

Стадії та етапи розробки наведені в таблиці А.1.

Таблиця А.1 — Стадії та етапи розробки

Стадії розробки	Етапи робіт	Зміст робіт	Строк виконання
1	2	3	4
1. Технічне завдання	Обґрунтування необхідності розробки програми	Постановка задачі Збір матеріалу Вибір й обґрунтування критеріїв ефективності та якості програми, що розроблюється.	02.09.2021
	Розробка й затвердження технічного завдання	Визначення вимог до програми. Визначення стадій, етапів й строків розробки програми та документації на неї. Вибір мови програмування.Узгодження й затвердження технічного завдання.	02.10.2021
2. Ескізний проект	Розробка ескізного проекту	Попередня розробка структури вхідних та вихідних даних. Уточнення методів розв'язку задачі. Розробка загального опису алгоритму розв'язку задачі	16.10.2021
	Затвердження ескізного проекту	Розробка пояснювальної записки. Узгодження й затвердження ескізного проекту.	23.10.2021
3. Технічний проект	Розробка технічного проекту	Уточнення структури вхідних та вихідних даних. Розробка алгоритму розв'язання задачі. Визначення форми представлення вхідних та вихідних даних. Визначення семантики та синтаксису мови. Розробка структури програми. Остаточне визначення конфігурації технічних засобів.	30.10.2021
	Затвердження технічного проекту	Розробка плану заходів з розробки та впровадження програм. Розробка пояснювальної записки. Узгодження та затвердження технічного проекту.	06.11.2021

Продовження таблиці А.1			
4. Робочий проект	Розробка програми	Програмування та налагодження програми.	17.11. 2021
	Розробка програмної документації	Розробка програмних документів	25.11. 2021
	Випробування програми	Розробка, узгодження та затвердження порядку та методики випробувань. Проведення випробувань. Коректування програми та програмної документації за результатами випробувань.	01.12. 2021
5. Впровадження	Підготовка та передача програми.	Підготовка та передача програми та програмної документації для супроводження та (або) виготовлення. Оформлення та затвердження акту про передачу програми на супроводження та (або) виготовлення.	15.12. 2021

6. Порядок прийому та контролю

Для контролю та прийому повинен бути наданий опис програмного продукту, а також сам програмний продукт і методика випробування. Порядок контролю та прийому даної розробки здійснюється представником розробника згідно з програмою та методикою випробувань. Якщо програма не пройшла випробування, виконавець зобов'язаний виправити помилки та недоліки у строк, не більш ніж один місяць від дня випробування. За результатами прийому складається акт, який підписується представником замовника та представником розробника та утверджується керівниками організації-замовника і організаціїрозробника. У випадку виявлення помилок під час прийому програмного продукту складаються акт про виявлені помилки, який підписується представниками замовника та розробника і утверджується керівниками організації – замовника й організації –

розробника. Розробник повинен у термін, який становить не більше як один місяць виправити вказані зауваження й повідомити замовника про повторне проведення перевірки, не пізніше як за два тижні до початку прийому програмного продукту

ДОДАТОК Б – ТЕКСТ ПРОГРАМИ

RegressionModel.vue

```
<template>
                <v-container>
                  <h3 class="comments-str">// Commits done from start.</h3>
                  <h3 style="position: fixed; width: 100px">x1 =</h3>
                  <v-text-field
                    outlined
                    style="width: 100px; margin-left: 50px"
                    v-model="x1"
                  ></v-text-field>


                  <h3 class="comments-str">// Spend work hours.</h3>
                  <h3 style="position: fixed; width: 100px">x2 =</h3>
                  <v-text-field
                    outlined
                    v-model="x2"
                    style="width: 100px; margin-left: 50px"
                  ></v-text-field>


                  <h3 class="comments-str">// Error coefficient.</h3>
                  <h3>ε = 0.05</h3>


                  <h3 class="comments-str">
                    // Calculate output total lines of code (PROJECT SIZE) using regression
                    model.
                  </h3>
                  <img src="../assets/formula1.png" />


                  <h3>Y = 10^(ε+1.04+{{ x1 }}^0.04+{{ x2 }}^0.25)</h3>
                  <h3>Y = {{ recalculate().predicted_y }}</h3>
                  <br />
                  <h3 class="comments-str">// Calculate intervals.</h3>
                  <h3>
                    {{ recalculate().trustInterval_Left }} &gt; Trusted_Interval &gt;
                    {{ recalculate().trustInterval_Right }}
                  </h3>
                  <h3>
                    {{ recalculate().predictedInterval_Left }} &gt; Predicted_Interval &gt;
```

```
      {{ recalculate().predictedInterval_Right }}
    </h3>
    <br />
    <br />
  </v-container>
</template>


<style scoped>
.comments-str {
  color: darkgreen;
}
</style>


<script>
export default {
  name: "RegressionModel",
  mounted() {},
  data() {
    return {
      x1: 1,
      x2: 2,
      Y: 0,
    };
  },
  methods: {
    toNumber() {},
    pow(a, b) {
      return Math.pow(a, b);
    },
    recalculate() {
      var x1 = Number(this.x1);
      var x2 = Number(this.x2);
      var epsilon = 0.05;
      //var sqrt = Math.sqrt;
      //var abs = Math.abs;
      var pow = Math.pow;
      //var Z = 16305.54;
      var student = 2.35183518; // Коеф. Стьюдента
      var predicted_y = pow(10, epsilon + 1.04 + pow(x1, 0.04) + pow(x2, 0.25));
//Предсказаное Y
      // var SZY = 0.120810582; // сумма квадратов разностей y
      // var SZX1 = 0.4378559; // сумма квадратов разницы x1 ????
      var trustedLastPart = 6240.073;
      var predictedLastPart = 86822.572;
      var trustInterval_Left = predicted_y - student * trustedLastPart;
//доверительный лев. гр.
```

```
        var trustInterval_Right = predicted_y + student * trustedLastPart;
      //доверительный прав. гр.
        var predictedInterval_Left = predicted_y - student * predictedLastPart;
      //предсказательный лев. гр.
        var predictedInterval_Right = predicted_y + student * predictedLastPart;
      //предсказательный прав. гр.
        var res = {
          predicted_y,
          trustInterval_Left,
          trustInterval_Right,
          predictedInterval_Left,
          predictedInterval_Right,
        };
        return res;
      },
    },
  };
  </script>
```

```
<template>
                                    <v-container>
                                      <v-card>
                                        <v-card-subtitle>Detailed stats</v-card-subtitle>
                                        <v-card-text>
                                          <div v-if="!repoLoaded">
                                            <v-text-field
                                              label="Enter repository url"
                                              v-model="searchingRepoUrl"
                                            ></v-text-field>
                                            <v-btn outlined @click="findRepo">Find</v-btn>
                                          </div>


                                          <div v-if="repoLoaded">
                                            <p id="statsStrEl"></p>


                                            <v-btn
                                              small
                                              outlined
                                              :color="getIsUsedInTrainingColor(repoInfo)"
                                              rounded
                                              @click="swithcUseInTraining(repoInfo)"
                                            >
```

```
            {{ getIsUsedInTrainingText(repoInfo) }}
          </v-btn>
          <h1 class="text-center">Info</h1>
          <h4>Id: {{ repoInfo.id }}</h4>
          <h4>Name: {{ repoInfo.name }}</h4>
          <h4>
            URL:
            <a :href="repoInfo.htmlUrl" style="color:
blue">{{
              repoInfo.htmlUrl
            }}</a>
          </h4>
          <h4>Language: {{ repoInfo.language }}</h4>
          <h4>Pull request: {{ repoInfo.pullRequestsCount
}}</h4>
          <h4>Opened issue: {{ repoInfo.openIssuesCount
}}</h4>
          <h4>Total issue: {{ repoInfo.issuesCount }}</h4>
          <h4>Forks: {{ repoInfo.forksCount }}</h4>
          <h4>Watchers: {{ repoInfo.watchersCount }}</h4>
          <h4>Size: {{ repoInfo.size }} bytes</h4>
          <br />
          <v-divider></v-divider>
          <br />
          <h1 class="text-center">Stats</h1>
          <h4>Total commits: {{ repoInfo.totalCommits
}}</h4>
          <h4>
            Total added lines: {{
repoInfo.totalAdditionsLinesCount }} bytes
          </h4>
          <h4>
            Total deleted lines: {{ -
repoInfo.totalDeletionsLinesCount }} bytes
          </h4>
          <h4>Total deleted lines: {{
repoInfo.totalNewLinesCount }} bytes</h4>
          <h4>
            Monthly average commits:
            {{ repoInfo.averageCommitsCount * 4 }} bytes
          </h4>
          <h4>
            Monthly average added lines:
            {{ repoInfo.averageAdditionsLinesCount * 4 }}
bytes
          </h4>
          <h4>
            Monthly average deleted lines:
```

```
            {{ -repoInfo.averageDeletionsLinesCount * 4 }}
bytes
          </h4>
          <h4>
            Monthly average deleted lines:
            {{ repoInfo.averageNewLinesCount * 4 }} bytes
          </h4>
          <h4>
            Monthly average authors:
            {{ repoInfo.averageAuthorsCount * 2 }} bytes
          </h4>
          <br />
          <v-divider></v-divider>
          <br />
          <div id="repoStatsChart" style="height: 370px;
width: 100%"></div>
        </div>
      </v-card-text>
    </v-card>
  </v-container>
</template>
```

GitHubRepositoryStats.vue

```
<script>
import Helpers from "../../libs/helpers.js";
import ApiClient from "../js/apiClient";
import Consts from "../js/consts";
export default {
  name: "GitHubRepositoryStats",
  async created() {
    var repoUrl = Helpers.getUrlParameter("repo_url");
    if (!repoUrl) {
      this.repoLoaded = false;
      return;
    }
    var repoShortInfo = await
ApiClient.github_getRepositoryByUrl(repoUrl);
    var repoInfo = await
ApiClient.github_getRepositoryInfo(repoShortInfo.id);
    this.repoInfo = repoInfo;
    this.initChart(this.repoInfo);
    //this.addStatsString(this.repoInfo);
  },
  data() {
    return {
      repoLoaded: true,
```

```
    searchingRepoUrl: null,
    repoInfo: {
      id: "Loading...",
      name: "Loading...",
      fullName: "Loading...",
      createdAt: "",
      updatedAt: "",
      description: "Loading...",
      forksCount: "Loading...",
      gitUrl: "",
      htmlUrl: "Loading...",
      language: "Loading...",
      openIssuesCount: "Loading...",
      issuesCount: "Loading...",
      size: "Loading...",
      watchersCount: "Loading...",
      pullRequestsCount: "Loading...",
      totalCommits: "Loading...",
      isUsingInTeaching: true,
      totalAdditionsLinesCount: 0,
      totalDeletionsLinesCount: 0,
      totalNewLinesCount: 0,
      averageAdditionsLinesCount: 0,
      averageDeletionsLinesCount: 0,
      averageNewLinesCount: 0,
      averageAuthorsCount: 0,
      averageCommitsCount: 0,
      weekCommitStats: [
        {
          weekNumber: 735202,
          weekDate: "2013-12-01T00:00:00Z",
          commitsCount: 1,
          additionsLinesCount: 346438,
          deletionsLinesCount: 0,
          newLinesCount: 346438,
          totalLinesCount: 346438,
          totalCommitsCount: 1,
          authorsCount: 1,
        },
      ],
    },
  };
},
methods: {
  getIsUsedInTrainingText(item) {
    if (item.isUsingInTeaching) {
      return "✓ Used in model training";
    } else {
```

```
          return "✕ Not used in model training";
        }
      },
      getIsUsedInTrainingColor(item) {
        if (item.isUsingInTeaching) {
          return "#006629";
        } else {
          return "#d18800";
        }
      },
      async swithcUseInTraining(item) {
        item.isUsingInTeaching = !item.isUsingInTeaching;
        await ApiClient.github_setUseInTeaching(item.id,
item.isUsingInTeaching);
      },
      findRepo() {
        if (this.searchingRepoUrl) {
          var url =
            Consts.RepositoryStats +
            "?repo_url=" +
            encodeURI(this.searchingRepoUrl);
          Helpers.redirect(url);
        }
      },
      addStatsString(repoInfo) {
        var splitter = "\t";
        var str =
          repoInfo.name +
          splitter +
          repoInfo.htmlUrl +
          splitter +
          repoInfo.totalCommits +
          splitter +
          repoInfo.totalAdditionsLinesCount +
          splitter +
          -repoInfo.totalDeletionsLinesCount +
          splitter +
          repoInfo.totalNewLinesCount;
        str +=
          splitter +
          repoInfo.averageCommitsCount * 4 +
          splitter +
          repoInfo.averageAdditionsLinesCount * 4 +
          splitter +
          -repoInfo.averageDeletionsLinesCount * 4 +
          splitter +
          repoInfo.averageNewLinesCount * 4 +
          splitter +
```

```
        repoInfo.averageAuthorsCount * 2;
      var statsStrEl =
document.getElementById("statsStrEl");
      statsStrEl.innerHTML = str;
      navigator.clipboard.writeText(str);
    },
    initChart(repoInfo) {
      //       averageAdditionsLinesCount: 0,
      // averageDeletionsLinesCount: 0,
      // averageNewLinesCount: 0,
      // averageAuthorsCount: 0,
      // averageCommitsCount: 0,
      // var commitsModifier = (repoInfo.averageCommitsCount
+ 1) / 200;
      // var additionsModifier =
(repoInfo.averageAdditionsLinesCount + 1) / 200;
      // var deletionsModifier =
(repoInfo.averageDeletionsLinesCount + 1) / 200;
      // var newLinesModifier =
(repoInfo.averageNewLinesCount + 1) / 200;
      // var authorsModifier = (repoInfo.averageAuthorsCount
+ 1) / 200;
      // var maxY = (repoInfo.averageAdditionsLinesCount /
additionsModifier) * 7;
      // var minY = -maxY / 100;
      //console.log(maxY + " " + minY);
      var labels = [];
      var commits = [];
      var additionsLinesCount = [];
      var deletionsLinesCount = [];
      var newLinesCount = [];
      var authorsCount = [];
      for (var index in repoInfo.weekCommitStats) {
        var week = repoInfo.weekCommitStats[index];
        var weekDate = week.weekDate.split("T")[0];
        labels.push(weekDate);
        commits.push(week.commitsCount);
        additionsLinesCount.push(week.additionsLinesCount);
        deletionsLinesCount.push(-week.deletionsLinesCount);
        if (week.newLinesCount < 0) {
          week.newLinesCount = -week.newLinesCount;
        }
        newLinesCount.push(week.newLinesCount);
        authorsCount.push(week.authorsCount);
      }
      var chart = new
window.CanvasJS.Chart("repoStatsChart", {
        animationEnabled: true,
```

```
title: {
  text: "Repo stats chart",
},
axisX: {},
axisY: {
  valueFormatString: " ",
},
legend: {
  cursor: "pointer",
  fontSize: 16,
},
toolTip: {
  shared: true,
},
zoomEnabled: true,
data: [
  {
    name: "Date",
    type: "spline",
    shoInLegend: false,
    dataPoints: [],
  },
  {
    name: "Commits",
    type: "spline",
    dataPoints: [],
  },
  {
    type: "spline",
    name: "Added lines",
    dataPoints: [],
  },
  {
    type: "spline",
    name: "Deleted lines",
    dataPoints: [],
  },
  {
    type: "spline",
    name: "New lines",
    dataPoints: [],
  },
  {
    type: "spline",
    name: "Authors",
    dataPoints: [],
  },
],
```

```
      });
      addDataPoints();
      chart.render();
      function addDataPoints() {
        for (var i = 0; i < labels.length; i++) {
          chart.options.data[0].dataPoints.push({
            x: i,
            y: labels[i],
          });
          chart.options.data[1].dataPoints.push({
            x: i,
            y: commits[i],
          });
          chart.options.data[2].dataPoints.push({
            x: i,
            y: additionsLinesCount[i],
          });
          chart.options.data[3].dataPoints.push({
            x: i,
            y: deletionsLinesCount[i],
          });
          chart.options.data[4].dataPoints.push({
            x: i,
            y: newLinesCount[i],
          });
          chart.options.data[5].dataPoints.push({
            x: i,
            y: authorsCount[i],
          });
        }
      }
    },
  },
};
</script>
```

GitHubUserStats.vue

```
 <template>
                <v-container>
                  <v-card>
                    <v-card-text>
                      <h2>Activity</h2>
                      <img :src="awesomeGithubStats" />
```

```
        <h2>Trophies</h2>
        <img :src="githubProfileTrophy" />
      </v-card-text>
    </v-card>
  </v-container>
</template>



<script>
import ApiClient from "../js/apiClient";
export default {
  name: "GitHubUserStats",
  async created() {
    var user = await ApiClient.getMe();
    this.GithubNickname = user.github_Login;
  },
  data() {
    return {
      GithubNickname: "",
    };
  },
  computed: {
    awesomeGithubStats: function () {
      var url =
        "https://awesome-github-
stats.azurewebsites.net/user-stats/" +
        this.GithubNickname +
        "?cardType=level";
      return url;
```

```
    },
    githubProfileTrophy: function () {
      var url =
        "https://github-profile-
trophy.vercel.app/?username=" +
        this.GithubNickname;
      return url;
    },
  },
  methods: {},
};
</script>
```

•

ДОДАТОК В – ПРОГРАМА І МЕТОДИКА ВИПРОБОВУВАНЬ

1. Об’єкт випробувань

   Назва розроблюваного проекту: «DiplomaStatsCounter».

   Галузь застосування – підприємства, які займаються розробкою програмного забезпечення, що. застосовують у розробці фреймворк .NET

2. Мета випробувань

   Мета проведення випробувань оцінки експлуатаційних характеристик програми, перевірка і підтвердження працездатності програми в умовах, максимально наближених до умов реальної експлуатації.

3. Вимоги до програми

   Програма має реалізувати функції, описані в технічному завданні, яке наведено в додатку А.

4. Вимоги до програмної документації

   Для проведення тестування програми, повинна бути надана наступна документація:

   - технічне завдання на розробку програми (вимоги до складання документу визначаються ДСТ 19.101-77);

   - текст програми (зміст і оформлення документу визначається ДСТ 19.101-77)

   - опис програми

   - інструкція користувача

   - програма і методика випробувань ПЗ (вимоги до складання документу визначаються ДСТ 19.101-77).

   5. Склад порядок та методи випробувань

   5.1 Перевірка програми на відповідність технічному завданню:

- авторизація з використанням аккаунту github;

- введення інформації про метрики додатку, що оцінюється з метою прогнозування його розмірів;

- перегляд списку додатків, що мають схожий функціонал;

- можливість додати та прибрати зі списку додатків, що мають схожий функціонал;

- обчислення інтервальних та точкових оцінок розміру додатку;

- перегляд звіту з оцінювання розміру майбутнього додатку.

ДОДАТОК Г – ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розроблене програмне забезпечення створене для оцінювання розміру . застосунків, що створюються з використанням фреймворку .Net з використанням багатофакторної регресійної моделі.

Програмне забезпечення доступне для використання за посиланням https://diploma-stats-counter.herokuapp.com/

Вихідні коди доступні за посиланням https://github.com/ExtremeDotneting/DiplomaStatsCounter.WebApp.

Рисунок Г.1 - Репозиторій з програмним забезпеченням

Програмне забезпечення дозволяє:

- авторизуватися з використаннням github –аккаунтів (рис Г.2);

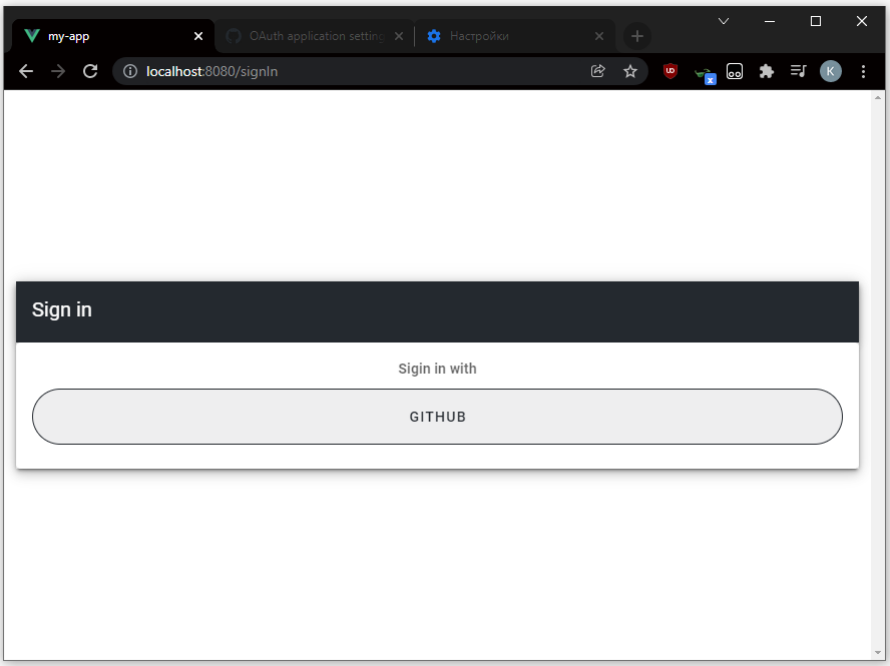


Рисунок Г2 – Авторизація за допомогою креденшіалів з github.com

- дивитися статистику по користувачах, репозиторіях (Рисунок Г.3),

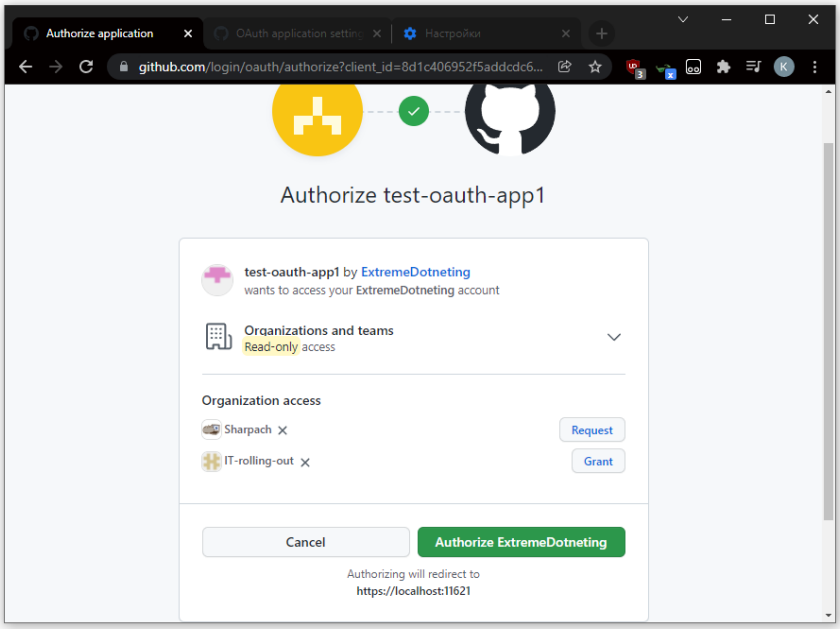


Рисунок Г.3 - Авторизація

- перегляд списку .Net додатків на базі яких ведеться розрахунок;

- редагування списку .Net додатків на базі яких ведеться розрахунок наведено на рисунку Г.4;

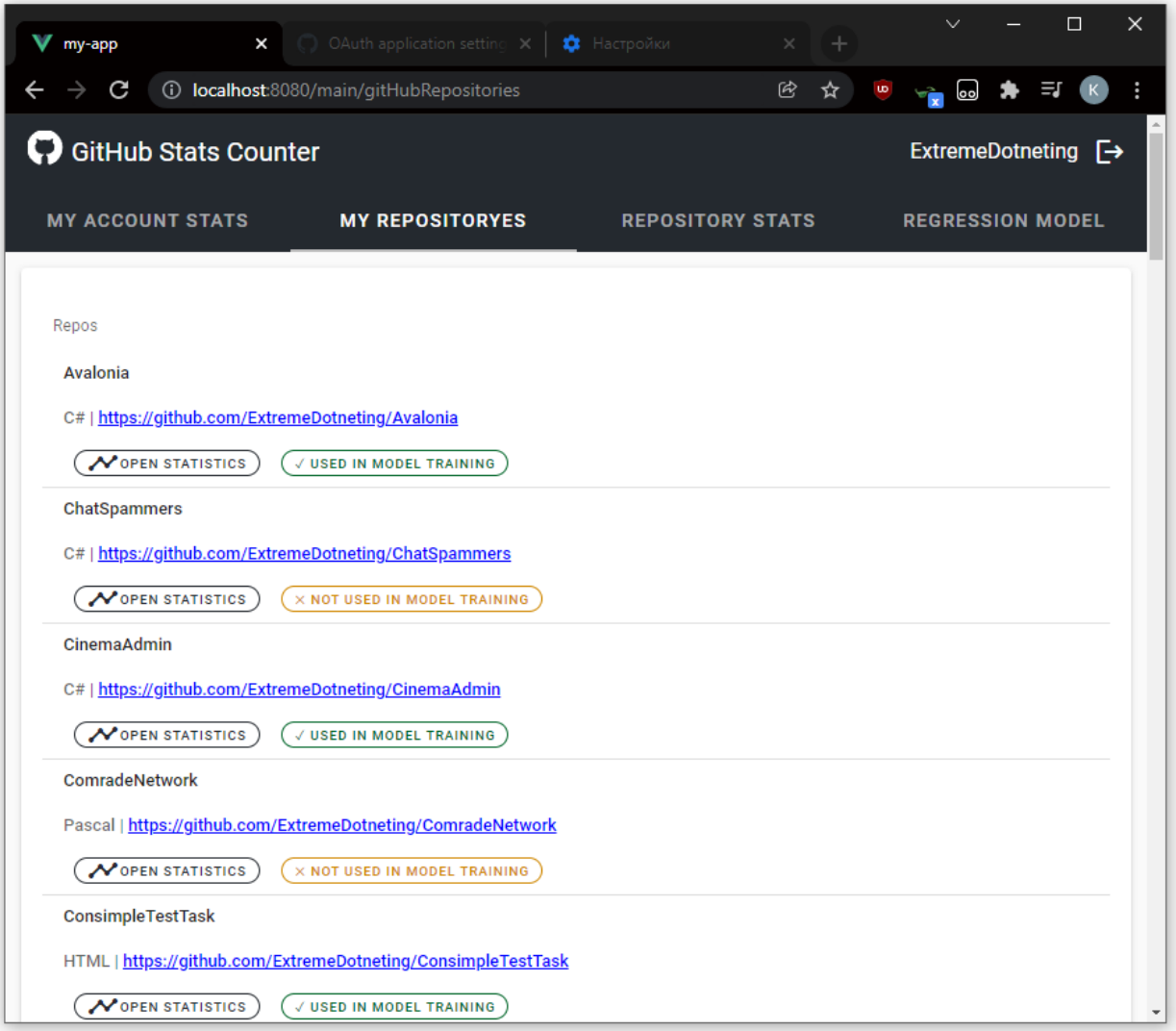


Рисунок Г.4 - Редагування списку .Net додатків на базі яких ведеться розрахунок

- введення інформації про метрики .Net додатків для оцінки його розміру та перегляд результатів розрахунку оцінки розміру майбутнього .Net додатку наведено на рисунку Г.5

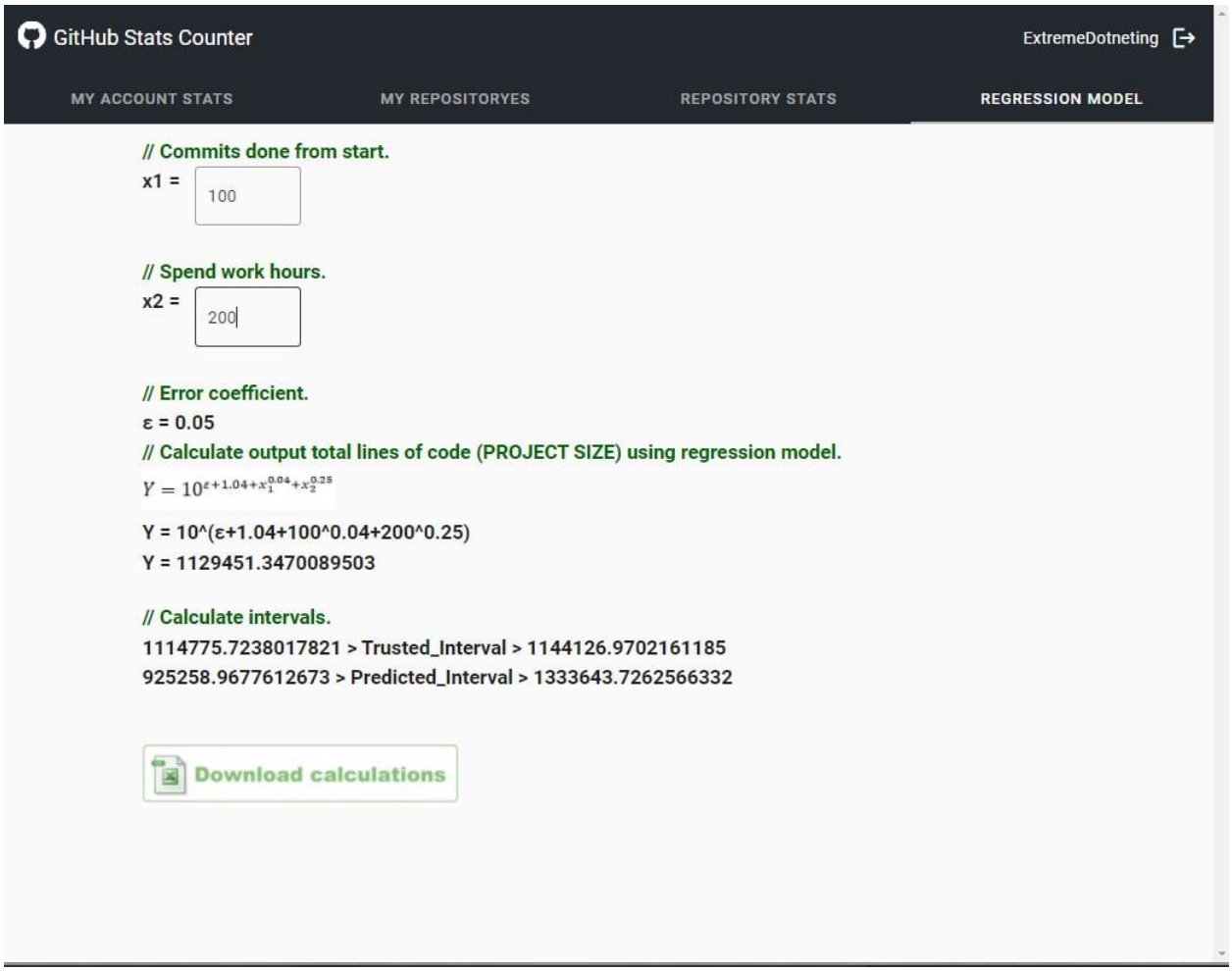
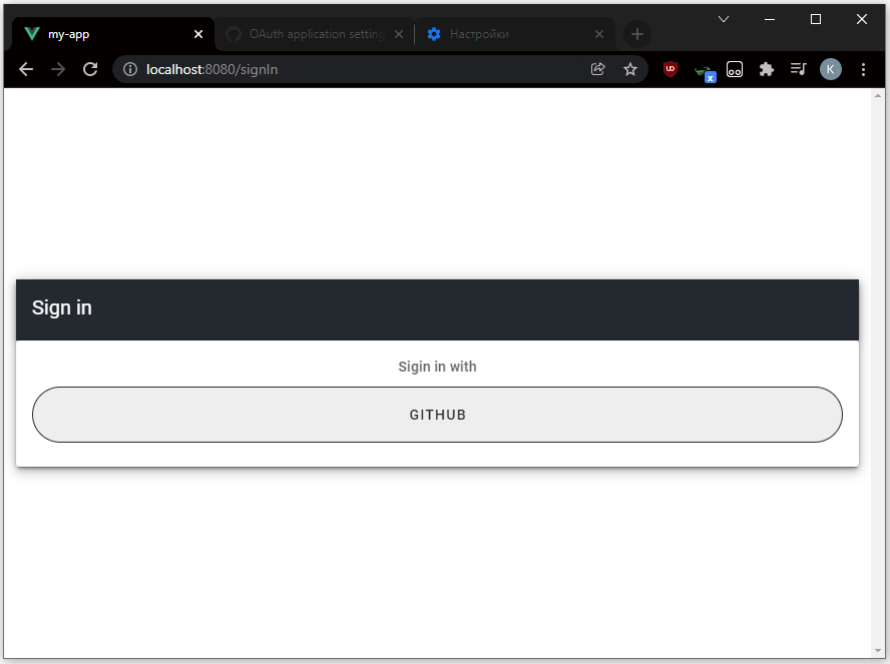


Рисунок Г.6 – Розрахунок розміру програмного забезпечення, що розроблене за допомогою фреймворку .Net

ДОДАТОК Д – ІНСТРУКЦІЯ КОРИСТУВАЧА

1.Першим кроком треба ввести логін та пароль аккауну github.com



2. Після авторизації потрапляємо на сторінку статистики облікового запису яку наведено на рисунку Д.1

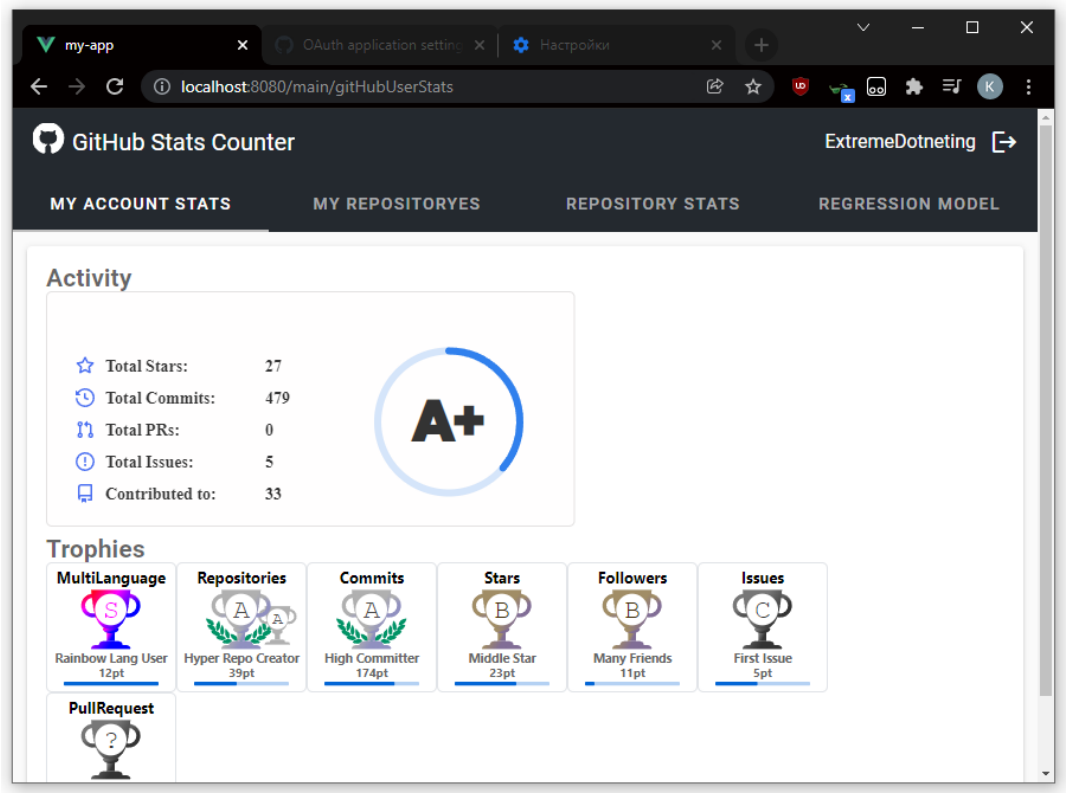
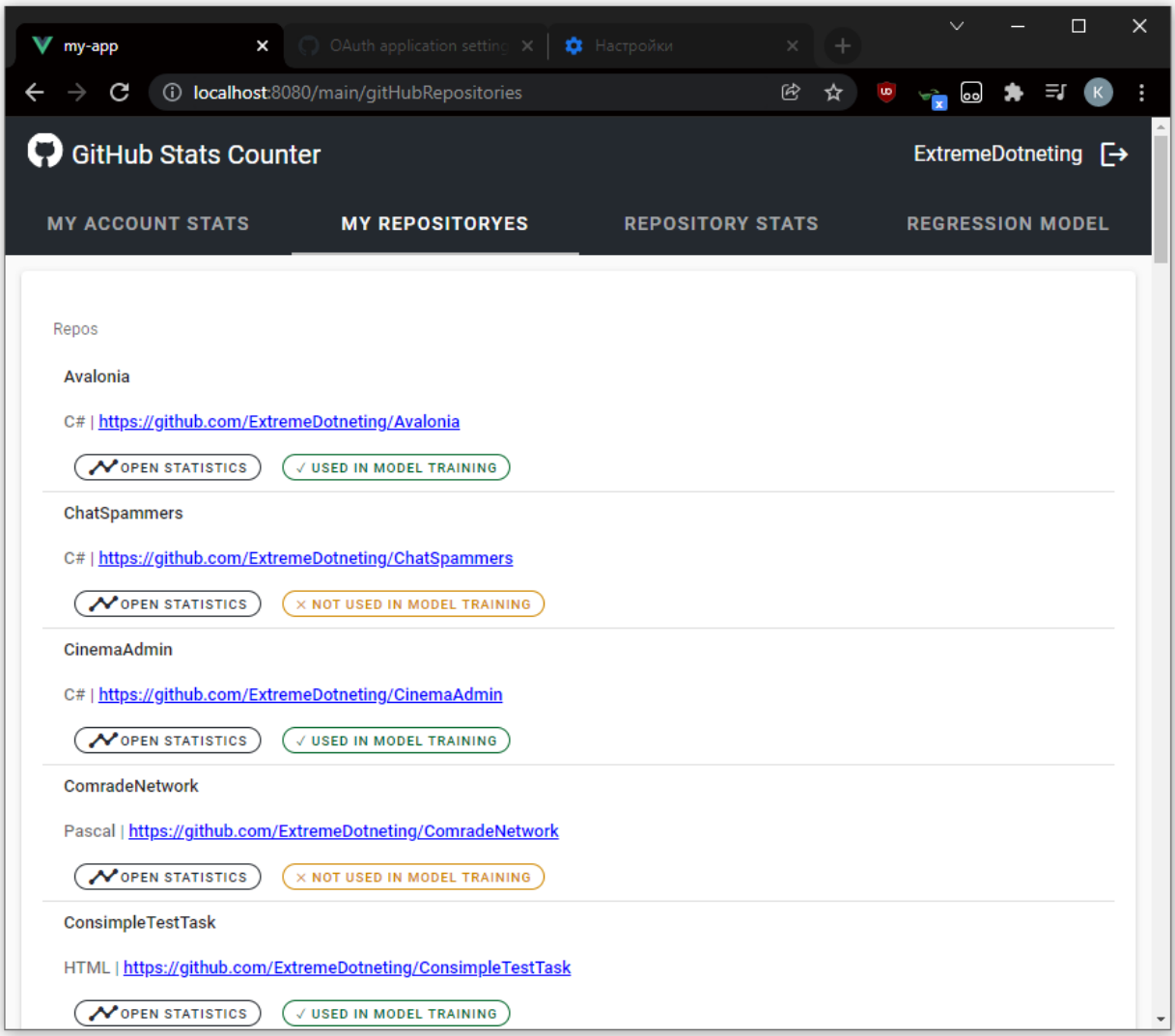


Рисунок Д.1. Сторінка статистики облікового запису Github



3. На сторінці "Мої репозиторії", яку наведено на рисунку Д.2 можна вибрати репозиторій зі списку, клікнути та перейти на статистику по ньому, або подивитися усі репозиторії, що доступні поточному користувачу. Є можливість додати або прибрати репозиторії, що не підходять до вибірки по тим чи іншим причинам.

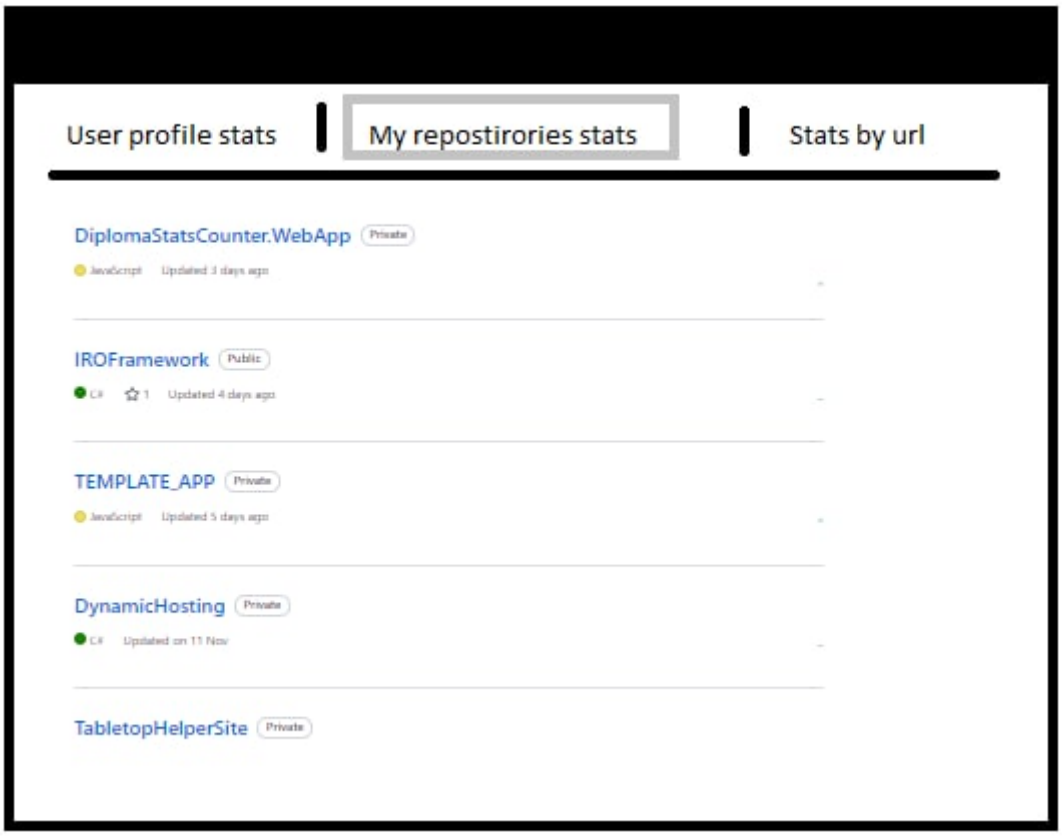


Рисунок Д.2. Список доступних репозиторіїв

4. На рисунку Д3 наведено сторінку статистики, де зверху можна ввести метрики проекту та подивитися результати оцінювання розміру майбутнього програмного забезпечення у графічній та табличній формах.

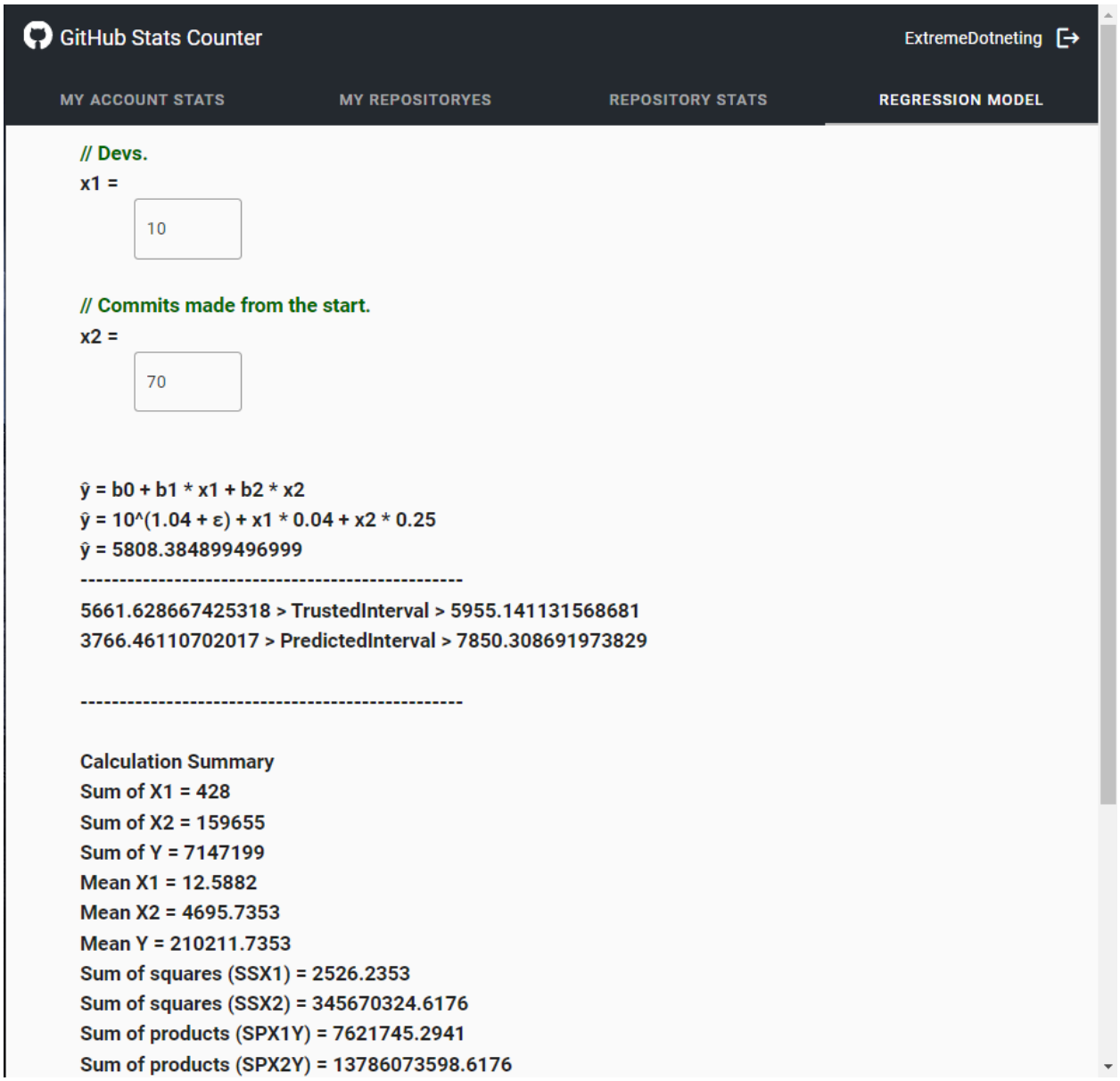


Рисунок Д.3 Розрахунок розміру з використанням математичної моделі