МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

Інститут прикладного системного аналізу .
(назва факультету, інституту)

Кафедра системного проектування .
(назва кафедри)

До захисту допущено
Завідувач кафедри

_____ А.І. Петренко .
(підпис) (ініціали, прізвище)

“___”_____2017 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проекту (роботи) освітньо-кваліфікаційного рівня “ **спеціаліст**”
(назва ОКР)

з напряму підготовки (спеціальності) **7.05010103 «Системне проектування».**
(код та назва напряму підготовки або спеціальності)

на тему: Веб додаток на фреймворку Spring Java для обробки та аналізу даних науковців по параметрам Google Scholar.

Студент групи ДА-52с Парамонов В. Д _____
(шифр групи) (прізвище, ім’я, по батькові) (підпис)

Керівник проекту доц. Цурін О. Ф _____
(вчені ступінь та звання, прізвище, ініціали) (підпис)

Консультанти:

нормоконтроль _____ к.т.н., доц. Стіканов В.Ю. _____
(назва розділу ДП (ДР)) (вчені ступінь та звання, прізвище, ініціали) (підпис)

рецензент _____ дир. КПІ телеком Ілляшенко А.М. _____
(назва розділу ДП (ДР)) (вчені ступінь та звання, прізвище, ініціали) (підпис)

Київ – 2017

Національний технічний університет України
“Київський політехнічний інститут”

Факультет (інститут) **Інститут прикладного системного аналізу** .
(повна назва)

Кафедра **Системного проектування** .
(повна назва)

Напрям підготовки **050101 Комп’ютерні науки** .
(код, назва)

Спеціальність **7.05010103 «Системне проектування».**
(код, назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ А.І. Петренко .
(підпис) (ініціали, прізвище)

“___”_____ 2017 р.

ЗАВДАННЯ
на передипломну практику та дипломний проект (роботу) освітньо-кваліфікаційного рівня
“спеціаліст” .
(назва рівня)

Студенту Парамонову Володимиру Денисовичу
(прізвище, ім’я, по батькові)

1. **Тема проекту** (роботи) Веб додаток на фреймворку Spring Java для відображення та аналізу даних науковців по параметрам Google Scholar

затверджена наказом по університету від _____

2. **Термін здачі** студентом закінченого проекту (роботи) _____

3. **Вихідні дані до проекту** (роботи)
Нефункціональні вимоги: ГС та його можливості, порівняння з іншими системами, мова програмування Java, аналіз актуальних фреймворків.
Функціональні вимоги: веб- додаток для аналізу даних науковців по параметрам Google Scholar, програмний код.

4. **Перелік питань, які мають бути розроблені**

1.Вступ та обгрунтування.

2.Місце GOOGLE SCHOLAR у вебометричних базах:
Його особливості, функції та порівняння з іншими подібними за функціоналом системами. Особисті кабінети та їх побудова у GOOGLE SCHOLAR.
3. База GOOGLE SCHOLAR університету (http://telef.kpi.ua) та її аналіз, програмна реалізація додатку.
4. Управління строками та ризиками виконання дипломного проекту.

Реалізація

- Розробити веб додаток за допомогою платформи Java, який буде оброблювати запити, параметри для яких задає користувач:
- Формування даних по цитованостям викладачів по кафедрам та факультетам – в табличному форматі та в виді гістограм;
-рейтингування з вилученням N найбільших та найменших значень
- Парсинг ID викладачів для визначення профілів що мають не підтвердженний заклад.
- Видання по запиту цих ID у форматі JSON
- розробити діаграму Ганта відповідно до плану дипломної роботи та розрахувати критичний шлях
- визначити основні ризики у проекті та розрахувати їх вплив на проект

1. **Перелік графічного (ілюстративного) матеріалу**

Структурна схема (креслення 1)
UML діаграма класів додатку (креслення 2)
Місце Google Scholar серед подібних систем (1-й плакат)
Скріншоти роботи програми (2-й плакат)
Основні висновки по роботі (3-й плакат)

6. **Консультанти**

з технічних вимог доц.к.т.н . Цурін О.Ф

7. **Дата видачі завдання** 01.09.2016

Керівник дипломного проекту (роботи) _____ Цурін О.П._____
(підпис) (ініціали, прізвище)

Завдання прийняв до виконання _____ _____
(підпис) (ініціали, прізвище)

ЗАТВЕРДЖУЮ

Керівник
дипломного проекту (роботи)
_____ Цурін О. П___
(підпис) (ініціали, прізвище)
“___”_____ 2017р.

КАЛЕНДАРНИЙ ПЛАН-ГРАФІК

виконання дипломного проекту (роботи)

студентом **Парамоновим Володимиром Денисовичем**
(прізвище, ініціали)

№ з/п	Назва етапів роботи та питань, які повинні бути розроблені відповідно до завдання	Термін виконання	Позначки керівника про виконання завдань
1	Ознайомлення з літературою і підготовка теоретичної частини роботи	15.09.2016 – 30.09.2016	
2	Аналіз вимог завдання, вибір методів і засобів розв’язання поставленої задачі	05.10.2016	
3	Розробка додатку	10.11.2016	
4	Тестування розробленого додатку, перевірка відповідності завданю.	30.12.2016	
5	Підготовка графічного матеріалу, оформлення пояснювальної записки, підготовка до захисту	12.01.2017	
6	Проходження нормоконтролю, отримання відгуку, рецензії, передача роботи в ДЕК	15.012.017	
7	Захист дипломної роботи	14.06 2017	

Студент _____
(підпис)

Анотація

Студент: Парамонов Володимир Денисович

*Тема: «*еВеб додаток на фреймворку Spring Java для відображення та аналізу даних науковців по параметрам Google Scholar»

Спеціальність:7.05010103 «Системне проектування»

НТУУ «КПІ»

Київ, 2017 рік.

Робота присвячена дослідженню актуальних фреймворків та розробці веб – додатку для відображення рейтингів цитованостей науковців НТУУ «КПІ ім. Ігоря Сікорського» в системі пошуку по науковим публікаціям Google Scholar.

Робота складається з розділів:

1. Google Schoolar.
2. Вибір засобів реалізації.
3. Програмна реалізація додатку.
4. Управління строками дипломної роботи.

У процесі роботи були зроблені додаток для збору та зберігання рейтингів, та відображення для нього у вигляді таблиць та гістограм.

Робота складається з 76 стор., 6 таблиць, 31 рисунок і 13 літературних джерел.

Аннотация

Студент: Парамонов Владимир Денисович

*Тема: «*Веб приложение на фреймворке спринг Java для отображения и анализа данных ученных по параметрам Google Scholar»

Специальность: 7.05010103 «Системное проектирование»

НТУУ «КПИ»

Киев, 2017 год.

Работа посвящена исследованию актуальных фреймворков и разработке веб – приложения для отображения рейтингов цитируемостей ученых НТУУ «КПИ им. Игоря Сикорского» в системе поиска по научным публикациям Google Scholar»

Работа состоит из разделов:

1. Google Schoolar.
2. Выбор средств реализации.
3. Программная реализация приложения.
4. Управление сроками дипломной работы.

В процессе работы было сделано пприложение для сбора и хранения рейтингов и отображение к нему в виде таблиц и гистограмм.

В работе есть 76 страниц 6 таблиц, 31 рисунков і 13 литературных источников.

Abstract

Student: Paramonov Volodymyr

Topic: " Web- based application on Java framework for processing and data analysis by scientists parameters of Google Scholar”

Specialty: 7.05010103 "System engeneering"

NTU "KPI"

Kyiv , 2017.

Diploma work is dedicated to research of relevant Java frameworks and development of web-based application for displaying ratings of NTUU «KPI named after Igor Sikorskiy» scientists citiation in system of search for scientific publications Google Scholar.

It consists of sections:

1.Google Schoolar

2.Choice methdos for implementation.

3.Programm implementation of application.

4.Timeline management for diploma work.

In the process, was developed server application and view for it as tables and histograms.

In this work there are 76 pages 6 tables, 31 pictures and 13 references.

Зміст

					ДА-52С 15 0004 001		
						Літера	Масса	Масштаб
Змін	Лист	№ документу	Підпис	Дата	Зміст			
Розробив		Парамонов В, Д				Лист 9	Листів 75	
Перевірив		Цурін О. П						
Рецензія						НТУУ "КПІ ім. Ігоря Сікорського" ІПСА ДА-52с		
Н. контр.		Стіканов В.Ю						
Затвердив		Петренко А.І.						

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

Web - система доступу до пов'язаних між собою документів на різних комп'ютерах, підключених до Інтернету.

HTML - HyperText Markup Language — Мова розмітки гіпертекстових документів.

CSS - Cascading Style Sheets (Каскадні таблиці стилів).

JS – прототипно – орієнтований язик програмування, що виконується на сторонні клієнта.

Патерн проектування - повторювана архітектурна конструкція, що представляє собою вирішеня проблеми проектування в рамках деякого часто виникаючого контексту.

ORM – технологія програмування, що зв’язує бази даних з концепціями об’єктно-орієнтваних мов програмування. Описує існуючу базу даних через створення об’єктів.

ІЦ – індекс дитування.

H – index – індекс Хірша. Наукометричний показник, використовується х 2005 року. Кількісна характеристика продутивності вченого, групи вчених, научної організації або країни. Оснований на кількості публікацій та кількості цитованостей цих публікацій.

ВСТУП

Упродовж останніх років послідовно набуває сили євроінтеграційний вектор державної політики України в сфері науки. У зв'язку з цим робляться кроки до впровадження кількісних та напівкількісних методів аналізу наукової активності.

"Стоючи на плечах у гігантів" – рекламний слоган сервісу пошуку по науковим статям та публікаціям Google Schoolar.

В певному сенсі науковці і викладачі нашої країни є гігантами науки та прогресуТож їхня праця і досягнення повинні бути визнані в Україні та по всьому світу.

З поширенням інтернету стало набагато легше відстежити, в яких джерелах є посилання на працю певного науковця.

Одна з пошукових систем, що з'явилася відносно недавно і дозволяє відстежувати цитованості науковців, як індивідуальних вчених, так і у зв'язку з організацією де вони працюють має назву Google scholar. Завдяки своїй відкритості, гнучкості, та вдалій назві система швидко наздогнала найкращіх конкурентів – Scopus та Web of Science, які на відміну від Scholar надає доступ до публікацій тільки після оформлення платної підписки.

Проте при наявності такої системи існує проблема для рейтингу нашого та інших університетів України, яка впливає на світові рейтинги наших закладів – не всі викладачі правильно оформили свої профілі, оскільки Google Scholar Citations враховує приналежність вченого до закладу тільки в тому випадку, якщо в публікаціях було зазначено офіційне найменування вузу і використаний адреса електронної пошти, прийнятий у ВУЗі. Це значно зменшує рейтинг нашого університету у світовому рейтингу.

Рішення достатньо просте, але часто ігнорується чи не помічається при заповненні персональних профайлів у системі.

Метою цієї роботи є розробка додатку для відображення поточного рейтингу цитованостей викладачів нашого ВУЗУ по структурним підрозділам – кафедрам та факультетам в різних форматах – табличному та гістограмному, з відображенням тих викладачів що неправильно заповнили свій профіль. Тобто наглядна демонстрація того, скільки факультет чи кафедра могли б мати рейтингу та скільки мають реально. Також описаний нижче та викладений в інтернеті шлях заповнення профіля таким чином, щоб було враховано належність викладача до ВНЗ, - для швидкого знайдення помилок та включення в загальний рейтинг.

Розділ I Google Schoolar

1.1 Опис

Академия Google (англ. *Google Scholar*) — безкоштовна пошукова система по повним текстам наукових публікацій усіх форматів та дисциплін. Проект стартував у 2004 році в статусі бета – версії. Індекс Академії містить дані з більшості рецензованих онлайн журналів найбільших наукових видавництв Європи та Америки.

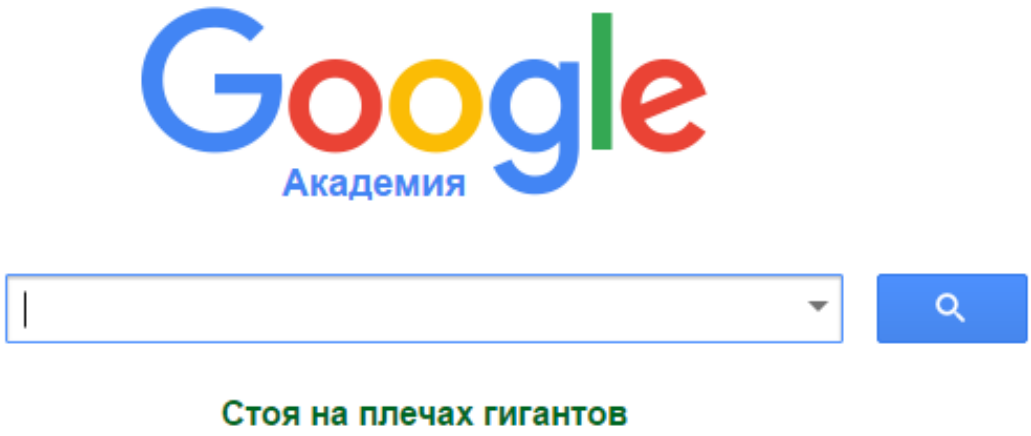


Рисунок 1.1 Гугл академія

Індекс цитування

Основна одиниця для оцінки вчених – індекс цитування(Google Scholar Citations). ІЦ - показник пошукової системи, обчислюваний на основі кількості посилань на даний ресурс з інших ресурсів мережі Інтернет. У простій різновидності індексу цитування враховується тільки кількість посилань на ресурс. Тематичний індекс цитування (ТІЦ) враховує також тематику сайтів, які посилаються на ресурс, а зважений індекс цитування - популярність сайтів, що посилаються (також у більшості випадків обчислюється на основі індексу

цитування). Спочатку, до того як з'явилися оптимізатори сайтів, індекс цитування реально відображав популярність відповідного ресурсу в інтернеті.

Першою великою пошуковою системою, яка почала активно використовувати індекс цитування, стала Google (алгоритм PageRank). У російськомовному сегменті Інтернету найбільшою популярністю користується ТІЦ «Яндекса».

Доступні метрики

Окрім цитованості, Scholar має ще три метрики:

1) H-index

Індекс рахується на основі розподілення цитованостей робіт науковця.

Згідно Хіршу:

Вчений має індекс h, якщо h з його N статей цитується як мінімум h разів кожна, в той час як статті, що залишаються - (N-h) цитуються не більше ніж h разів кожна.

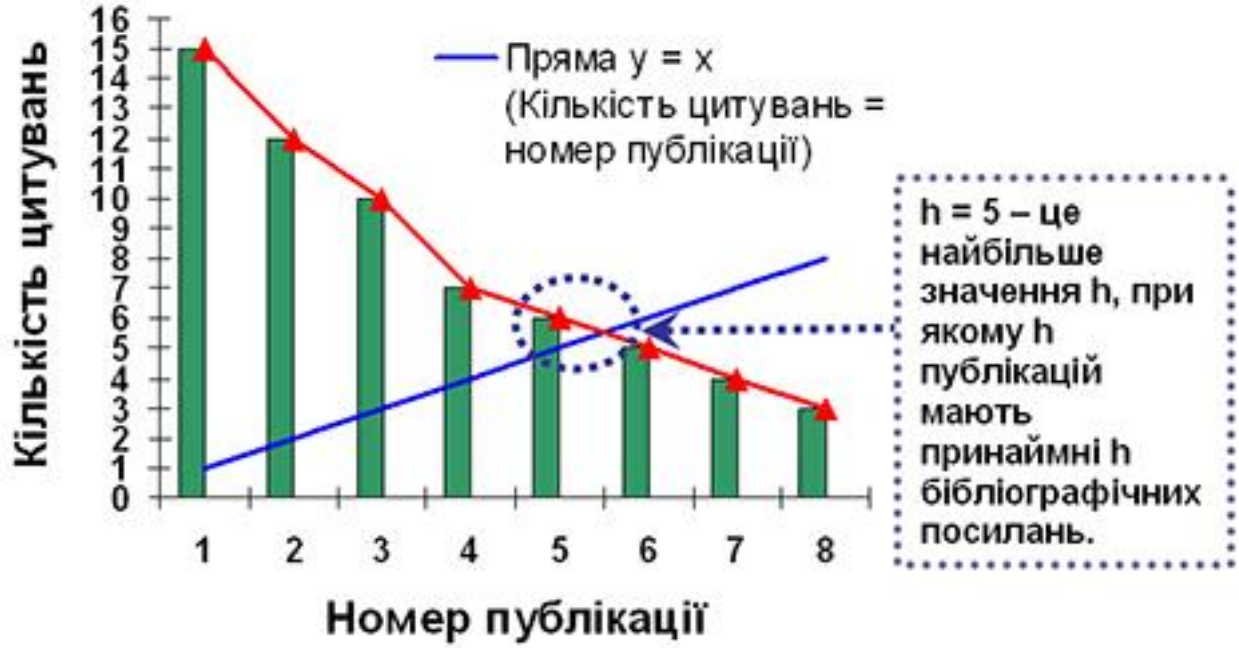


Рисунок 1.2 Індекс Хірша [8]

Тобто, якщо вчений з індексом h опублікував h статей, кожну з яких процитували h разів. Наприклад, індекс Хірша, що дорівнює 5 означає, що вченим опубліковано щонайменше 5 робіт, кожна з яких процитована не менше ніж 5 разів.

2) **h-core.** Це є набір з топ цитуємих h статей з публікації, на яких базється індекс Хірша. Наприклад, публікація, що має п’ять статей з цитованостями 25, 18, 9, 3, 2 відповідно має h-core 25,18, 9.
3) **H-median.** Це середнє значення цитованостей, що базується на H-core. Якщо взяти h-core, зазначений вище, то значення його h-median буде дорівнювати 5.
4) **h5-index, h5-core**, та **h5-median.** Показує перші три показника тільки для тих статей, які були опубліковані у п’ять останніх років.
5) **I-10 індекс.** Кількість публікацій, що цитувалися не менше ніж десять разів.

Scholar враховує приналежність вченого до закладу тільки в тому випадку, якщо в публікаціях було зазначено офіційне найменування вузу і використаний адреса електронної пошти, прийнятий у ВУЗі. Дані розраховуються з топ-10 профілів вчених кожного університету. Вчений з найвищим рейтингом в ВУЗі виключається з загального рейтингу, для поліпшення результатів.

У рейтинг по версії академії потрапили 38 ВУЗів України, з них 6 знаходяться в Києві[6]:

1) National Taras Shevchenko University of Kyiv (1525 місце)
2)National Technical university of Ukraine “Ihor Sikorsky Kyiv Polytechnic Institute” (1538 місце).
3)Kyiv National Economic University Vadim Hetman (1562 місце).
4) Borys Grinchenko Kyiv University (2642 місце)

5) Kyiv University of Economics and Law of the CRIC Krok University (2782 місце)

6) National Aviation University Kyiv International University of Civil Aviation (2934 місце)

По функціям Schoolar схожий на сайти Scirus(англ.) від видаництва Elsevier, CiteSeerX та GETCITIED. Також Schoolar схожий на сайти, що надають доступ до публікацій після оформлення платної підписки, наприклад Scopus видавництва Elsevier та Web of Science від інституту Thomson ISI. Рекламний слоган «Академії Google» - «стоючи на плечах гигантів» - частина знаменитого вислову Ньютона - «Якщо я бачив даліінших, то це тому, що стояв на плечах гигантів», дань вченим, шо вносили вклад в розвиток науки на протязі віків та забеспечившим основу для нових відкриттів та досягнень.

1.2 Історія створення проекту

Google Scholar виник з обговорення між Алексом Верстаком (Alex Verstak) і Анурагом Ачарья (Anurag Acharya), обидва потім працювали над створенням основного веб-індексу Google.

У 2006 році у відповідь на випуск Windows Live Academic Search від Microsoft, потенційного конкурента для Google Scholar, була реалізована функція імпорту цитат з використанням бібліографічних менеджерів (таких як RefWorks, RefMan, EndNote і BibTeX). Подібні можливості також реалізовані в інших пошукових системах, таких як CiteSeer і Scirus.

У 2007 році Ачарья оголосив, що Google Scholar почав програму з оцифрування та хостингу журнальних статей за угодою з видавцями, окремо від Google Books, чиї скани старих журналів не включають метадані, необхідні для пошуку конкретних статей у конкретних областях.

У 2011 році Google переробив верхню панель сайту google.com і видалив з неї посилання на «Scholar». Це ускладнило доступ до проекту.

1.3 Особливості та функції

«Академія Google» виконує пошук не тільки за статтями, доступним онлайн, але і за статтями, доступним тільки в бібліотеках або за гроші. «Наукові» результати пошуку генеруються з використанням посилань з «повнотекстових журнальних статей, технічних звітів, препринтів, дисертацій, книг та інших документів, в тому числі обраних веб-сторінок, які вважаються "науковими". У результатах пошуку «Scholar» виводить посилання на статті. Більшість з посилань ведуть на сторінки, що містять коротку інформацію про статті; можливо, за доступ до повного тексту статей доведеться заплатити. «Розширений пошук» дозволяє шукати в конкретних журналах або статтях.

Результати пошуку упорядковано:
• по рейтингу автора;
• кількості посилань на статтю;
• рейтингу статей, що посилаються на знайдену статтю;
• рейтингу журналів, в яких опубліковані посилаються статті;
• рейтингу журналу, в якому опублікована знайдена стаття.

З блоку «Цитується в» можна дізнатися список статей, в яких цитується розглянута стаття. Ця функція, зокрема, забезпечує індекс цитування, раніше доступний тільки в Scopus і Web of Knowledge. Даний індекс може бути використаний при вебометріческом ранжируванні сайтів.

У блоці «Статті по темі» виводиться список статей, схожих за змістом з даної статті. Статті впорядковані за ступенем схожості з даної статті і за своєю значимістю.

Ранжування

Процес вибудовування результатів пошукув в порядку наібільшої відповідності до пошукового запросу[1].

Більшість академічних баз даних та пошукових систем дозволяє користувачам вибирати один фактор для ранжування результатів (наприклад актуальність, кількість цитат, або дата публікації) Schoolar ранжує результати за допомогою комбінованного алгоритму ранжування, який діє «так як це роблять дослідники, враховуючи повний текст кожної статті, автора, видання, в якому стаття опублікована, і як часто вона була процитована в іншій науковій літературі»

1.4 Створення персонального кабінету

Одна з задач данної дипломної роботи – це виявлення некоректно заповнених профілей викладачів – потрібно навести приклад правильної регістрації персонального кабінета науковця у Google scholar.

1. Заходимо на головну сторінку Scholar - https://scholar.google.com.ua/

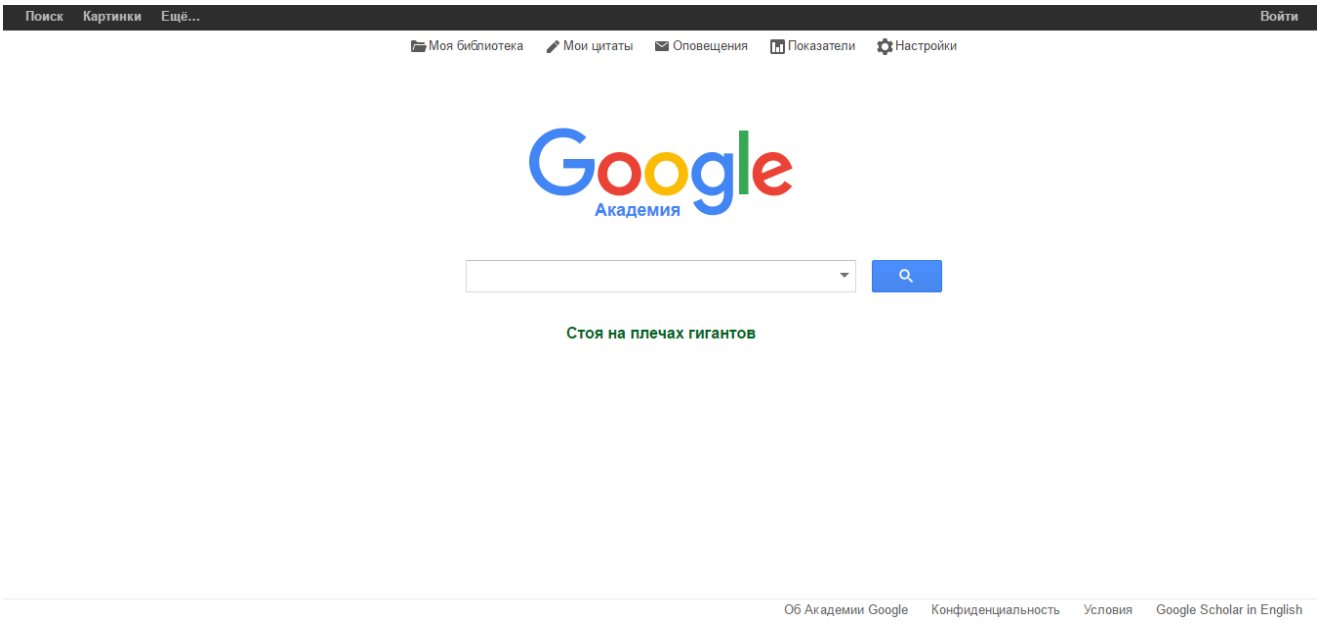


Рисунок 1.3 Головна сторінка академії [5]

Щоб створити кабінет потрібно мати почту від google. Якщо її немає – можна створити далі. Наступний крок - натиснути одне з посилань – зайти, мої бібліотеки чи мої посилання. Всі три направлять на сторінку, де потрібно ввести дані свого гугл акуанта.

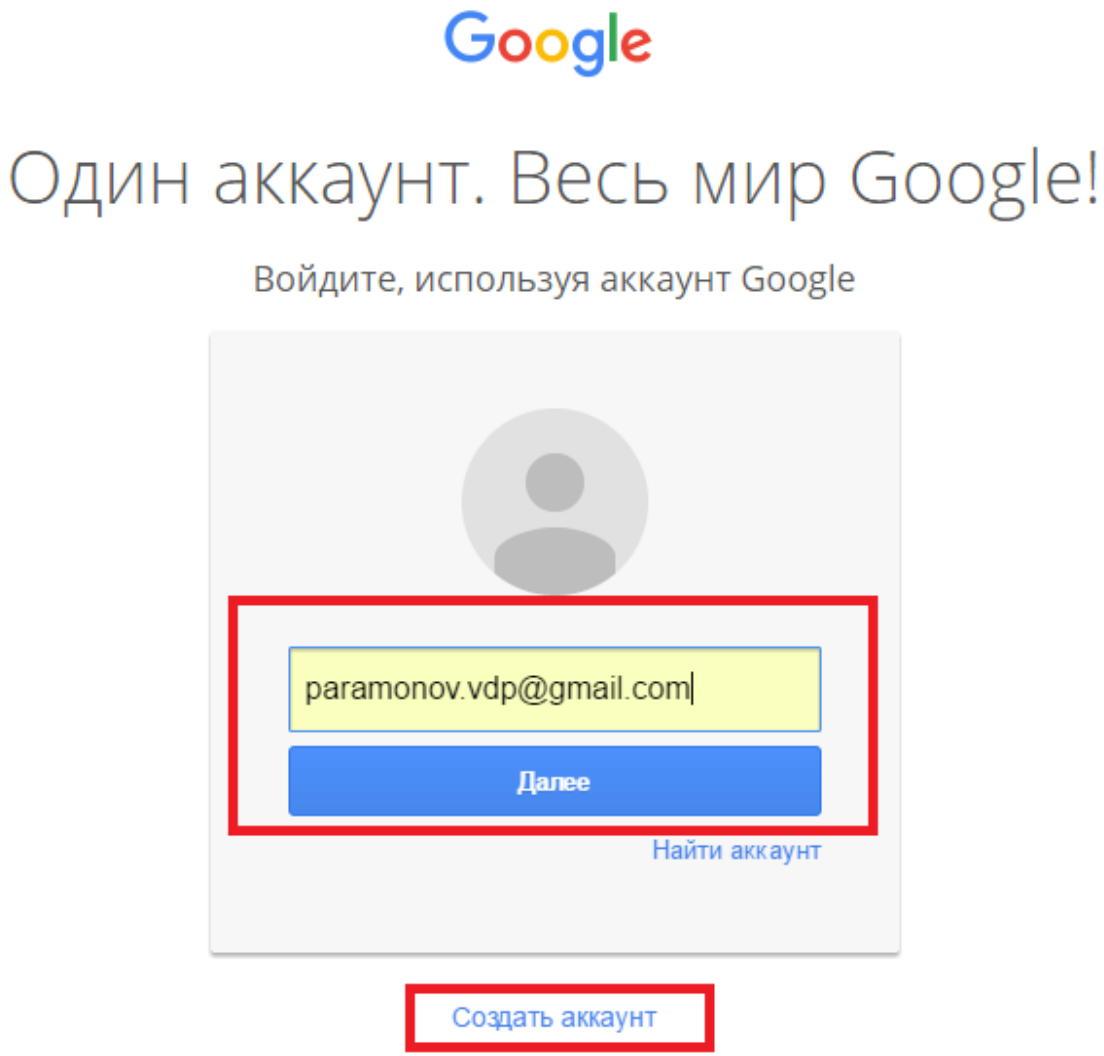


Рисунок 1.4 Варіанти входу [5]

Припустимо, що пошта гугл у нас є та заповнимо поле. Якщо пошти немає – натискаємо на посилання створити акуант. Реєстрація в пошті від гугл стандартна та мало чим відрізняється від реєстрації на інших почтових сервісах.

Після натискання кнопки далі потрібно ввести свій пароль від пошти. Якщо пароль було введено вірно то ми знову попадаємо на головну сторінку сервісу, та в правому верхньому куту теперь буде відображатися наша почта. Далі потрібно заповнити свій профіль. Для цього натискаємо «мої цитати».

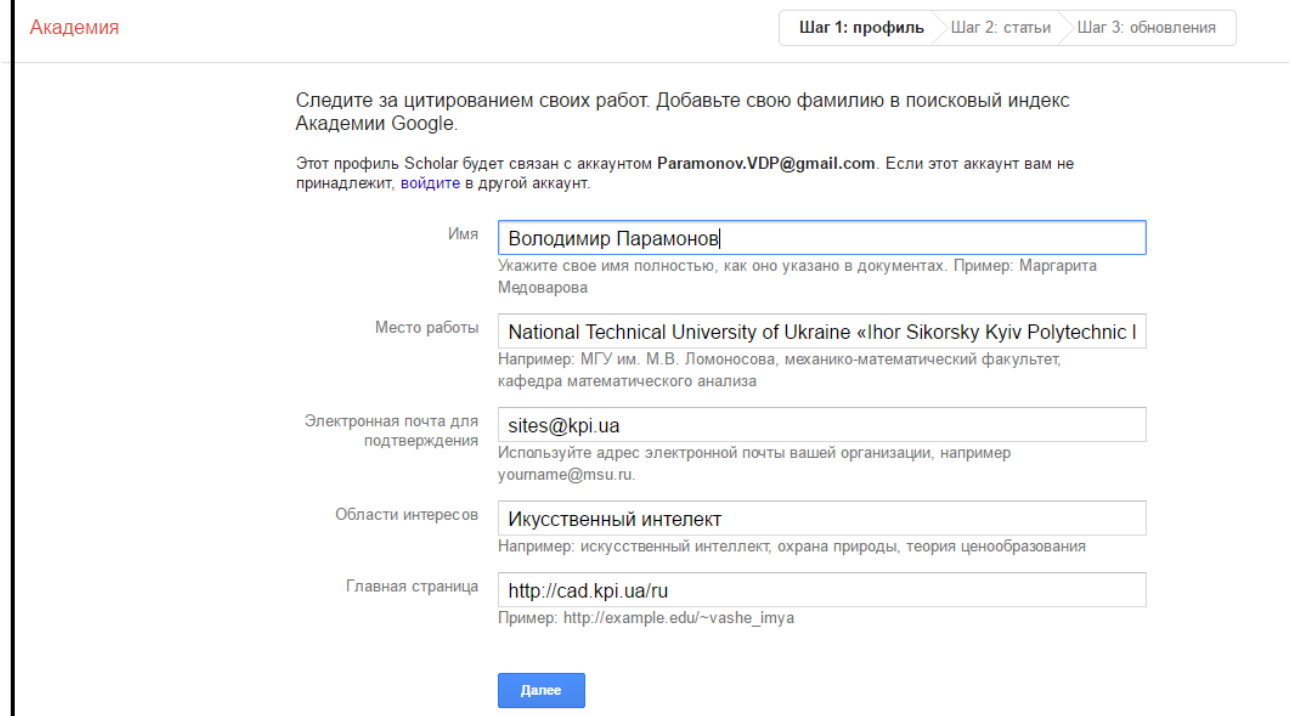


Рисунок 1.5 Заповнення персонального профілю [5]

Рекомандації щодо заповнення інформації профілю:

- В поле ім’я заносимо Ім’я та призвище.
- В поле місце роботи – треба ввести назву університету, яка зафіксована ГУГЛ , для КПІ ім. Ігоря Сікорського це **National Technical University of Ukraine**

- В поле електрона пошта для підтвердження потрібно обов’язково ввести почту від КПІ, інакше профіль не буде з підтвердженним навчальним закладом.
- В поле головна сторінка можна ввести сайт кафедри, або якщо є особисту сторінку, яка зв’язана з науковою діяльністю.
- Для того, щоб не зробити помилки краще керуватися http://phone.kpi.ua/schol

Натискаємо кнопку далі. Попадаємо на сторінку з вибором статей. Можна вибрати группу чи по одній статі, пошук ведеться по вашим імені та призвищу.

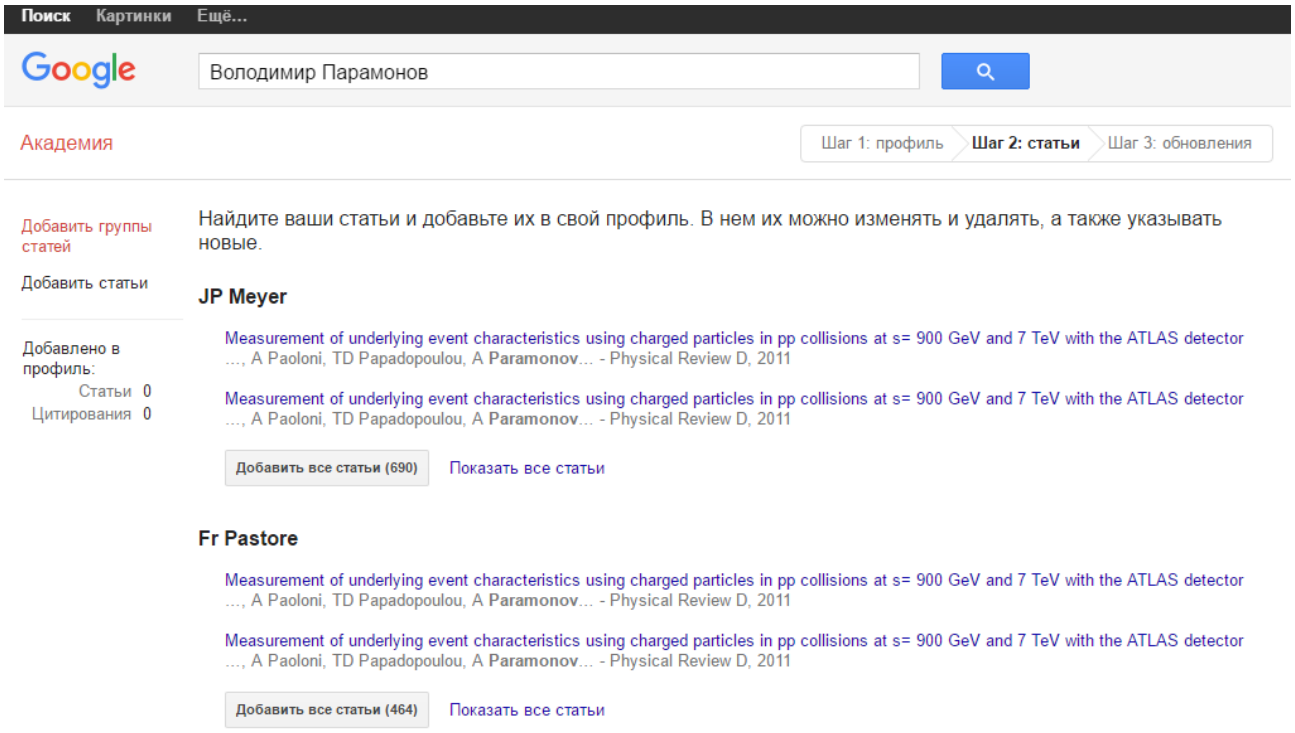


Рисунок 1.6 Вибір статей [5]

Цей крок можна пропустити, натиснувши внизу «Пропустити цей крок».

На завершальному третьому кроці пропонується вибрати як оновлювати ваш особистий профіль. Пропонується автоматичне додавання нових статей до

вашого профілю по вашим даним, оновлення бібліографічних даних та виявлення однакових матеріалів з їх подальшим видаленням чи об’єднанням.

При цьому зберігається можливість додавання статей в ручному режимі – автоматичне оновлення не торкнеться зроблених вами змін.

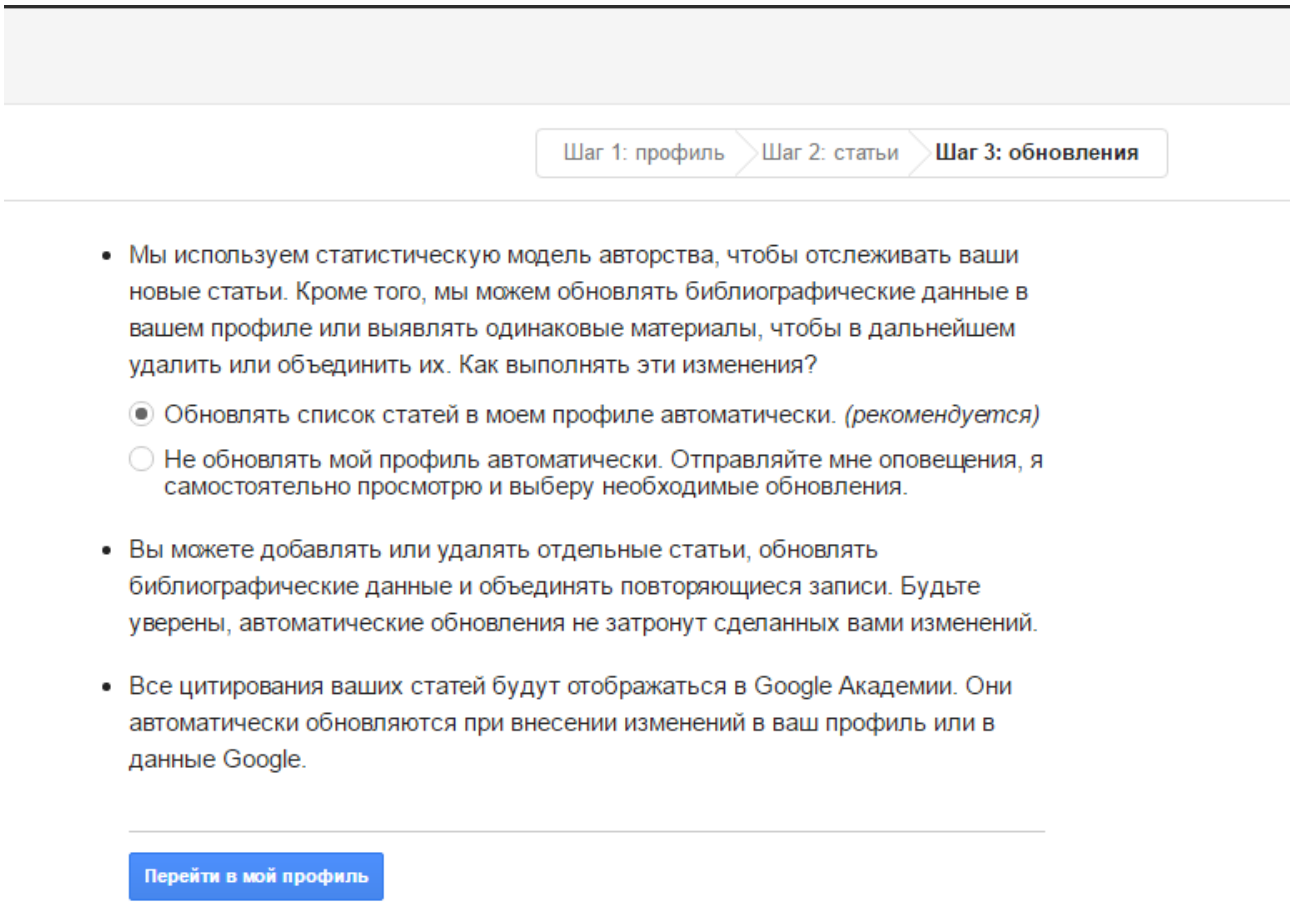


Рисунок 1.7 Вибір варіанту оновлення [5]

Автоматичне оновлення рекомендується ввімкнути, якщо ви маєте доволі рідкісне призвище, інакше до вашого профілю будуть додані матеріали від людей, чиї ім’я та фамілія співпадають з вашими.

Після цього кроку ми потрапляємо в особистий кабінет, як показано на малюнку нижче.

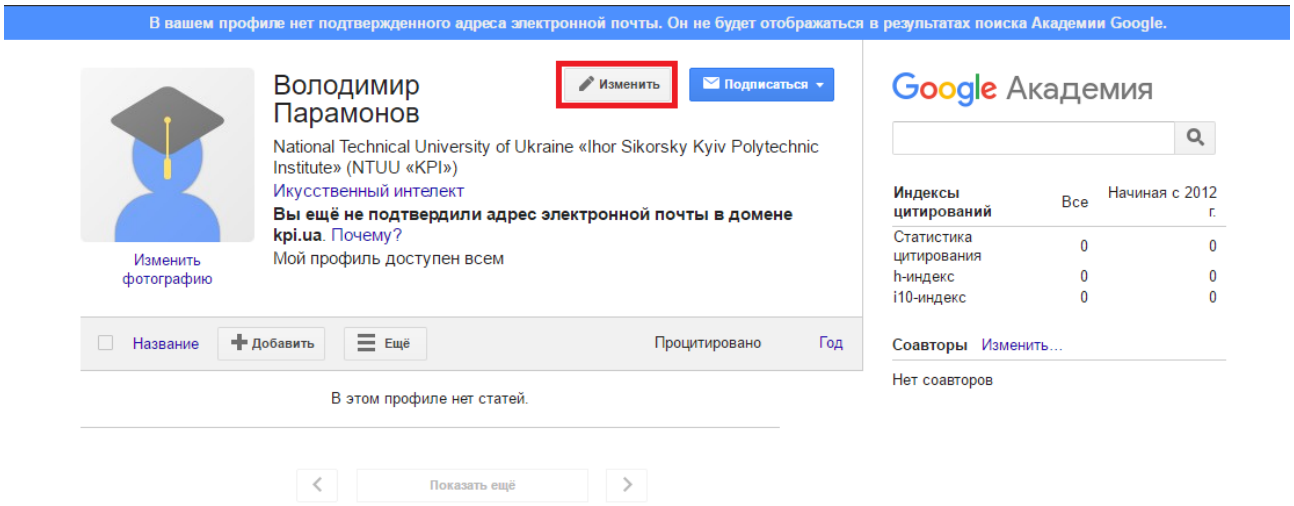


Рисунок 1.8 Персональний профіль

Далі рекомендується натиснути на кнопку «Змінити» та редагувати свої особисті дані:

- В поле Ім’я можна ввести свої прізвище та ініціали різними мовами, це потрібно для того щоб ваш профіль було легше знайти з різних точок світу.
- Також обов’язково треба відкрити спільний доступ до профілю, як показано на малюнку нижче:

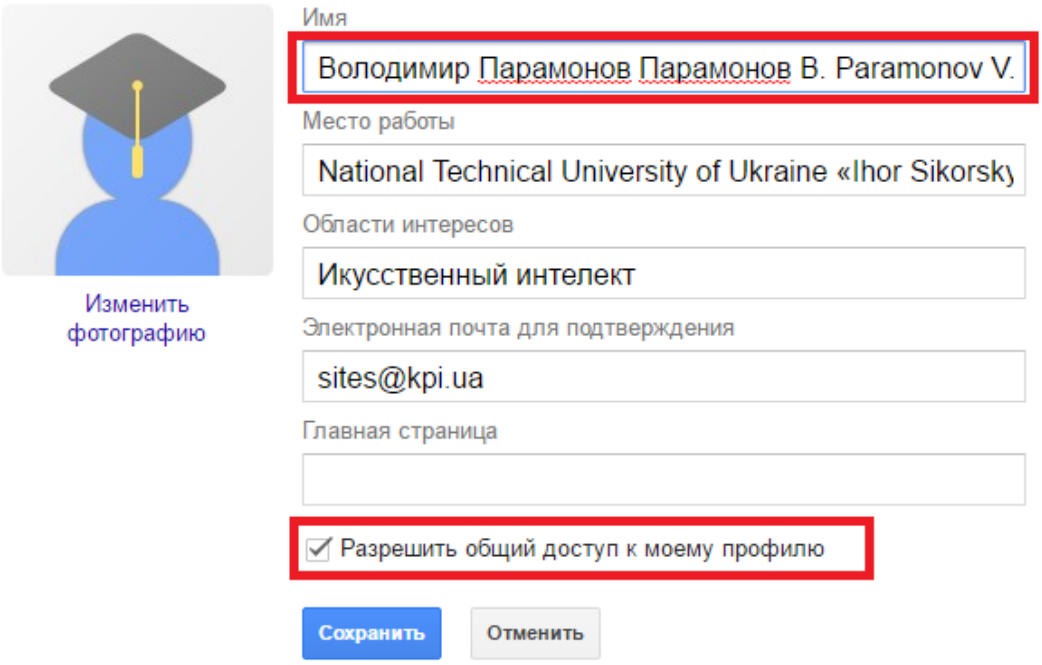


Рисунок 1.9 Надання загального доступу [5]

Якщо цього зробити – ваш профіль буде неможливо знайти. Також треба обов'язково підтвердити пошту, яку ви вказували як пошту для підтвердження – на неї прийде лист з посиланням для підтвердження.

Після підтвердження пошти назва навчального закладу повинна стати синього кольору та мати посилання на сторінку цього закладу, де будуть наведені всі науковці, що до нього належать.

Щоб знайти людину яка вам потрібна використовується поле пошуку з правої сторони кабінету:

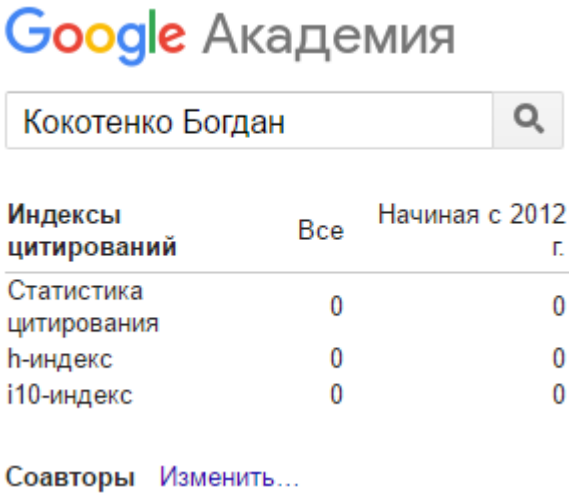


Рисунок 1.10 Пошук публікацій [5]

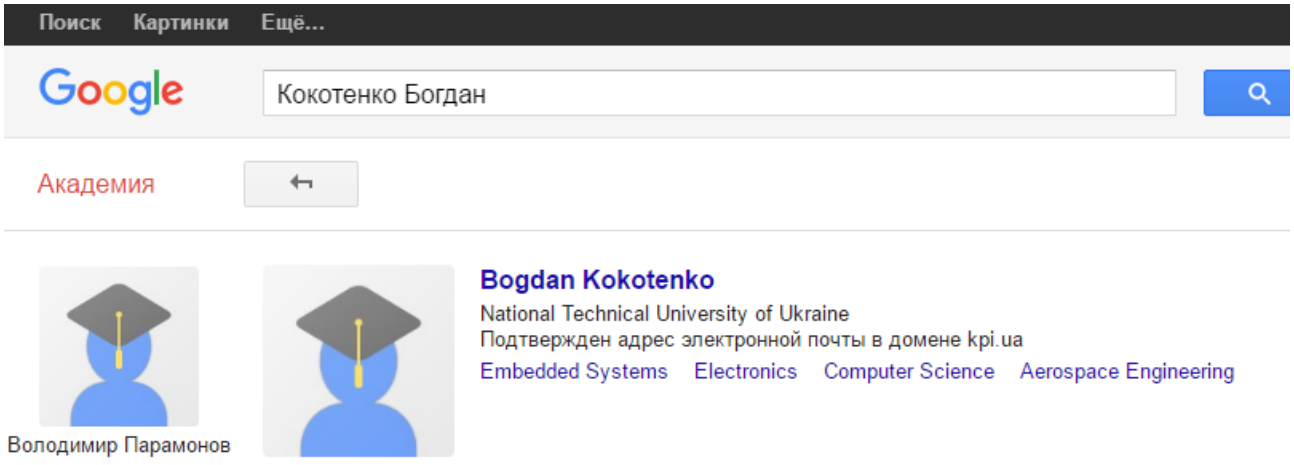


Рисунок 1.11 Результати пошуку [5]

Майте на увазі – якщо в вводите частину ініціалів, якою немає в профілі у людини яку ви шукаєте – пошук нічого не дасть. Наприклад якщо замість Кокотенко Богдан ввести Кокотенко Богдан Валерійович – пошук не дасть результатів.

Далі, якщо ви маєте публікації, їх можна додати. Для прикладу додамо в профіль декілька робіт. Натискаємо на кнопку «додати», що знаходиться на головній сторінці кабінету. Портапимо на сторінку, яку ми бачили на другому кроці регістрації, за вийнятком того що з’явиться додатковий розділ – додати статтю вручну. Сторінка має вісім категорій статей з усіма необхідними параметрами:

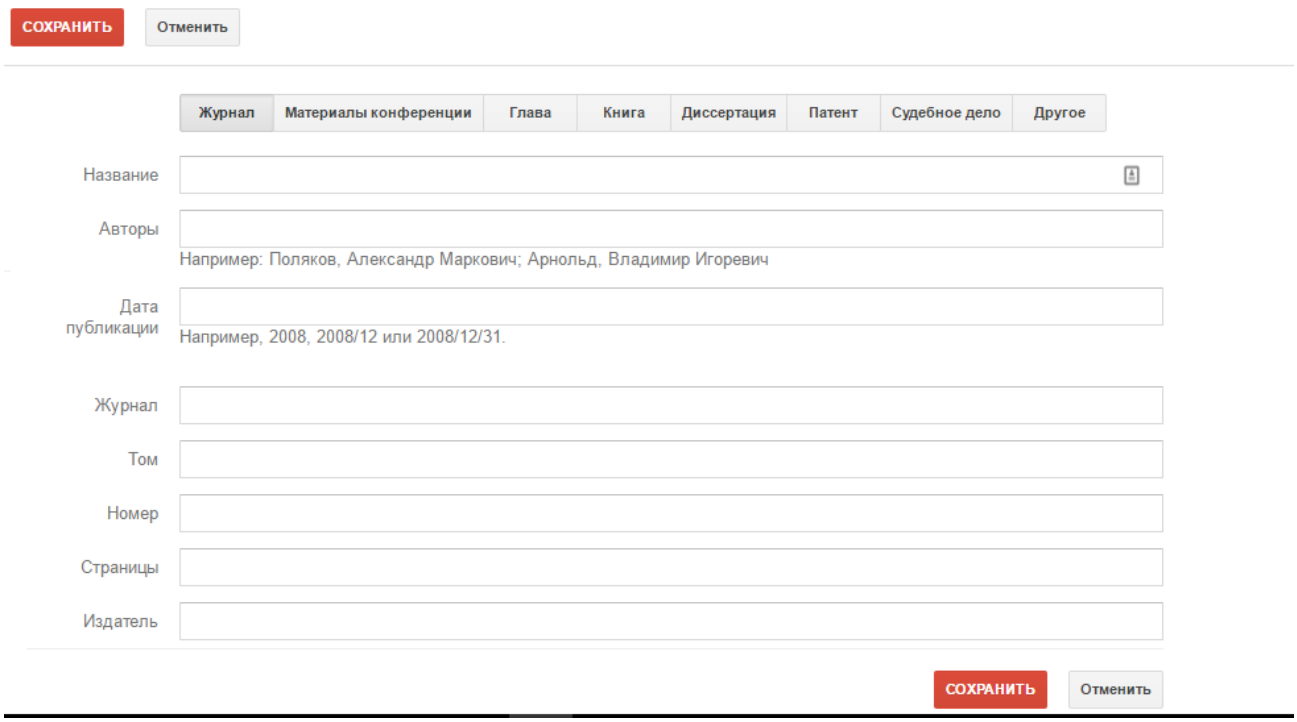


Рисунок 1.12 Додавання публікацій [5]

Для прикладу я використаю пункт меню «Додати статю», та додам декілька. Після цього в моєму профілю зростуть цитування та індекси:

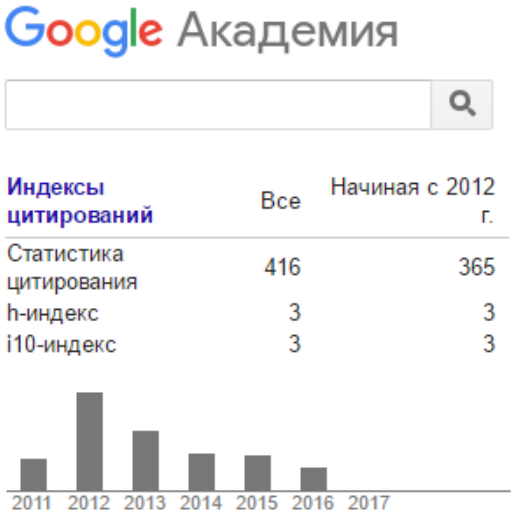


Рисунок 1.13 Цитування [5]

Звичайно це некоректні дані, адже я додав публікації, які мені не належать. Система дозволяє додавати до свого профілю будь-які публікації, проте через деякий час ці дані видаляються, якщо вони не належать людині яка їх дійсно написала. Тому треба бути уважним під час додавання своїх матеріалів до профілю.

1.5 Порівняння з іншими подібними системами

Наукові публікації можуть бути розміщені в трьох варіантах баз – відкриті, з обмеженим доступом та закриті.

Кожний журнал при розробці свого Web-сайту попадає в категорію видимих з InterNet, тобто відноситься до категорії відкритих. Браузери дозволяють його продивитись, а індексованість у пошукових системах дозволяє знайти певні (необхідні) матеріали, що надруковано у журналах.

До публікацій з обмеженим доступом доступом відносяться:

- Google Scholar
- ScienceDirect
- GetCITIED
- SiteSeerx

Щоб отримати доступ до публікацій в таких системах потрібно пройти реєстрацію.

Scholar є всесвітньовідомим, та на відміну від аналогів в сегменті має найбільшу кількість категорій наукових статей. Science direct має лише 9 категорій, а SiteSeerX взагалі обмежується лише комп'ютерними науками та інформаційними технологіями. Глобальна пошукова платформа Google Scholar, зокрема її апарат підрахунку наукової цитованості, спеціалізується на індексації наукових публікацій не лише онлайнових, а і друкованих науково-інформаційних продуктів (статей, монографій, препринтів, звітів та інших академічних матеріалів Реальними його конкурентами є представники закритого сегменту, доступ до яких відкривається тільки після оформлення платної підписки – це Scopus та Web of Science.

Їх основний сервіс- пошук наукових статей у базах відповідних систем та оцінка індекса цитувань.

Google Scholar з'явився пізніше, але його відкритість, гнучкість та популярність дозволила наздогнати ці популярні бренди.

Scopus – є найбільшою базою даних наукових публікацій, з великим тематично- типологічним покриттям.

Рисунок 1.14 Логотип Scopus [9]

Класифікаційна система Scopus включає 24 тематичних розділи. Тематичний охват розподіляється наступним чином:

- Фізичні науки
- Медичні науки
- Науки про життя
- Соціогуманітарні науки

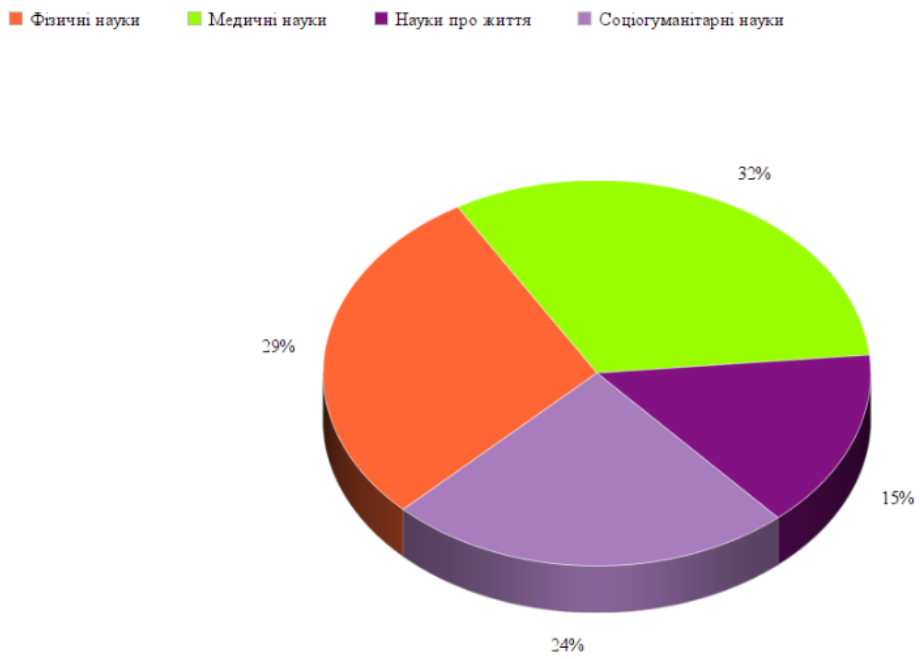


Рисунок 1.15 Гістограма тематичного охвату Scopus[2]

База даних Scopus налічує понад 60 млн записів, що включають:

- Понад 21,500 рецензованих журналів, з яких понад 4200 цілком відкриті.
- Більше ніж 360 торгових публікацій.
- Публікації з більш ніж 5000 міжнародних видавництв

Функціонал Scopus включає багато функцій для пошуку, аналізу та відкриттів:

- Пошук документів
- Пошук авторів
- Пошук належності - виявлення та оцінка аффілірованності наукової продукції.
- Система сповіщень
- Огляд джерел в алфавітному порядку
- Перегляд посилань – перегляд списку посилань обраної статті.

- Аналіз результатів пошуку (візуалізація метрик у вигляді графіків)
- Огляд цитувань. Аналіз тенденції цитованостей для будь – якої статті або списку авторських документів.
- Сторінки авторів. Можна легко аналізувати і відстежувати цитовоності окремо взятого автора.

Scopus дозволяє проводити пошук по більш ніж 15 тис. наукових видань, найбільш авторитетних в науковому середовищі. Оцінюється індекс цитованостей і Індекс Хірша.

Scopus – платна база, в яку журналу потрапити досить складно. Він індексує наукові джерела, що видаються на різних мовах, тільки за умови наявності у них англомовних версій анотацій.

Web of Science - об'єднує реферативні бази даних публікацій в наукових журналах та патентів, в тому числі бази, що враховують взаємне цитування публікацій, що розробляється і надається компанією Thomson Reuters.

Рисунок 1.16 Логотип Web of science[10]

Тематичний охват Web of science:

- Технічні науки
- Суспільні науки

- Природничі науки
- Гаманітарні науки
- Мистецтво

WOS – має велику кількість баз даних, що допомогають вченим на етапі проведення досліджень, коли їм потрібно перебрати велику кількість інформації.

WOS core collection[13] – головна база даних, з недесциплінарною інформацією. Містить шість архівів, найстарший з яких ведеться з 1840 року. Це перша база даних, в якій науковець шукає потрібну інформацію. Core collection фільтрує журнали по якості, з поміж усіх, якіх юільше 100000 у світі відбирає приблизно 13000. Критеріями для відбору є:

- Видаіничі стандарти(англійська мова, номер ISSN журналу)
- Автори з різних країн
- Перевірка вмісту журналу
- Аналіз цитування

Інші 11 баз є орінтованими на область науки чи регіон.

4 з них мають в своїх показниках імпакт- фактор:

- Derwent Innovations Index
- Data Citiation Index
- Biosis Citiation Inxex
- Zoological Record

7 баз використовують виключно індекс цитування:

- Meedline
- SciELO Citiation Index
- Russian Science Citiation index
- KCI- Korean Journal Database
- Current Contens Connect
- CABI
- FSTA

WOS, навідміну від Scopus та Google Scholar має додатковий показник, що називається імпакт-фактор. Він відображає рівень наукового журналу, що зареєстрований в базі WOS. Імпакт – фактор показує середню кількість цитованостей кожної опублікованої статті за два роки після її публікації і являє собою відношення числа цитованостей статей до числа всіх опублікованих статей в даному журналі. При розрахунках імпакт – фактору не враховується нотатки редакції, кореспонденція від читачів, новини, довідкова інформація і тому подібна інформація.

1.6 Постановка задачі

Розробити веб – додаток, що буде збирати та оброблювати дані цитованостей викладачів КПІ, будувати графіки по цитованостях, рахувати цитованості по кафедрах, факультетах, виділяти викладачів які мають профіль без належності до КПІ у Google scholar, відображення рейтингу з урахуванням неправильно заповнених профайлів.

1.7 Висновки

В цьому розділі було розглянуто три найпопулярніші бази електроних публікацій їхні особливості та функціонал. Оскільки ця робота присвячена Google scgolar, було розглянуто історію його створення та принцип роботи.

Також наглядно показано, як коректно створити особистий кабінет та додати до нього публікації. Підсумуємо головні дії, які необхідні для того, щоб профіль був коректний та вносив рейтинг науковця в відповідний підрозділ ВНЗ:

- Необхідно заповнити назву ВНЗ латинецею.
- Використовувати пошту від ВНЗ для підтвердження профілю, та обов’язково підтвердити її, перейшовши за посиланням у листі.

- Рекомендовано зробити свій профіль публічно доступним, щоб інші науковці мали змогу вас знайти.

Також в розділі поставлена задача по реалізації додатку.

Розділ ІІ Вибір засобів реалізації

2.1 Вибір платформи для реалізації додатку

Вхідними даними для додатку є:

- Сторінка КПІ на сайті академії (National Technical University of Ukraine)
- Сайт з цитованостями викладачів telef.kpi.ua
- Файл формату xls з ФІО викладачів та посиланнями на їх профілі.

В додаток входять наступні функції:

1) Обробка файлу з ФІО викладачів та парсинг їхній профілів у системі Schoolar з метою виявлення тих, хто некоректно заповнив свій профіль, збереження цієї інформації в пам’яті.
2) Обробка сайту telef з метою отримання кількісних показників цитованостей викладачів, збереження інформації в пам’яті.
3) Видача результатів цитованостей по вибраній кафедрі чи факультету, як кожного викладача, так і суми по кафедрі чи факультету, з урахуванням та без урахування цитованостей тих викладачів, що не кореектно заповнили свій профіль.
4) Відображення у виді гістограм 15 кафедр та факультетів, що мають найкращі показники, також з та без урахуванням некоректних даних.

Для реалізації додатку була обрана платформа Java, оскільки вона містить велику кількість готових бібліотек, що полегшить задачі парсингу.

2.2 Вибір фреймворків та бібліотек для реалізації додатку

Оскільки додаток повинен бути розміщений в веб, та мати серверну частину, найкраще для його реалізації підійде архітектурний патерн проектування, що має назву MVC.

MVC патерн

Шаблон проектування MVC передбачає розділення даних додатку, інтерфейсу користувача та керуючої логіки на три окремих компоненти:

- Model(модель)
- View(відображення)
- Controller(контролер)

Таким чином, модифікація кожного компоненту може виконуватись незалежно від інших. Під компонентом слід розуміти окрему частину коду, кожна з яких грає одну з ролей контролера, моделі, чи відображення.

Компонент модель містить в собі функціональну бізнес-логіку додаткуб не знає нічого про елементи дизайну та про те яким чином він буде відображений.

В обов'язки відображення входить відображення даних, отриманих від моделі, однак відображеня не може напряму впливати на модель, тільки зчитувати з неї дані. Найчастіше модель – HTML сторінка.

Контролер в свою чергу забспечує зв'язок між користувачем та системою, використовує модель та відображення для реалізації необхідних дій.

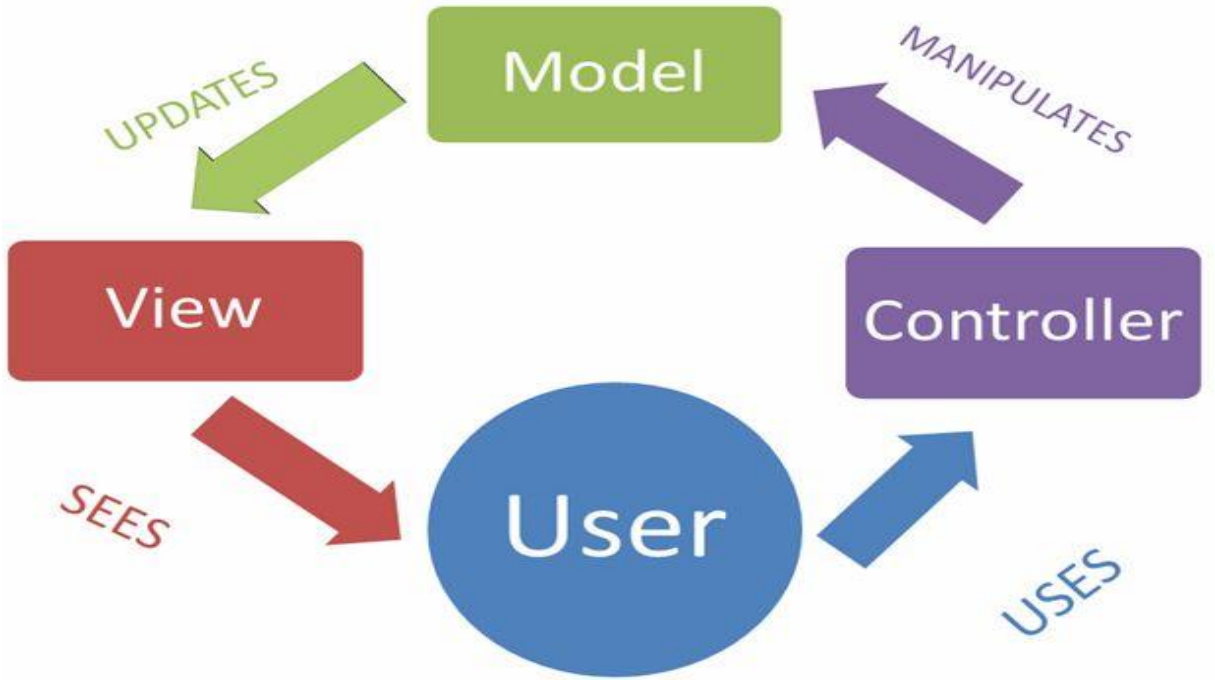


Рисунок 2.1 Взаємодія компонентів в MVC

Фреймворки MVC для платформи JAVA

На сьогодні існує декілька фреймворків, що підтримують MVC, найпопулярніші з них:

- Spring framework
- JSF
- Vaadin
- Play
- Grails

Spring framevork на разі є найпопулярнішим фреймворком на даний момент, тому що має чітку структуру та велику кількість модулів для вирішення будь – яких задач.

Рисунок 2.2 Логотип Spring[11]

JSF є частиною JAVA EE, офіційно підтримується Oracle. Незважаючи на легкість в використанні, цей фреймворк не дуже підходить для швидкої розробки, оскільки має великий поріг входження через складність його архітектури.

Рисунок 2.3 логотип JSF [11]

Vaadin базується на Google Web Toolkit, що добавляє архітектурі цього фреймворку складності. Однак зананя Google web toolkit не є обов'язковим для розробки. Головна перевага фреймворку – режим дизайну, в якому можливо будувати інтерфейс користувача в режимі WYSIWIG та писати логіку окремо для кожного компоненту.

Рисунок 2.4 логотип Vaadin [11]

Play написаний на Scala і Java (при розробці можна використовувати обидві мови, але спочатку фреймворк призначений для Scala). Даний фреймворк вкрай простий для вивчення - вже через 10 хвилин читання документації можливо написати перший перший своєрідний «Hello, world!».

Рисунок 2.5 логотип play [11]

Grails - фреймворк, написаний на скриптовій мовою Groovy, створений під впливом Ruby on Rails. Використовувати даний фреймворк можна як с Java, так і с Groovy, але останній варіант все-таки краще, так як при написанні коду на Java ви не зможете використовувати більшість чудових можливостей Grails

Рисунок 2.6 логотип Grails

IOC патерн

Інший принцип, який часто використовується разом з MVC – IOC(інверсія контролю), також використовується в розробці даної роботи. Суть принципу в тому, що кожен компонент системи має бути якнайбільше ізольованим від інших, не полагаючись в своїй роботі на деталі роботи конкретної реалізації інших компонентів.

Dependency Injection (впровадження залежностей) - це одна з реалізацій цього принципу. Работа фреймворка, обеспечивающая внедрение зависимости, описывается следующим образом. Приложение, независимо от оформления, исполняется внутри контейнера IoC, предоставляемого фреймворком. Часть объектов в программе по-прежнему создается обычным способом языка программирования, часть создается контейнером на основе предоставленной ему конфигурации.

Внедрение зависимости более гибко, потому что становится легче создавать альтернативные реализации данного типа сервиса, а потом указывать, какая именно реализация должна быть использована в, например, конфигурационном файле, без изменений в объектах, которые этот сервис используют. Это особенно полезно в юнит-тестировании, потому что вставить реализацию «заглушки» сервиса в тестируемый объект очень просто.

Порівняння фреймворків та вибір найбільш підходящого.

Для того щоб зробити коректний вибір фреймворку, враховуючи їхню схожість буде практично зробити порівняльну таблицю.

Таблиця 1 Порівняння фреймворків за функціоналом

Функція / Фреймворк	Spring	Jsf	Vaadin	Play	Grails
MVC	+	+	-	+	+
IOC	+	Дуже спрощений	-	+	-
Виконання функціоналу за розкладом	+	+	+	+	+
Додаткові модулі	+	-	-	Тільки один	+
ORM	Hibernate, Ibatis, JPA	+	+	JPA, Hibernate	Hibernate, Gorm
Частота використання в корпоративному сегменті	Найбільш часто використовуємий у світі	Дуже популярний	Не дуже Популя-рний	Популя-рний	Популяр-ний
Склайдність	Середня	Середня	Простий	Простий	Середня

Для розробки додатку найважливішими є три компоненти, які мають бути присутніми у фреймворку:

- Наявність MVC патерну.
- Наявність IOC патерну.
- Виконання функцій за розкладом.

Останній пункт необхідний у зв’язку з тим, що за алгоритмом роботи додатку він робить парсинг різних джерел, дані в яких в перспективі змінюються. Тому треба повторювати основні функції через деякий час, щоб мати актуальні дані. Звичайно можна розробити такий функціонал власноруч, проте це буде зайвим витрачанням часу, оскільки в більшості фреймворків це вже реалізовано, і є можливість використовувати цей функціонал написавши одну строчку коду.

По трьом головним компонентам відразу можна відкинути Vaadin та Grails frameworks.

З точку зору майбутнього розширення функціоналу додатку можуть знадобитися додаткові модулі, яких у Spring багато на всі можливі випадки,а у JSF зовсім немає. Також можливим розширенням роботи додатку може стати робота за базами даних. По цьому пункту підійдуть обидва фреймворки, проте Spring використовує більш популярні методи по роботі з БД. Тож, в підсумку для реалізації додадтку підійдуть два фреймворки, але враховуючи перспективи розширення та частоту використання Spring безперечно попереду, тож його і було обрано і використано при реалізації додатку.

2.3 Spring framework

2.3.1 Загальні відомості

Spring Framework (або коротко **Spring**) універсальный фреймворк відкритим кодом для Java-платформы. Фреймворк був вперше випущений під ліцензією Apache 2.0 license в червні 2003 року. Перший стабільний реліз 1.0 був випущений в березні 2004. Spring 2.0 був випущений в жовтні 2006, Spring 2.5 - в листопаді 2007, Spring 3.0 в грудні 2009, Spring 3.1 в грудні 2011. Поточна версія - 4.2.6. Наразі вважається головним фреймворком, що необхідно знати Java- розробнику. Spring забезпечує вирішення багатьох завдань, з якими стикаються Java-розробники і організації, які хочуть створити інформаційну систему, засновану на платформі Java. Через широку функціональності важко визначити найбільш значущі структурні елементи, з яких він складається. Spring не повністю пов'язаний з платформою Java Enterprise, незважаючи на його масштабну інтеграцію з нею, що є важливою причиною його популярності.

Spring, ймовірно, найбільш відомий як джерело розширень (features), потрібних для ефективної розробки складних бізнес-додатків поза великовагових програмних моделей, які історично були домінуючими в промисловості. Ще одне його достоїнство в тому, що він ввів раніше невикористовувані функціональні можливості в сьогоднішні панівні методи розробки, навіть поза платформи Java.

2.3.2 Модулі

Spring може бути розглянутий як колекція менших фреймворків або фреймворків у фреймворку. Більшість цих фреймворків може працювати незалежно один від одного, проте вони забезпечують більшу функціональність при спільному їх використанні. Ці фреймворки діляться на структурні елементи типових комплексних програм:

- Inversion of Control-контейнер
- MVC- фреймворк – заснований на http і сервлетах, що надає безліч можливостей для розширення та налаштування
- Фреймворк віддаленого доступу -дозволяє налаштування передачі Java – обєктів через мережу в стилі RPC, що підтримує RMI, CORBA, HTTP- based протокол, включаючи веб – сервіси.
- Фреймворк аспектно – орієнтованого проектуваня - працює з функціональністю, яка не може бути реалізована можливостями обєктно – орієнтованого програмування на Java без витрат.
- Фреймворк доступу до даних – працюэ з системами керуваня баз даних на Java –платформі, використовуючи Jdbc I ORM засоби, і забеспечує вирішеня завдань, які повторюются в великому числі Java-based додатків.
- Фреймворк управління транзакціями об’єднує декілька API управління транзакціями та координує операції для Java – об’єктів.
- Аутентифікація та авторизація – настроювані процеси безпеки, які підтримують цілий ряд стандартів, протоколів, інструментів і практик.
- Віддалене управління – конфігураційний вплив та управління Java – об’єктами для місцевої або віддаленої конфігурації через JMX
- Тестування – підтримка класів для написання юніт-тестів.

2.4 Висновки

В цьому розділі було розглянуто та порівняно найпопулярніші фреймворки для розробки на платформі Java, порівняно їх функціонал необхідний для розробки та можливого розширення додатку в майбутньому. Обраний оптимальний для розробки програмного сервісу для дипломної роботи, також наведено його опис.

Розділ ІІІ Програмна реалізація додатку

3.1 Веб – додаток

Під час написання дипломної роботи було розроблено програмний продукт, що складається з серверної та front-end частини.

Для реалізації інтерфейсу було використовано HTML та CSS фреймворк Bootstrap, скриптова мова Javascript для відправки даних на серверну частину додатку та побудови графіків.

Ці технології є своєрідним стандартом в побудові веб інтерфейсів, та забеспечують стабільну та швидку роботу.

3.1.1 Опис функціоналу

Доступ до додатку можливий за адресою:

- 192.168.31.80:8085/index

Нижче наведений скріншот головного окна додатку, де потрібно вибрати кафедру з випадаючого списку. Якщо потрібно отримати цитованості по факультетам, потрібна нажати на посилання «факультети», сторінка за цим посиланням виглядає так самао, тільки в випадаючому списку замість кафедр потрібно обрати факультет.

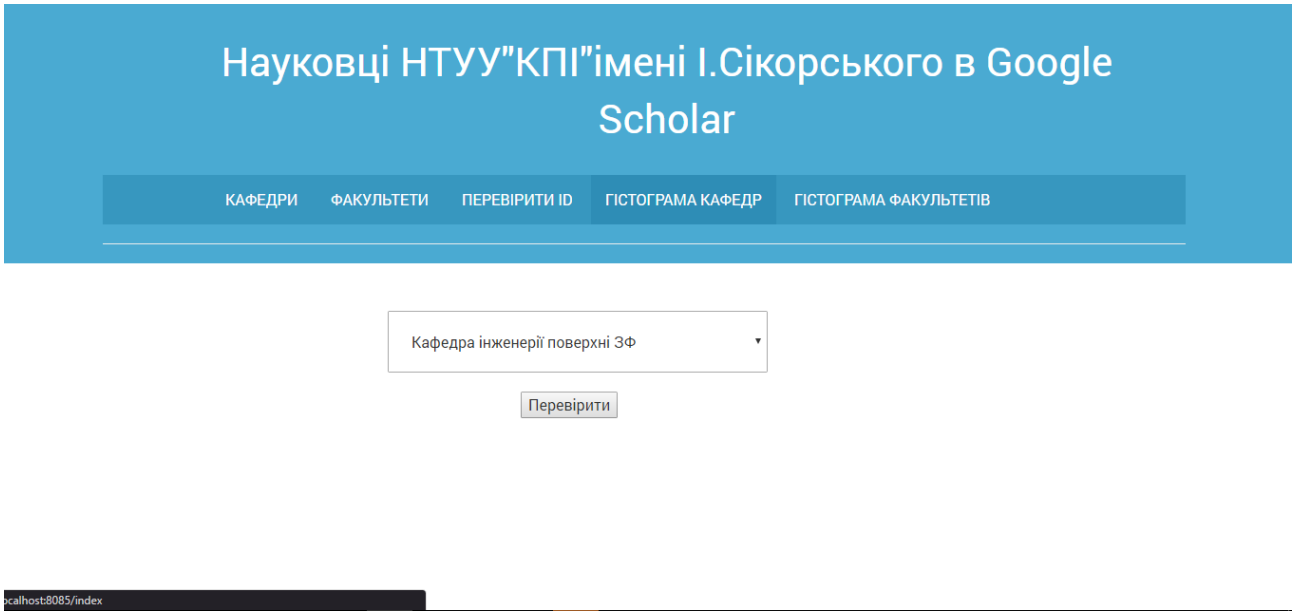


Рисунок 3.1 Головна сторінка сервісу

Щоб отримати результати, нажимаємо кнопку перевірити, після чього попадаємо на сторінку з результатами. Викладачі, що мають непідтверджені профілі у системі Schoolar виділені в результуючій таблиці червоним кольором.

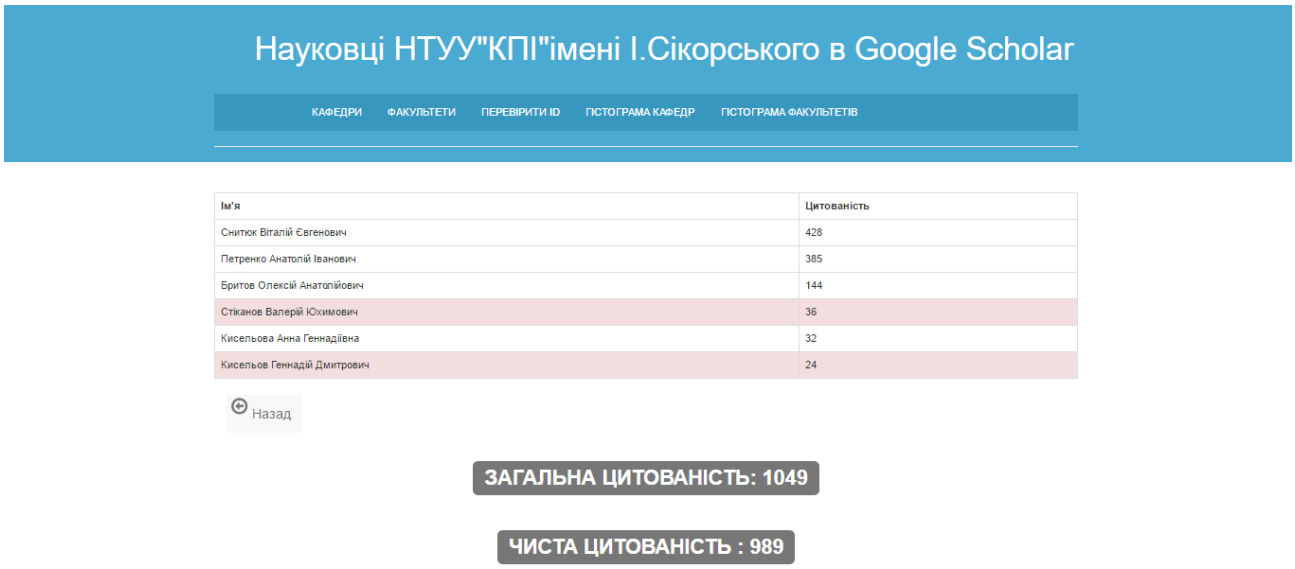


Рисунок 3.2 Результат по кафедрі

Алгоритм роботи додатку:

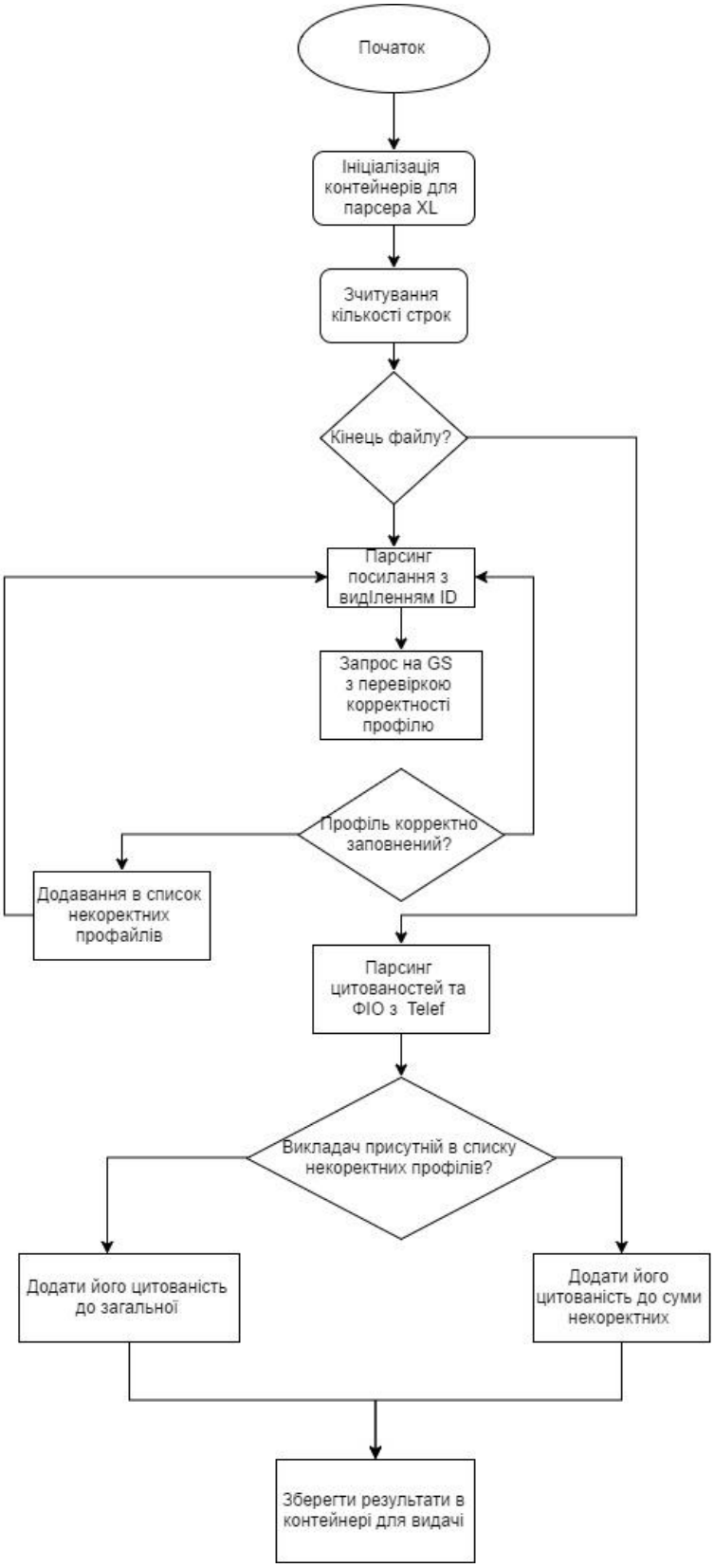


Рисунок 3.3 Алгоритм роботи додатку

На малюнку видно дві суми цитованостей: перша – сума без урахування некоректно заповнених профілей, в другій – вищезазначені профілі не сумуються.

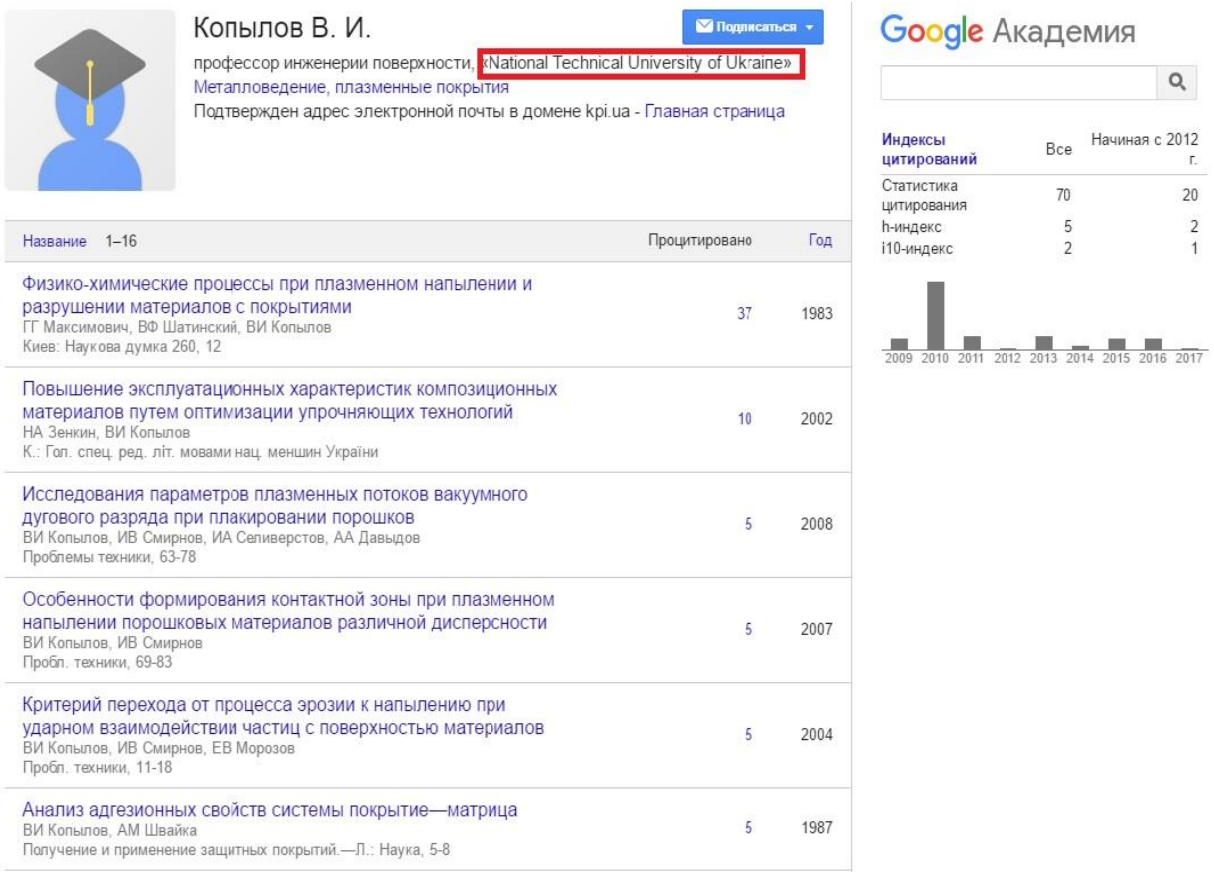
Повний список викладачів береться з файлу, що лежить в архіві з веб сторінками в додатку, та періодично оновлюються автоматично.

1	ПІП	факультет	Кафедра	Посада	Ставка	Профіль
2	Володарський Євген Ти	faks	Кафедра автоматизації експериментальних д	професор	1	user=NS6ChT0AAAAJ
3	Назарчук Ірина Василів	iasa	Кафедра математичних методів системного	старший ви	1	user=_2FZHCYAAAAJ
4	Гомонай Олена Василів	fti	Кафедра інформаційної безпеки ФТІ	професор	1	user=_2SzWC0AAAAJ
5	Маріц Дарія Олександр	fsp	Кафедра інформаційного права та права інте	доцент	1	?user=_3Q_jXUAAAAJ
6	Лошицький Павло Павл	fel	Кафедра фізичної та біомедичної електроніки		1	user=_4rVUSAAAAAJ
7	Концевой Андрій Леоні	xtf	Кафедра технології неорганічних речовин та	доцент	1	user=_5dUA2EAAAAJ
8	Шалденко Олексій Вік	tef	Кафедра автоматизації проектування енерге	асистент	0,25	user=_8HeTSgAAAAJ
9	Задерей Петро Василь	fmf	Кафедра математичного аналізу та теорії йм	професор	0,5	?user=_8rCbPsAAAAJ
10	Передерієнко Надія Ів	vpi	Кафедра організації видавничої справи, полі	доцент	0,93	?user=_9jSz5cAAAAJ
11	Ярощук Людмила Дем'	ihf	Кафедра автоматизації хімічних виробництв	доцент	1	?user=_EESI0IAAAAJ
12	Глоба Олександр Васи	mmi	Кафедра інтегрованих технологій машинобуд	доцент	1	?user=_gL3IQ8AAAAJ
13	Городкова Ганна Сергії	fl	Кафедра англійської мови гуманітарного сп	викладач	1	user=_GnENtsAAAAJ
14	Жураковський Ярослав	ihf	Кафедра автоматизації хімічних виробництв	старший ви	1	user=_HFx9M8AAAAJ
15	Огієнко Валентина Пет	fl	Кафедра англійської мови технічного спряму	старший ви	1	ser=_HRN2HMAAAAJ
16	Ковбаса Сергій Микола	fea	Кафедра автоматизації електромеханічних си	доцент	1	user=_nuLQVEAAAAJ
17	Матеєнко Юрій Петрови	fea	Кафедра електричних станцій ФЕА	доцент	1	ser=_O1PBWoAAAAJ
18	Багінський Андрій Вла	fsp	Кафедра соціології ФСП		0,75	user=_PACVn8AAAAJ
19	Шибецький Владислав	fbt	Кафедра біотехніки та інженерії ФБТ	асистент	1	?user=_PI77icAAAAJ
20	Розен Віктор Петрович	iee	Кафедра автоматизації управління електроте	завідувач	1	ser=_pr0WSMAAAAJ

Рисунок 3.4 Список всіх викладачів.

Після першого запуску додатку автоматично формується список з тих викладачів, що мають не підтверджену установу і тримається в пам'яті під час роботи додатку. Список оновлюється автоматично раз на місяць.

Нижче наведене зображення профылю викладача з обраної кафедри, що був виділений червоним кольором. Можемо переконатись, що на цьому профілі дійсно не зазначений заклад та цитованість співпадає.



Копылов В. И.

Подписаться

профессор инженерии поверхности, «National Technical University of Ukraine»

Металловедение, плазменные покрытия

Подтвержден адрес электронной почты в домене kpi.ua - Главная страница

Google Академия

Индексы цитирований	Все	Начиная с 2012 г.
Статистика цитирования	70	20
h-индекс	5	2
i10-индекс	2	1

2009 2010 2011 2012 2013 2014 2015 2016 2017

Название 1–16	Процитировано	Год
Физико-химические процессы при плазменном напылении и разрушении материалов с покрытиями ГГ Максимович, ВФ Шатинский, ВИ Копылов Киев: Наукова думка 260, 12	37	1983
Повышение эксплуатационных характеристик композиционных материалов путем оптимизации упрочняющих технологий НА Зенкин, ВИ Копылов К.: Гол. спец. ред. літ. мовами нац. меншин України	10	2002
Исследования параметров плазменных потоков вакуумного дугового разряда при плакировании порошков ВИ Копылов, ИВ Смирнов, ИА Селиверстов, АА Давыдов Проблемы техники, 63-78	5	2008
Особенности формирования контактной зоны при плазменном напылении порошковых материалов различной дисперсности ВИ Копылов, ИВ Смирнов Пробл. техники, 69-83	5	2007
Критерий перехода от процесса эрозии к напылению при ударном взаимодействии частиц с поверхностью материалов ВИ Копылов, ИВ Смирнов, ЕВ Морозов Пробл. техники, 11-18	5	2004
Анализ адгезионных свойств системы покрытие—матрица ВИ Копылов, АМ Швайка Получение и применение защитных покрытий.—Л.: Наука, 5-8	5	1987

Рисунок 3.5 Профіль з не підтвердженим закладом [12]

Інша частина сервісу присвячна гістограмному відображенню цитованостей по кафедрам та факультетам. Щоб їх побачити, нажимаємо посилання «гістограма кафедр» та «гістограма факультетів» відповідно.

Гістограми відображають 10 кафедр з найбільшими показниками цитованостей, та мають по дві шкали на кожну кафедру:

- Синя – сума цитованостей без урахування профілей з непідтвердженним закладом.
- Червона – сума цитованостей, з якої віднімається сума з некоректно заповненими профілями.

Тобто можливо наглядно подивитись, скільке саме рейтингу реально втрачає кафедра чи факультет через ці профілі, і стимулювати викладачів до коректного завовнення профілю.

Дані для відображення додаток бере з сайту telef.kpi.ua, збирає їх під час першого запуску та синхронізує їх з telef приблизн в той час, коли він оновлює свої дані, роблячи парсинг сайту.

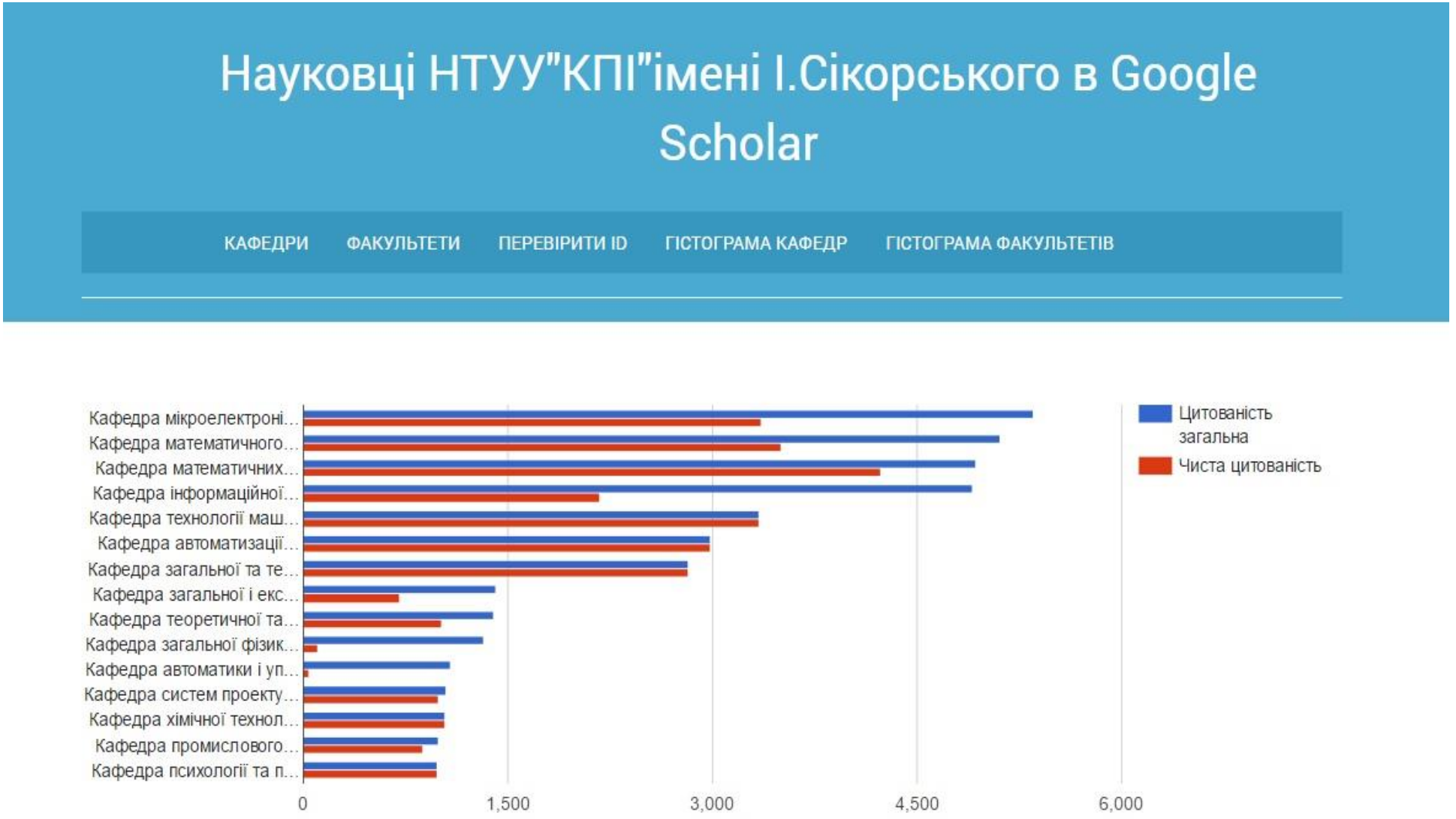


Рисунок 3.6 Гістограма кафедр

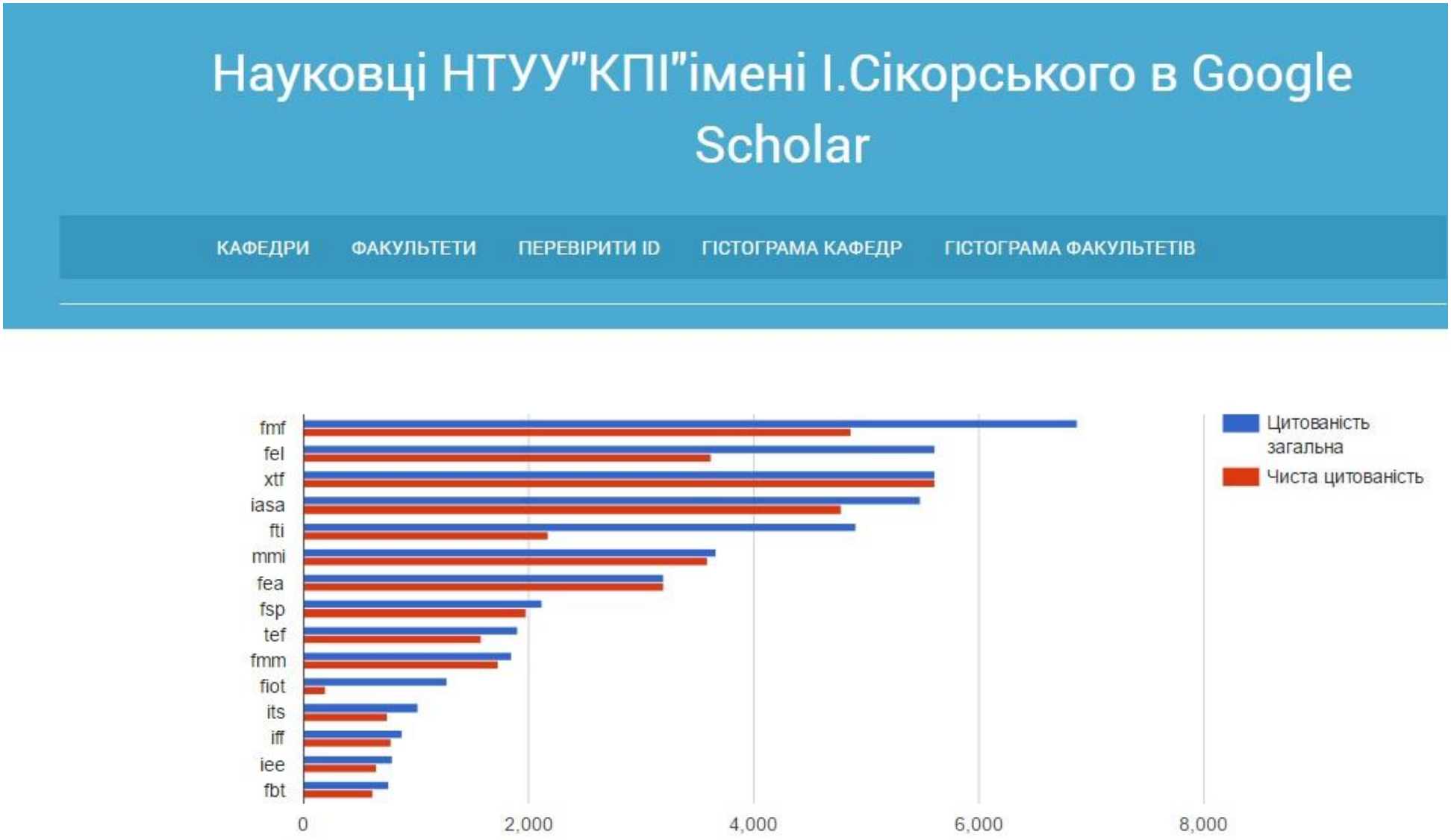


Рисунок 3.7 Гістограма факультетів.

3.2 Додаткові засоби реалізації

Для створення HTML - сторінок додатку було використоно безкоштовний редактор від Adobe, що має назву Brackets, тому що в ньому він орієнтований на роботу одразу з тьома технологіями: HTML, CSS, Javascript, та маэ велику кількість доповнень, та досить стабільне та зручне вбудоване живе відображення сторінок під час редагування коду.

Для створення гістограм була використана JS та Jquery бібліотека від гугл – Google charts. Має велику кількість графіків та доволі простий спосіб їх будування.

3.3 Висновки

В ході виконання дипломної роботи було розроблено програмний продукт – веб сервіс для аналізу та відображення даних про науковців КПІ.
Це сайт, що дозволяє перевіряти цитованості науковців з коректними та некоректними профайлами у системі Scholar, можливістю подивитись всі некоректно заповнені профілі.

Розділ IV Управління строками та ризиками виконання дипломного проекту

4.1 Задачі

1. Дослідити систему Google scholar та індекси що в ньому використовуються.
2. Дослідити подібні системи.
3. Обрати мову програмування для реалізації додатку.
4. Обрати засоби для розробку додатку.
5. Розробити веб- додаток

	Етап підготовки	21 дней?	**05.09.16 8:00**	**03.10.16 17:00**	
	Визначення цілей та задач роботи	5 дней?	05.09.16 8:00	09.09.16 17:00	
	Пошук інформації по темі	12 дней?	12.09.16 8:00	27.09.16 17:00	2
	Пошук додаткових джерел інформації	5 дней?	12.09.16 8:00	16.09.16 17:00	3SS
	Коригування цілей і завдань з урахуванням наявних д	3 дней?	28.09.16 8:00	30.09.16 17:00	4;3
	формування списку літератури	1 день?	03.10.16 8:00	03.10.16 17:00	5
	Робота над першим розділом	25 дней?	**04.10.16 8:00**	**07.11.16 17:00**	**6**
	Аналіз інформації та літератури про Schollar	5 дней?	04.10.16 8:00	10.10.16 17:00	
	Аналіз індексів Scholar	2 дней?	11.10.16 8:00	12.10.16 17:00	8
	Аналіз функцій та можливостей	2 дней?	13.10.16 8:00	14.10.16 17:00	9
	Вивчення проблеми некоректних профілів і способів її	7 дней?	17.10.16 8:00	25.10.16 17:00	10
	Виклад способу коректної реєстрації	1 день?	26.10.16 8:00	26.10.16 17:00	11
	Пошук подібих систем	3 дней?	27.10.16 8:00	31.10.16 17:00	12
	Аналіз їх функціоналу та можливостей	3 дней?	01.11.16 8:00	03.11.16 17:00	13
	Оформлення пункту про порівняння систем	2 дней?	04.11.16 8:00	07.11.16 17:00	14
	Постановка задачі	1 день?	04.11.16 8:00	04.11.16 17:00	15SS
	Робота над другим розділом	22 дней?	**08.11.16 8:00**	**07.12.16 17:00**	**7**
	Визначення функціоналу додатку	3 дней?	08.11.16 8:00	10.11.16 17:00	
	Аналіз вхідних даних для додатку	2 дней?	11.11.16 8:00	14.11.16 17:00	18
	Аналіз популярних фреймворків	10 дней?	15.11.16 8:00	28.11.16 17:00	19
	Порівняння фреймворків, вибір найбільш підходящого	4 дней?	29.11.16 8:00	02.12.16 17:00	20
	Оформлення розділу	3 дней?	05.12.16 8:00	07.12.16 17:00	21
	Робота над додатком	22 дней?	**08.12.16 8:00**	**06.01.17 17:00**	**17**
	Створення алгоритму роботи	7 дней?	08.12.16 8:00	16.12.16 17:00	
	Створення контроллерів	2 дней?	19.12.16 8:00	20.12.16 17:00	24
	Написання коду	12 дней?	19.12.16 8:00	03.01.17 17:00	25SS
	тестування	3 дней?	30.12.16 8:00	03.01.17 17:00	26FF
	Відлагодження недоліків	3 дней?	04.01.17 8:00	06.01.17 17:00	27
	Робота над третім розділом	1 день?	**09.01.17 8:00**	**09.01.17 17:00**	**23**
	Формування тексту	1 день?	09.01.17 8:00	09.01.17 17:00	
	Завершальна робота, підготовка	7 дней?	**10.01.17 8:00**	**18.01.17 17:00**	**30**
	Створення креслень та додатків.	1 день?	10.01.17 8:00	10.01.17 17:00	
	Проходження контролю норм	1 день?	11.01.17 8:00	11.01.17 17:00	32
	Підготовка до захисту	5 дней?	12.01.17 8:00	18.01.17 17:00	33

Рисунок 4.1 Всі задачі

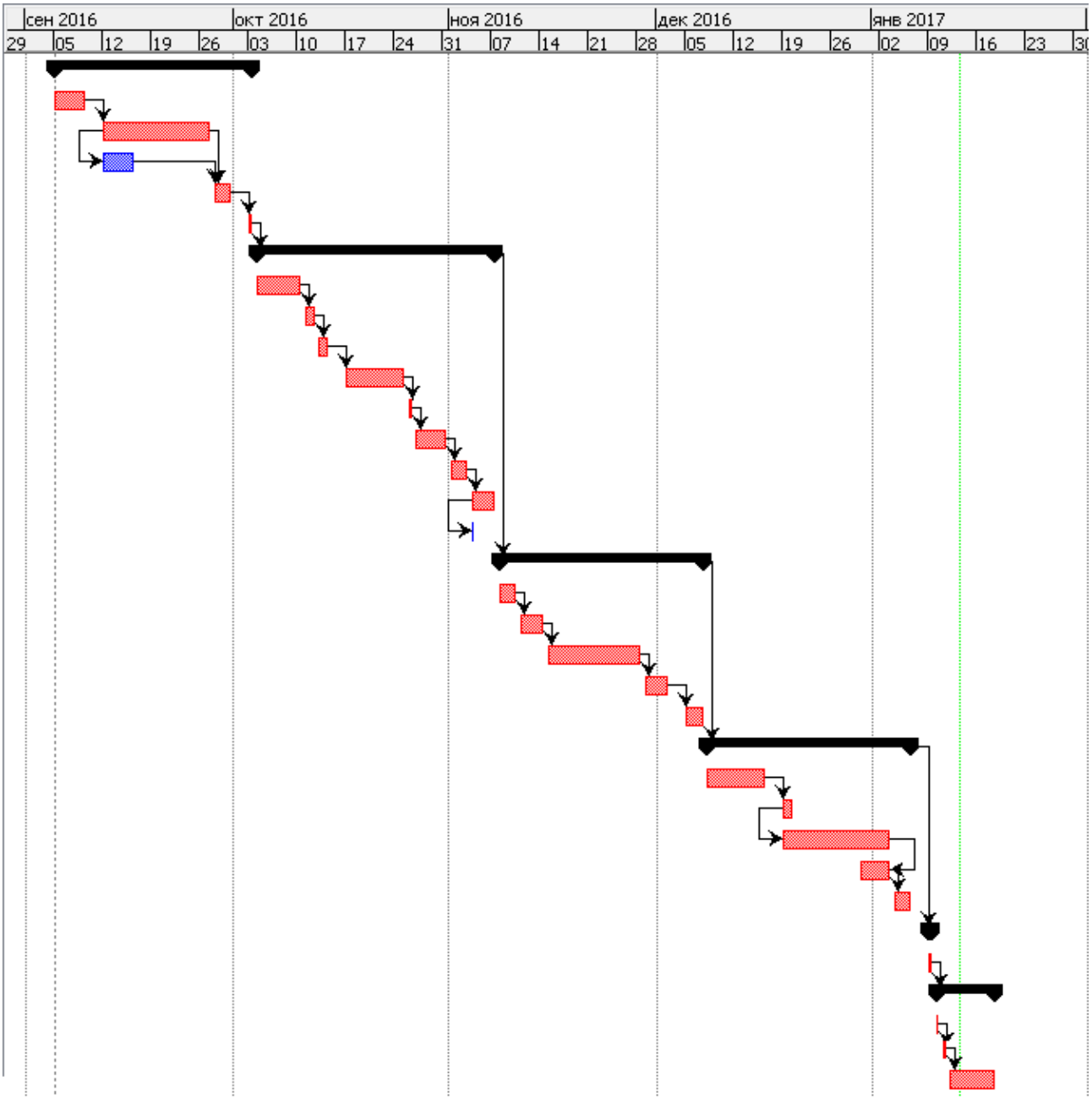


Рисунок 4.2 Діаграма Ганта

Далі необхідно провести оцінку тривалості робіт по методу PERT. Для кожного Пункту роботи необхідно визначити три оцінки її діяльності:

- Оптимістична
- Пессемістична
- Найбільш ймовірна

Критичный шлях та час виконання (песемістичне – реалістичне - оптимістичне)

1) Етап підготовки 30 – 21 -18 днів.
2) Робота над першим розділом 30 -22 -20 днів.
3) Робота над другим розділом 26 – 20 – 15 днів.
4) Робота над додатком 30 – 20 -18 днів.
5) Робота над третім розділом 3 –2 – 1 днів.
6) Завершальна робота, підготовка до захисту 10 – 6 – 5.

Таблиця 2 Оцінка тривалості виконання

Пункт дипломної роботи	Оптимістична оцінка, днів (**О**)	Песемістична оцінка, днів (**Р**)	Наібільш ймовірна, днів (**М**)	Час виконання, днів (**Те**)
1	18	25	21	18.6
2	20	27	22	23
3	15	24	20	20.1
4	18	30	20	21.3
5	1	3	2	2
6	5	9	6	6.5
Сума	77	118	91	91.5

4.2 Визначення ймовірності завершення дипломного проекту до певного моменту часу

1.Виначеня дисперсії:

$$\sigma^2 = \left(\frac{P-O}{6}\right)^2 \quad = \mathbf{46.7}$$

Стандатртне відхилення

$$\sigma = \sqrt{\sigma^2} = \sqrt{\sum \left(\frac{P-O}{6}\right)^2}. \quad = \mathbf{6.8}$$

Z	**0.88**
F(z)	**0.78**

Одже, з верогідністю 78% проект буде завершено за 97 днів.

Таблиця 3 Джерела даних про тривалість робіт

Пункт	Джрело даних		
	Песемістична оцінка	Оптимістична оцінка	Найбільш ймовірна оцінка
1 – 6	Досвід, нестача часу	Обмеження по строкам, додатковий вільний час	Досвід
2	Досвід, нестача часу	Обмеження по строкам, додатковий вільний час	Досвід
3	Досвід, нестача часу	Обмеження по строкам, додатковий вільний час	Досвід

Таблиця 3 продовження

4	Досвід, нестача часу	Обмеження по строкам, додатковий вільний час	Досвід
5	Досвід, нестача часу	Обмеження по строкам, додатковий вільний час	Досвід
6	Досвід, нестача часу	Обмеження по строкам, додатковий вільний час	Досвід

4.3 Управління ризиками

Таблиця 4 Ієрархічна структура ризиків

Пункт роботи	Ризик
1	Нестача інформації
2	Відсутність в інтернеті літератури по темі, Вихід зі строю комп'ютеру
3	Вихід зі строю комп'ютеру
4	Нестача часу, Помилки в розробці, Прорахунки в алгоритмі Прорахунки в розробці фронт-енду, не знання обраного фреймворку
5	Зміна правил оформлення дипломної роботи чи розділів, інформація про необхідність додавання нових розділів
6	Проблеми зі своєчасним підписанням роботи

Таблиця 5 Перелік заходів по усуненню ризиків.

Ризик	Заходи по усуненню
Нестача інформації	Пошук інформації заздалегіть.
Вихід зі строю комп’ютеру	Постійне збереження файлів та коміт коду.
Нестача часу	Менше спати, частіше передивлятись план.
Помилки в розробці	Використання сучасних та перевірених методів розробки ПЗ.
Прорахунки в алгоритмі	Приділити якумога більше часу на розробку алгоритмів та їх коректировку.
Прорахунки в розробці фронт-енду	Перерозробка
Не знання обраного фреймворку	Виділення додаткового часу на його вивчення якщо критично. Обрати інший фреймворк.
Зміна правил оформлення дипломної роботи чи розділів, інформація про необхідність додавання нових розділів	Бути готовим, постаратися закінчити роботу якумога раніше, щоб був час на переробку.
Проблеми зі своєчасним підписанням роботи	Оговорювати сроки підписання своєчасно.

Таблиця 6 Матриця впливу ризиків на показники дипломного проекту

Показатель	Номер риска								
	1	2	3	4	5	6	7	8	9
Сроки	сильное	сильное	сильное	слабое	слабе				
Качество	слабое	среднее	сильное	сильное	середнє				

4.5 Висновок

В цьому розділі було сплановано процес підготовки дипломного проекту, для планування було використано діаграму Ганта. Розраховано ймовірність закінчення дипломної роботи в визначений час, - за 97 днів, вона склала 78%. Були визначені основні ризики, іх вплив на проект та методи запобігання цим ризикам.

Висновок по роботі

Дипломна робота присвячена вибору фреймворку та розробці серверного додатку, що дозволяє аналізувати ситуацію з цитованостями науковців НТУУ «КПІ ім. Ігоря Сікорського» у системі Google Scholar.

В першому розділі було розглянуто саму систему Google Scholar, її функціонал та історію створення, вебометричні характеристики та місце серед подібних систем. Scholar займає перше місце серед подібних систем завдяки своїй гнучкості та відкритості. Також були наведені рекомендації для викладачів щодо коректного створення особистого кабінету, ці рекомендації розміщені в інтернеті на сайті - http://phone.kpi.ua/schol/.

Другий розділ присвячений вибору засобів для реалізації додатку. Після аналізу найпопулярніших на сьогодні рішень було прийняте рішення використати фреймворк Spring для спрощення розробки, через ряд причин:

- Наявність необхідних інструментів
- Найпопулярніший в корпоративній розробці
- Велика кількість модулів допоможе при потребі швидко розширити додаток додатковим функціоналом
- Простота реалізації Rest api додатку.

Для відображення зібраної в серверній частині інформації були використані HTML, Javascript та бібліотеки для будування гістограм від Google.

Останньою частиною роботи стала практична реалізація додатку з використанням всіх досліджень зроблених в ході роботи – розробка та корекція алгоритму з урахуванням поставлених цілей, потім написання коду по алгоритму, тестування та налогодження..Третій розділ присвячено опису реалізації додатку, наведено приклади роботи додатку.

Останній четвертий розділ присвячено використаню методів управління термінами, де були застосовані отриманні під час навчання знання та навички побудови діаграми Ганта, визначення критичного шляху, визначення ризиків та рахування вероїдності закінчення роботи вчасно.

Список використаних джерел

1. Teaching Google Scholar: A Practical Guide for Librarians - Paige Alfonzo

2. Офіційний сайт Elsevier Scopus. Режим доступу: https://www.elsevier.com/solutions/scopus/content

3. Онлайн діаграми. Режим доступу: http://www.onlinecharts.ru/

4. Наукометричні показники дя оцінки журналів. Режим доступу: http://library.unecon.ru/publ-aktivnost/scopus-web-science-rinc

5. Офіційний сайт Google Scholar. Режим доступу: https://scholar.google.com.ua/

6. Рейтинг університетів світу за вервіє. GOOGLE SCHOLAR CITATIONS Режим доступу: http://aphd.ua/reitynh-universytetiv-svitu-za-versiieiu-google-scholar-citations/

7. Топ - 1200 Вузов мира (индекс цитирования Google Scholar Citations) – 2016. Режим доступу: http://hotuser.ru/monitoring-vuzov-kakoj-vuz-vybrat/3047-top-1200-vuzov-mira-indeks-czitirovaniya-google-scholar-citations

8. Чернігівський національний технічний університет. Про Індекс Хірша. Режим доступу: http://www.stu.cn.ua/news_view/1012/

9. Академичне письмо. Журнал по підготовці в Scopus. Режим доступу: https://writing-skills-development.blogspot.com/2015/03/zhurnal-po-pedagogike-indeksiryemyj-v-scopus.html

10. Loughborough university. Web of science update. Режим доступу: https://blog.lboro.ac.uk/adlib/it/web-of-science-upgrade-2

11. Java веб фреймворки. Режим доступу: https://tproger.ru/digest/java-web-frameworks/

12. Профіль викладача Копилова В.І. Режим доступу: https://scholar.google.com.ua/citations?user=lNfIodoAAAAJ

13. Clarivate analitycs. About web of science. Режим доступу: http://clarivate.com/?product=web-of-science

Додаток А. Лістинг програми

AcademicsIdParser.java

```
public interface AcademicsIDParser {

  void fetchID();
  Map <String, String> getResults();

}
```

XLParser.java

```
public class XLParser implements AcademicsIDParser {
   private Map<String, String> academicsNameIDMap;
   private String source = "/pages/adcedemics.xls";

   public XLParser() {
      this.academicsNameIDMap = new HashMap<> ();
   }

   @Override
   public void fetchID() {

      InputStream in = null;
      HSSFWorkbook wb = null;
      List<String> blacklist = new ArrayList<>();
      try {
         in = new FileInputStream(this.source);
         wb = new HSSFWorkbook(in);

         Sheet sheet = wb.getSheetAt(0);
         int i = sheet.getPhysicalNumberOfRows()-1;

         for (int j = 1; j<i; j++) {
            Row cells =   sheet.getRow(j);
            Cell namecell = cells.getCell(1);
            Cell idcell = cells.getCell(6);
             academicsNameIDMap.put(namecell.getStringCellValue(),idcell.getStringCellValue());
         }

      } catch (Exception e) {
         try {
            throw new CustomException("error fetch data");
         } catch (CustomException e1) {
            e1.getMessage();
         }
      }

   }

   @Override
   public Map<String, String> getResults() {
      return this.academicsNameIDMap;
   }
}
```

WebParser.java

```
public interface WebParser {

    boolean doParse(String url) throws IOException;

}
```

TelefParser.java

```
@Component
public class TelefParser implements WebParser {
   private static boolean isFirstInit = true;
   private String File;
   List<String> cafedras;
   List<String> faculty;
   private DepartmentsParser departmentsParser;
   private ResultContainerAndProcessor resultContainerAndProcessor;
   private WrongIDManager wrongIDManager;
   private String site = "http://telef.kpi.ua/kpi/cafedra.php";


   public TelefParser() {
      if (isFirstInit) {
         isFirstInit = false;
         return;
      } else
         System.out.println("telef" + this.toString());
      cafedras = new ArrayList<>();
      faculty = new ArrayList<>();
      wrongIDManager = new WrongIDManager();
      System.out.println("teleffirst");

   }


   public void parseAllDepratments() {
      parseCafedrasNames();
      parseFacultiesName();
   }

   private void parseFacultiesName() {
      try {
         Document doc = Jsoup.connect(site).get();
         String caffedrasArray[] = doc.text().split("Факультет");

         faculty = new ArrayList(Arrays.asList(caffedrasArray));
         faculty.remove(0);

         String toChange = cafedras.get(cafedras.size() - 1);

         if (toChange.contains("Copyright")) {

            String[] toSplit = toChange.split("Copyright");

            faculty.add("Факультет" + toSplit[0]);

         }
```

```
        } catch (IOException e) {
            try {
                throw new CustomException("Network problems");
            } catch (CustomException e1) {
                e1.getMessage();
            }

        }
    }


    private void parseCafedrasNames() {
        try {
            Document doc = Jsoup.connect(site).get();
            String caffedrasArray[] = doc.text().split("Кафедра");

            cafedras = new ArrayList(Arrays.asList(caffedrasArray));
            cafedras.remove(0);

            String toChange = cafedras.get(cafedras.size() - 1);

            if (toChange.contains("Copyright")) {

                String[] toSplit = toChange.split("Copyright");

                cafedras.add("Кафедра" + toSplit[0]);

            }
        } catch (IOException e) {
            try {
                throw new CustomException("Network Problems");
            } catch (CustomException e1) {
                e1.printStackTrace();
            }
        }
    }


    public boolean doParse(String site) {

        Set<Department> cafedraes = new TreeSet<>();
        Set<Department> faculties = new TreeSet<>();


        for (String cafedra : cafedras) {
            Department department = new Department(cafedra, (Integer)
resultContainerAndProcessor.getSum(departmentsParser.
                    parseCafedra("Кафедра" + cafedra)), resultContainerAndProcessor.getTempDepartmentBlackSum());
            cafedraes.add(department);

        }

        for (String s : faculty) {
            Department department = new Department(s, resultContainerAndProcessor.getSum(departmentsParser.
                    parseCafedra(s)), resultContainerAndProcessor.getTempDepartmentBlackSum());
            faculties.add(department);
        }

        resultContainerAndProcessor.setForChartInformationCafedras(new ArrayList<>(cafedraes));
        resultContainerAndProcessor.setForChartInformationFaculty(new ArrayList<>(faculties));
        return true;
    }
```

```
    public void setSite(String site) {
        this.site = site;
    }

    public List<String> getCafedras() {
        return cafedras;
    }

    public List<String> getFaculty() {
        return faculty;
    }

    public void setDepartmentsParser(DepartmentsParser departmentsParser) {
        this.departmentsParser = departmentsParser;
    }

    public void setResultContainerAndProcessor(ResultContainerAndProcessor resultContainerAndProcessor) {
        this.resultContainerAndProcessor = resultContainerAndProcessor;
    }
}
```

ScholarParser.java

```
@Component
public class SchoolarParser  implements  WebParser {

    public boolean doParse(String url) throws IOException {
        Document doc = Jsoup.connect(url).get();
        String code = doc.html();
        if (code.contains("=55961170570326719 97"))
        {
            return true;
        }
        return false;
    }

}
```

IDManager.java

```
@Component
public class WrongIDManager  {
    private ResultContainerAndProcessor resultContainerAndProcessor;
    private WebParser webParser;
    private static boolean firstInit = false;
    private AcademicsIDParser academicsIDParser;


    public WrongIDManager() {
        if (firstInit){
        }
        else firstInit = true;
        return;
    }
```

```
    public void fillResultContainer() {
        academicsIDParser = new XLParser();
        academicsIDParser.fetchID();
       Map<String, String>  map = new HashMap<>();
      for (Map.Entry<String, String> stringStringEntry : academicsIDParser.getResults().entrySet()) {
         try {
            if (!webParser.doParse(stringStringEntry.getValue())) {
               resultContainerAndProcessor.addToBlackList(stringStringEntry.getKey());
               System.out.println(stringStringEntry.getKey());
            }
         } catch (IOException e) {
            e.printStackTrace();
         } {

         }
      }

   }

   public void setResultContainerAndProcessor(ResultContainerAndProcessor resultContainerAndProcessor) {
      this.resultContainerAndProcessor = resultContainerAndProcessor;
   }

   public void setWebParser(WebParser webParser) {
      this.webParser = webParser;
   }
}
```

DepartmentParser.java

```
@Component
public class DepartmentsParser {

   private Map<String, String> parsedfaculty;
   private Set<Academic> sortedAcademics;
   private List<Academic> toReturn;
   private ResultContainerAndProcessor resultContainerAndProcessor;

   public DepartmentsParser() {
      parsedfaculty = new HashMap<>();
      sortedAcademics = new TreeSet<>();
      toReturn = new ArrayList<>();

   }

   public List parseCafedra(String department) {
      String url = "";
      if (department.contains("Кафедра")) {
         url = "http://telef.kpi.ua/kpi/cafedra_table.php";
      } else {
         url = "http://telef.kpi.ua/kpi/facultet_table.php";
      }

      toReturn = new ArrayList<>();
      sortedAcademics = new TreeSet<>();
      try {
```

```
        Connection.Response res2 = Jsoup.connect(url)
                .header("Content-Type", "application/x-www-form-urlencoded;charset=UTF-8")
                .data("cafedra_name", department)
                .method(Connection.Method.POST)
                .execute();
        Document document = res2.parse();

        Elements table = document.select("table");
        Iterator<Element> ite = table.select("td").iterator();

        for (int i = 0; i < document.select("td").size(); i += 3) {
           if (ite.hasNext()) {
              String name = ite.next().text();
              String r1 = ite.next().text();
              String r2 = ite.next().text();
              String r3 = ite.next().text();
              Academic academic = new Academic(name, r1, r2, r3);
              if (resultContainerAndProcessor.getBlacklist().contains(academic.getName()))
              {
                 academic.putinblackList();
              }
              sortedAcademics.add(academic);

           }
        }
        toReturn.addAll(new ArrayList<>(sortedAcademics));
     } catch (Exception e) {
        try {
           throw new CustomException("fail parsing cafedra");
        } catch (CustomException e1) {
           e1.getMessage();
        }
     }
     resultContainerAndProcessor.addToTableMap(department, toReturn);
     return toReturn;
  }


  public void setResultContainerAndProcessor(ResultContainerAndProcessor resultContainerAndProcessor) {
     this.resultContainerAndProcessor = resultContainerAndProcessor;
  }
}
```

Academic.java

```
public class Academic implements  Comparable<Academic>{
    private String name;
    private String schoolarRaitting;
    private String citation;
    private String index;
    private boolean isInBlackList;
```

```
    public Academic(String name, String schoolarRaitting, String citation,
String index) {
        this.name = name;
        this.schoolarRaitting = schoolarRaitting;
        this.citation = citation;
        this.index = index;
        isInBlackList = false;
    }

    @Override
    public int compareTo(Academic second) {

        int thisRaiting = Integer.parseInt(this.citation);
        int secondRaiting = Integer.parseInt(second.citation);
        if (secondRaiting == thisRaiting){
            return this.name.compareTo(second.name);
        }

        return secondRaiting - thisRaiting;

    }


    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getSchoolarRaitting() {
        return schoolarRaitting;
    }

    public void setSchoolarRaitting(String schoolarRaitting) {
        this.schoolarRaitting = schoolarRaitting;
    }

    public String getCitation() {
        return citation;
    }

    public void setCitation(String citation) {
        this.citation = citation;
    }

    public String getIndex() {
        return index;
    }

    public void setIndex(String index) {
        this.index = index;
    }
```

```
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        Academic academic = (Academic) o;

        return name.equals(academic.name);

    }

    @Override
    public int hashCode() {
        return name.hashCode();
    }


    public void putinblackList() {
        isInBlackList = true;
    }

    public void removeFromBlackList(){
        isInBlackList = false;
    }

    public boolean getisInBlackList() {
        return isInBlackList;
    }

    public void setInBlackList(boolean inBlackList) {
        isInBlackList = inBlackList;
    }


}
```

Department.java

```
public class Department implements Comparable<Department> {

   private String name;
   private Integer citiation;
   private Integer citiationWithBlackList;


   public Department(String name, Integer citiation, Integer citiationWithBlackList) {
      this.name = name;
      this.citiation = citiation;
      this.citiationWithBlackList = citiationWithBlackList;

   }


   public String getName() {
      return name;
   }

   public void setName(String name) {
      this.name = name;
   }
```

```
    public Integer getCitiation() {
        return citiation;
    }

    public void setCitiation(Integer citiation) {
        this.citiation = citiation;
    }

    public Integer getCitiationWithBlackList() {
        return citiationWithBlackList;
    }

    public void setCitiationWithBlackList(Integer citiationWithBlackList) {
        this.citiationWithBlackList = citiationWithBlackList;
    }

    @Override
    public int compareTo(Department second) {
        if (second.citiation == this.citiation){
            return this.name.compareTo(second.name);
        }
      return   second.citiation - this.citiation;
    }
}
```

ResultContainerAndProcessor.java

```
@Component
public class ResultContainerAndProcessor {

    private List<Department> forChartInformationFaculty;
    private List<Department> forChartInformationCafedras;
    private List<String> blacklist;
    List<Academic> results;
    private List<Integer> sum;
    private Integer tempDepartmentBlackSum;
    private Map<String, List<Academic>> forTaableInformation;
    private List<Academic> allBLl;
    private Integer totalLost = 0;

    public ResultContainerAndProcessor() {
        blacklist = new ArrayList<>();
        forTaableInformation = new HashMap<>();
        allBLl = new ArrayList<>();
    }

    private void Sum() {
        sum = new ArrayList<>();
        Integer sumWithoutBlacklist = 0;
        Integer sumwithBlackList = 0;
        for (Academic academic : results) {
            Integer currentPrepodCitiation = Integer.parseInt(academic.getCitation());
            sumWithoutBlacklist += currentPrepodCitiation;
            if (academic.getisInBlackList()) {
                sumwithBlackList += currentPrepodCitiation;
            }
        }
```

```
    sum.add(sumWithoutBlacklist);
    sum.add(sumWithoutBlacklist - sumwithBlackList);
}

public void setResults(List<Academic> results) {
    this.results = new ArrayList<>(results);
    Sum();
}


public Integer getSum(List<Academic> academicList) {
    Integer result = 0;
    tempDepartmentBlackSum = 0;
    for (Academic academic : academicList) {
        result += Integer.parseInt(academic.getCitation());
        if (!academic.getisInBlackList()) {
            tempDepartmentBlackSum += Integer.parseInt(academic.getCitation());
        }
    }
    return result;
}

public void addToTableMap(String department, List academics) {
    forTaableInformation.put(department, academics);
}

public void getTableMap(String department) {
    setResults(forTaableInformation.get(department));
}

public List<Integer> getSum() {
    return sum;
}

public void addToBlackList(String prepodName) {
    blacklist.add(prepodName);
}

public List<Academic> getResults(Integer number) {
    ArrayList toReturn = new ArrayList();
    if (results.size() >= 10) {
        for (int i = 0; i < number; i++) {
            toReturn.add(results.get(i));

        }
    } else return results;

    return toReturn;

}

public List<Department> getForChartInformationFaculty() {
    return forChartInformationFaculty;
}

public void setForChartInformationFaculty(List<Department> forChartInformationFaculty) {
    this.forChartInformationFaculty = forChartInformationFaculty;
}
```

```
    public List<Department> getForChartInformationCafedras() {
        return forChartInformationCafedras;
    }

    public void setForChartInformationCafedras(List<Department> forChartInformationCafedras) {
        this.forChartInformationCafedras = forChartInformationCafedras;
    }

    public List<String> getBlacklist() {
        return blacklist;
    }

    public List<Academic> getAllLostCitiations() {

        if (allBLl.size() == 0) {
            for (Map.Entry<String, List<Academic>> entry : forTaableInformation.entrySet()) {
                List<Academic> tmp = entry.getValue();
                for (Academic academic : tmp) {
                    if (academic.getisInBlackList()) {
                        allBLl.add(academic);
                        totalLost += Integer.parseInt(academic.getCitation());
                    }
                }

            }
        }

        return allBLl;
    }

    public Integer getTempDepartmentBlackSum() {
        return tempDepartmentBlackSum;
    }

    public Integer getTotalLost() {
        return totalLost;
    }
}
```

CustomException.java

```
public class CustomException extends Exception {

    public CustomException(String message) {
        super(message);
    }
}
```

Wrapper.java

```
public class Wrapper {
    private String department;
    private String numberOfScientistsToDisplay;

    public String getDepartment() {
        return department;
    }

    public void setDepartment(String department) {
        this.department = department;
    }
```

```
    public String getNumberOfScientistsToDisplay() {
        return numberOfScientistsToDisplay;
    }

    public void setNumberOfScientistsToDisplay(String numberOfScientistsToDisplay) {
        this.numberOfScientistsToDisplay = numberOfScientistsToDisplay;
    }
}
```

IService.java

```
public interface IService {

    List getDepartmentSum();

    List getPeople();

    void process(String departmentName);

    List getResultsForFacultiesChart();

    List getResultsForCafedrasChart();

    void parseAll();

    void setNumberOfScientistsToDisplay(String number);

    Integer getTotalLost();

    List getAllIncorectProfilesList();
}
```

PostconstructWorkService.java

```
@Service
public class PostConstructWorkService implements IService {

    private String site = "http://telef.kpi.ua/kpi/cafedra.php";
    private static boolean firstinit = true;
    private Integer numberToDisplay = 10;

    public PostConstructWorkService() {
        System.out.println("service" + this.toString());
    }

    @Autowired
    ResultContainerAndProcessor resultContainerAndProcessor;


    @Autowired
    DepartmentsParser departmentsParser;

    @Autowired
    WrongIDManager wrongIDManager;

    @Autowired
    SchoolarParser schoolarParser;

    @Autowired
    TelefParser telefParser;
```

```
@Override
public List getDepartmentSum() {
    return resultContainerAndProcessor.getSum();
}

@Override
public List getPeople() {
    return resultContainerAndProcessor.getResults(numberToDisplay);
}

@Override
public void process(String departmentName) {
    resultContainerAndProcessor.getTableMap(departmentName);
}

@Override
public List getResultsForFacultiesChart() {
    return resultContainerAndProcessor.getForChartInformationFaculty();
}

@Override
public List getResultsForCafedrasChart() {
    return resultContainerAndProcessor.getForChartInformationCafedras();
}

@Override
@PostConstruct
@Scheduled(cron = "10 0 1 * *")
public void parseAll() {
    if (firstinit) {
        firstinit = false;
        return;
    } else {

        wrongIDManager.setResultContainerAndProcessor(resultContainerAndProcessor);
        wrongIDManager.setWebParser(schoolarParser);
        wrongIDManager.fillResultContainer();

        departmentsParser.setResultContainerAndProcessor(resultContainerAndProcessor);
        telefParser.setDepartmentsParser(departmentsParser);
        telefParser.setResultContainerAndProcessor(resultContainerAndProcessor);
        telefParser.parseAllDepratments();
        telefParser.doParse("php");
    }

}

@Override
public void setNumberOfScientistsToDisplay(String number) {
    this.numberToDisplay = Integer.parseInt(number);
}

@Override
public Integer getTotalLost() {
    return resultContainerAndProcessor.getTotalLost();
}


@Override
public List getAllIncorectProfilesList() {
    return resultContainerAndProcessor.getAllLostCitiations();
}
```

MainController.java

```
@Controller
public class CollectorsController {

    @Autowired
    IService service;

    @RequestMapping(path = "/index", method = RequestMethod.GET)
    @ResponseBody
    public ModelAndView test2() {
        ModelAndView modelAndView = new ModelAndView();
        modelAndView.setViewName("cafedras");
        return modelAndView;
    }


    @RequestMapping(path = "/post", method = RequestMethod.POST)
    @ResponseBody
    public List test(@RequestBody Wrapper wrapper) {
        service.process(wrapper.getDepartment());
        return new ArrayList();
    }


    @RequestMapping(path = "/faculty", method = RequestMethod.GET)
    @ResponseBody
    public ModelAndView faculty() {
        ModelAndView modelAndView = new ModelAndView();
        modelAndView.setViewName("faculty");
        return modelAndView;
    }

    @RequestMapping(path = "/result", method = RequestMethod.GET)
    public ModelAndView tes3() {
        ModelAndView modelAndView = new ModelAndView();
        modelAndView.addObject("results", service.getPeople());
        modelAndView.addObject("sum", service.getDepartmentSum());
        modelAndView.setViewName("resulttable");
        return modelAndView;
    }


    @RequestMapping(path = "/chart_cafedras", method = RequestMethod.GET)

    public ModelAndView chart() {
        ModelAndView modelAndView = new ModelAndView();
        modelAndView.setViewName("chartcafedr");
        return modelAndView;
    }

    @RequestMapping(path = "/chart_faculty", method = RequestMethod.GET)

    public ModelAndView chartfaculty() {
        ModelAndView modelAndView = new ModelAndView();
        modelAndView.setViewName("chartfaculty");
        return modelAndView;
    }
```

```
    @RequestMapping(path = "/chart_faculty_json", method = RequestMethod.GET)
    @ResponseBody
    public List facultychart() {
        return service.getResultsForFacultiesChart();
    }


    @RequestMapping(path = "/chart_cafedras_json", method = RequestMethod.GET)
    @ResponseBody
    public List cafedraschart() {
        return service.getResultsForCafedrasChart();
    }

    @RequestMapping(path = "/set_number", method = RequestMethod.POST)
    @ResponseBody
    public void setNumber(@RequestBody Wrapper wrapper) {
        service.setNumberOfScientistsToDisplay(wrapper.getNumberOfScientistsToDisplay());
    }

    @RequestMapping(path = "/showuser", method = RequestMethod.GET)
    public ModelAndView showusers() {
        ModelAndView modelAndView = new ModelAndView();
        modelAndView.addObject("results", service.getAllIncorectProfilesList());
        System.out.println(service.getTotalLost());
        modelAndView.addObject("sum", service.getTotalLost());
        modelAndView.setViewName("users");
        return modelAndView;
    }

}
```