Національний технічний університет України

Київський політехнічний інститут ім. Ігоря Сікорського

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

ЗВІТ

ПРО ПРОХОДЖЕННЯ ПЕРЕДДИПЛОМНОЇ ПРАКТИКИ

Студента 4 курсу групи ІТ-92

Спеціальності 121 «Інженерія програмного забезпечення»

Татарина Миколи Сергійовича

(прізвище, ім'я, по батькові студента)

Термін практики з «17» квітня 2023 р. по «21» травня 2023 р.

База практики: ПП Тильда

(назва підприємства)

Керівник від підприємства

директор

(науковий ступінь, вчене звання, посада)

Ксенія Бондар

(прізвище, ім'я, по батькові, підпис)

Керівник практики від кафедри ІПІ

(науковий ступінь, вчене звання, посада,)

(прізвище, ім'я, по батькові, підпис)

Київ-2023р.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

На сьогоднішний день у світі існує величезна кількість текстової інформації (книг, статей, блогів, і т.д.). Через це виникає проблема пошуку потрібного потрібної інформації. Те, наскільки легко чи важко можна знайти той чи інший шматок інформації, залежить від декількох факторів. Один із таких факторів - це метадані, як от: заголовок, опис, ключові слова. Останні особливо корисні, бо дозволяють користувачу пошукової системи суттєво звузити пошук.

В ідеалі, ключові слова повинні надавати автори творів, але іноді цього не трапляється. Або ж автор це зробив, але метадані були якимось чином втрачені.

У будь-якому випадку, для того щоб отримати ключові слова для певного тексту, цей текст повинен хтось прочитати. Існують платформи (наприклад, goodreads), що дозволяють читачам колективно редагувати теги, проте було б непогано, якби це можна було робити автоматично, за допомогою застосунку-класифікатора.

1.2 Змістовний опис і аналіз предметної області

Основною задачою цієї роботи є розробка веб-застосунку, який дозволяє користувачу класифікувати за жанрами художні твори, у вигляді простого тексту, або у вигляді файлу, використовуючи для цього або веб-інтерфейс, або API.

1.3 Аналіз існуючих технологій та успішних IT-проєктів

Проаналізуємо відоме на сьогодні алгоритмічне забезпечення у даній області та технічні рішення, що допоможуть у реалізації застосунку для категоризації художніх творів. Далі будуть розглянуті допоміжні програмні засоби, засоби розробки та готові програмні рішення.

1.3.1 Аналіз відомих алгоритмічних та технічних рішень

1.3.2 Аналіз допоміжних програмних засобів та засобів розробки

Для розробки застосунку було обрано Python – високорівневу інтерпретовану мову програмування загального використання. Такий вибір зумовлений наступними причинами:

1. Як вже було сказано, Python – високорівнева інтерпретована мова, що пришвидшує розробку.
2. Для цієї мови існує декілька великих фреймворків для машинного навчання.
3. Для цієї мови існує декілька веб фреймворків.
4. Python має досить багато сторонніх бібліотек, наприклад бібліотеки для скрейпінгу веб сторінок, або бібліотеки для обробки і конвертації різних файлових форматів.

Розглянемо основні фреймворки для машинного навчання, що доступні для обраної мови:

1. Tensorflow – розроблений компанією Google, надає високорівневі програмні інтерфейси для побудови моделей (Keras) і підтримки розподіленого навчання на декількох пристроях.
2. PyTorch – розроблений компанією Facebook, цей фреймворк аналогічний Tensorflow, але має більш інтуїтивний інтерфейс.
3. MXNet – ще одна альтернатива, розроблена Apache Software Foundation. MXNet підтримує купу інших мов програмування, а також надає інстументи для комп’ютерного зору, обробки природних мов та часових рядів.

Для навчання класифікатора буде використовуватись Tensorflow, оскільки автор вже мав досвід його використання.

Розглянемо кандидати на роль веб фреймворка, що буде використаний для розробки веб застосунка:

1. Django – веб фреймворк для перфекціоністів з дедлайнами. Це високорівневий фреймворк, заснований на архітектурі model-view-template (MVT). Відзначається своєю швидкістю, безпекою та можливостями до масштабування.
2. Flask – легкий і гнучкий WSGI фреймворк, спроєктований для швидкої розробки застосунків з можливістю подальшого їх розширення у більш копмлексні рішення.
3. Bottle – швидкий, простий, легкий мікро-фреймворк. Розповсюджується як однофайловий модуль що залежить тільки від стандартної бібліотеки мови Python.

У якості веб фреймворка автор обрав Django, оскільки вже працював з ним до цього.

Для збору датасету буде використано модулі gutenbergpy і beautifulsoup4. Перший надає API для зручного завантаження текстів з бібліотеки Project Gutenberg, що містить книги з публічного домену. Другий дозволяє парсити веб сторінки, що буде використано для збору тегів.

Для конвертації файлів у звичайний текст буде використано бібліотеку Pandoc. Вона підтримує великий набір файлових форматів і має інтерфейс для обраної мови (сама бібліотека написана на мові Haskell).

У якості середовища розробки веб сервісу буде використано Emacs, для розробки класифікатору – Jupyter Notebook.

1.3.3 Аналіз відомих програмних продуктів

Напевно, слід виділити два типи програмних продуктів або сервісів, що могли б бути використані для вирішення нашої задачі:

1. Онлайн бібліотеки і каталоги, що індексують літературу і збирають метадані, у тому числі теги. Існує вірогідність, що потрібний твір уже є у подібному каталозі, і тоді задача зводиться до того, щоб витягнути

звідти уже існуючі метадані. Звичайно, що такий трюк не спрацює для довільного тексту, якого немає в каталозі.

2. Онлайн класифікатори і API для навчання моделей.

Розглянемо декілька готових програмних продуктів першого типу:

1. Goodreads – популярна платформа для соціального каталогування. Має досить велику базу книг, їх метаданих і коментарів. До 2020 року мала API, але його закрили.
2. LibraryThing – схожа на Goodreads платформа. На відміну від останньої, все ще має API, але вибірка книг і якість тегів не найкраща.

І декілька програмних продуктів другого типу:

1. MonkeyLearn – сервіс текстової аналітики, що орієнтується на бізнес. Надає досить багато інструментів, у тому числі класифікатори для топік лейбелінгу, проте цей класифікатор пропонує трохи інші теги, ніж ті, що нам потрібні. Окрім цього, дозволяє вчити свої моделі.
2. uClassify – безкоштовний веб-сервіс, що дозволяє створювати і використовувати класифікатори тексту. Має доволі великий набір готових класифікаторів.

Занесемо усі переваги і недоліки кожного продукту до таблиці, щоб порівняти з нашим проєктом (таблиця 1.3). Як можна бачити, наш проєкт надає API для роботи з тегами і підходящий класифікатор, чого не роблять інші продукти.

Таблиця 1.1 – Порівняння функціоналу дипломної роботи і аналогів

Програмний продукт	Наявність API	Якість тегів	Наявність тегів для довільного тексту	Можливість навчання своїх моделей	Наявність підходящого класифікатора

Дипломна робота	✓		✓	✗	✓
Goodreads	✗	✓	✗	n/a	n/a
LibraryThing	✓	✗	✗	n/a	n/a
MonkeyLearn	✓	✗	✓	✓	✗
uClassify	✓	✗	✓	✓	✗

1.4 Аналіз вимог до програмного забезпечення

Головною функцією програмного забезпечення є класифікація тексту користувачами (реєстрація не потрібна) у ручному режимі, і за допомогою АПІ (рисунок 1.3).

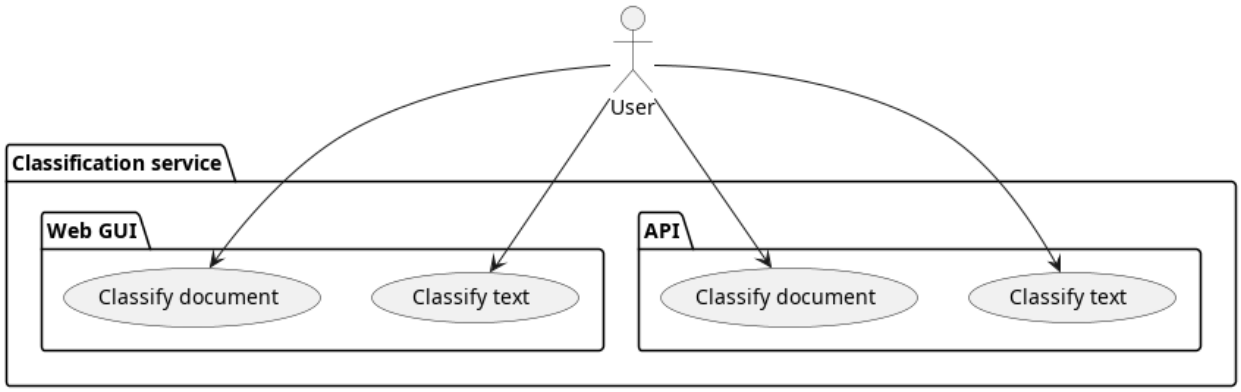

Рисунок 1.1 – Діаграма варіантів використання

В таблицях 1.2 - 1.5 наведені варіанти використання програмного забезпечення.

Таблиця 1.2 – Варіант використання UC-01

Use case name	[GUI] Класифікація довільного тексту
Use case ID	UC-01
Goals	Класифікація довільного тексту
Actors	Гість (незарєстрований користувач)
Trigger	Користувач хоче категоризувати довільний текст

Pre-conditions	-
Flow of Events	Користувач заходить на головну сторінку, обирає пункт «Категоризація довільного тексту» і переходить на відповідну сторінку. Там він вводить свій текст у поле для вводу і натискає кнопку «Категоризувати».
Extension	-
Post-Condition	Користувач бачить список тегів у інтерфейсі.

Таблиця 1.3 – Варіант використання UC-02

Use case name	[GUI] Класифікація файлу
Use case ID	UC-02
Goals	Класифікація файлу
Actors	Гість (незарєстрований користувач)
Trigger	Користувач хоче категоризувати документ
Pre-conditions	Файл повинен бути формату pdf (з текстовим шаром).
Flow of Events	Користувач заходить на головну сторінку, обирає пункт «Категоризація документу» і переходить на відповідну сторінку. Там він завантажує файл і натискає кнопку «Категоризувати».
Extension	У випадку, якщо завантажено невалідний файл, користувач бачить помилку.
Post-Condition	Користувач бачить список тегів у інтерфейсі.

Таблиця 1.4 – Варіант використання UC-03

Use case name	[API] Класифікація довільного тексту
Use case ID	UC-03
Goals	Класифікація довільного тексту за допомогою АПІ
Actors	Гість (незараєстрований користувач)
Trigger	Користувач хоче класифікувати довільний текст

Pre-conditions	-
Flow of Events	Користувач надсилає запит з текстом, що потрібно категоризувати, у тілі запиту.
Extension	-
Post-Condition	Користувач отримує відповідь, що містить список тегів.

Таблиця 1.5 – Варіант використання UC-04

Use case name	[API] Класифікація файлу
Use case ID	UC-04
Goals	Класифікація файлу за допомогою АПІ
Actors	Гість (незараєстрований користувач)
Trigger	Користувач хоче зареєструвати текст
Pre-conditions	Файл повинен бути формату pdf (з текстовим шаром).
Flow of Events	Користувач надсилає запит з файлом, що потрібно категоризувати, у тілі запиту.
Extension	У випадку, якщо надіслано невалідний файл, користувач отримує помилку.
Post-Condition	Користувач отримує відповідь, що містить список тегів.

1.4.1 Розроблення функціональних вимог

На рисунку 1.2 наведено загальну модель вимог, а в таблицях 1.18 – 1.27 наведений опис функціональних вимог до програмного забезпечення. Матриця трасування вимог у таблиці 1.11.

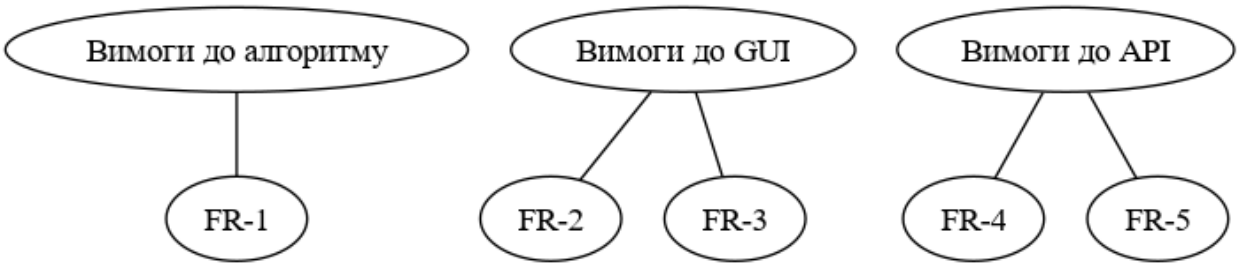

Рисунок 1.2 – Модель вимог у загальному вигляді

Таблиця 1.6 – Функціональна вимога FR-1

Назва	Коректна категоризація звичайного тексту
Опис	Система повинна коректно категоризувати текст за жанрами з достатньою точністю.

Таблиця 1.7 – Функціональна вимога FR-2

Назва	Категоризація довільного тексту за допомогою графічного користувацького інтерфейсу.
Опис	Система повинна надавати можливість категоризувати текст за допомогою GUI.

Таблиця 1.8 – Функціональна вимога FR-3

Назва	Категоризація файлів за допомогою графічного користувацького інтерфейсу.
Опис	Система повинна надавати можливість категоризувати файли у форматі pdf за допомогою GUI.

Таблиця 1.9 – Функціональна вимога FR-4

Назва	Категоризація довільного тексту за допомогою API.
Опис	Система повинна надавати можливість категоризувати довільний текст за допомогою API.

Таблиця 1.10 – Функціональна вимога FR-5

Назва	Категоризація файлів за допомогою API.
Опис	Система повинна надавати можливість категоризувати файли у форматі pdf за допомогою API.

Таблиця 1.11 – Матриця трасування вимог

	FR-1	FR-2	FR-3	FR-4	FR-5
UC-1	x	x			
UC-2	x		x		
UC-3	x			x	
UC-4	x				x

1.4.2 Розроблення нефункціональних вимог

Нефункціональні вимоги наступні:

1. Сервіс повинен бути легким у розгортанні.
2. Веб-інтерфейс повинен працювати з браузерами Chrome та Firefox
3. Сервіс повинен розгортатись на Linux-сумісних системах.

1.5 Постановка задачі

Загалом, було виділено наступні цілі, що повинні бути досягнуті при виконанні даної роботи:

1. Підібрати існуючий датасет категоризованих за тегами текстів, або ж створити свій, використовуючи літературу з публічного домену.

2. Навчити на цьому датасеті обрану модель.

3. Створити веб сервіс для автоматичної категоризації художніх творів.

Висновки до розділу

У даному розділі ми сформулювали загальні положення, описали і проаналізували предметну область застосунку для категоризації тексту. Було розглянуто існуючі алгоритми та моделі для категоризації тексту, і визначено,

що розробляти своє рішення не потрібно. Також було складено список допоміжних програмних засобів та засобів розробки, що будуть використані у процесі виконання роботи. Після цього було проведено аналіз вимог до програмного забезпечення, що розробляється, описано варіанти використання, функціональні і нефункціональні вимоги. У кінці було сформульовано задачі, що мають бути виконані у процесі написання даної роботи.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Усього сервіс передбачає 2 процеси:

1. Категоризація тексту (рис. 2.1).
2. Категоризація файлу (рис. 2.2).

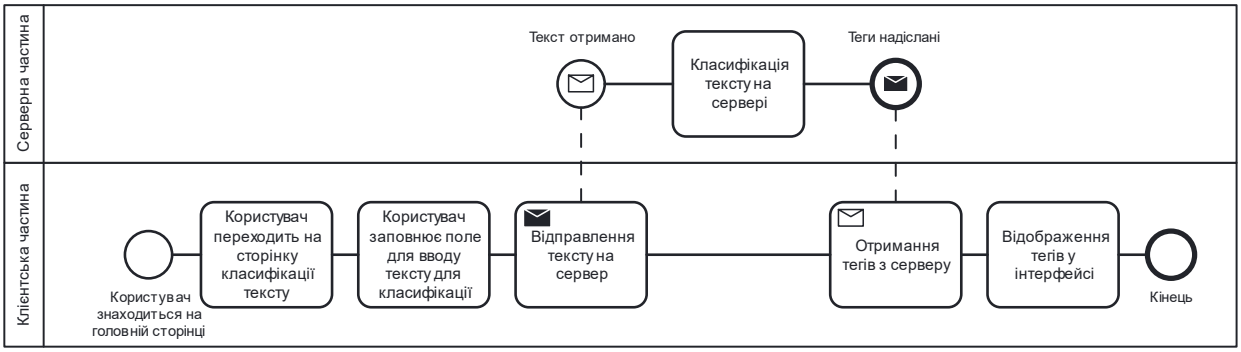

Рисунок 2.1 – BPMN діаграма процесу «Класифікація тексту»

Опис процесу категоризації тексту:

1. Користувач переходить з головної сторінки на сторінку класифікації тексту.
2. Користувач вводить текст у поле для вводу.
3. Користувач натискає кнопку «Класифікувати».
4. Введений текст відправляється на сервер.
5. Сервер класифікує текст і відправляє теги клієнту.
6. Клієнт відображає результати категоризації.

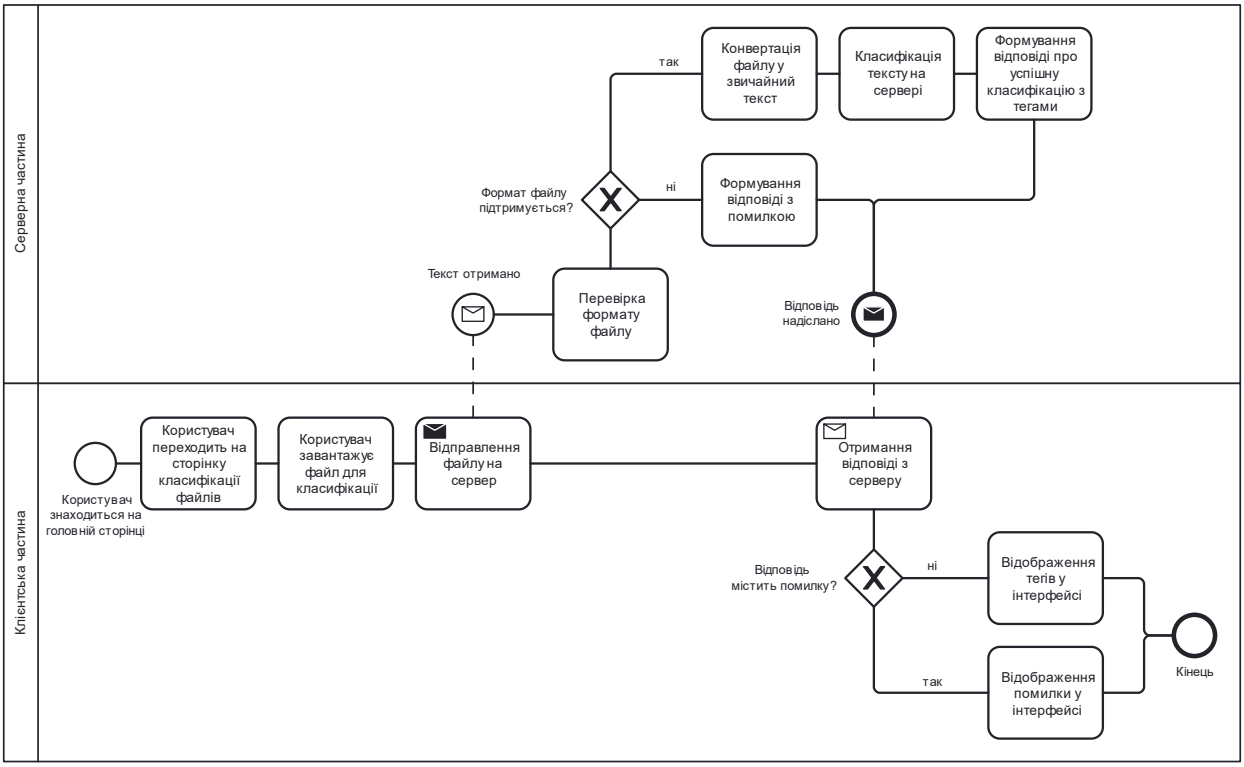

Рисунок 2.2 – BPMN діаграма процесу «Класифікація файлу»

Опис процесу категоризації файлу:

1. Користувач переходить з головної сторінки на сторінку класифікації файлів.
2. Користувач завантажує файл.
3. Користувач натискає кнопку «Класифікувати».
4. Сервер конвертує файл у звичайний текст, класифікує його і надсилає результати класифікації клієнту (або помилку, якщо файл у форматі, що не підтримується).
5. Клієнт відображає або результати класифікації, або помилку, в залежності від того, що надіслав сервер.

2.2 Архітектура програмного забезпечення

Розглянемо архітектуру веб-сервісу, що буде розроблено для взаємодії з класифікатором.

Веб сервіс буде розроблено з використанням монолітної архітектуру за допомогою фреймворку Django, який в свою чергу дотримується шаблону MVT (model-view-template), тому для планування взаємодії між компонентами системи буде доречно використовувати саме цей шаблон.

Model-view-template – це шаблон, дуже схожий на MVC (model view controller) але з деякими відмінностями. На відміну від MVC, частина функцій представлення зміщена у шаблон – це те, що побачить користувач. Представлення у свою чергу, вміщає в себе частину функцій контролера, і відповідає за рендеринг шаблонів і взаємодію з моделлю. Модель залишається як є.

Розглянемо схему взаємодії компенентів системи (рис. 2.3):

1. IndexView – представлення головної сторінки застосунку. Воно не взаємодії з моделлю, тому для його роботи потрібен тільки відповідний шаблон.
2. TextClassifierView, FileClassifierView – представлення сторінок класифікації тексту і класифікації файлів. Окрім відповідних шаблонів, для роботи їм також потрібен класифікатор, який відноситься до моделі. Представлення FileClassifierView також потребує компонент DocumentConverter для конвертації файлів.
3. TextClassifierAPIView, FileClassifierAPIView – представлення ендпоінтів API. Для роботи їм не потрібен шаблон, вони взаємодіють тільки з моделлю.

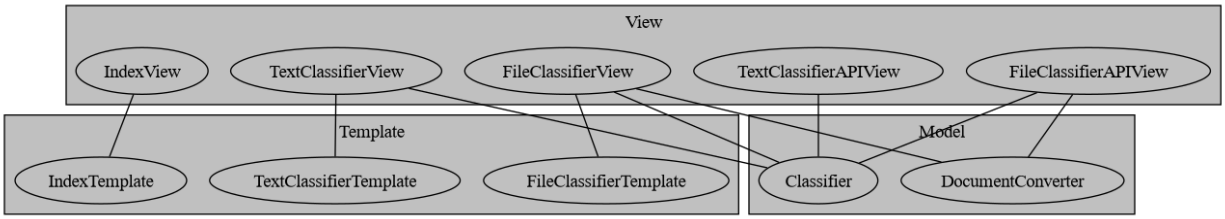

Рисунок 2.3 – Схема взаємодії компонентів сервісу

2.3 Конструювання програмного забезпечення

<Описати реалізацію класифікатора>

Для реалізації компонентів, що було описано у попередньому розділі, було використано абстракції, що надає Django і Django REST Framework. Останній надає більш зручний функціонал для роботи з REST API.

Розглянемо класи, з яких складається система (рис. 2.4):

1. TextClassifierView і FileClassifierView – Представлення графічного інтерфейсу для роботи з класифікатором. Реалізують наступні методи:
    a. get() – метод, що викликається, коли користувач переходить на сторінку з головної.
    b. post() – метод, що викликається, коли користувач надсилає дані для класифікації.
2. TextClassifierAPIView і FileClassifierAPIView – Представлення API для роботи з класифікатором. Реалізують наступні методи:
    a. post() – метод, що викликається, коли користувач робить запит POST з даними класифікації.
3. View, APIView – Базові класи для класів з перших двох пунктів, що надає Django і Django REST Framework.
4. DocumentConverter – клас, що відповідає за конвертацію документів з підтримуваних форматів у звичайний текст. Реалізує наступні публічні методи:
    a. convert() – конвертує документ у звичайний текст.
5. Classifier – клас, що відповідає за взаємодію з моделлю для класифікації. Реалізує наступні публічні методи:
    a. load() – ініціалізує модель для класифікації і завантажує її ваги з диску.
    b. classify() – класифікує текст і повертає список тегів.

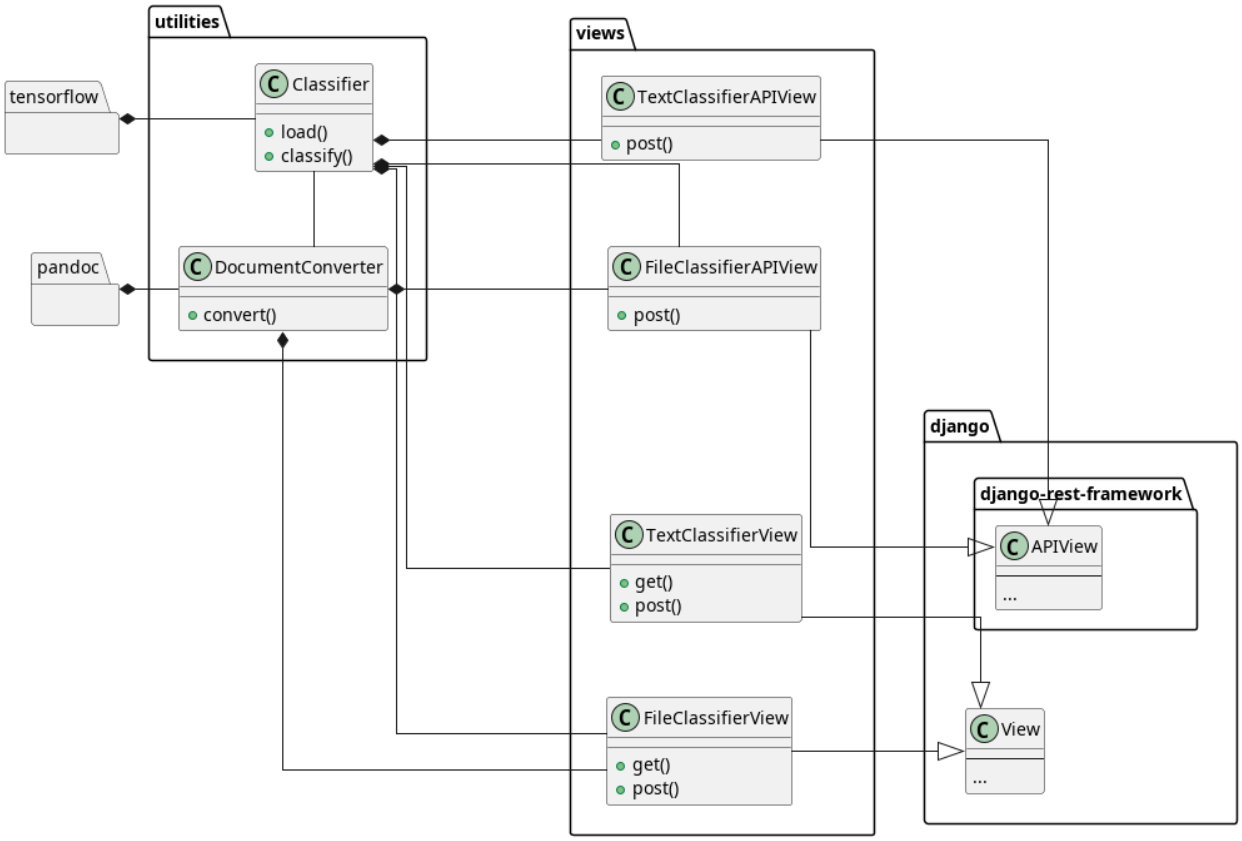

Рисунок 2.4 – Діаграма класів сервісу

Більш детальний опис утиліт, бібліотек та іншого стороннього програмного забезпечення, що було використано під час розробки наведено в таблиці 2.1.

Таблиця 2.1 – Опис утиліт

№ п/п	Назва утиліти	Опис застосування
1	Emacs	Головне середовище розробки програмного забезпечення серверної частини курсової роботи.
2	Jupyter Notebook (і Google Collab)	Головне середовище розробки моделі на основі машинного навчання, що потрібна для роботи класифікатора.

2.4 Аналіз безпеки даних

Персональні дані користувачів знаходяться у безпеці, оскільки сервіс їх не використовує.

Висновки до розділу

У даному розділі було змодельовано основні процеси, які реалізує програмне забезпечення, побудовано його архітектуру та виділено його компоненти. Після цього було описано деталі імплементації ПЗ, і проаналізовано загрози безпеці даних.