Київський національний торговельно-економічний університет
Кафедра інженерії програмного забезпечення та кібербезпеки

ВИПУСКНИЙ КВАЛІФІКАЦІЙНИЙ ПРОЄКТ

на тему:

«Проектування архітектури інформаційно-аналітичного програмного продукту на основі машинного навчання»

Студента 2м курсу, 2 групи, спеціальності 121 «Інженерія програмного забезпечення» спеціалізації «Інженерія програмного забезпечення» _____ підпис студента — Свидригєлло Максон Олегович

Науковий керівник pHd, доцент кафедри інженерії програмного забезпечення та кібербезпеки _____ підпис керівника — Десятко Альона Миколаївна

Гарант освітньої програми доктор економічних наук, професор кафедри інженерії програмного забезпечення та кібербезпеки _____ підпис гаранта — Токар Володимир Володимирович

КИЇВ - 2021

Київський національний торговельно-економічний університет

Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення та кібербезпеки
Освітній ступінь магістр
Спеціальність 121 «Інженерія програмного забезпечення»
Спеціалізація «Інженерія програмного забезпечення»

Затверджую
Зав. кафедри інженерії програмного забезпечення та кібербезпеки
Криворучко О. В.
"29" грудня 2020 р.

Завдання
на випускний кваліфікаційний проєкт студентки
Свидригєлла Максона Олеговича
(прізвище, ім'я, по батькові)

Тема випускного кваліфікаційного проєкту «Проектування архітектури інформаційно-аналітичного програмного продукту на основі машинного навчання»

Затверджена наказом ректора від "28" грудня 2020 р. № 3923

2. Строк здачі студентом закінченої проєкту 25 листопада 2021р.

3. Цільова установка та вихідні дані до проєкту

Мета проєкту проектування архітектури інформаційно-аналітичної системи особистісних вподобань користувачів.

Об'єкт дослідження рекомендаційна система, котра визначає рівень сумісності між користувачами.

Предмет дослідження алгоритми рекомендаційних систем на базі машинного навчвання.

4. Консультанти проєкту із зазначенням розділів, які консультують:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв

5. Зміст випускного кваліфікаційного проєкту (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1. АНАЛІЗ ТА ХАРАКЕРИСТИКА РЕКОМЕНДАЦІЙНИХ СИСТЕМ

1.1. Дослідження призначення рекомендаційних систем

1.2. Загальні відомості про машинне навчання

1.3. Розгляд призначення нейронних мереж

1.4. Нейронні мережі нового покоління

1.5. Огляд кращих рекомендаційних систем сучасного ринку

1.6. Бізнес вимоги до рекомендаційних систем

1.7. Висновок до розділу 1

РОЗДІЛ 2. РОЗГЛЯД ОСНОВНИХ ЕТАПІВ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ТЕХНІЧНЕ ЗАВДАННЯ НА РОЗРОБКУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1. Розподіл ролей у розробці програмного забезпечення

2.2. Життєвий цикл розробки ПЗ

2.3. Технічне завдання на розробку інформаційно-аналітичної системи особистісних вподобань користувачів

2.4. Висновок до розділу 2

РОЗДІЛ 3. РОЗРОБКА АРХІТЕКТУРИ ІНФОРМАЦІЙНО-АНАЛІТИЧНОЇ СИСТЕМИ

3.1. Перший рівень архітектурної деталізації: опис загального уявлення про систему

3.2. Другий рівень архітектурної деталізації: декомпозиція рекомендаційної системи на підсистеми

3.3 Третій рівень архітектурної деталізації: розділення підсистем на класи

3.4. Четвертий рівень архітектурної деталізації: розподіл класів на методи за функціональними особливостями

3.5. Висновок до розділу 3

6. Календарний план виконання проєкту

№ пор.	Назва етапів випускного кваліфікаційного проєкту	Строк виконання етапів проєкту	
		за планом	фактично
1	2	3	4
1.	*Вибір теми випускного кваліфікаційного проєкту*	*21.09.2020*	*21.09.2020*
2.	*Розробка та затвердження завдання на проєкт магістра*	*22.12.2020*	*22.12.2020*
3.	*Вступ та перелік літературних джерел*	*27.02.2021*	*27.02.2021*
4.	*Розробка технічного завдання*	*20.03.2021*	*20.03.2021*
5.	*Розділ 1. Аналіз та харакеристика рекомендаційних систем*	*16.04.2021*	*16.04.2021*
6.	*Розділ 2. Розгляд основних етапів розробки програмного забезпечення та технічне завдання на розробку програмного забезпечення*	*24.05.2021*	*24.05.2021*
7.	*Розділ 3. Розробка архітектури інформаційно-аналітичної системи*	*21.06.2021*	*21.06.2021*
8.	*Розділ 4. Формування бази даних та розробка алгоритмів рекомендаційної системи*	*20.09.2021*	*20.09.2021*
9.	*Розробка програми та методики тестування*	*18.10.2021*	*18.10.2021*
10.	*Написання наукової статті*	*22.05.2021*	*22.05.2021*
11.	*Керівництво користувача*	*21.10.2021*	*21.10.2021*
12.	*Висновки та пропозиції*	*01.11.2021*	*01.11.2021*
13.	*Здача випускного кваліфікаційного проєкту на кафедру (перша перевірка)*	*03.11.2021*	*03.11.2021*
14.	*Підготовка автореферату та презентації доповіді*	*03.11.2021*	*03.11.2021*
15.	*Попередній захист випускного кваліфікаційного проєкту*	*22.11.2021 –25.11.2021*	*24.11.2021*
16.	*Здача зброшурованої випускного кваліфікаційного проєкту*	*25.11.2021*	*25.11.2021*
17.	*Зовнішнє рецензування випускного кваліфікаційного проєкту*	*26.11.2021*	*26.11.2021*
18.	*Підготовка до публічного захисту випускного кваліфікаційного проєкту*		

7. Дата видачі завдання «10» листопада 2021 р.

8. Науковий керівник випускного кваліфікаційного проєкту Десятко А.М.

(прізвище, ініціали, підпис)

9. Гарант освітньої програми Токар В.В.

(прізвище, ініціали, підпис)

10. Завдання прийняв до виконання студент Свидригєлло М.О.

(прізвище, ініціали, підпис)

11. Відгук керівника випускного кваліфікаційного проєкту

Науковий керівник випускного кваліфікаційного проєкту

_____

(підпис, дата)

Відмітка про попередній захист _____Десятко А.М._____

(ПІБ, підпис, дата)

12. Висновок про випускний кваліфікаційний проєкт

Випускний кваліфікаційний проєкт студента _____Свидригєлло М.О._____

(прізвище, ініціали)

може бути допущена до захисту екзаменаційній комісії.

Гарант освітньої програми _____Токар В. В._____

(прізвище, ініціали, підпис)

Завідувач кафедри _____Криворучко О. В._____

(підпис, прізвище, ініціали)

«_____»_____20 ___ р.

АНОТАЦІЯ

Відповідно до мети дослідження робота присвячена розробці архітектури інформаційно-аналітичної системи, технологіям обробки датасетів за допомогою машинного навчання.

В результаті дослідження розроблено концептуальні моделі інформаційно-аналітичної системи особистісних вподобань користувачів, моделі розгортання додатків за допомогою сучасних систем контейнеризації.

Під час дослідження спроєктовано та описано архітектуру інформаційно-аналітичної системи особистісних вподобань користувачів за чотирма основними рівнями деталізації.

Автором описані сучасні підходи до розгортання додатків, алгоритми збору і обробки даних, алгоритми генерації рекомендацій.

Ключові слова: машинне навчання, інформаційно-аналітична система, конвенції програмування, контейнеризація, збір даних, датасет, моделі даних.

ABSTRACT

According to the purpose of the study, the work is devoted to the development of information-analytical system, dataset processing technologies based on machine learning.

As a result of the research, conceptual models of information-analytical system of users personal tastes, models of deployment apps using modern containerization systems were developed.

During the research was designed and described the architecture of the information-analytical system of users personal tastes using four main detailization levels.

The author described modern methods to deploy apps, data collection and data processing algorithms, recommendations generation algorithms.

Keywords: machine learning, recommendation-analytical system, programming conventions, containerization, data collection, dataset, data models.

ЗМІСТ

ВСТУП

Актуальність. Сьогодні практично неможливо знайти більш-менш серйозну організацію, котра не використовує машинне навчання. Тема великих даних та штучного інтелекту останні 3 роки є дуже актуальною через плоди, котрі приносяться компаніям. Зараз кожна корпорація має власні розробки штучного інтелекту у своєму розпорядженні, а пропозиції роботи для фахівців у сфері великих даних ростуть з неймовірною швидкістю. Навіть ті компанії, котрі не можуть дозволити собі розробку власних рекомендаційних систем, залучають готові рішення. Навіть кожен типовий користувач смартфону не підозрюючи використовує штучний інтелект, адже ШІ вбудований в більшість інтернет сервісів, додатків і навіть в ядро операційної системи смартфонів. Тенденція на використання штучного інтелекту настільки розрослась, що ШІ вбудовують в ті програми, котрі і без штучного інтелекту добре працюють.

Зараз у світі IT є тенденція на “Великі Дані”, або “Big Data” — це позначення даних величезного розміру зі значним різноманіттям структурованого або неструктурованого типу, що є ефективно оброблені та мають підтримку горизонтального масштабування за допомогою програмних інструментів. Цей напрям має витоки з кінця 2000-х років нашого століття та походить від альтернативних систем управління базами даних [17].

До Big Data можна віднести три основні пункти:

1. Обробляти “ненормовано великі” (якщо порівняти зі стандартними сценаріями обробки) масиви даних [17]

2. Мати змогу опрацювати велике надходження даних на вході (налаштувати потокове надходження інформації) [17]. 3. Мати можливість обробити паралельно інформацію різної степені структурованості [17].

З вищевказаного випливає, що сама концепція Big Data — це складний комплекс з кількох галузей, що являє собою симбіоз багатьох технологій,

практик, та сукупність тенденцій, що вже представляють повсякденність [17].

Визначаючий постулат в Big Data — це дотримання головних принципів роботи з великими даними. Основні — це обсяг, швидкість, різноманітність. Обсяг — це фізичний об'єм, швидкість — це швидкість приросту з необхідністю блискавичного аналізу, різноманітність — це можливість паралельного запуску обробки даних різного типу [17].

Чому великі данні грають велику роль у сучасній економіці дає відповідь поняття "економіка довгого хвоста". Поняття довгого хвоста ввів Кріс Андерсон у своїй статті для журналу Wired у 2004 році, а у 2006 році на основі цієї статті була написана книга.

У статті Кріс Андерсон описав нову бізнес-модель, яка часто зустрічається в інтернеті. Основна ідея бізнес-моделі полягала в тому, що, якщо підприємець має магазин не в інтернеті, у підприємця обмежений простір для зберігання товарів і, що ще важливіше, обмежений простір для демонстрації товарів покупцям. Також присутнє невелике коло покупців, оскільки людям доводиться йти в магазин. Якби не було цих обмежень, підприємцям не довелося б продавати тільки популярні товари, як часто буває в традиційній торговій бізнес-моделі. В оффлайн-магазинах тримати на складі непопулярні товари вважається програшною стратегією, оскільки

треба платити за необхідне місце для зберігання великої кількості товарів, які можливо ніхто і не купить. Але якщо у підприємця інтернет-магазин, то у його розпорядженні місце для безлічі товарів, оскільки плата за

інтернет-магазин невисока Більше того, якщо продавати цифровий контент, то складський простір взагалі не потрібен, тобто плата або мінімальна або дорівнює нулю. Суть економіки “довгого хвоста” зводиться до того, що можна отримувати дохід, продаючи багато різних товарів, кожен з них – по кілька екземплярів безлічі різних людей. Величезний каталог товарів – це чудово, але питання, на яке важко дати відповідь, полягає в тому, як саме користувачі знаходять те, що їм потрібно? Саме тут допомагають рекомендаційні системи. Саме такі системи допомагають людям знаходити різні речі, про існування яких ті навіть не здогадувалися. У мережі інтернет на сьогоднішній день титанами у відношенні контенту та рекомендацій вважаються сервіси Amazon і Netflix [18].

Машинне навчання і штучний інтелект – це терміни котрі є синонімами сучасної епохи технологій. Машинне навчання застосовується для створення алгоритмів по обробці великих даних. Глибоке навчання – це складова машинного навчання, де штучні нейронні мережі, алгоритми, подібні до людського мозку, вчаться на великій кількості даних. Машинне навчання використовує спосіб, подібний до людського. Людина вчиться новому вмінню через практику та досвід. Машинне навчання використовує той самий спосіб – спочатку практикується на великій кількості даних, отримує досвід, потім використовує набутий досвід для виконання певних дій з інформацією.

Мета дослідження: проектування архітектури інформаційно-аналітичної системи особистісних вподобань користувачів.

Об’єкт дослідження: рекомендаційна система, котра визначає рівень сумісності між користувачами.

Предмет дослідження: алгоритми рекомендаційних систем на базі машинного навчвання.

Завдання дослідження:

- проаналізувати сутність роботи рекомендаційних систем;
- проаналізувати наявні на ринку рішення;
- дослідити основи роботи машинного навчання;
- розробити вимоги до рекомендаційної системи особистісних вподобань користувачів;
- розробити архітектуру інформаційно-аналітичної системи особистісних вподобань користувачів;
- зібрати дані для машинного навчання;
- розробити моделі на яких базуватимуться характеристики користувачів;
- розробити систему генерації рекомендацій.

Методи дослідження:

- аналіз — дозволяє розкласти досліджуваний матеріал на одиниці, вивчити окремі частини елементів;
- індукції — це перехід від часткового до загального;
- дедукції — це перехід від загального до часткового;
- моделювання — переносить реальний об'єкт в створювану модель;
- статистичний — висновки робляться на основі статистичних даних;
- емпіричний — дозволяє робити висновки на основі практичних спроб.

РОЗДІЛ 1
АНАЛІЗ ТА ХАРАКТЕРИСТИКА РЕКОМЕНДАЦІЙНИХ СИСТЕМ

1.1. Дослідження призначення рекомендаційних систем

Рекомендаційна система — це система, котра підбирає і пропонує користувачам релевантний контент, ґрунтуючись на відомих даних про користувача та на взаємодії користувача з контентом. Релевантність — це розташування контенту у відповідності з тим, що найбільше підходить користувачеві в даний момент часу.

Останнім часом рекомендаційні системи набули дуже широкого поширення, в сучасному світі є безліч прикладів використання. Найчастіше рекомендаційні системи розроблені під фільми, музику, книги, новини, дослідні статті та і взагалі під більшість товарів. Але рекомендаційні системи застосовуються також і в багатьох інших сферах, включаючи цифрові дані, пошук роботи, фінансові послуги, страхування життя, тобто, майже всюди, де потрібно зробити вибір [1].

Застосування рекомендаційних систем вимагає знань про користувачів в першу чергу, а вже в другу чергу відповідний алгоритм для обробки даних про користувачів. Рекомендаційні системи найчастіше передбачають домашнє використання інтернет сервісів, оскільки так можна не тільки охопити окремих користувачів, але і отримати дані про їх поведінку [1].

Існують три основних типи рекомендацій:

- неперсоналізовані
- напівперсоналізовані;
- персоналізовані [1]

Нижче на Рис.1.1. описано види рекомендацій різного ступеня персоналізації та що відображає кожен вид рекомендацій.

Рис. 1.1. Ступені персоналізації контенту

Джерело:побудовано на основі[1]

Неперсональні рекомендації передбачають для прикладу список найпопулярніших об'єктів. В даному прикладі розрахунок ведеться на те, що даному користувачеві сподобаються ті самі об'єкти, що і більшості користувачів. До неперсональних рекомендацій також відноситься вибудовування об'єктів у списку за датою їх появи, для прикладу коли найпершими відображаються найновіші об'єкти. Будь-який користувач, який взаємодіє з рекомендаційною системою, бачить ті ж рекомендації, що й інші користувачі. До таких рекомендацій можна також віднести спецпропозиції, коли відвідувачам ресторанів пропонуються алкогольні напої у п'ятницю ввечері, або каву вранці [1].

Напівперсональні рекомендації — це рекомендації, що передбачають поділ користувачів за групами. Користувачів можна розділити на групи за різними ознаками, наприклад: за віком, за національністю, за статтю. Також існує класифікація за характерною ознакою, наприклад: бізнесмени, робочі різних спеціалізацій, студенти, водії різних видів транспорту. Приклад системи напівперсоналізованих рекомендацій — це система для продажу квитків на концерти, котра рекомендує ґрунтуючись на країні чи місті

перебування користувача. Інший приклад — це прослуховування музики на пристрої, рекомендаційна система може спробувати визначити за допомогою датчиків, переміщується цей пристрій чи ні. Якщо пристрій переміщується, то на основі даних про інтенсивність переміщення система може визначити чи займається людина спортом. Якщо пристрій нерухомий, тор користувачу, ймовірно більше підійде спокійна музика. Якщо пристрій рухається з високою інтенсивністю, то ймовірно користувач займається спортом і система порекомендує більш інтенсивну музику. Такі рекомендації будуть корисні користувачу, адже давно доведено, що інтенсивна музика збільшує ефективність занять спортом на 20 відсотків. Напівперсоналізована рекомендаційна система не знає нічого особисто про користувача, для такого типу рекомендаційних систем користувач є частиною певної групи чи сегменту. Інші користувачі, котрі входять до цієї ж групи, отримують ті ж самі рекомендації [1].

Персональні рекомендації — це рекомендації, котрі базуються на даних про конкретного користувача, а саме на інформації про те, яким чином користувач взаємодіяв із системою раніше. Так формуються рекомендації персонально для даного користувача. Більшість рекомендаційних систем при складанні персональних рекомендацій також враховують приналежність користувача до групи та популярність контенту. Приклад персональних рекомендацій можна побачити на сайтах інтернет-магазинів, де в особистому кабінеті користувача є розділ “Рекомендовано для вас”. Головна сторінка сервісу Netflix — це найяскравіший приклад персональних рекомендацій, котрий рекомендує лише персональні рекомендації. Але зазвичай інтернет сервіси комбінують різні типи рекомендацій. На Amazon також можна побачити розділ “Найбільше продаються товари — це розділ де відсутня персоналізація, а також розділ “Разом з цим купують ці товари”, тобто вибіркові рекомендації. Ці рекомендації пропонуються вибірково –

наприклад, тим людям, які дивляться цей товар. У розділі “Рекомендовано для вас” вже видаються персональні рекомендації зареєстрованим користувачам. Потреба в персональних рекомендаціях пов'язана з тим, що людям цікаві не тільки популярні товари, але і ті, які не входять до числа найбільш продаваючих [1].

Більшість рекомендаційних систем намагаються тим чи іншим чином застосовувати дані, показані на Рис1.2.

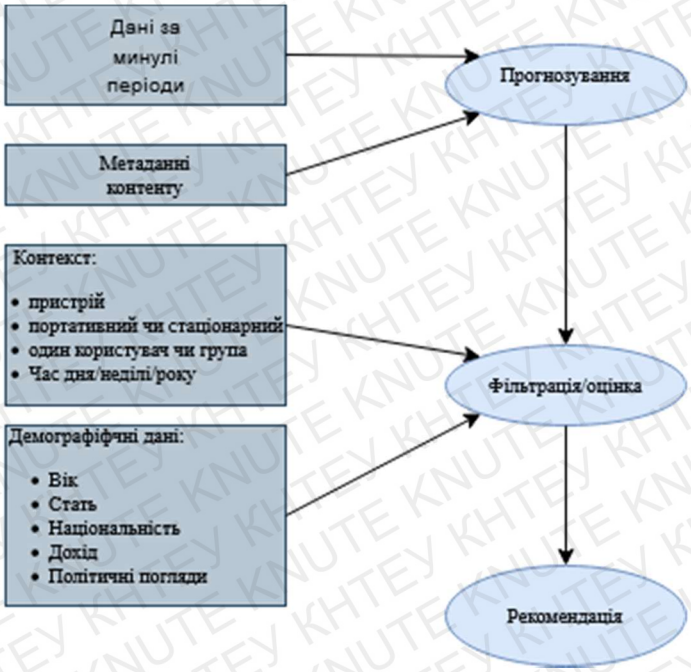

Рис. 1.2. Дані на які може спиратись рекомендаційна система

Джерело:побудовано на основі[1]

Крім того, Рис. 1.2 ілюструє ще один факт, який необхідно враховувати: прогнозування оцінок чи рейтингів — це лише одне з безлічі завдань рекомендаційної системи. Інші фактори також можуть істотно впливати на те, що саме система показуватиме користувачеві [2].

1.2. Загальні відомості про машинне навчання

Машинне навчання базується на нейронних мережах. “Н” — це послідовність нейронів, з'єднаних між собою синапсами [19].

Структура нейронних мереж була запозичена індустрією інформаційних технологій з біології. Завдяки аналогічній структурі, комп'ютер отримує можливість аналізувати і навіть запам'ятовувати різну інформацію. Нейронні мережі здатні не тільки аналізувати вхідну інформацію, а й відтворювати дану інформацію з пам'яті у майбутньому. Іншими словами, нейромережа це машинна інтерпретація мозку людини, в якому знаходяться мільйони нейронів котрі передають інформацію у вигляді електричних імпульсів.

1.3. Розгляд призначення нейронних мереж

Наразі нейронні мережі використовуються для вирішення складних завдань, котрі вимагають аналітичних обчислень подібних тим, котрі опрацьовує людський мозок. Найпоширенішими застосуваннями нейронних мереж є:

- класифікація - розподіл даних по параметрах. Наприклад, на вхід дається набір даних про людей і потрібно вирішити, кому з них давати кредит, а кому ні. Цю роботу може зробити нейронна мережа, аналізуючи таку інформацію як: вік, платоспроможність, кредитна історія і т.д.;
- передбачення – можливість прогнозувати наступний крок. Наприклад, зростання або падіння акцій, грунтуючись на ситуації на фондовому ринку;
- розпізнавання – в даний час, саме широке застосування нейронних мереж. Використовується в Google, Microsoft, банківських системах коли люди шукають фото або в камерах телефонів, коли воно визначає положення вашого обличчя і виділяє його і багато іншого [20].

Базовими одиницями нейронних мереж є нейрони. Нейрон – це обчислювальна одиниця, яка отримує інформацію, робить над нею прості обчислення і передає її далі. Нейрони діляться на три основних типи: вхідний (синього кольору), прихований (червоного кольору) і вихідний (зеленого). Також є нейрон зміщення і контекстний нейрон. У тому випадку, коли нейромережа складається з великої кількості нейронів, вводять термін шару.

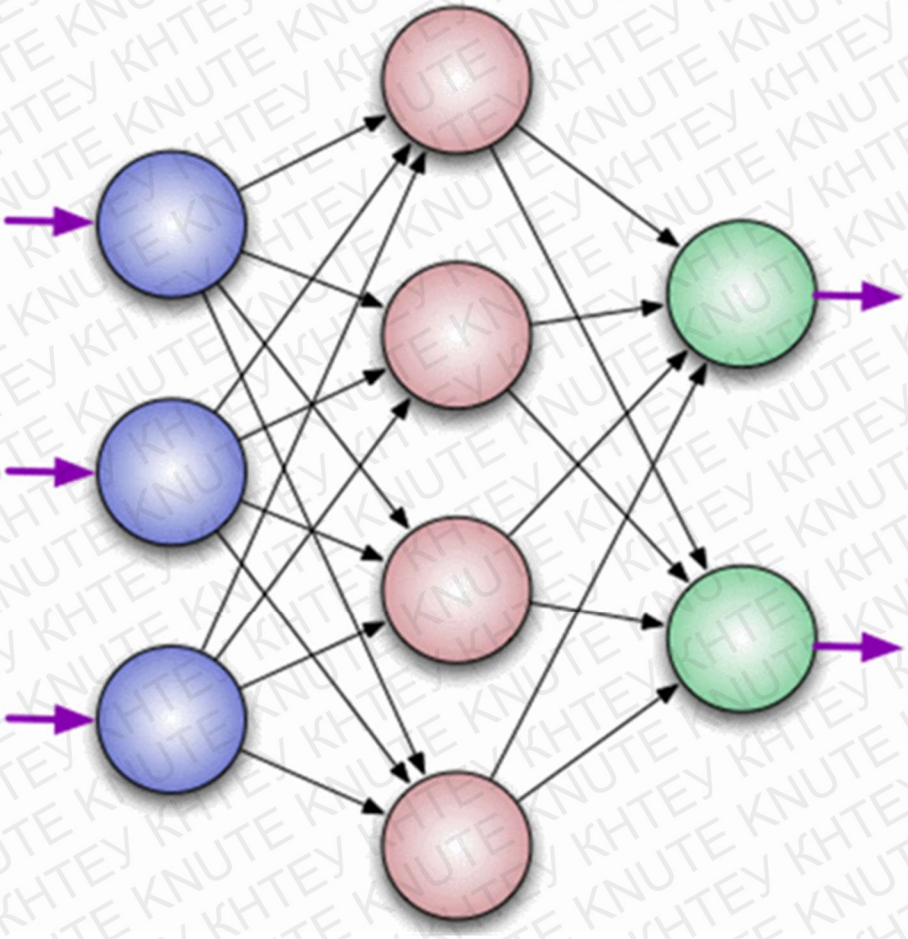

Рис. 1.3. Базова структура нейромережі

Джерело:побудовано на основі[19]

Відповідно, є вхідний шар, який отримує інформацію, n прихованих шарів (зазвичай не більше трьох), які обробляють інформацію і вихідний шар, який виводить результат. У кожного з нейронів є дві основні параметри: вхідні дані (input data) і вихідні дані (output data). У разі вхідного нейрона: input = output. В інших, в поле input потрапляє сумарна інформація всіх нейронів з попереднього шару, після чого, вона нормалізується, за

допомогою функції активації (поки що просто уявімо її f (x)) і потрапляє в поле output [19]. Важливо пам'ятати, що нейрони оперують числами в діапазоні [0,1] або [-1,1]. Щоб обробляти числа, які виходять з даного діапазону необхідно розділити 1 на це число. Даний процес називається нормалізацією і дуже часто використовується в нейронних мережах [19].

1.4. Нейронні мережі нового покоління

Щодо нового типу обчислень, який імітує роботу людського мозку, вже змінив те, як вчені могли вирішувати деякі з найскладніших проблем обробки інформації. Дослідники з “Researchgate” знайшли спосіб змусити так звані резервуарні обчислення працювати від тридцяти трьох до мільйона разів швидше, зі значно меншими обчислювальними ресурсами і меншим обсягом даних. Фактично, в одному з випробувань цього нового покоління нейронних мереж дослідники вирішили складне обчислювальне завдання за секунду на настільному комп'ютері. Деніел Готьє, провідний автор дослідження та професор фізики в Університеті штату Огайо визначив, що при використанні сучасної технології для вирішення тієї ж проблеми з обчисленням необхідний суперкомп'ютер, причому навіть на суперкомп’ютері на це йде набагато більше часу. Готьє зауважив, що з новим поколінням можна виконувати дуже складні завдання з обробки інформації за короткий проміжок часу, використовуючи набагато менше комп'ютерних ресурсів порівняно з тими технологіями, що поширені на сьогодення. За словами Готьє, резервуарні обчислення — це алгоритм машинного навчання, розроблений на початку 2000-х років і використовується для вирішення найскладніших обчислювальних завдань, таких як прогнозування еволюції динамічних систем, котрі змінюються з часом [21].

Динамічні системи, такі як погода, важко передбачити, бо лише одна невелика зміна в одній умові може мати серйозні наслідки. Одним із відомих прикладів є “ефект метелика“, в якому в одній метафоричній ілюстрації – зміни, викликані помахом крил метелика, можуть зрештою вплинути на погоду за кілька тижнів. Попередні дослідження показали, що резервуарні обчислення добре підходять для вивчення динамічних систем і можуть дати точні прогнози про те, як вони будуть поводитися в майбутньому. Це можливо за допомогою штучної нейронної мережі, схожої на людський мозок. Вчені завантажують дані з динамічної мережі в “резервуар” випадково пов'язаних штучних нейронів в мережі. Мережа дає корисний результат, який вчені можуть інтерпретувати і передавати назад у мережу, створюючи більш точний прогноз того, як система розвиватиметься у майбутньому. Чим більша і складніша система і чим точнішим має бути прогноз, тим більше має бути мережа штучних нейронів і тим більше обчислювальних ресурсів і часу потрібно для виконання завдання. Одна проблема полягала в тому, що резервуар штучних нейронів — це “чорна скринька” і вчені точно не знають, що відбувається всередині резервуару. Вчені тільки знають, що система працює. Готьє пояснив, що штучні нейронні мережі, котрі лежать в основі обчислень резервуарів, побудовані на математиці. У цьому дослідженні Готьє та його колеги досліджували це питання та виявили, що всю обчислювальну систему резервуара можна значно спростити, що різко знизить потребу в обчислювальних ресурсах та значно заощадить час. Вони перевірили свою концепцію на задачі прогнозування з використанням погодної системи, розробленої Едвардом Лоренцем, робота якого призвела до розуміння ефекту метелика. Обчислення резервуарів нового покоління стало явним переможцем перед сучасними досягненнями у вирішенні завдання прогнозування Лоренца. В одному відносно простому моделюванні, виконаному на настільному комп'ютері, нова система була в 33-163 рази

швидше за поточну модель. Але коли мета полягала у високій точності прогнозу, обчислення резервуарів нового покоління були приблизно в один мільйон разів швидше. І обчислення нового покоління досягли такої самої точності з еквівалентом всього 28 нейронів у порівнянні з 4000, необхідними для моделі поточного покоління. Важливою причиною прискорення є те, що "мозок", котрий стоїть за цим новим поколінням резервуарних обчислень, вимагає набагато менше розминки та навчання порівняно з нинішнім поколінням для отримання тих самих результатів. Розминка — це тренувальні дані, які необхідно подати як вхідні дані в комп'ютер резервуара, щоб підготувати до реального завдання [21].

В даний час вченим доводиться вводити 1000 або 10000 точок даних або більше, щоб розігріти алгоритм нейроннио мережі. І це всі дані, які втрачені, які не потрібні для реальної роботи. Необхідно запровадити лише одну, дві чи три точки даних для ефективного використання даних.

І як тільки дослідники будуть готові навчити резервуарний комп'ютер робити прогнози, знову ж таки, в системі наступного покоління буде потрібно набагато менше даних для розминки. У тесті завдання прогнозування Лоренца дослідники могли отримати самі результати, використовуючи 400 точок даних, як і поточне покоління, отримане з допомогою 5000 точок даних чи більше, залежно від бажаної точності [21].

1.5. Огляд кращих рекомендаційних систем сучасного ринку

Netflix – це стрімінговий сервіс. Основний контент сервісу – це фільми та серіали, добірка яких постійно оновлюється. Завдання, яке виконують рекомендації Netflix, полягає в тому, щоб підтримувати інтерес користувача до представленого контенту якнайдовше, стимулюючи клієнта оплачувати підписку кожного місяця. Причому сервіс працює на різних платформах.

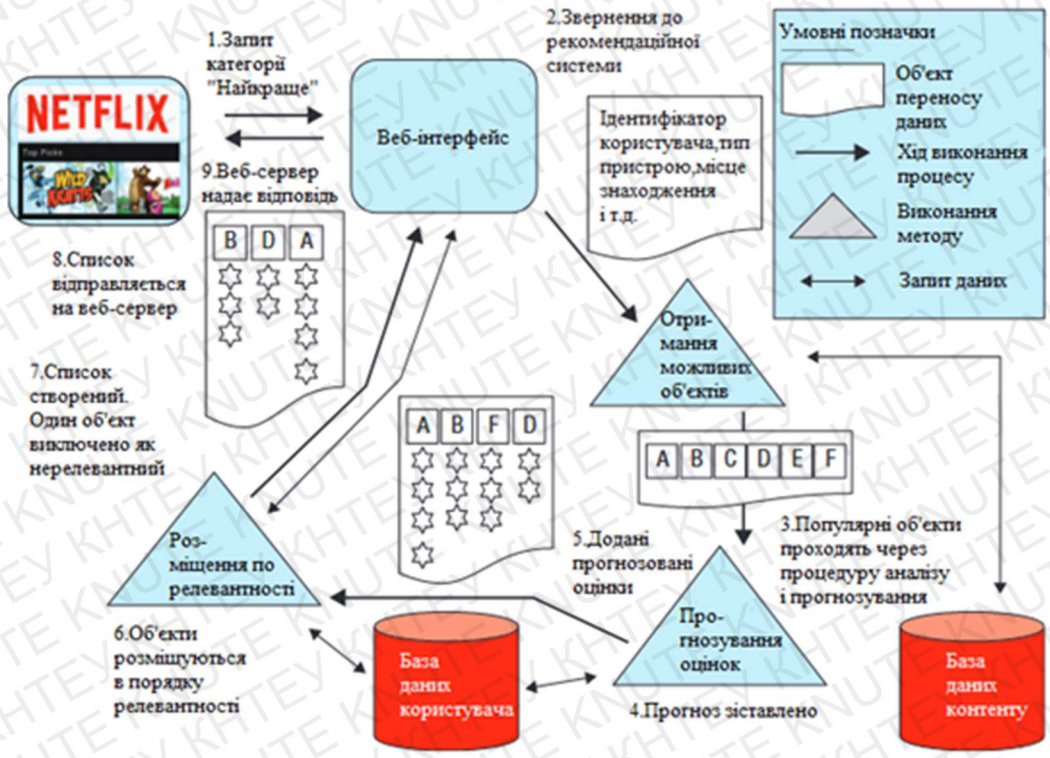

Рис. 1.4. Процес видачі персоналізованих рекомендацій рекомендаційною системою Netflix

Джерело:побудовано на основі[1]

Вивчивши рис. 1.4, легко зрозуміти, що при роботі з рекомендаційними системами необхідно враховувати безліч аспектів. У вищевказаному описі відсутні етапи збору даних і побудови моделей.

Категорія “Зараз у тренді” — це приклад неперсоналізованих рекомендацій. Тренд — це досить розмите поняття, котре може мати декілька значень, але в даному випадку йдеться про контент, який користувався популярністю до недавнього часу. Категорія “Популярне на Netflix” також включає популярний контент, але період часу, протягом якого цей контент користується популярністю довше, десь близько тижня і відображається за релевантністю [1].

Персональними рекомендаціями є добірка “Найкраще”, котра відображає користувацькі уподобання. В даній категорії показаний контент,

який, за прогнозами рекомендаційної системи Netflix, користувач захоче подивитися найближчим часом [1].

Часто прогнози є вірними. Завдяки використанню профілів користувачів, рекомендаційні системи збирають необхідні дані про користувачів і видають більш точні рекомендації. Звісно декілька користувачів можуть використовувати один і той же профіль, але сучасні алгоритми добре навчені і вміють розпізнати поведінку користувачів навіть коли ті користуються сервісом під іншим профілем.

1.6. Бізнес вимоги до рекомендаційних систем

Іноді рекомендаційна система не може робити все за користувача. Рекомендаційна система не може зрозуміти, які рекомендації надавати, якщо наприклад люди купують мультфільми та фільми жахів. Зрештою у системи виникають асоціації. І зрештою, рекомендаційна система може видати дорослі фільми жахів дітям. Одним зі способів цього уникнути є фільтрація контенту, що дозволяє обмежити рекомендації певних типів контенту в залежності від того, що користувач зараз дивиться. Такі фільтри можуть заважати алгоритму працювати, але фільтрація є вкрай необхідною. Бізнес-правила можуть бути визначені позитивно або негативно. В рекомендаційній системі “its-tar” бізнес правила визначають підбір користувачів в першу чергу по віковим категоріям і в другу чергу за статтю і професією, в залежності від того кого користувач хоче знайти.

1.4 Висновки до розділу 1

В даному розділі було дослідження призначення рекомендаційних систем і тенденції використання у сучасних додатках. Рекомендаційні системи в більшості випадків використовують машинне навчання для обробки даних. Було розглянуто найбільш розвинені рекомендаційні системи сучасного ринку і встановлено способи взаємодії таких систем з користувачами.

З’ясовано роль і вектор розвитку нейронних мереж нового покоління резервуарного типу. Даний тип нейронних мереж дозволяє використовувати значно менше апаратних ресурсів порівняно з традиційними нейронними мережами. Також значною привілегією подібного типу нейромереж є необхіднисть значно меншої кількості даних для розминки системи, в залежності від точності прогнозів. Також було аргументовано використання бізнес правил у рекомендаційних системах. Подібні бізнес правила необхідно встановлювати в першу чергу, так як веб-додатками можуть користуватись зовсім різні цільові категорії людей.

РОЗДІЛ 2
РОЗГЛЯД ОСНОВНИХ ЕТАПІВ РОЗРОБКИ ПЗ ТА ТЕХНІЧНЕ ЗАВДАННЯ НА РОЗРОБКУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1. Розподіл ролей у розробці програмного забезпечення

Будь-яка інформаційна система не може бути реалізована без знання проектною командою предметної області, проектна команда має розуміти для чого і для кого розробляє продукт, що даний продукт повинен робити. Якщо проект пов’язаний з розробкою інформаційної системи для автоматизації продажів, то відповідно проектна команда має бути ознайомленою з тим як працюють продажі, що треба автоматизувати, хто буде використовувати систему, як система буде виглядати для користувачів.

У будь-якій команді проекту чи то малій IT компанії чи великій корпорації, кожному учаснику відведено свою роль у проекті. Бізнес аналітики вивчають предметну область з якою пов’язаний проект, якщо наприклад проект — це бухгалтерська система, то бізнес аналітики знають область бухгалтерської діяльності, процеси даної діяльності, і загалом як і що роблять бухгалтери. Менеджери з продажів займають велику роль у будь-якому проекті, так як шукають клієнтів, безпосередньо спілкуються з клієнтами і проводять угоди між сторонами. Бухгалтери і економісти займаються фінансовими питаннями проекту, а юристи юридичними. Рекрутери шукають необхідний та підходящий для проекту персонал.

Проектні менеджери керують проектною командою. Архітекторипрограмного забезпечення розробляють архітектуру системи.

Конструктори ПЗ або ж програмісти безпосередньо реалізують інформаційну систему. Тестувальники тестують ПЗ на наявність несправностей, щоб споживач отримав робочий та якісний продукт у використання.

З вищесказаного можна відмітити основних членів команди, необхідних для реалізації проекту, пов'язаного з розробкою ПЗ. У малих проектах зазвичай ці ролі нікуди не пропадають, адже виконання декількох ролей бере на себе одна людина, або залучаються люди з аутсорсу для одноразового виконання певної роботи. Наприклад директор невеликої компанії може виконувати ще і роль менеджера з продажів — спілкуватися з замовниками безпосередньо, укладати угоди. Проектний менеджер може виконувати роль бізнес аналітика і досліджувати предметну область проекту. Програмісти часто стають одночасно і архітекторами і програмістами, і тестувальниками ПЗ. Бухгалтери, юристи, рекрутери часто залучаються з аутсорсу.

2.2. Життєвий цикл розробки ПЗ

Будь-яка розробка проекту починається з визначення проблеми. У клієнта є проблема, яку програмне забезпечення має вирішити, тому власне клієнт і шукає компанію котра вирішить дану проблему. Бізнес аналітик знає діяльність, якою займається клієнт, методи роботи бізнесу клієнта. Тому перший етап циклу розробки ПЗ — це виявлення проблеми. Правильно виявлена проблема ще до початку реалізації проекту сприяє правильному вирішенню — це найважливіший етап, так як ПЗ написане для вирішення нецільової для проекту проблеми простіше викинути і написати нове, а це великі гроші та час витрачені дарма [3].

Далі розробляються вимоги до ПЗ. Це також надважливий етап втілення ПЗ. Ключові для системи вимоги повинні бути вироблені ще до початку проектування архітектури і тим паче втілення проекту [4].

Нижче на Табл 2.1 зображено у скільки разів збільшується вартість проекту при виявленні дефектів у ПЗ на кожному з етапів реалізації проекту.

Таблиця 2.1.

Ціна дефектів у вимогах на різних етапах реалізації проектування

Час внесення дефекту	Вироблення вимог	Проектування архітектури	Конструювання	Тестування системи	Після релізу ПЗ
Вироблення вимог	1	3	5-10	10	10-100
Проектування архітектури	-	1	10	15	25-100
Конструювання	-	-	1	10	10-25

Джерело:побудовано на основі[3]

Як видно з рисунку вище — чим раніше буде знайдено дефект у вимогах, тим меншою буде вартість його виправлення. Тому вимогам до ПЗ приділяється особлива увага, аби вся команда розробки знала що розробляє, і що саме дане ПЗ має робити. У невеликих проектах плануванню часто приділяють менше часу і більше реалізації, тільки через те, що проекти менше за розміром.

Ключове правило розробки ПЗ — чим більше часу буде витрачено на планування і правильну розробку вимог до ПЗ, тим менше часу піде у

програмістів на саму розробку ПЗ. Вимоги до ПЗ визначені у технічному завданні.

2.3. Технічне завдання на розробку ПЗ

Загальні відомості

Розробка вимог до системи починається з визначення проблеми. Визначення проблеми — це фундамент всього процесу програмування. Неправильно визначена проблема призводить до неправильного програмного забезпечення. У випадку рекомендаційної системи проблема полягає в рекомендування найбільш підходящих людей для користувача веб-сайту. Інформаційно-аналітична система користувацьких вподобань призначена для підбору найбільш підходящого користувача для іншої особи. Дана система рекомендуватиме користувачів котрі підходять для користувача системи за такими ознаками як вік, стать, інтереси та інші дані в залежності від використаних моделей.

Далі після визначення проблеми слідує розробка вимог до інформаційної системи. Вимоги докладно описують, що повинна робити програмна система, а їх розробка — це перший крок до вирішення проблеми. Зазвичай вимоги розробляються на основі потреб майбутніх кінцевих користувачів.

Вимоги поділяються на явні і неявні. Явні вимоги — це ,якщо узагальнити, бізнес-вимоги. Явні вимоги — це вимоги до самого функціоналу ПЗ котрі узгоджуються з замовником. Явні вимоги повинні бути чітко визначені та формалізовані, підписані замовником у додатку до договору аби надалі уникнути зміни ключових вимог зі сторони замовника [3].

Неявні вимоги — це вимоги більше до функціонування ПЗ у зовнішньому середовищі, наприклад як система повинна працювати під час навантаження, як оброблятиме ту чи іншу кількість запитів, як буде себе

поводити у різних середовищах функціонування, наскільки система відмовостійка і інші вимоги, котрі не стосуються самого функціоналу системи. Вцілому, якщо провести метафору про явні вимоги, то даний тип вимог має бути подібним воді — на заморожену воду легше опиратись, а якщо вимоги рідкі як вода то на такі вимоги неможливо опиратись. Але на практиці не може бути заморожених вимог, так як клієнту необхідно побачити деяку частину ПЗ, аби сформулювати більш точні вимоги. Та зазвичай такі зміни не стосуються ключових вимог, котрі затверджені ще до початку втілення проекту. Тому власне ключові вимоги до ПЗ затверджуються документально. Важливість явного набору вимог пояснюється декількома причинами:

- явні вимоги допомагають гарантувати, що функціональність системи визначається користувачем, а не програмістом;

- якщо вимоги сформульовані явно то користувач може проаналізувати і затвердити їх;

- якщо явних вимог немає, програмісту зазвичай самому доводиться виробляти їх під час програмування;

- явні вимоги дозволяють не вгадувати, а точно знати, чого хоче користувач [3].

Також варто враховувати, що багато проблем до вимог зникають при згадці про комерційні передумови проекту. Тобто вимоги, які спочатку здавалися прекрасними ідеями, можуть виявитися жахливими при оцінці затрат на їх реалізацію. Розробники ПЗ, які беруть до уваги комерційні наслідки своїх рішень, високо цінуються [5].

Вимоги за функціональністю поділяються на функціональні та нефункціональні [4]. Функціональні вимоги визначають функції та функціональні можливості всередині системи програмного забезпечення. До функціональних вимог належать:

а) бізнес вимоги;

б) користувацькі вимоги;

в) функціональні вимоги [22].

Нефункціональні – це вимоги, які не належать до функціонального аспекту програмного забезпечення. Вони не є явними або очікуваними характеристиками програмного забезпечення, на які розраховує користувач. До нефункціональних вимог належать:

а) бізнес-правила - визначають обмеження, що виникають з предметної області і властивостей об'єкта, що автоматизується (збір даних і генерація рекомендацій);

б) системні вимоги та обмеження - визначають елементарні операції, які повинна мати система, а також різні умови, яким система може задовольняти [22]. До системних вимог і обмежень відносяться:

1) обмеження на програмні інтерфейси, в тому числі до зовнішніх систем;

2) вимоги до обладнання та комплектуючих, що застосовуються, а також програмне середовище в якому буде працювати система;

3) вимоги до атрибутів якості;

4) вимоги до документування;

5) вимоги до дизайну і зручності керування;

6) вимоги до безпеки та надійності;

7) вимоги до відмовостійкості та відновлюваності системи;

8) вимоги до продуктивності;

9) вимоги до експлуатації і персоналу;

10) інші вимоги та обмеження (зовнішні впливи, мобільність, автономність, тощо).

Технічне завдання розробляється згідно GOST 34.602-89 [6].

1.1. Найменування системи

“Інформаційно-аналітична система особистісних вподобань користувачів”.

1.1.1. Повне найменування системи

“Інформаційно-аналітична система особистісних вподобань користувачів на базі машинного навчання”.

1.1.2. Скорочене найменування системи:

“Інформаційно-аналітична система its-tar”.

1.2 Планові терміни початку та закінчення робіт

Початок робіт відбувся 20.11.2020. Кінець робіт заплановано на 30.10.2021.

1.3. Порядок оформлення і пред’явлення результатів робіт

Система “ Інформаційно-аналітична система its-tar” повинна бути передана замовнику з результатами розробки у вигляді:

а) фінансовий акт приймання-передачі наданих послуг (у двох екземплярах);

б) технічний акт приймання-передачі наданих послуг (у двох екземплярах);

в) програмне забезпечення (у одному екземплярі);

г) ліцензію або ліцензійну угоду на супроводжувальне програмне забезпечення.

1.2. Головний бенефіціар та потенційні користувачі системи

Головним бенефіціаром системи виступає замовник інформаційно-аналітичної системи its-tar. Потенційними користувачами системи “ Інформаційно-аналітична система its-tar” є користувачі веб-сайтів, до яких підключена дана система.

2. Мета та призначення створення системи

2.1. Призначення системи

Система “ Інформаційно-аналітична система its-tar” призначена для підбору найбільш підходящих користувачів особі котра використовує систему.

2.2. Мета створення системи

Метою створення системи є розробка рекомендаційної системи для пошуку підходящих пар.

3. Вимоги до системи

3.1. Вимоги до системи в цілому

Вимоги до системи в цілому є бізнес вимогами [4]. Інформаційно-аналітична система ist-tar передбачає:

а) для веб-сайтів – це надання даних про користувачів;

б) для користувачів – видача рекомендацій.

3.1.1. Вимоги до структури та функціонування системи, перелік підсистем

Система повинна мати структуру пов’язаних між собою підсистем. До системи повинні бути включені такі підсистеми:

а) підсистема збору даних;

б) підсистема зберігання тимчасових даних для тренування моделей даних;

в)підсистема бази даних для зберігання постійних даних про користувачів інформаційно-аналітичної системи;

г)підсистема аналізу даних;

ґ) підсистема роботи з моделями даних;

д)підсистема конфігурації;

е)підсистема запуску;

є)підсистема генерації рекомендацій;

ж)підсистема інтеграції з веб-сайтами.

3.1.1.1. Вимоги до способів і засобів інформаційного обміну між компонентами системи

Сполучення характеризує силу зв'язку одного компоненту програми з іншими. Повинні бути створені компоненти системи, що мають нечисленні, безпосередні, явні і гнучкі відносини з іншими класами, що називається “слабким сполученням” [3]. У контекстах класів і методів концепція сполучення одна і та ж, сполучення між методами і класами називаються сполученнями між «модулями».

Сполучення модулів повинно бути досить слабким, щоб одні модулі могли з легкістю використовувати інші. Тому модулі системи повинні бути створені з мінімальним числом залежностей.

3.1.1.2. Вимоги до режимів функціонування системи

Система повинна працювати в режимі онлайн цілодобово без вихідних окрім випадків запланованих актів технічного обслуговування чи незапланованих надзвичайних ситуацій. Повинен бути резервний сервер, на якому система працюватиме поки на основному сервері відбуватиметься обслуговування.

3.1.1.3. Вимоги до діагностування системи

Діагностування системи повинно відбуватися на стороні сервера під час запланованих перевірок, або незапланованих збоїв системи [6].

3.1.1.4. Вимоги до режимів управління системою

У інформаційно-аналітичній системі повинно бути розмежування прав доступу користувачів [8]. Тому система матиме 2 режими управління: адміністратор, користувач.

3.1.2. Показники призначення

3.1.2.1. Параметри, що характеризують ступінь відповідності системи призначенням

Параметрами, котрі характеризують ступінь відповідності системи призначенню є бізнес-правила проєкту [6].

3.1.2.2. Вимоги до пристосовності системи до змін

Система повинна пристосовуватись до змін інтернет з'єднання та під час оновлення [9].

3.1.2.3. Вимоги до збереження працездатності системи в різних ймовірних умовах

Система " Інформаційно-аналітична система its-tar" повинна бути стійкою до відмов у роботі. Навіть при поганому з'єднанні з інтернетом система повинна працювати, а при розривах з'єднання використовувати резервні канали зв'язку.

Відновлюваність системи. Завдяки "Cookie-файлам", збереженим під час сеансу веб-браузера система повинна відновлюватись з моменту останньої вдалої конфігурації, при умові, що не було здійснено процедуру виходу з інтегрованого веб-сайту [9].

3.1.3. Вимоги до надійності

До системи " Інформаційно-аналітична система its-tar" висуваються такі вимоги, щодо надійності: система повинна відповідати концепції "п'ять дев'яток".

3.1.3.1. Склад показників надійності до системи в цілому

В цілому показники надійності до системи визначаються вимогами до безпеки, відмовостійкості та відновлюваності.

3.1.3.2. Вимоги до надійності технічних засобів і програмного забезпечення

Технічні засоби повинні справно працювати на стороні клієнта за наявності електроживлення пристрою. На стороні сервера додатково повинні бути встановлені джерела безперебійного живлення для доступності системи. Програмне забезпечення повинно відповідати системним вимогам [7].

3.1.3.3. Вимоги до методів оцінки і контролю показників надійності на різних стадіях створення системи

На стадії розробки технічного завдання надійність системи оцінюється дотриманням параметрів відповідності системи призначенню.

3.1.5. Вимоги до експлуатації, технічного обслуговування, ремонту і зберігання компонентів системи

Система “ Інформаційно-аналітична система its-tar” може бути експлуатована на пристрої з програмним середовищем та інтернет з’єднанням, котрий задовольняє системні вимоги.

Перед використанням системи коритувач повинен дати згоду на забір і використання користувацьких даних системою.

Вимоги технічного обслуговування та ремонту: технічне обслуговування повинне бути плановим, а ремонт необхідно виконувати за наявності збоїв у роботі системи [9].

3.1.6 Вимоги до захисту інформації від несанкціонованого доступу

Щодо захисту інформації від несанкціонованого доступу висуваються такі вимоги:

а) облікові записи адміністратора і користувачів в інтегрованому веб-сайті повинні бути захищені надійними паролями, котрі відповідають політиці використання паролів;

б) доступ до керування системою на сервері повинен мати лише адміністратор інтегрованого веб-сайту;

в) адміністратор системи не повинен передавати стороннім особам дані автентифікації та ключі стороннім особам, котрі не мають повноважень для керування системою.

3.1.6.1. Вимоги до інформаційної безпеки

До системи “ Інформаційно-аналітична система its-tar” висуваються такі вимоги, щодо інформаційної безпеки: вимоги до цілісності, доступності, конфіденційності.

Вимоги до цілісності: система повинна забезпечувати зберігання логів та виконувати резервне копіювання за розкладом. Контроль та перевірка резервних копій має виконуватись адміністратором системи. Перевіряти цілісність баз даних необхідно автоматизовано [8].

Вимоги до доступності: система повинна бути доступною цілодобово при наявності інтернет з’єдняння та пристрою, що відповідає системним вимогам.

Вимоги до конфіденційності:

а) інформація у базі даних повинна бути захищеною від використання сторонніми особами;

б)дані про користувачів не повинні надаватися третім особам без згоди на те користувачів;

в) повинен бути передбачений захист від вставки скриптів для протидії атакам з використанням міжсайтовому скриптингу на інтегрованому веб-сайті.

Політика використання паролів. Пароль повинен бути мінімум 8 символів, максимум 32 символи, містити тільки латинські букви, цифри, спеціальні символи. Обов’язково повинен містити хоча б одну велику літеру.

3.1.6.2. Вимоги до антивірусного захисту

Для безпечного використання системи “ Інформаційно-аналітична система its-tar” веб-браузер користувача повинен мати останні оновлення безпеки.

3.1.6.3. Розмежування відповідальності ролей при доступі

До системи “Інформаційно-аналітична система its-tar” розмежовано доступ з правами адміністратора веб-сайту та адміністратора бази даних [8].

Адміністратор відповідає за функціонування і правильну інтеграцію інформаційно-аналітичної системи з веб-сайтом на стороні сервера, забезпечує безпеку системи вцілому. Користувач відповідає за користування системою на стороні клієнта, тому відповідає за власні автентифікаційні дані для входу на інтегрований веб-сайт.

3.1.7. Вимоги до захисту від впливу зовнішніх факторів

Пристрій користувача повинен бути захищений від негативних факторів зовнішнього середовища, котрі можуть спричинити виведення з ладу чи порушення нормальної роботи пристрою. Система повинна бути захищена від несанкціонованого доступу ззовні мережі.

3.1.8. Вимоги безпеки

Вимоги до безпеки включені в пунктах 3.1.6 “Вимоги до захисту інформації від несанкціонованого доступу”, 3.1.6.1 “Вимоги до інформаційної безпеки” та 3.1.6.2 “Вимоги до антивірусного захисту” даного технічного завдання.

Програма повинна мати в кожному модулі описані алгоритми обробки помилок, так як помилка програми може порушити функціонування операційної системи і навіть фізичні компоненти — це залежить від типу ПЗ. Виключення теж повинні бути описані у необхідних випадках.

Захисту програми від неправильних вхідних даних приділяється особлива увага. Програма повинна функціонувати так, щоб навіть при неправильних вхідних даних програма видавала правильні вихідні дані або правильно оброблювала помилки.

Також для захисту програми використовуються барикади методів, або ж іншими словами ізоляція модулів.

Не варто перевіряти всі вхідні у всіх місцях, так як завелика кількість перевірок сповільнює програму.

3.2. Перелік підсистем системи

Кожна система ділиться на підсистеми, або ж пакети. У випадку системи “Інформаційно-аналітична система its-tar” повинні бути такі підсистеми:

а) модулі збирання даних;

б) модуль роботи з базою даних;

в) натреновані моделі;

г) модуль конфігурації;

д)модуль запуску системи;

е)модуль генерації рекомендацій;

є)модуль інтеграції рекомендаційної системи з веб-додатками.

3.3. Вимоги до видів забезпечення

3.3.1. Вимоги до математичного забезпечення

В математичних операціях зі змінними потрібно враховувати, щоб не було ділення на нуль, не тільки якщо ділення на 0 вказано явно, а й формула може привести до ділення на 0. В такому випадку необхідно робити обробку помилки ділення на 0. Це стосується і інших математичних правил.

Вимоги до математичного забезпечення проявлятимуться в виборі функції активації для нейромережі, вагових коефіцієнтів сиснапсів.

3.3.2. Вимоги до інформаційного забезпечення

Вимоги до інформаційного забезпечення вказані в пункті 4 “Вимоги до програмного забезпечення” технічного завдання.

3.3.2.1. Вимоги до складу, структури і способів організації даних в системі

Склад даних являє собою записи у базі даних системи “Інформаційно-аналітична система its-tar”. Структура даних повинна бути визначена в модулях системи. Тимчасові дані про користувачів котрі збиратимуться будуть записуватись в файли формату “.csv”, а постійні дані про користувачів системи зберігатимуться в файлі бази даних SQLite під назвою “db.sqlite”.

3.3.2.2. Вимоги до інформаційного обміну між компонентами системи

Дані вимоги описано в пункті технічного завдання 3.1.1.1. “Вимоги до способів і засобів інформаційного обміну між компонентами системи“, де охарактеризовано “слабке сполучення” модулів системи.

3.3.2.3. Вимоги до інформаційної сумісності із суміжними системами

Система “Інформаційно-аналітична система its-tar” передбачає інтеграцію з іншими суміжними системами. Інтеграція — це взаємодія компонентів програми між собою або системи з іншою системою. Інтеграція ПЗ повинна йти в правильному порядку. Висхідна інтеграція походить від інтегрування класів до об’єднання цілих пакетів програм, а низхідна від пакетів програм до класів як найменших частин інтеграції.

Інтеграція функцій — це інтеграція модулів функціоналу програми.

У випадку інформаційно-аналітичної системи особистісних вподобань користувачів, інтеграція виконується на рівні пакетів.

3.3.2.4. Вимоги використання класифікаторів та уніфікованих документів

Кожен користувач повинен мати унікальний ідентифікатор в базі даних системи. Користувачі системи мають класифікуватися за заданими параметрами в моделях.

3.3.2.5. Вимоги щодо застосування систем управління базами даних

Для роботи системи на стороні сервера вимагається наявність встановленого SQLite та необхідних бібліотек мови програмування python. На стороні клієнта вимагається підтримка браузером Javascript та Python на стороні сервера.

3.3.2.6. Вимоги до структури процесу збору, обробки, передачі даних в системі представлення даних

Структура процесу збору, обробки, передачі даних повинна бути чітко зазначена в фізичній моделі бази даних та на діаграмах модулів системи [10].

3.3.2.7. Вимоги до захисту даних від руйнувань при аваріях і збоях в електроживленні системи

На стороні сервера повинні бути встановлені джерела безперебійного живлення для уникнення проблем при збоях в електроживленні системи, повинна бути передбачена система резервного копіювання та відновлення даних [8].

3.3.2.8. Вимоги до контролю, зберігання, оновлення та відновлення даних

На стороні сервера повинна бути передбачена система резервного копіювання та відновлення даних. Система управління базами даних відповідає за контроль, зберігання, оновлення та видлення даних.

3.3.2.9. Вимоги до процедури надання юридичної сили документам, що продукуються технічними засобами системи

Інформаційно-аналітична система особистісних вподобань користувачів не продукуватиме документи.

4. Вимоги до програмного забезпечення

Вимоги до програмного забезпечення є складовими системних вимог. До програмного забезпечення висуваються такі вимоги:

- підтримка інтернет протоколів TCP/IP, HTTP, HTTPS;
- операційна система з підтримкою веб-браузерів;

1. Chrome від версії 55;
2. Firefox від версії 50;
3. Internet Explorer від версії 11;
4. Microsoft Edge від версії 26;
5. Safari від версії 6;

- підтримка веб-браузером HTML5;
- підтримка веб-браузером Javascript;

- підтримка веб-браузером технологій на котрих базується інтегрований веб-сайт;

- увімкнені Cookie-файли у браузері.

5. Вимоги до технічного забезпечення

Вимоги до технічного забезпечення є складовими системних вимог. До технічного забезпечення висуваються такі вимоги:

- об’єм оперативної пам’яті: мінімум 2 Гб для пристроїв під управлінням ОС Windows, Linux, MacOS та мінімум 1Гб для пристроїв під управлінням ОС iOS, Android;

- наявність Інтернет з’єднання (швидкість від 10 Мбіт/с. для нормальної роботи).

6. Вимоги до методичного забезпечення

Система “Інформаційно-аналітична система its-tar” повинна комплектуватись інструкцією з розгортання системи та інтеграції з веб-сайтом. Дана інструкція включає: системні вимоги, інструкція з розгортання системи, опис технологій, необхідних для інтеграції з веб-сайтом. Системні вимоги вказано в пунктах “4. Вимоги до програмного забезпечення” та “6. Вимоги до технічного забезпечення” технічного завдання.

2.4. Висновки до розділу 2

Під час дослідження другого розділу автором було досліджено яким чином розподіляються ролі у проектних командах. Виявилося, що розробники ПЗ є ключовими фігурами у проекті, але за рахунок тільки розробників успішно завершити проект майже неможливо, адже у сучасному ІТ бізнесі з'явилося багато нових аспектів, котрих треба дотримуватись.

Розглянуто і з'ясовано життєвий цикл розробки ПЗ для кращого розуміння процесу розробки ПЗ.

Було розроблено і затверджено вимоги до програмного забезпечення з використанням перевірених часом стандартів. Створено технічне завдання до інформаційно-аналітичної системи за стандартом "GOST 34.602-89 Information technology".

РОЗДІЛ 3
РОЗРОБКА АРХІТЕКТУРИ ІНФОРМАЦІЙНО-АНАЛІТИЧНОЇ СИСТЕМИ

3.1. Перший рівень архітектурної деталізації: опис загального уявлення про систему

Сам принцип розробки архітектури ПЗ — це розбити складну проблему на більш прості елементи. Проектування необхідно робити 2 рази за проект — перший раз до розробки системи, другий раз під час розробки системи аби правильно осмислити всі аспекти програми. Проектування ПЗ — це розробка схеми перетворення специфікації програми в готову програму. На початку проектування ПЗ необхідно зв'язати елементи програми з об'єктами реального світу та їх атрибути.

Мінімальна але повна функціональність — це важлива вказівка, котрою необхідно керуватись, щоб правильно розробити ПЗ за обмеженої кількості можливостей.

При розробці архітектури необхідно керувати складністю проекту, так як при складному вирішенні простої проблеми втрачається час, а при простому вирішенні складної проблеми програмна система може взагалі не запрацювати що призведе до втрати грошей і часу на перерозробку програми. При конструюванні ПЗ необхідно слідувати за типовими рішеннями типових проблем для економії часу. Використання типових перевірених рішень допомагає краще розуміти і супроводжувати ПЗ [11].Зазвичай використовуються внизхідне і висхідне проектування. У першому випадку

метод дедукції(від загального до часткового) у другому випадку, індукції(від часткового до загального). Інформаційно-аналітична система створюється за використання внизхідного проектування, тобто від загального до часткового

Кожна програма має свою систему бізнес-правил, систему зберігання даних і точно систему взаємодії з користувачем. Кожна підсистема пов'язана з 1 і більше підсистем [3]. Це, наприклад зображено на рисунку нижче.

Рис. 3.1. Концептуальна модель основних модулів ПЗ та взаємодія між ними

Джерело:побудовано на основі[3]

У випадку інформаційно-аналітичної системи особистісних вподобань користувачів концептуальна модель основних модулів ПЗ дуже схожа на загальну, котра описана на Рис.3.1., але не має користувацького інтерфейсу, так як інформаційно-аналітична система інтегрується у сторонній веб-сайт з власним користувацьким інтерфейсом. В даний час час новизною стало інтеграція окремих модулів одних програм в інші програми. Дана

закономірність виникла через швидкозмінні вимоги до сучасних систем та необхідність пришвидшеня розробки програм. Нижче на Рис.3.2. зображено концептуальну модель інформаційно-аналітичної системи особистісних вподобань користувачів.

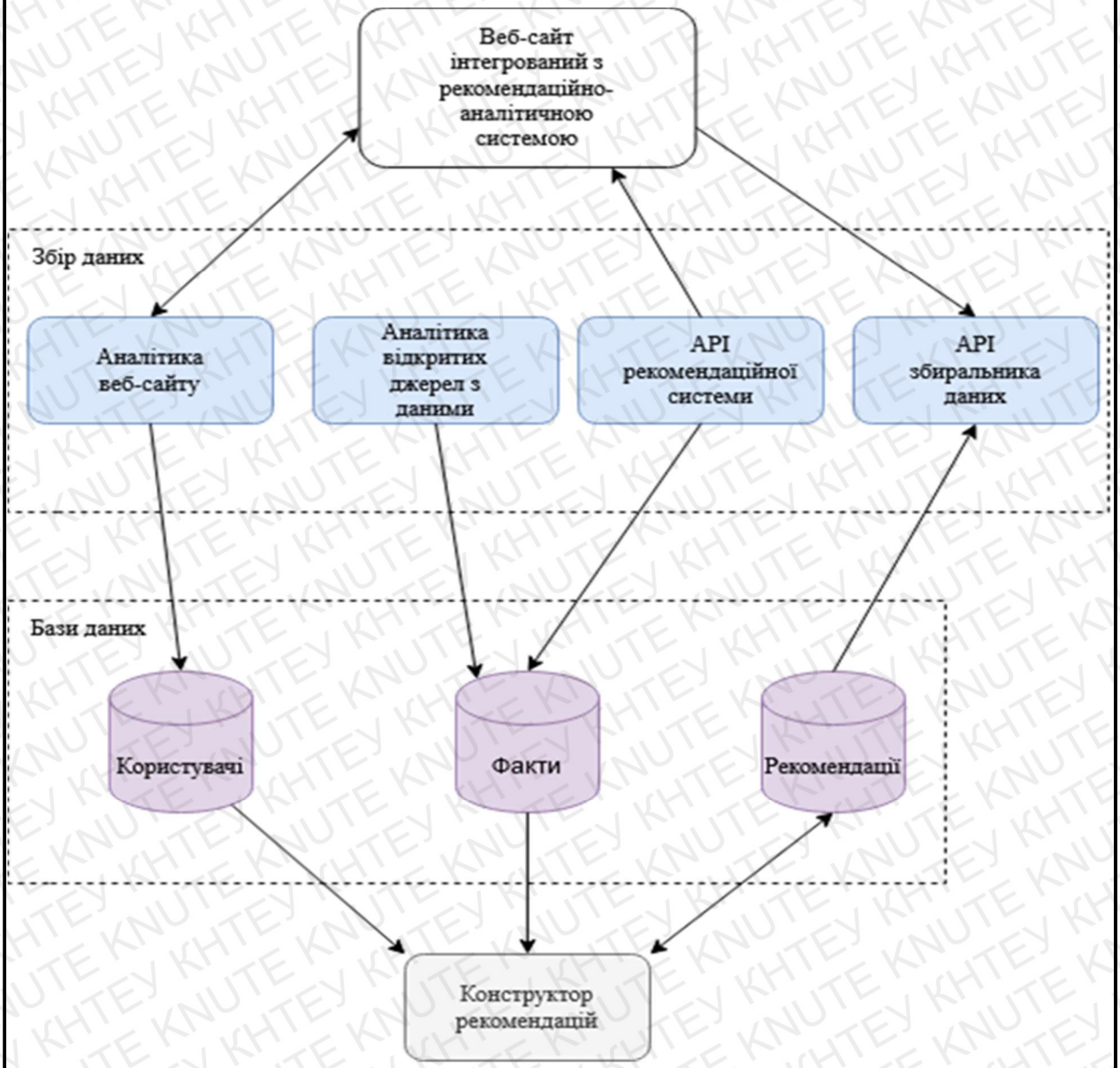

Рис. 3.2. Концептуальна модель інформаційно-аналітичної системи "its-tar"

Джерело:побудовано автором в системі "draw.io"

Пакети – це одиниця в ієрархії програмного забезпечення, котра збирає в собі класи та інші файли в одне ціле за функціональним призначенням.

Діаграма пакетів відображає наявні в системі підсистеми за допомогою пакетів.

У мові UML пакет – це загальноцільовий механізм для організації різних елементів моделі в множину, який реалізує системний принцип декомпозиції моделі складної системи і допускає вкладеність пакетів один в одного. Підпакет у UML – це пакет, котрий є складовою частиною іншого пакету [12].

На рисунку нижче зображено модель системи у вигляді пакетів в форматі UML.

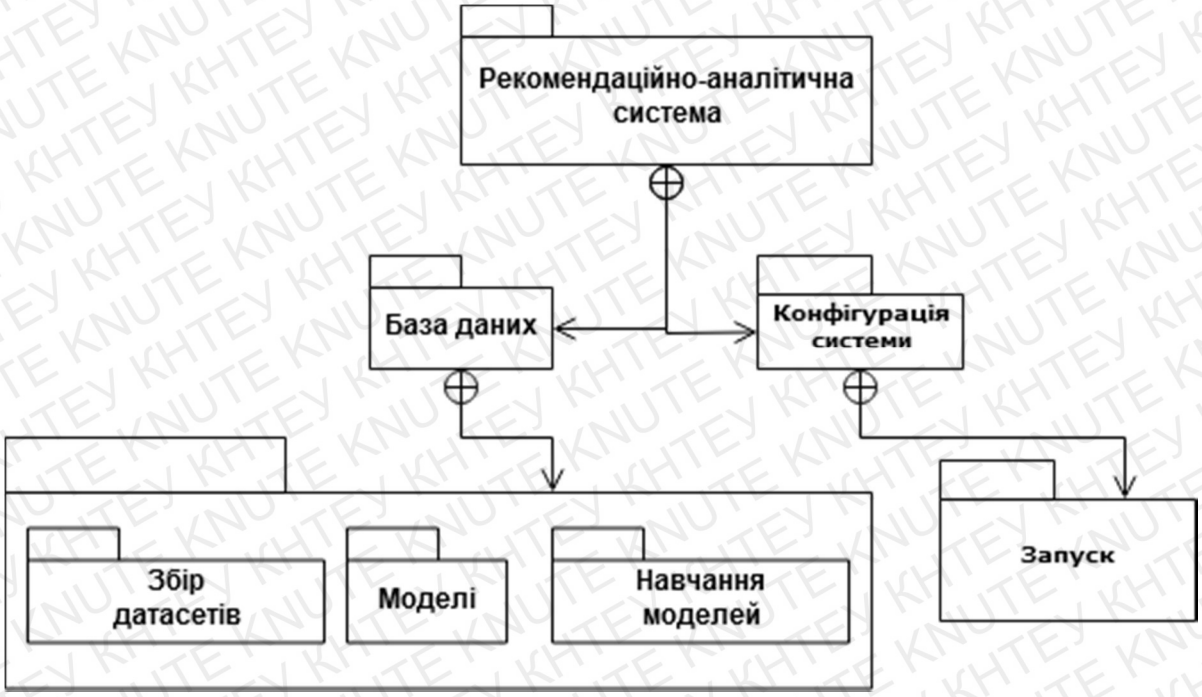

Рис. 3.3 Діаграма пакетів в форматі UML

Джерело:побудовано автором в системі "draw.io"

Діаграма пакетів дозволяє на більш високому рівні представити ієрархію системи.

В мові UML відношення пакетів може бути зображене за допомогою відрізків ліній, схожих на графічне зображення дерева. Найбільш загальний пакет або іншими словами контейнер зображується у верхній частині малюнку, а підпакети зображуються рівнем нижче. Контейнер з'єднується з підпакетами суцільною лінією, на кінці якої є спеціальний символ кола з

перехрестними лініями всередині, що примикає до контейнера. Даний символ нотації діаграми пакетів означає, що підпакети є частиною контейнера і, окрім цих підпакетів, контейнер не містить жодних інших пакетів [13].

3.2. Другий рівень архітектурної деталізації: розділення системи на підсистеми

У загальному розумінні більшість програм розділяються на підсистеми, схожі за функціональними ознаками.

Необхідно враховувати, що збільшення зв'язків між підсистемами ще називається "сильним спряженням", а в технічному завданні вказано, що система повинна мати "слабке спряження". З рівнем спряження пов'язано контроль рівня розгалуження — це контроль кількості звернень одного класу до інших класів.

Тому для розміщення підсистем використовуватиметься система контейнеризації "Docker", а для керування контейнерами система оркестрації "Docker Swarm". На рисунку нижче зображено еволюцію розгортання додатків у середовищі існування.

Від традиційного виду розгортання "одна програма на один фізичний сервер" з часом відбувся перехід на системи віртуалізації, що значно зменшило навантаження на фінансування закупівель апаратних засобів. Віртуальні машини дозволили встановлювати декілька віртуальних ОС на основну ОС встановлюючи вручну необхідну кількість апаратних ресурсів і більше того, вирішили багато проблем з безпекою, так як віртуальні машини ізольовані від основної ОС. Системи віртуалізації звісно допомогли вирішити проблему використання значного обсягу коштів на апаратні ресурси і питання з безпекою, але під кожну віртуальну машину необхідний час на встановлення ОС, більше того, віртуальні машини потребують багато пам'яті

у сховищі тільки для самої віртуальної ОС. Тому на зміну прийшло нове покоління рішень для зручного розгортання сервісів — це системи контейнеризації.

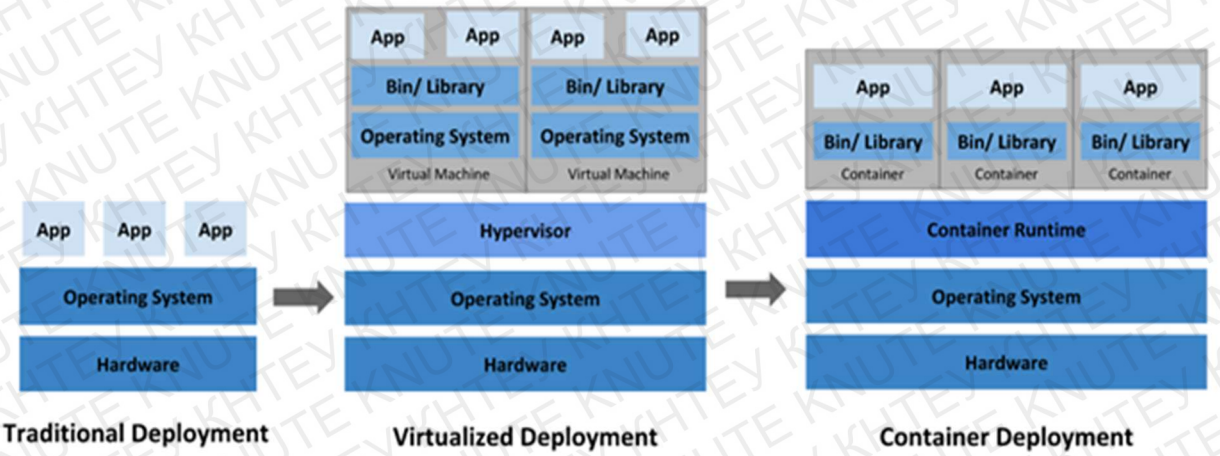

Рис. 3.3 Еволюція розгортання додатків

Джерело:побудовано на основі[20]

Даний тип систем потребує меншого простору в сховищі, так як всі розгорнуті додатки ізольовані в контейнерах, а контейнери працюють як окремі процеси в одній ОС. Виходячи з вищесказаного, з'ясувалось, що системи контейнеризації не потребують встановлення ОС для кожного окремого сервісу і більше того додатки безпечно ізольовані в контейнери в межах однієї ОС. Але основні досягнення систем контейнеризації — це легкість і зручність масштабування, балансування навантаження на фізичні ресурси.

Рекомендаційні системи часто базуються на машинному навчанні, а як відомо машинне навчання потребує значної кількості вхідних даних, а тому значно навантажує апаратні ресурси. Причому інформаційно-аналітичні системи часто знаходяться на одному і тому ж сервері, що й веб-сайт, а це вимагає великих зусиль на оптимізацію роботи і балансування навантаження на сервері. Тому, виходячи з вищевказаних можливостей систем

контейнеризації, вважається цілком доцільним розміщувати інформаційно-аналітичну систему в контейнери.

У окремих контейнерах розгорнуто:

- підсистема бази даних “SQLite”;
- середовище виконання python і необхідні пакети для роботи інформаційно-аналітичної системи;
- фреймворки для роботи з машинним навчанням;
- бек-енд фреймворки для роботи веб-сайту;
- веб-сервер для роботи додатку;
- модуль Git.

На рисунку нижче зображено концептуальну модель архітектури на рівні деталізації підсистем в контейнерах.

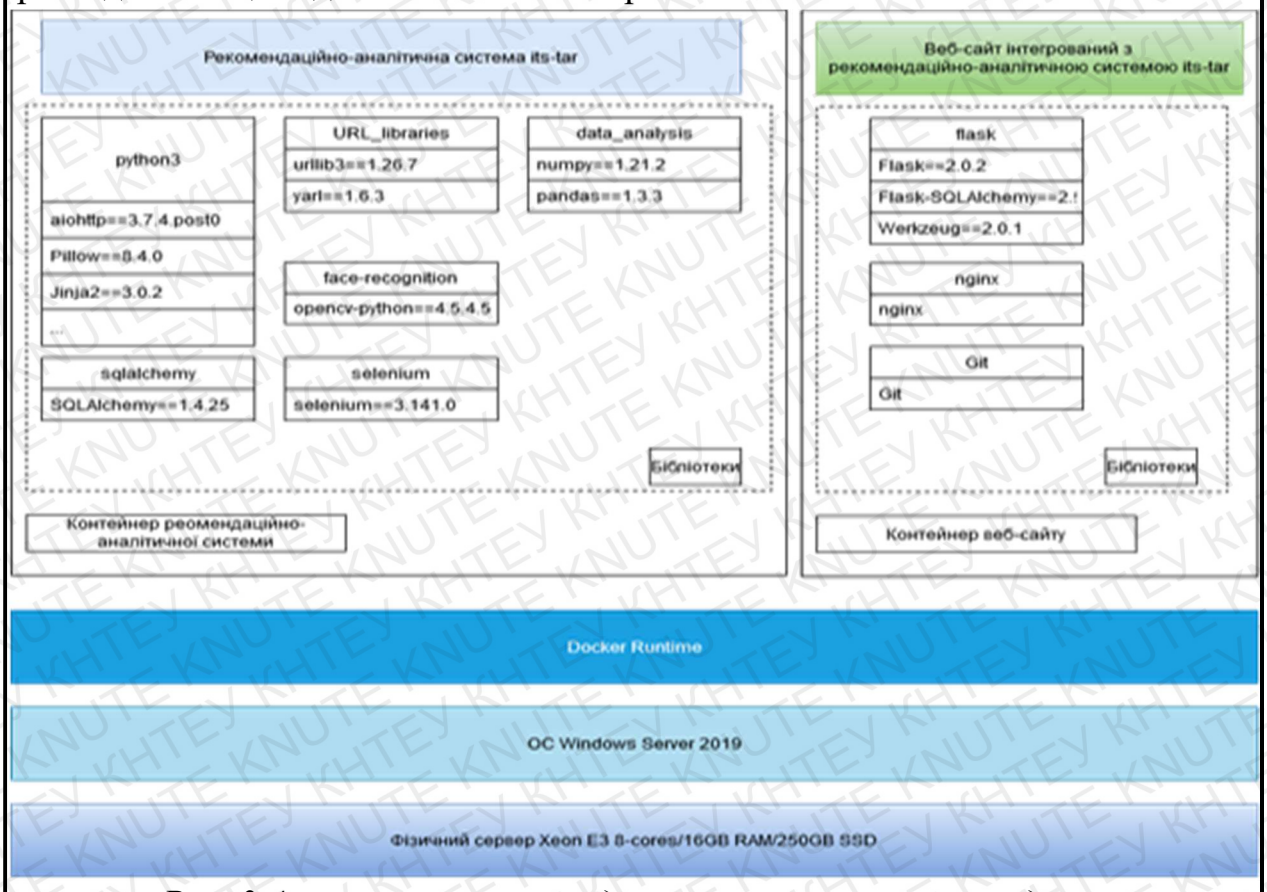

Рис. 3.4. концептуальна модель архітектури на рівні деталізації підсистем в контейнерах

Джерело:побудовано автором в системі “draw.io”

Інформаційно-аналітична система розміщена в контейнерах дозволяє не тільки балансувати навантаження на сервер, а також вирішує проблеми інтеграції з веб-сайтом.

Діаграма варіантів використання (use case diagram) – це вихідна концептуальна модель системи в процесі проектування і розробки, на якій зображуються відношення між зовнішніми агентами, або іншими словами акторами (actors) і варіантами використання [12].

Варіант використання (use case) – це опис загальних функцій модельованої системи без розгляду внутрішньої структури даної системи. Окремий варіант використання позначається на діаграмі за лопомогою еліпса, усередині якого міститься коротке ім'я варіанту використання. Кожен варіант використання відповідає окремому сервісу, який надає модельована система по запиту зовнішнього агента, тобто визначає один із способів використання системи. Сервіс, який ініціалізувався по запиту зовнішнього актора, має бути закінченою послідовністю дій. Це означає, що система повинна повернутися у вихідний стан, після того, як закінчить обробку запиту зовнішнього актора, в якому знову буде готова до виконання наступних запитів [12].

Діаграма варіантів використання містить кінцеву множину варіантів використання, які в цілому повинні визначати всі можливі сторони очікуваної поведінки системи. Зовнішній агент (actor) — це будь-який зовнішній по відношенню до модельованої системи елемент, який взаємодіє і використовує функціональні можливості системи. Зовнішнім агентом може бути людина, зовнішня база даних, програма або будь-яка інша система, яка служить джерелом дії на модельовану систему. Варіант використання визначає набір дій, що здійснюється системою при взаємодії із зовнішнім агентом. Стандартним графічним позначенням зовнішнього агента на діаграмах варіантів використання є фігурка "людини", під якою записується ім'я зовнішнього агента [12].

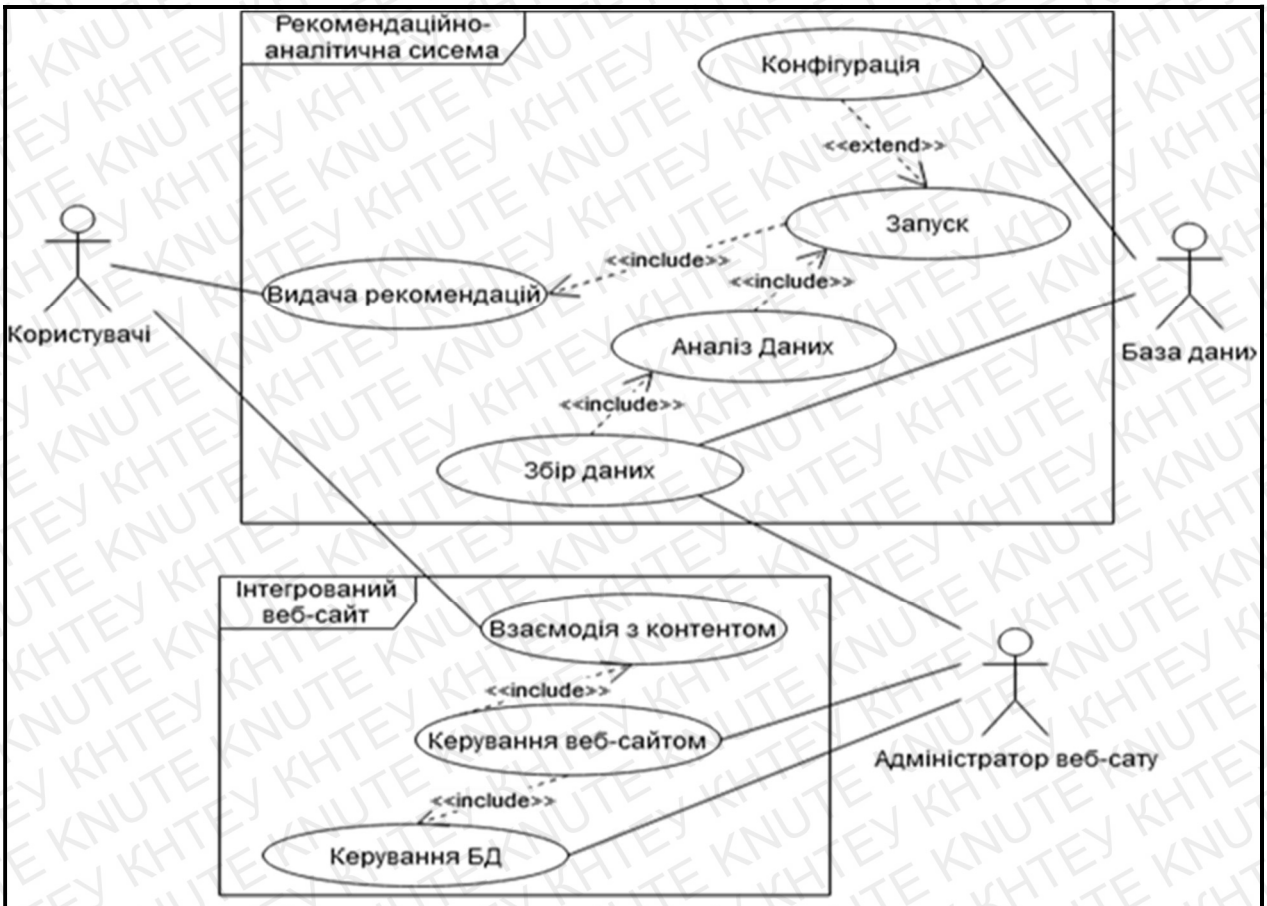

Рис. 3.5. Діаграма варіантів використання інформаційно-аналітичної системи особистісних вподобань користувачів

Джерело:побудовано автором в системі "draw.io"

На Рис.3.5. зображено діаграму варіантів використання інформаційно-аналітичної системи особистісних вподобань користувачів.

Включення (include) – це різновид відношення залежності між базовим варіантом використання і спеціальним випадком варіанту використання. Відношення включення встановлюється лише між двома варіантами використання і вказує на те, що задана поведінка для одного варіанту використання включається як складовий фрагмент в послідовність поведінки іншого варіанту використання [12].

На діаграмі варіантів використання, що зображена на Рис.3.5. зв'язки включення позначені у формі пунктирної лінії із стрілкою з надписом "<<include>>", котра направлена від базового варіанту використання до варіанту використання, котрий включається. Зовнішніми акторами

виступають користувачі, внітрішніми акторами вийступають база даних та адміністратор веб-сайту.

3.3. Третій рівень архітектурної деталізації: розділення підсистем на класи

Клас, або ж іншими словами компонент, можна розглядати як деяке збірне поняття складного об’єкта. Класи переносять сутності реального світу на мову програмування.

Класи — це абстрактний тип даних. Імена класів повинні бути іменниками і чітко відображати сутність об’єкта реального світу. Класи і об’єкти — це основа об’єктно-орієнтованого програмування. Клас можна порівняти з сукупність об’єктів одного виду. У кожного об’єкта є свій стан і поведінка — при описі об’єкта в класі відповідно задаються стан і поведінка. Хорошим тоном у написанні класів є розміщення до 7 сутностей, аби розробнику було легше запам’ятовувати сутності в класі [4].

При проектуванні програми обов’язково виявляються істотні та неістотні властивості об’єктів. Істотні властивості об’єкта — це основоположні елементи певного об’єкта, без котрих об’єкт не може функціонувати, або функціонуватиме неповністю. Неістотні — це властивості об’єкта, котрі доповнюють об’єкт. Наприклад якщо провести аналогію з автомобілем — істотні елементи об’єкта автомобіль це двигун, колеса, кузов, підвіска, сидіння, кермо так як без цього автомобіль не може функціонувати. Неістотні властивості об’єкта автомобіль — це колір кузова, ширина сидінь і тому подібне. Відповідно дану аналогію можна перенести на область програмування, виділити в кожному пакеті істотні класи в котрих описано бізнес логіку, для неістотних елементів наприклад графічний інтерфейс можна виокремити окремі класи користувацького інтерфесу, цей

спосіб допоможе легко змінювати програму при зміні вимог клієнта або при розширенні функціоналу програми.

При конструюванні класів використовують методології об’єктно орієнтованого програмування(ООП). ООП в загальному використовується для того, щоб перенести сутності з реального світу в віртуальний та якісно їх описати. Абстракція — це одна з шести методологій ООП для формування збірних понять про сутність реального світу. Абстрактні класи використовуються частіше для прототипування систем. Інкапсуляція — це методологія ООП, за використання якої приховуються деталі реалізації певних елементів програми. Наслідування — це методологія, котра допомагає позбавитись від багаторазового використання одного коду. Поліморфізм — це можливість динамічної підміни об’єктів під час виконання програми. Посилка повідомлень — це методологія ООП, котра визначає взаємодію програми з системними викликами, іншими словами взаємодію з API. Повторне використання — це методологія ООП, котра закликає розробляти програми або окремі модулі так, щоб їх можна було використати повторно.

Клас в мові UML зображується у вигляді прямокутника, в якому має бути вказане ім'я відповідного класу. В процесі опрацювання окремих компонентів діаграми опис класів доповнюється атрибутами і операціями класу. Четверта секція в прямокутнику є не обов'язковою і служить для розміщення додаткової інформації довідкового характеру, наприклад, про виключення або обмеження класу. Передбачається, що остаточний варіант діаграми містить якнайповніший опис класів, котрі складаються з трьох або чотирьох секцій [12].

В деяких випадках необхідно вказати, до якого пакету відноситься той чи інший клас. Наприклад при використанні однакових імен класів у різних

пакетах. Тому для цього використовується спеціальний символ роздільник – подвійна двокрапка і записується “Ім'я пакету::Ім’я класу” [12].

Навіть якщо секції атрибутів і операцій порожні, в позначенні класу порожні секції мають бути виділені горизонтальною лінією, для того, щоб відрізнити клас від інших елементів мови UML [12].

Під час проектування діаграм класів для інформаційно-аналітичної системи особистісних вподобань користувачів у першій секції вказано імена класів, у другій секції об’єкти класів, у третій секції вказані методи класів.

На Рис.3.6. зображено діаграму класів, котра описує класи для збирання датасетів даних. Відношення агрегації, позначене на Рис.3.6. між модулями parser_main.py та parser_imdb.py відповідно.

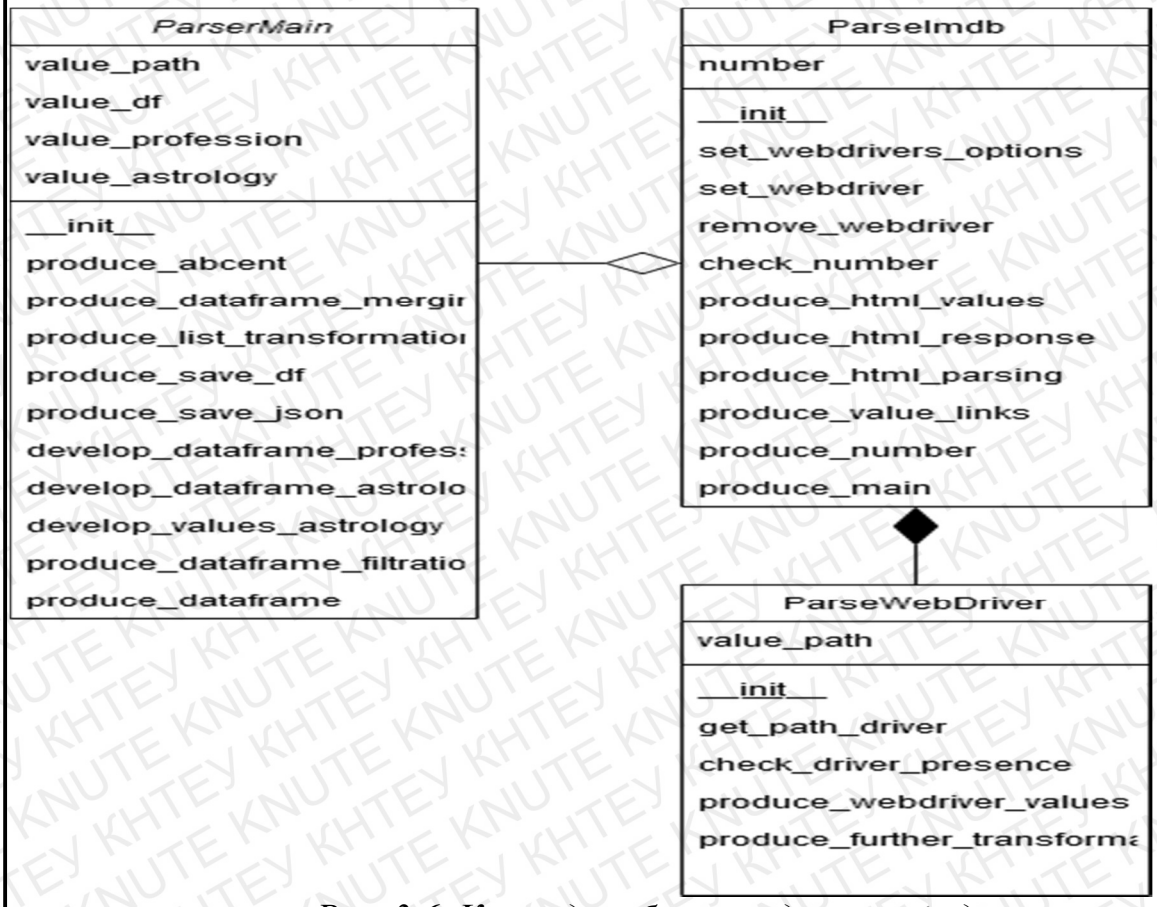

Рис. 3.6. Класи для збирання датасетів даних

Джерело:побудовано автором в системі “draw.io”

Агрегація (aggregation) – це спеціальна форма асоціації, яка служить для зображення відношення типу «від цілого до частини». Відношення

агрегації має місце між декількома класами в тому випадку, якщо один з класів включає в якості складових частини інші елементи [12].

Композиція (composition) – це різновид відношення агрегації, при якому складові частини цілого мають такий же час життя, що і саме ціле. При цьому типі відношення складові частини знищуються разом із знищенням цілого. Особливість даного взаємозв'язку полягає в тому, що частини не можуть виступати у відриві від цілого. Графічно відношення композиції зображається суцільною лінією, один з кінців якої є зафарбованим усередині ромбом. Зафарбований ромб вказує на той клас, який є класом цілої частини [12].

Відношення композиції, позначене на Рис.3.6. між модулями parser_webdriver.py та parser_imdb.py відповідно.

На Рис.3.7. вказано класи для роботи з моделями.

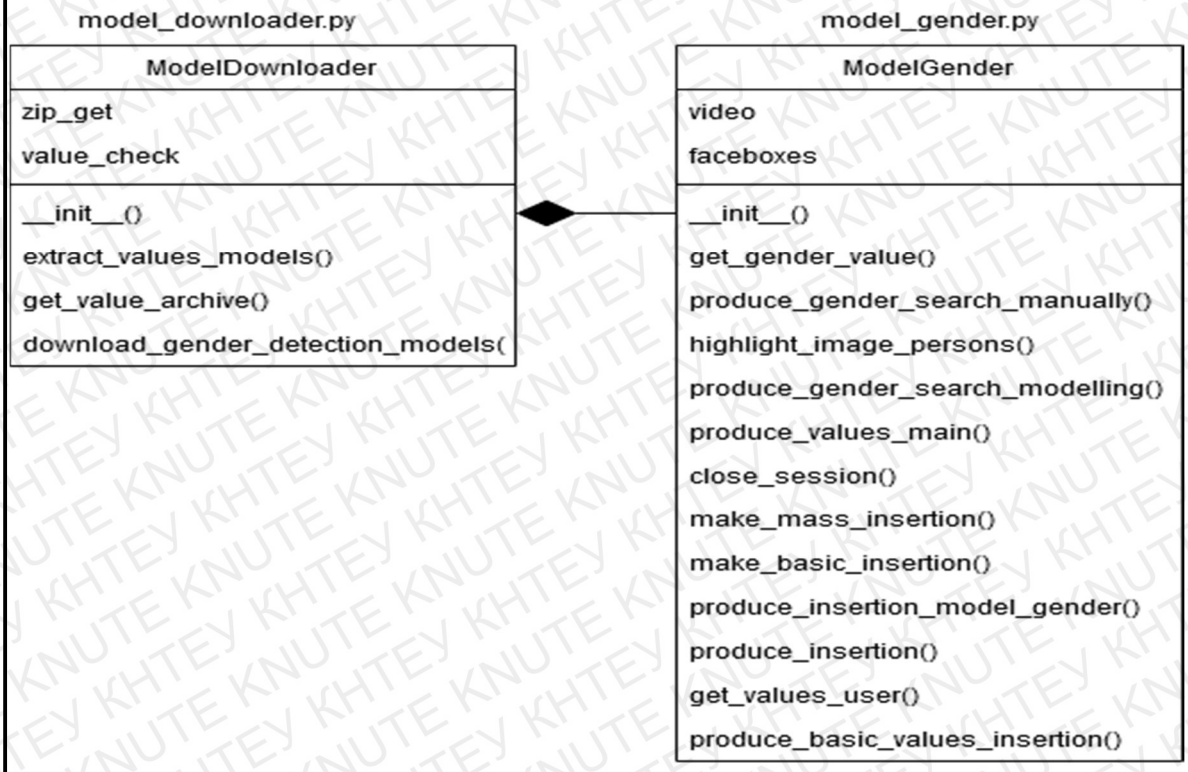

Рис. 3.7. Модулі роботи з моделями

Джерело:побудовано автором в системі "draw.io"

Клас model_downloadaer.py завантажує натреновані за допомогою машинного навчання моделі розпізнавання статі та приблизного віку по

фотографії користувача. Відношення композиції зображено між модулями, так як модель визначення статі і віку по фотографії користувача model_gender.py не працюватиме якщо модуль model_downloader.py не завантажить натреновані моделі.

На Рис.3.8. зображено діаграму класів модулю config.py для конфігурації інформаційно-аналітичної системи.

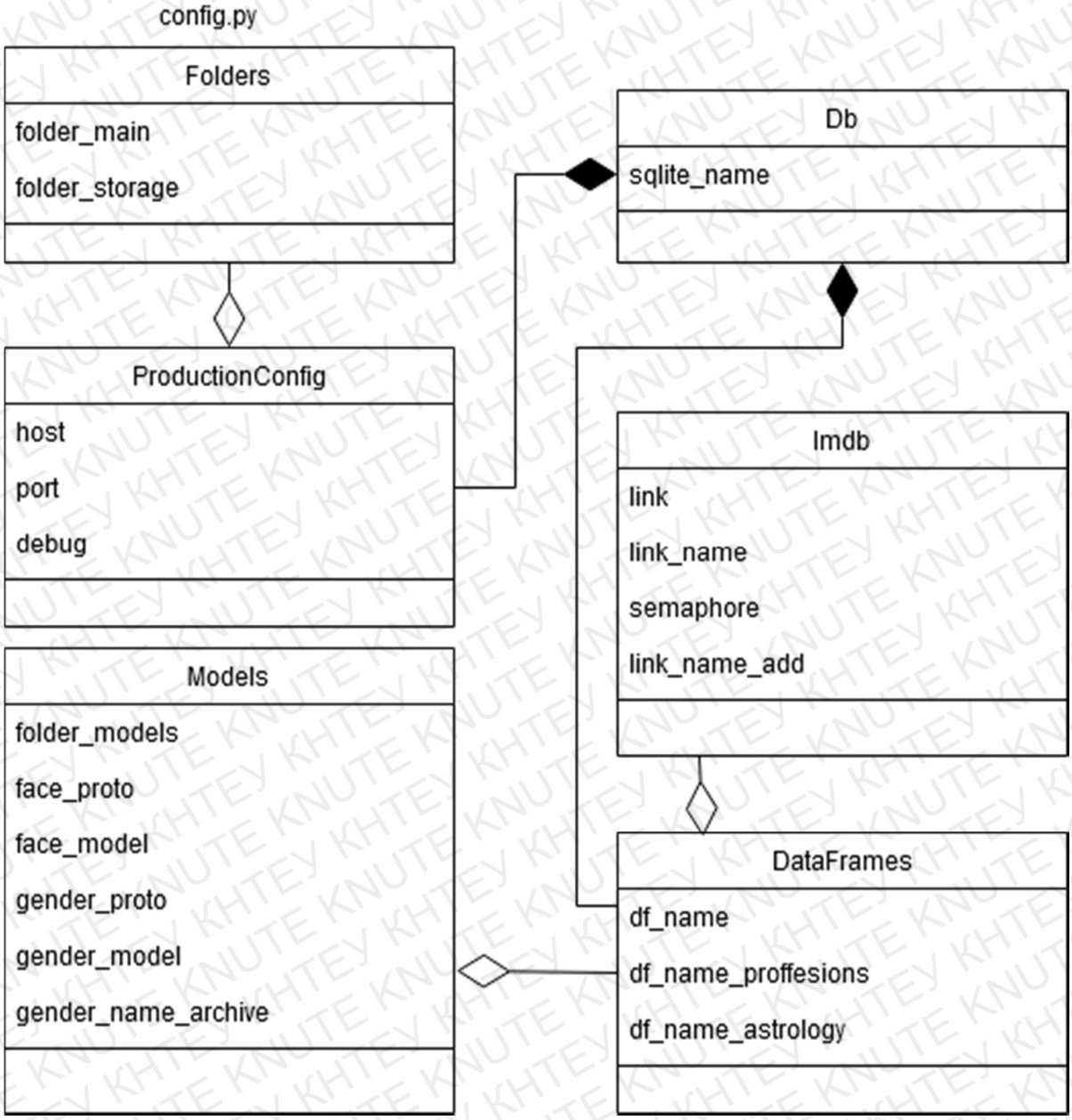

Рис. 3.8. Модуль конфігурації системи

Джерело:побудовано автором в системі "draw.io"

Відношення агрегації відображено між класами folders та production_config так як в файлі конфігурації має міститися конфігурація для структури директорій проекту.

Відношення композиції позначено між класами db та production_config, так як конфігурація бази даних зберігається в класі production_config.

Також відношення композиції зображено між класом folders та production_config, так як модель визначення статі і віку по фотографії користувача model_gender.py не працюватиме якщо.

Клас dataframes є частиною класу models, тому між ними позначене відношення агрегації.

Клас dataframes залежить від класу db так як дані характеристик користувачів необхідно зберігати в базі даних, тому між даними класами позначено відношення композиції.

3.4. Четвертий рівень архітектурної деталізації: розподіл класів на метод за функціональними особливостями

Наразі методи, в багатьох мовах програмування називаються функціями — це та одиниця програмування, котра задає поведінку об’єктам. Методи в першу чергу служать для зниження складності програми.При моделюванні функціонування системи виникає необхідність деталізувати особливості алгоритмічної і процедурної реалізації виконуваних системою операцій. Для цього, як правило, використовуються блок-схеми або структурні схеми алгоритмів. Кожна така схема акцентує увагу на послідовності виконання певних процедур або елементарних операцій, які в сукупності приводять до отримання бажаного результату. Для моделювання

процесу виконання операцій в мові UML використовуються діаграми діяльності [12].

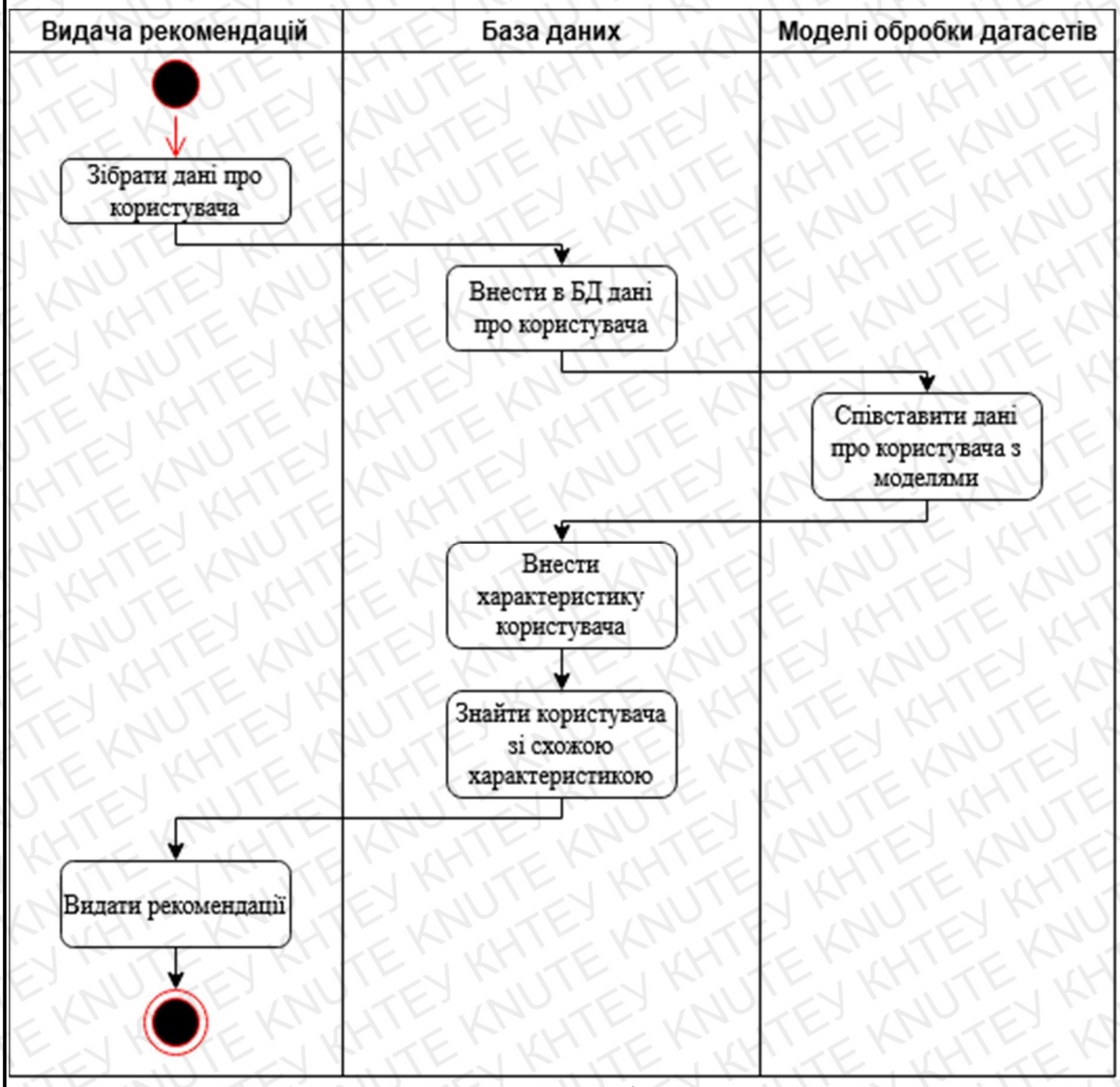

Рис. 3.9. Діаграма діяльності інформаційно-аналітичної системи "its-tar"

Джерело:побудовано автором в системі "draw.io"

На діаграмі діяльності з Рис.3.9. визначено окремі доріжки для кожного етапу діяльності інформаційно-аналітичної системи. Зафарбоване коло вгорі доріжки "Видача рекомендацій" означає початок роботи системи, а напівзафарбоване коло означає кінець діяльності системи. Стрілки на даній діаграмі позначають перехід від однієї дії до іншої по порядку.

3.5. Висновки до розділу 3

В даному розділі було розроблено архітектуру інформаційно-аналітичної системи особистісних вподобань користувачів. Розбиття архітектури на рівні деталізації допомогло полегшити розробку архітектури, полегшило розуміння архітектури. Розробка архітектури подібним методом дозволяє швидко орієнтуватись у необхідних для конструювання компонентах, допомагає розібратись у предметній області програмної системи.

Автором в даному розділі було розроблено діаграми в форматі UML, так як даний тип нотації стандартизований і використовується компаніями вже на протязі кількох десятиліть, та постійно оновлюється. Діаграми відображають особливості реалізації інформаційно-аналітичної системи на кожному рівні деталізації архітектури.

РОЗДІЛ 4
ФОРМУВАННЯ БАЗИ ДАНИХ ТА РОЗРОБКА АЛГОРИТМУ ІНФОРМАЦІЙНО-АНАЛІТИЧНОЇ СИСТЕМИ

4.1. Конвенції програмування для інформаційно-аналітичної системи

Конструювання ПЗ — це етап розробки проекту, де узгоджуються основні конвенції програмування для даного типу проекту, обирається мова програмування, розробляється архітектура модулів. Найперше, що треба контролювати при конструюванні ПЗ — це версії інструментів розробки. Інструменти розробки необхідно уніфікувати для того, щоб не було проблем з запуском програми у інших членів команди розробки. Навіть при розробці однією людиною необхідно використовувати найбільш підходящі на даний час інструменті розробки. Це робиться через те, що найновіші інструменти ще не підтримують багато сторонніх модулів. Під час конструювання програми розробники повинні розуміти предметну область проекту, так як незнання предметної області призводить до розробки неправильної програми.

В другу чергу необхідно вимірювати кількісні та якісні показники на кожному етапі розробки для відстеження успішності проекту і прогрессу [11].

Форматування є надважливим елементом програмування так як теорема форматування зазначає — хороше візуальне форматування показує логічну структуру програми. Хороші програмісти пишуть код, котрий добре зрозуміти людям, котрі цей код будуть супроводжувати [3].Описувати мету блоку коду в коментарях — це найперша мета коментування коду. Описувати

чому, а не як. Коментарі повинні бути без скорочень чи з мінімальним числом скорочень. Коментувати діапазони значень методу, коментувати початок і завершення складного блоку коду для виконання однієї задачі. Щодо коментування методів варто відзначити необхідність написання коментарів цілі методу близько до об'явлення самого методу, коментування параметрів методу та мети кожного параметру. Щодо коментування класів важливо коментувати інтерфейси класу, конструктори класу, паттерни з допомогою яких спроектовано клас, також можна вказувати в коментарях класів контакти автора класу і юридичну інформацію [3].

При виборі мови програмування необхідно дотримуватись старого правила — використання підходящої для даного типу проекту мови програмування, з котрою знайомі програмісти сприяє підвищенню продуктивності команди розробки майже вдвічі [3].

Для даного проекту узгоджено такі конвенції: використовувати ООП, у Python хорошим тоном вважається ставити 5 пробілів заміть клавіші Tab під час написання конструкції if-else, використовувати , бібліотеки для роботи з даними, використовувати back-end бібліотеки мови Python для інтеграції веб-сайту Flask, на кожному етапі розробки тестувати роботу модулів так як система працюватиме з великою кількістю даних.

Системи контролю версій стали невід'ємною частиною сучасної розробки. Тому проект має бути підключено до системи контролю версій Git. У файлі ".gitignore" має бути вказано директорії, котрі не будуть поміщатись в проект на Github.

Псевдокод — це дуже потужна методика програмування, так як допомагає представити конструювання класів на більш високому рівні не вдаючись в особливості деталей реалізації. Програма з псевдокодом значно простіше супроводжується ніж програма з детальною документацією [3]. Більшість класів і методів інформаційно-аналітичної системи

користувацьких вподобань спочатку було описано на псевдокоді. У псевдокоді виділено передумови і задача, котру виконуватиме клас чи метод, вирішується як тестуватиметься програма, виділяється який функціонал реалізовуватиметься за допомогою стандартних бібліотек.

Написання блоку коду в коментарях відкривається трьома відкриваючими лапками і закривається трьома закриваючими лапками [24].

```
68  def develop_database(self):
69      """
70      Method which is dedicated to produce the database it that cases
71      Input:  None
72      Output: we created values from the db_creator file and produced
73      the database
74      """
75      try:
76          Base.metadata.create_all(self.engine)
77      except Exception as e:
78          print(f"We faced problems with Base: {e}")
79          print('----------------------------------------')
80
```

Рис. 4.1. Приклад використання псевдокоду в модулі "db_main.py"

Джерело:побудовано автором в системі " Sublime text 3"

Інкапсуляція необхідна при розробці програми — це приховування деталей реалізації програми [15]. Якщо провести аналогію того як працює інкапсуляція, можна уявити, що є айсберг, вершина котрого є програмний інтерфейс який використовує користувач, а реалізація програми це нижня частина айсберга, котра прихована під водою.

Для полегшення супроводу класів необхідно очищати автозгенеровані методи, тобто ті які генеруються середовищем розробки автоматично. Також якщо один параметр передається в декілька методів, правильно в такому разі об'єднати методи в клас, так як значно швидше робити зміни в одному місці замість декількох. Якщо в класі є дані, але не описана поведінка об'єктів то це є незавершений клас [3].

Для зручності написання і супроводу методів необхідно керуватись такими правилами написання методів:

• задавати правильні імена методів, краще всього у назві методу коротко написати що даний метод виконує, а не те як виконує;

• документувати метод — писати короткі коментарі того, що робить метод;

• правильно відформатований;

• метод не змінює глобальні змінні;

• цілі методу розмиті;

• метод не захищений від отримання неправильних вхідних даних [3].

Методи повинні бути покриті Unit тестами хоча б на 30% [14].

Передавати параметри в методи краще в такому порядку:

• вхідні значення;

• змінні значення;

• вихідні значення [5].

Мінімум звернень методами до глобальних змінних грає велику роль, так як глобальну змінну може використовувати інший метод, і відповідно змінювати дані оброблені попереднім методом. Ізоляція нестабільних елементів — це вежливий момент при проектуванні і розробці [5].

Об’явлення змінної — це процес задання імені та типу змінної. Ініціалізація змінної — це присвоєння значення змінній. Виходячи з вищевказаного, так як змінні містять в собі дані, роботі зі змінними необхідно приділяти особливу увагу. У змінних містяться дані. Змінні у коді краще за все ініціалізувати одразу після створення. Повторне використання змінних в одному методі формує вікно вразливості між даними використаннями змінних. Повторне використання змінної в одному методі спричиняє зміну значення змінної. Тому краще за все задавати одну змінну для виконання однієї задачі, хоча і повторна ініціалізація не є помилковою. Для прикладу можна привести змінну, яка задана вгорі методу на двісті строк

коду, а сам код, котрий використовує змінну знаходиться вгорі, коли інший програміст проводить рефакторинг коду то може помітити, що змінна не задана для даної області коду і створить змінну, але не візьме до уваги, що змінна вже задана в іншій області коду, відповідно змінна буде задана двічі і програма видасть помилку [3].

Ще однією важливою деталлю є задання змінних для циклічних конструкцій у самому циклі. У більшості мов програмування у циклі “for” змінна записується в самій умові циклу. Хоча не є помилкою об’явлення змінної перед циклом і вже ініціалізація змінної в самому циклі.

Імена змінних повинні бути максимально конкретизовані, наприклад змінна “дата” не повністю конкретизує область для якої буде використовуватись, краще назвати “датаПочаткуРоботи” як приклад, так як дане ім’я більш конкретне розуміння того, для чого ця змінна використовуватиметься. Але занадто довгі імена змінних можуть незручно читатись, як і занадто короткі імена не будуть давати достатнього уявлення про об’єкт. В циклах невеликого розміру можна задавати короткі однобуквенні іменна змінних, наприклад “i,j,k”, а в більш великих циклах і вкладених циклах задавати короткі але більш конкретні імена змінних.

Масиви — це структуровані набори даних. При роботі з масивами необхідно враховувати кількість елементів масиву і не звертатись до виходячих за діапазон значень масиву. Також у багатовимірних масивах необхідно повністю розуміти до якого саме масиву йде звернення [3].

Умовні оператори — це більша частина коду. В умовному операторі if-else правильну умову необхідно задавати в блоці if, а неправильно залишати на блок else.

Цикли дуже часто використовуються при конструюванні. Один цикл в ідеалі повинен виконувати одну задачу. В циклах не варто використовувати оператора goto і break, маніпуляції з лічильником циклу повинні бути

заборонені. лічильник цикла повинні бути цілочисловими без плаваючої точки. Називати осмислено імені лічильників циклів а не просто i,j. У циклах не варто робити більше 3 вкладених циклів, так як це ускладнює роботу цикла і приводить до потенційних помилок [3].

Необхідно писати відкриваючі і закриваючі дужки одразу,а потім всередині дужок писати умову для того, щоб не натрапити на помилку через непоставлену дужку.

Відладка і тестування ПЗ займають більше 50 відсотків всього циклу розробки ПЗ. Але інспекція коду, або ж “Code review” у багато разів зменшує витрати часу порівняно з тестуванням, зменшує затрати на проект.

Багатато тестування не виправить помилок, якщо привести аналогію — постійне тестування те саме що намагатись зменшити вагу об’єкта за рахунок частого звішування. Тому власне інспекція коду, написання правильного коду має значно більше значення, ніж просто багато тестування для виходу якісного продукту. Розробники під час розробки програми більшу частину часу використовують “Smoke Testing” – це тестування того чи запускається взагалі програма, та “Critical Path Testing” – це перевірка того чи працює мінімально необхідний набір функцій програми [14].

Самодокументований код — це ідеал всіх ідеалів у програмуванні. Приклади самодокументованого коду: назви змінних котрі відображають область застосування змінної, імена методів відображають що робить даний метод, оператори названо по аналогії з об’єктами реального світу. В хорошому коді подібне використовується, але в реальності досягти ідеалу завдання неможливе, можливо тільки приблизитись до хорошого коду. Також відмовитись від коментування в реальності теж неможливо, так як не всю суть програми можна пояснити в іменуванні елементів мови програмування. Більше того, коментарі допомагають орієнтуватися в логічній структурі програми.

4.2. Створення бази даних інформаційно-аналітичної системи

У будь-якій системі є своя база даних. У випадку інформаційно-аналітичної системи особистісних вподобань користувачів створено базу даних користувачів. База даних інформаційно-аналітичної системи має містити не тільки статичні дані про характеристики користувачів на основі яких генеруватимуться рекомендації, а ще мати проміжний прошарок для зберігання тимчасових даних про користувачів, зібраних з відкритих джерел.

Факти – це дані, що відображають уподобання користувача. Коли ми говоримо про збирання фактів, мається на увазі збір інформації про події та дії, які є ключами до розуміння смаків користувача. У більшості книг про рекомендаційні системи описуються алгоритми та способи оптимізації цих алгоритмів. На старті вже має бути великий обсяг даних, необхідних для заповнення алгоритму. До цих даних входить каталог реальних користувачів. Для того, щоб зібрати необхідний набір фактів, потрібно чимало часу та зусиль. Відома метафора програмістів “Сміття на вході – сміття на виході” дуже актуальна для рекомендаційних систем і машинного навчання в цілому. На жаль, дані, які підходять для однієї системи, можуть не підійти для іншої. Як правило, від користувачів системи отримують зворотний зв'язок двох типів: явну (оцінки або лайки) і неявну (активно фіксується в процесі спостереження за користувачем). Користувач може дати явний зворотний зв'язок у вигляді заповнених анкет або лайків [1].

На Рис. 3.6. у розділі 3 було зображено і описано діаграму класів підсистеми збору даних. Підсистема збору даних записуватиме в файли типу “.csv” зібрані датасети про користувачів з відкритих джерел. Вихідний код даної підсистеми розміщено в додатках А, Б, В.

Для постійного зберігання даних і фактів про користувачів системи використовуватиметься база даних “SQLite”.

На Рис. 4.2. зображено логічну модель бази даних інформаційно-аналітичної системи особистісних вподобань користувачів.

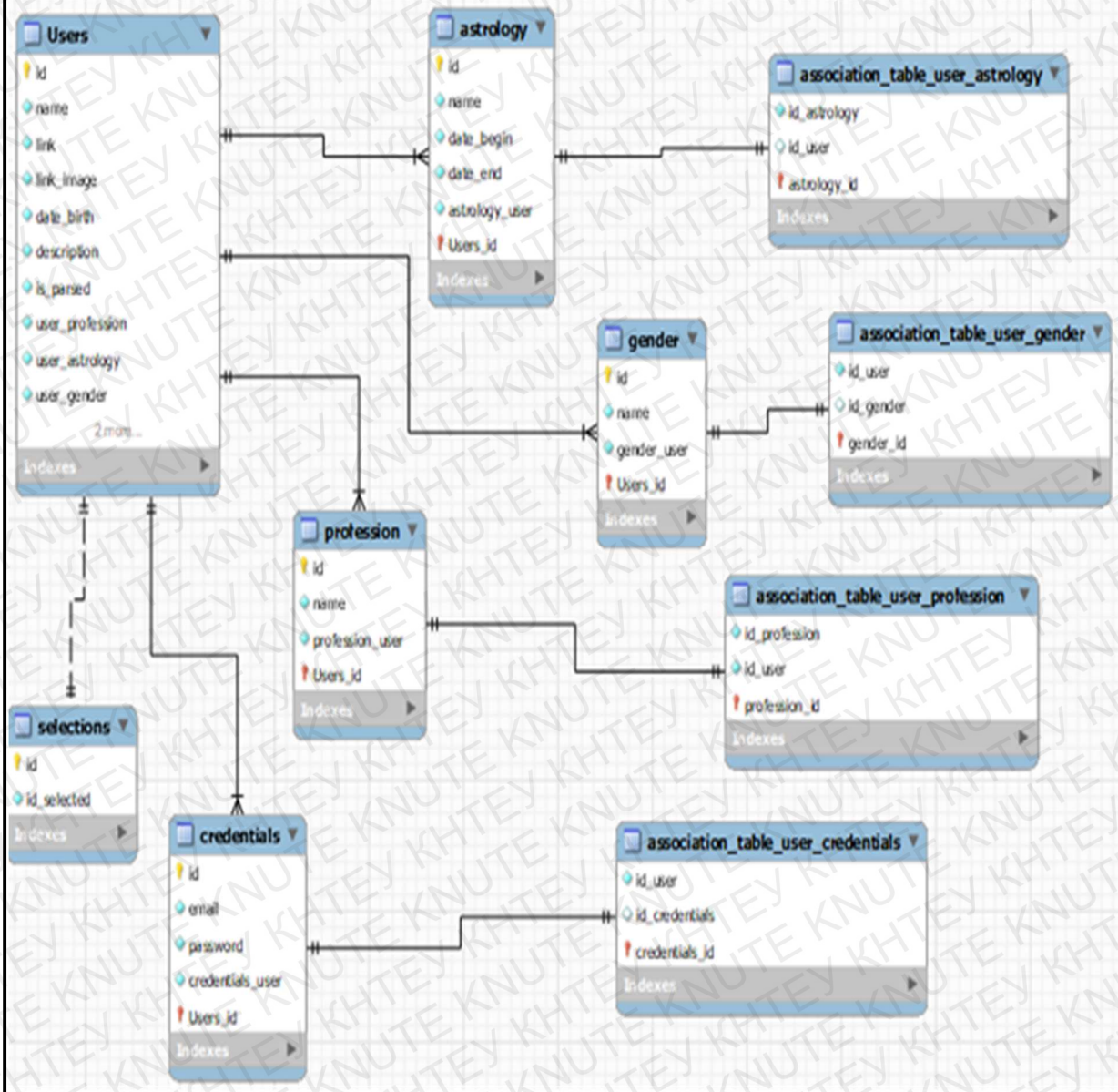

Рис. 4.2. Логічна модель бази даних

Джерело:побудовано автором в системі "MySQL Workbench"

Фізична модель бази даних більш повно відображає базу даних. На Рис. 4.3. зображено фізичну модель бази даних.

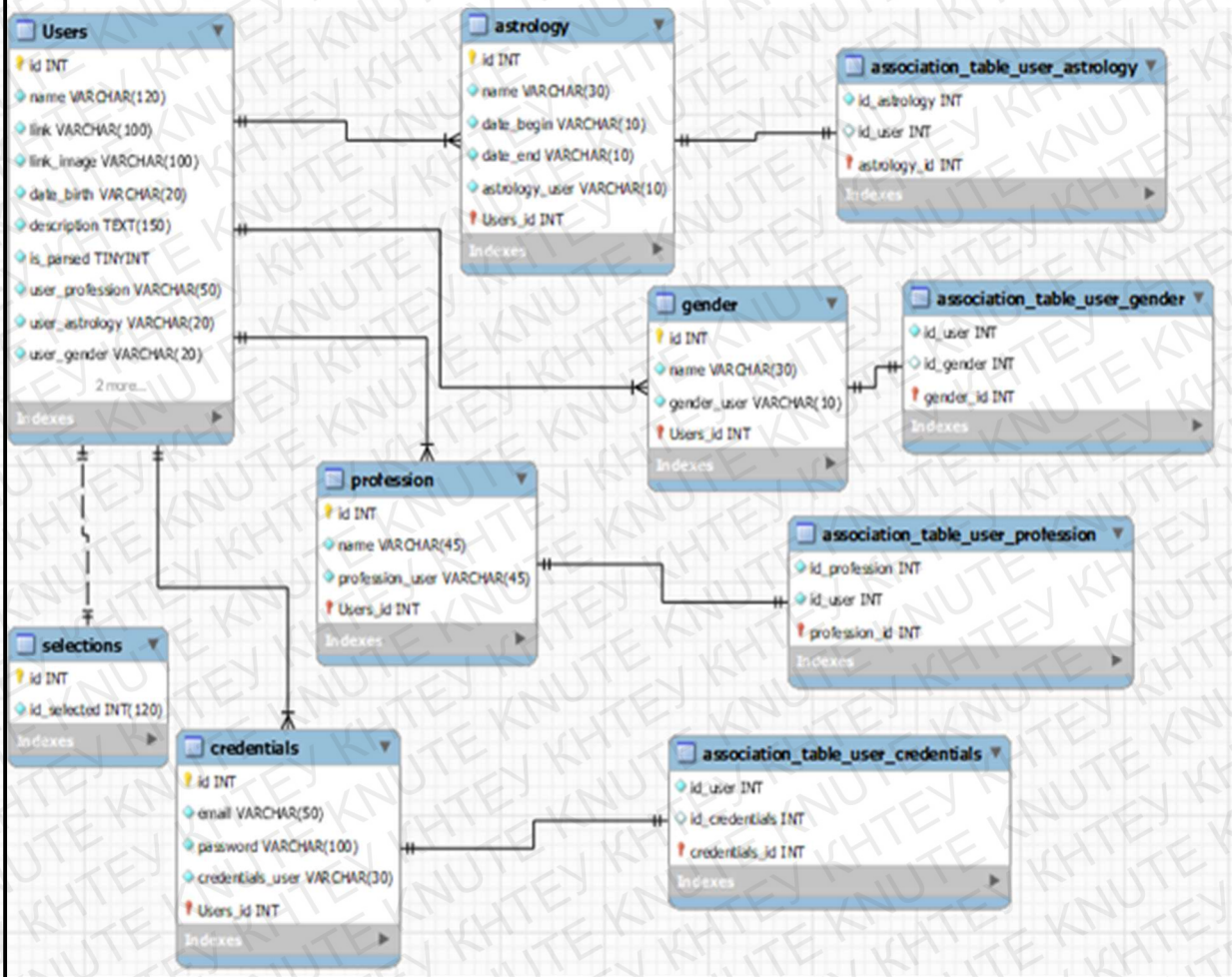

Рис. 4.3. Фізична модель бази даних

Джерело:побудовано автором в системі "MySQL Workbench"

У фізичній моделі бази даних відображено типи даних для кожної колонки та найбільш підходящі кількості символів в колонці з урахуванням особливостей інформаційно-аналітичної системи.

4.3. Створення моделей рекомендаційної системи

Моделі є елементами, на основі яких рекомендаційна система надаватиме рекомендації. Моделі визначення статі та віку по фотографії було створено на основі машинного навчання з допомогою відкритих джерел та бібліотек мови програмування python.

Система розпізнавання віку та статі користувача працює за методом “чорного ящика” як і більшість сучасних систем розпізнавання образів [16].

Нейронна мережа базується на основі бібліотеки opencv мови програмування python. Вагові коефіцієнти синапсів нейронної мережі було визначено з допомогою типових рішень у бібліотеках для машинного навчання мови програмування python. У випадку інформаційно-аналітичної системи “its-tar” задані вагові коефіцієнти мають такі значення:

w1=78.4263377603;

w2=87.7689143744;

w3=114.895847746.

Дані коефіцієнти задано у модулі model_gender.py котрий розміщено в додатку Є.

4.4. Створення алгоритмів взаємодії компонентів системи

Серце системи – це модуль конфігурації. Даний модуль описує взаємодію всіх компонентів інформаційно-аналітичної системи. У файлі “config.py” розроблено взаємодію компонентів системи між собою.

В даному модулі вказано класи для обробки даних, котрі отримані на основі моделей датасетів. Об’єкт user_numbers задає кількість вхідних точок даних для нейронної мережі. Клас Db описує місце для зберігання бази даних користувачів інформаційно-аналітичної системи. Клас ProductionConfig описує конфігурацію інформаційно-аналітичної системи для роботи з веб-

серверомю Клас DataFrames записує дані по кожному з датасетів в файли формату ".csv". Клас Models вказує шляхи до кожної з моделей датасетів.

config.py

```python
import os
from attr import dataclass
from dotenv import load_dotenv


load_dotenv()

link_webdriver = os.getenv("LINK_WEBDRIVER")
link_model_gender = os.getenv("LINK_MODEL_GENDER")

default_chunk = 10
user_numbers = 10000

@dataclass
class Db:
    sqlite_name = 'sqlite.db'
    echo = False

@dataclass
class ProductionConfig(object):
    host = '0.0.0.0'
    port = 5001
    debug = True
    # SECRET_KEY = ''

@dataclass
class Models:
    folder_models = 'models'
    face_proto = 'opencv_face_detector.pbtxt'
    face_model = 'opencv_face_detector_uint8.pb'
    gender_proto = 'gender_deploy.prototxt'
    gender_model = 'gender_net.caffemodel'
    gender_name_archive = 'gad.zip'

@dataclass
class Folders:
    folder_main = os.getcwd()
    folder_storage = 'storage'

@dataclass
class Imdb:
    link = 'https://www.opendb.com'
    link_name = 'name'
    semaphore = 10
    link_name_add = 'nm'

@dataclass
class DataFrames:
    df_name = 'names_db.csv'
    df_name_proffesions = 'professions.csv'
    df_name_astrology = 'astrology.csv'

dictionary_gender = {
    'Male': 1,
    'Female': 2,
    'Unknown': 3,
}

dictionary_astrology = [
```

Рис. 4.2. Модуль конфігурації проекту

Джерело:побудовано автором в системі "Sublime text 3"

Архітектуру модулю конфігурації було зображено на Рис.3.8. у розділі 3. Модуль конфігурації config.py знаходиться в додатку З.

4.5. Розгортання проекту

В даному пункті описується керівництво користувача. Майбутній користувач має керуватися нижчевказаною інструкцією для розгортання проекту. Для розгортання проекту на сервері у середовищі Docker повинен бути розгорнутий контейнер ubuntu з встановленим ”Python” версії 3. Для цього в ОС сервера необхідно відкрити командний рядок і ввести команду “docker pull Ubuntu” для встановлення контейнера і запустити контейнер за допомогою команди “docker run ubuntu” [25].

Найступним кроком є встановлення python. Для цього необхідно запустити командний рядок контейнера і виконати команду “sudo apt-get install python3”.

Після вищевказаного в контейнері необхідно встановити менеджер пакетів для мови програмування python котрий називається “pip”. Це робиться за допомогою команди “python install pip”.

Далі необхідно перейти в кореневу папку з проектом під назвою “its-tar_production” за допомогою командного рядка контейнера. Перейшовши в директорію з проектом спочатку необхідно створити віртуальне середовище для проекту. Це робиться за допомогою команди “ python -m venv venv/Scripts/activate”.

Після розгортання віртуального сереловища за допомогою менеджера пакетів “pip” необхідно встановити такі бібліотеки:

- aiohttp==3.7.4.post0
- async-timeout==3.0.1
- attrs==21.2.0
- beautifulsoup4==4.10.0

- bs4==0.0.1
- certifi==2021.5.30
- chardet==4.0.0
- charset-normalizer==2.0.6
- click==8.0.1
- fake-useragent==0.1.11
- filelock==3.3.1
- Flask==2.0.2
- Flask-SQLAlchemy==2.5.1
- gdown==4.2.0
- greenlet==1.1.2
- idna==3.2
- itsdangerous==2.0.1
- Jinja2==3.0.2
- lxml==4.6.3
- MarkupSafe==2.0.1
- multidict==5.1.0
- numpy==1.21.2
- opencv-python==4.5.4.58
- pandas==1.3.3
- Pillow==8.4.0
- PySocks==1.7.1
- python-dateutil==2.8.2
- python-dotenv==0.19.0
- pytz==2021.1
- requests==2.26.0
- selenium==3.141.0

- six==1.16.0
- soupsieve==2.2.1
- SQLAlchemy==1.4.25
- tqdm==4.62.3
- typing-extensions==3.10.0.2
- urllib3==1.26.7
- Werkzeug==2.0.1
- yarl==1.6.3

Встановити бібліотеки можна за допомогою файлу “requirements.txt” вказавши шлях менеджеру пакетів “pip” командою “pip install -r requirements.txt”. І для запуску проекту необхідно повернутися в кореневу директорію з проектом і виконати команду “python start.py”.

4.6. Висновки до розділу 4

В даному розділі на самому початку було встановлено конвенції програмування для більш ефективного конструювання ПЗ. Це допомогло одразу окреслити необхідні правила для етапу конструювання. В конвенціях програмування було досліджено і затверджено кращі методики написання елементів коду.

Було описано створення бази даних інформаційно-аналітичної системи особистісних вподобань користувачів, розроблено логічну і фізичну моделі бази даних.

Автором було описано методи і засоби для роботи машинного навчання у інформаційно-аналітичній системі.

Також було представлено порядок розгортання системи і необхідні для цього ресурси.

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

Зниження складності — це головний технічний імператив розробки ПЗ. Чим простіше розробка ПЗ, тим швидше програмна система з’явиться на світ і швидше принесе користь користувачам, а виробникам прибуток. Використання шаблонів проектування пришвидшує розробку і покращує якість ПЗ.

Метафори дозволяють проводити аналогії складних речей з області розробки ПЗ у ситуації з повсякденного життя. Використання метафор дозволяє полегшити розуміння проекту не тільки командою розробки, а й розуміння замовником того, як система будується і що буде робити.

Програмування — це процес автоматизованого вирішення проблем в рамках певної предметної області. Більше того, концентрація уваги на предметній області покращує розуміння і використання принципів ООП, адже власне ООП побудовано на використанні властивостей і поведінки об’єктів реального світу.

У ході дослідження з’ясувалося, що рекомендаційні системи увійшли в життя більшості користувачів інтернету, адже у більшості випадків такі системи застосовуються у домашньому використанні. Нейронні мережі на котрих базуються інформаційно-аналітичні системи стали сучасним трендом розробки програмного забезпечення.

Розробка архітектури інформаційно-аналітичної системи завдяки розкладанню на рівні детелізації значно допомогла знизити складність розробки такого складного ПЗ як інформаційно-аналітична система особистісних вподобань користувачів на базі машинного навчання.

Також було з’ясовано, що використання систем контейнеризації є сучасним рішенням. Адже наразі будь-який проект прагне за меншу кількість ресурсів і часу мати систему, котра буде добре захищена, збалансована і що найголовніше вимагатиме менше фізичних ресурсів для розміщення проекту.

Під час дослідження було з’ясовано, що як великі так і малі компанії використовують конвенції програмування для стандартизації процесу крнструювання ПЗ всередині компанії. Тому для полегшення супроводу системи і покращення якості коду було розроблено конвенції програмування під проект інформаційно-аналітичної системи особистісних вподобань користувачів з урахуванням досвіду великих компаній.

Розробку інформаційно-аналітичної системи було втілено завдяки використанню типових рішень і бібліотек як основи при написанні базових модулів котрі не стосуються ідеї продукту. Використання такої методики є поширеною практикою у компаніях будь-якого розміру на сьогоднішній день.

У ході дослідження усі поставлені завдання були вирішені. Мету розробки інформаційно-аналітичної системи особистісних вподобань користувачів на базі машинного навчання було досягнуто.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ким Фальк. Рекомендательные системы на практике./ К.Фальк – Москва: ДМК, 2020 – 270с.

2. John Riel, Joseph A. Konstan., Eric Vrooman. Word of Mouse: The marketing power of collaborative filtering./ Riel J., Konstan A.J., Vrooman E. – New York:Business Plus, 2002 – 25p.

3. Макконнелл Стив. Совершенный код 2 издание./ С.Макконнелл – Москва: Microsoft Press, 2004 – 886с.

4. Вигерс К. Разработка требований к программному обеспечению/ К.Вигерс. – Москва: Microsoft Press, 2004 – 20с.

5. Martin R.C. Clean Architecture A Craftsman's Guide to Software Structure and Design/ R.C. Martin. – Boston: Pearson Education Inc., 2018 – 25с.

6. ISO/IEC/IEEE 15288:2015. Systems and software engineering – System life cycle processes, 2015 – 126p.

7. GOST 34.602-89. Information Technology Set of standards for automated systems. Technical directions for developing of automated system, 1989 – 12p.

8. Santos O. CCNA Security 210-260 Official Cert Guide/Omar Santos, John Stuppi, – Indianapolis: Cisco Press, – 2015 25с.

9. Lowe D. Networking All For Dummies 7th Edition/ D. Lowe – New Jersey: John Wiley & Sons Inc., 2018 – 77с.

10. Nixon R. Learning PHP MySQL JavaScript 5th Edition/Robin Nixon, – Sebastopol: O'Reilly, 2018 – 800с.

11. Rasmusson J.The Agile Samurai How Agile Masters Deliver Great Software/ Jonathan Rasmusson, – Dallas: The Pragmatic Bookshelf, 2010 – 165с.

12. В.М.Дубовой, С.М.Москвіна, О.Д.Никитенко. Моделювання процесів і систем керування / Дубовой В.М., Москвіна С.М., Никитенко О.Д. – Вінниця: ВНТУ, 2009, - 103с.

13. Фаулер М., Скотт К., UML. Основы - СПб: "Символ-Плюс", 2002. - 192 с.

14. Myers G.J. The Art of Software Testing, 3rd Edition/ Glenford J. Myers, Corey Sandler, Tom Badgett, – New Jersey: John Wiley & Sons Inc. 2012 – 23с.

15. Chollet, F. Deep Learning with Python/F.Chollet – New York: Manning, 2017. — 384 p.

16. Адитья Бхаргава. Грокаем алгоритмы Иллюстрированное пособие для программистов и любопытствующих/ Бхаргава А. – СПб: Питер, 2017 – 52с.

Інтернет ресурси

17. Big Data/останнє оновлення 01.11.2021[Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Big_data

18. The Long Tail/останнє оновлення 01.11.2021[Електронний ресурс]. – Режим доступу: https://www.wired.com/2004/10/tail/

19. Machine learning/останнє оновлення 01.11.2021[Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Machine_learning

20. Artificial neural network /останнє оновлення 01.11.2021[Електронний ресурс]. – Режим доступу:https://en.wikipedia.org/wiki/Artificial_neural_network

21. Быстрее суперкомпьютера – ученые создают новый тип вычислений/ останнє оновлення 01.11.2021[Електронний ресурс]. – Режим

доступу: https://techno.bigmir.net/technology/1621058-Bystree-superkomp-jutera--Uchenye-sozdajut-novyj-tip-vychislenij

22. Software requirements specification/останнє оновлення 11.11.2021[Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Software_requirements_specification

23. What is Kubernetes?/ останнє оновлення 01.11.2021[Електронний ресурс]. – Режим доступу: https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/

24. PCAP: Programming Essentials in Python/останнє оновлення 01.11.2021[Електронний ресурс]. – Режим доступу: https://www.netacad.com/courses/programming/pcap-programming-essentials-python

25. Что такое Docker, и как его использовать? Подробно рассказываем/останнє оновлення 01.11.2021[Електронний ресурс]. – Режим доступу: https://www.proglib.io/p/docker/

ДОДАТКИ

Додаток А

Модуль збору даних parser_main.py

```
import os
import time
import json
import asyncio
import pandas as pd
from pprint import pprint
from datetime import datetime
from parser.parser_imdb import ParseImdb
from utillities.check_all import (check_storage,
                                  produce_chunks,
                                  produce_storage,
                                  check_file_presence)
from config import (Folders,
                    DataFrames,
                    user_numbers,
                    dictionary_astrology)


class ParserMain:
    """
    class which is dedicated to develop values of the parser and store it as a dataframe
    """
    def __init__(self) -> None:
        self.folder_storage = os.path.join(Folders.folder_main,
Folders.folder_storage)
```

```
        self.dataframe_storage = os.path.join(self.folder_storage,
DataFrames.df_name)
        self.dataframe_professions = os.path.join(self.folder_storage,
DataFrames.df_name_proffesions)
        self.dataframe_astrology = os.path.join(self.folder_storage,
DataFrames.df_name_astrology)
        self.columns_professions = ['id', 'name']
        self.columns_astrology = ['id', 'name', 'date_begin', 'date_end']
        self.columns = ['id', 'type', 'url', 'name', 'image',
                        'description', 'birthdate', 'deathdate', 'jobs']
        self.columns_full = self.columns + ['jobs_indexes', 'astrology_index']

    def produce_absent(self) -> list:
        """
        Method which is dedicated to produce values of the id values which were not
used
        Input:  None
        Output: list with values which were previously non-created
        """
        value_repeat = pd.read_csv(self.dataframe_storage).id.values
        values_id = [i for i in range(1, user_numbers+1) if i not in value_repeat]
        return produce_chunks(values_id)

    def produce_dataframe_merging(self, value_df:pd.DataFrame) ->
pd.DataFrame:
        """
        Method which is dedicated to produce values of the merge
        Input:  value_list = list of the dictionaries
        Output: we merged values to the datafrane
        """
```

```
        if not os.path.exists(self.dataframe_storage) or not
os.path.isfile(self.dataframe_storage):
            value_old = pd.DataFrame([], columns=self.columns)
        else:
            value_old = pd.read_csv(self.dataframe_storage)
        value_df = pd.concat([value_old, value_df])
        value_df.drop_duplicates(subset=self.columns, keep='first', inplace=True)
        self.produce_save_df(self.dataframe_storage, value_df)

    def produce_list_transformations(self, value_list:list) -> pd.DataFrame:
        """
        Method which is dedicated to make list transformations
        Input:  value_list = list values of the development
        Output: we created list transformation to this
        """
        value_list_old = []
        for value_dict in value_list:
            job_title = value_dict.get('jobTitle', []) if \
                isinstance(value_dict.get('jobTitle', []), list) else
[value_dict.get('jobTitle', [])]
            value_list_old.append(
                [
                    value_dict.get('ID', 0),
                    value_dict.get('@type', 'Person'),
                    value_dict.get('url', ''),
                    value_dict.get('name', ''),
                    value_dict.get('image', ''),
                    value_dict.get('description', ''),
                    value_dict.get('birthDate', ''),
                    value_dict.get('deathDate', ''),
```

```
                '|'.join(job_title)
            ]
        )
    value_df = pd.DataFrame(value_list_old, columns=self.columns)
    return value_df

@staticmethod
def produce_save_df(value_path:str, value_df:pd.DataFrame) -> None:
    """
    Method which is dedicated to save dataframe values
    Input:  value_path = path do dataframe where to save
            value_df = dataframe which is saved
    Output:
    """
    value_df.to_csv(value_path, index=False)

def produce_save_json(self, value_list:list, value_name:str='example.json') ->
None:
    """
    Method which is dedicated to store values for it as a json
    Input:  value_list = list values
            value_name = name which is required to work with
    Output: we saved value as a json
    """
    value_path = os.path.join(self.folder_storage, value_name)
    if os.path.exists(value_path):
        return
    with open(value_path, 'w') as json_wrote:
        json.dump(value_list, json_wrote, indent=4)
```

```
    def develop_dataframe_professions(self, value_list:list) -> pd.DataFrame:
        """
        Method which is dedicated to develop values of the
        Input:  value_list = list of the values of the selected professions
        Output: dataframe of the proffesion values
        """
        value_professions = []
        for value in value_list:
            if isinstance(value, str):
                value_professions.extend(value.split('|'))
        value_professions = list(set(value_professions))
        value_professions = [(index + 1, profession) for index,
                             profession in enumerate(value_professions)]
        return pd.DataFrame(value_professions, columns=self.columns_professions)

    def develop_dataframe_astrology(self) -> pd.DataFrame:
        """
        Method which is dedicated to develop astrology values
        Input:  None
        Output: we developed dataframe for the astrology
        """
        value_list = [(index+1, f.get('name', ''),
                       f.get('begin', ''), f.get('end', ''))
                      for index, f in enumerate(dictionary_astrology)]
        df_astrology = pd.DataFrame(value_list, columns=self.columns_astrology)
        self.produce_save_df(self.dataframe_astrology, df_astrology)

    def develop_values_astrology(self, value_string:str, list_astrology:list=[]) ->
int:
        """
```

```
        Method which is dedicated to return values to the astrology
        Input:  value_string = string of the birthdate
                list_astrology = dataframe lists is returns
        Output: we created values for the astrology index
        """
        if not list_astrology:
            list_astrology = [[datetime.strptime(f.get('begin'), "%m.%d"),
                               datetime.strptime(f.get('end'), "%m.%d")]
                              for f in dictionary_astrology]
        if isinstance(value_string, float):
            return 0
        value_date = datetime.strptime(value_string,"%Y-%m-%d")
        value_add = list_astrology[0][0].year if not \
            (value_date.month==1 and value_date.day < 20) else
list_astrology[0][0].year + 1
        value_date = value_date.replace(day=28) if value_date.day == 29 \
            and value_date.month == 2 else value_date
        value_date = value_date.replace(year=value_add)
        for value_begin, value_end in list_astrology:
            if value_begin <= value_date <= value_end:
                return list_astrology.index([value_begin, value_end]) + 1
        return 0

    def produce_dataframe_filtration(self,
df_value:pd.DataFrame=pd.DataFrame([])) -> pd.DataFrame:
        """
        Method which is dedicated to filtrate values of the selected dataframe
        Input:  df_value = value which is dedicated to produce values
        Output: we created values of the filtrated dataframe
        """
```

```
        if df_value.empty:
            df_value = pd.read_csv(self.dataframe_storage)
        df_value = df_value[df_value['type'] == 'Person']
        df_value_professions =
    self.develop_dataframe_professions(df_value['jobs'].values)
        self.produce_save_df(self.dataframe_professions, df_value_professions)
        list_indexes = []
        list_search = list(df_value_professions['name'].values)
        for values in df_value['jobs'].values:
            if isinstance(values, str):
                list_indexes.append(
                    '|'.join([str(list_search.index(f) + 1) for f in values.split('|')]))
            else:
                list_indexes.append('')
        df_value['jobs_indexes'] = list_indexes
        if not os.path.exists(self.dataframe_astrology):
            self.develop_dataframe_astrology()
        list_astrology = [[datetime.strptime(f.get('begin'), "%m.%d"),
                           datetime.strptime(f.get('end'), "%m.%d")]
                          for f in dictionary_astrology]
        list_astrology[9][1] =
    list_astrology[9][1].replace(year=list_astrology[0][0].year +1)
        list_index_astrology = []
        for value_date in df_value['birthdate'].values:
            list_index_astrology.append(self.develop_values_astrology(value_date,
    list_astrology))
        df_value['astrology_index'] = list_index_astrology
        self.produce_save_df(self.dataframe_storage, df_value)
        return df_value
```

```
    def produce_dataframe(self) -> pd.DataFrame:
        """
        Method which is dedicated to produce values of it
        Input:  None
        Output: we returned values of it
        """
        begin = time.time()
        if not check_storage(self.folder_storage):
            produce_storage(self.folder_storage)
        if check_storage(self.folder_storage) and not
    check_file_presence(self.dataframe_storage):
            values_id = [i+1 for i in range(user_numbers)]
            values_id = produce_chunks(values_id)
        else:
            values_id = self.produce_absent()
        parse_imdb = ParseImdb()
        for value_id in values_id[:]:
            loop = asyncio.get_event_loop()
            value_list = loop.run_until_complete(parse_imdb.produce_main(value_id))
            self.produce_save_json(value_list)
            [f.update({'ID': i}) for f, i in zip(value_list, value_id)]
            value_df = self.produce_list_transformations(value_list)
            self.produce_dataframe_merging(value_df)
            print(f"{value_id[0]}-{value_id[-1]} finished creating values")
            print('_____________________________________________________')
        print(f"{time.time() - begin} seconds")
        print('ccccccccccccccccccccccccccccccccccccccccccccccccccccccc')
```

Додаток Б

Модуль збору даних parser_webdriver.py

```
import os
import zipfile
import requests
from config import link_webdriver, Folders


class ParseWebDriver:
    """
    class which is dedicated to work with selenium and install selenium manually
    """
    def __init__(self) -> None:
        self.name_archive = link_webdriver.split('/')[-1]
        self.path, self.path_webdriver, \
            self.path_archive  = self.get_path_driver()
        self.presence_driver, \
            self.presence_archive = self.check_driver_presence()

    def get_path_driver(self) -> set:
        """
        Method which is dedicated to produce the returnla of the values of the driver
        Input:  value which is required ro be used
        Output: we created our link to the webdriver for it
        """
        value_path = os.path.join(Folders.folder_main,
                                  Folders.folder_storage)
        return value_path, os.path.join(value_path, 'chromedriver'), \
               os.path.join(value_path, self.name_archive)
```

```
    def check_driver_presence(self) -> set:
        """
        Method which is dedicated to check drivers presence of the values
        Input:  None
        Output: boolean value which is required to be used
        """
        os.path.exists(self.path) or os.mkdir(self.path)
        return (os.path.exists(os.path.join(self.path_webdriver, 'chromedriver')) and \
                os.path.isfile(os.path.join(self.path_webdriver, 'chromedriver')) or \
                os.path.exists(self.path_webdriver)                                    and
os.path.isfile(self.path_webdriver)), \
                os.path.exists(self.path_archive) and os.path.isfile(self.path_archive)

    def produce_webdriver_values(self) -> None:
        """
        Method which is dedicated to produce webdriver for the selenium
        Input:  None
        Output: we created and unarchived values
        """
        if self.presence_driver:
            return os.path.join(self.path_webdriver, 'chromedriver')
        elif not self.presence_driver and self.presence_archive:
            self.produce_further_transformations()
        else:
            archive_value = requests.get(link_webdriver, stream=True)
            with open(self.path_archive, 'wb') as archive_new:
                archive_new.write(archive_value.content)
            self.produce_further_transformations()
        return os.path.join(self.path_webdriver, 'chromedriver')
```

```
    def produce_further_transformations(self) -> None:
        """
        Method which is dedicated to produce values out of the archive
        Input:  path of the result
        Output: we created values of the
        """
        try:
            with zipfile.ZipFile(self.path_archive) as myzip:
                if 'chromedriver' in myzip.namelist():
                    myzip.extract('chromedriver', self.path_webdriver)
                    os.chmod(os.path.join(self.path_webdriver, 'chromedriver'), 755)
        except Exception as e:
            print(e)

print('>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>')
        finally:
            os.remove(self.path_archive)
```

Додаток В

Модуль збору даних parser_imdb.py

```
import json
import os
import time
import aiohttp
import asyncio
import requests
from pprint import pprint
from bs4 import BeautifulSoup
from selenium import webdriver
from fake_useragent import UserAgent
from parser.parser_webdriver import ParseWebDriver
from config import Imdb


class ParseImdb:
    """
    class which is dedicated to produce
    """
    def __init__(self) -> None:
        self.driver = None
        self.webdriver_path = ParseWebDriver().produce_webdriver_values()

    def set_webdrivers_options(self) -> None:
        """
        Method which is dedicated to create options of the webdriver
        Input:  None
        Output: we developed options to this
```

```
        """
        self.options = webdriver.ChromeOptions()
        self.options.add_argument("--headless")
        self.options.add_argument(f"user-agent={UserAgent().random}")

    def set_webdriver(self) -> None:
        """
        Method which is dedicated to create webdriver in all cases
        Input:  None
        Output: we created webdriver with randomized options
        """
        self.set_webdrivers_options()
        self.driver = webdriver.Chrome(self.webdriver_path, options=self.options)

    def remove_webdriver(self) -> None:
        """
        Method which is dedicated to create webdriver in all cases
        Input:  None
        Output: we created webdriver
        """
        self.driver.close()
        self.driver.quit()

    @staticmethod
    async def check_number(number:int) -> str:
        """
        Static method which is dedicated to check numbers
        Input:  number = number of values which was previously created
        Output: we returned number as a string
        """
```

```
        return str(number) if isinstance(number, int) or isinstance(number, float) \
            or not isinstance(number, str) else number

    async def produce_html_values(self, session:object, link:str) -> str:
        """
        Method which is dedicated to work with html values
        Input:  session = session which was previously developed
                link = link value which is required to work with
        Output: html value which is dedic
        """
        async with session.get(link, headers={'user-agent': UserAgent().random,}) as
r:
            if r.status == 200:
                return await r.text()
            return link

    async def produce_html_response(self, link:str) -> str:
        r = requests.get(link)
        if r.status_code == 200:
            return r.text
        return link

    async def produce_html_parsing(self, html:str, value_link:str) -> dict:
        """
        Method which is dedicated to produce values of the actor/actress differentiating
from it
        Input:  html = html which was previously parsed from it
                value_link = link value which was previously parsed
        Output: we successfully parsed values from the page as a dictionary
        """
```

```
        value_dict = {'link': value_link}
        if len(html) < 1000:
            return value_dict
        try:
            soup = BeautifulSoup(html, 'html.parser')
            soup = soup.find("script", type="application/ld+json")
            value_dict = json.loads(soup.text)
        except Exception as e:
            print(e)
            print('>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>')
        return value_dict

    @staticmethod
    async def produce_value_links(number:str) -> str:
        """
        Static method which is dedicated to produce values
        Input:  number = number which is requited to work with
        Output: we created link value for developing new values of it
        """
        def produce_number(number):
            if len(number) >= 7:
                return number
            number = ''.join(['0' for _ in range(7 - len(number))]) + number
            return number
        number = produce_number(number)
        return                    '/'.join([Imdb.link,                    Imdb.link_name,
f'{Imdb.link_name_add}{number}'])

    async def produce_main(self, numbers:list) -> list:
        """
```

Method which is dedicated to produce values of the actors and get all of their information

Input: numbers = numbers of IDs of the Kinopoisk which possibly could be used as a parsing

Output: we created lists of the dictionaries and developed the database for it

"""

numbers = [asyncio.create_task(self.check_number(number)) for number in numbers]

numbers = await asyncio.gather(*numbers)

tasks = [asyncio.create_task(self.produce_value_links(number)) for number in numbers]

links = await asyncio.gather(*tasks)

semaphore = asyncio.Semaphore(Imdb.semaphore)

async with semaphore:

async with aiohttp.ClientSession(trust_env=True) as session:

tasks = [asyncio.create_task(self.produce_html_values(session, link)) for link in links]

htmls = await asyncio.gather(*tasks)

tasks = [asyncio.create_task(self.produce_html_parsing(html, link)) for html, link in zip(htmls, links)]

return await asyncio.gather(*tasks)

Додаток Г

Модуль створення бази даних db_creator.py

```
import re
from sqlalchemy.orm import relationship
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy import (Table,
                        Text,
                        Column,
                        Integer,
                        String,
                        ForeignKey,
                        PrimaryKeyConstraint)
from sqlalchemy.sql.sqltypes import Boolean


Base = declarative_base()

association_table_user_astrology = Table('user_astrology', Base.metadata,
    Column('id_astrology', ForeignKey('astrology.id')),
    Column('id_user', ForeignKey('user.id')),
    PrimaryKeyConstraint('id_astrology', 'id_user')
)

association_table_user_profession = Table('user_profession', Base.metadata,
    Column('id_profession', ForeignKey('profession.id')),
    Column('id_user', ForeignKey('user.id')),
    PrimaryKeyConstraint('id_profession', 'id_user')
)
```

```
association_table_user_gender = Table('user_gender', Base.metadata,
    Column('id_user', ForeignKey('user.id')),
    Column('id_gender', ForeignKey('gender.id')),
    PrimaryKeyConstraint('id_gender', 'id_user')
)

association_table_user_credentials = Table('user_credentials', Base.metadata,
    Column('id_user', ForeignKey('user.id')),
    Column('id_credentials', ForeignKey('credentials.id')),
    PrimaryKeyConstraint('id_credentials', 'id_user')
)


class User(Base):
    __tablename__ = 'user'
    id = Column(Integer, primary_key=True)
    name = Column(String(120))
    link = Column(String(100))
    link_image = Column(String(200))
    date_birth = Column(String(20))
    date_death = Column(String(20))
    description = Column(Text)
    is_parsed = Column(Boolean, default=True)
    user_profession = relationship("Profession",
        secondary=association_table_user_profession,
        back_populates="profession_user")
    user_astrology = relationship("Astrology",
        secondary=association_table_user_astrology,
        back_populates="astrology_user")
    user_gender = relationship("Gender",
```

```
        secondary=association_table_user_gender,
        back_populates="gender_user")
    user_credentials = relationship('Credentials',
        secondary=association_table_user_credentials,
        back_populates='credentials_user')

class Astrology(Base):
    __tablename__ = 'astrology'
    id = Column(Integer, primary_key=True)
    name = Column(String(30))
    date_begin = Column(String(10))
    date_end = Column(String(10))
    astrology_user = relationship("User",
        secondary=association_table_user_astrology,
        back_populates="user_astrology")

class Profession(Base):
    __tablename__ = 'profession'
    id = Column(Integer, primary_key=True)
    name = Column(String(100))
    profession_user = relationship("User",
        secondary=association_table_user_profession,
        back_populates="user_profession")

class Gender(Base):
    __tablename__ = 'gender'
    id = Column(Integer, primary_key=True)
    name = Column(String(10))
    gender_user = relationship("User",
        secondary=association_table_user_gender,
```

```
    back_populates="user_gender")

class Credentials(Base):
    __tablename__ = 'credentials'
    id = Column(Integer, primary_key=True)
    email = Column(String(50), nullable=False)
    password = Column(String(100), nullable=False)
    credentials_user = relationship('User',
        secondary=association_table_user_credentials,
        back_populates='user_credentials')

class Selection(Base):
    __tablename__ = 'selection'
    id = Column(Integer, primary_key=True)
    id_selected = Column(Integer)
```

Додаток Д

Модуль заповнення бази даних db_main.py

```
import os
import pandas as pd
from pprint import pprint
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker
from parser.parser_main import ParserMain
from db.db_creator import  (Base,
                            User,
                            Gender,
                            Astrology,
                            Profession,
                            association_table_user_gender,
                            association_table_user_astrology,
                            association_table_user_profession)
from utillities.check_all import (check_storage,
                                  produce_storage,
                                  check_file_presence)
from config import (Db,
                    Folders,
                    dictionary_astrology)


class DataBaseMain:
    """
    class which is dedicated to make the basic tests insertion and operations for it
    """
    def __init__(self) -> None:
        self.session = None
```

```
        self.parser_main = ParserMain()
        self.folder_path = os.path.join(Folders.folder_main, Folders.folder_storage)
        self.file_path = os.path.join(self.folder_path, Db.sqlite_name)
        self.engine = self.produce_engine_file()

    def check_route(self) -> bool:
        """
        Method which is dedicated to develop check of the route of sqlite
        Input:  None; all what we have created
        Output: we developed path to the database and produced everything
        """
        if not check_storage(self.folder_path):
            produce_storage(self.folder_path)
            return False
        if not check_file_presence(self.file_path):
            return False
        return True

    def produce_engine_file(self) -> object:
        """
        Method which is dedicated to create engine from the file
        Input:  None
        Output: we created engine files
        """
        self.check_route()
        return create_engine(f"sqlite:///{self.file_path}", echo=Db.echo)

    def check_database(self) -> bool:
        """
        Method which is dedicated to develop checking the database
```

```
    Input:
    Output: boolean value which signify that database is developed
    """
    check_route = self.check_route()
    if not check_route:
        print('Base is completely new')
        self.develop_database()
    return bool(self.session)

def develop_database(self):
    """
    Method which is dedicated to produce the database it that cases
    Input:  None
    Output: we created values from the db_creator file and produced the database
    """
    try:
        Base.metadata.create_all(self.engine)
    except Exception as e:
        print(f"We faced problems with Base: {e}")
        print('----------------------------------------')

def return_session(self) -> object:
    """
    Method which is dedicated to develop session
    Input:  None
    Output: We created session of the values
    """
    try:
        Session = sessionmaker(bind=self.engine)
        return Session()
```

```
        except Exception as e:
            print(e)
            print('----------------------------')

    def close_session(self) -> None:
        """
        Method which is dedicated to close session of the sql alchemy
        Input:  None values
        Output: we close this session
        """
        if self.session:
            self.session.close()

    def make_mass_insertion(self, value_list:list) -> None:
        """
        Method for inserting objects at one iterations
        Input:  value_list = list of the selected objects
        Output: we commited to the database
        """
        if value_list:
            self.session.add_all(value_list)
            self.session.commit()

    def make_basic_insertion(self, value_list:list) -> None:
        """
        Method which is dedicated to insert basic without add_all command
        Input:  value_list = list of the commands
        Output: we inserted values of it
        """
        if value_list:
```

```
            for f in value_list:
                self.session.execute(f)
                self.session.commit()

    def produce_insertion_model_gender(self, list_gender:list, list_gender_id:list) ->
None:
        """
        Method which is dedicated to make insertion of the selected values
        Input:  list_gender = set of the values to insert genders
                list_gender_id = list of lists of the values which is innerconnected to it
        Output: we developed insertion values which developed
        """
        if not self.check_database():
            self.session = self.return_session()
        gender_used = [f[0] for f in self.session.query(Gender.id).all()]
        list_gender = [[id, name] for id, name in list_gender if id not in gender_used]
        gender_specified = [f[0] for f in self.session.query(User.id).filter(
                        association_table_user_gender.c.id_user== User.id).all()]
        list_gender_id = [[id_user, id_gender] for id_user, id_gender
                        in list_gender_id if id_user not in gender_specified]
        objects = [Gender(id=id, name=name) for id, name in list_gender]
        self.make_mass_insertion(objects)
        objects     =     [association_table_user_gender.insert().values(id_user=id_user,
id_gender=id_gender)
                    for id_user, id_gender in list_gender_id]
        self.make_basic_insertion(objects)
        self.close_session()
        print(f'Finished inserting for {list_gender_id[0][0]}-{list_gender_id[-1][0]}')
        print('==========================================================')
```

```
    def produce_insertion(self, *args:set) -> None:
        """
        Method which is dedicated to insert all values to the database
        Input:  args = set with used values of the created database:
                list_astrology = list with the astrology table
                list_profession = list with the profession table
                list_users = list with the users table
                list_id_astrology = list of the connecting user/astrology
                list_id_professions = list of the connecting id/professions
        Output: we inserted values to the database if it is necessary
        """
        list_astrology, list_profession, list_users, \
            list_id_astrology, list_id_professions = args
        if not self.check_database():
            self.session = self.return_session()

        list_id_astrologies = [[list_users[i-1][1], list_ids] for i, list_ids in
list_id_astrology]
        list_id_profession = [[list_users[i-1][1], list_ids] for i, list_ids in
list_id_professions]

        value_links_prev = [f[0] for f in self.session.query(User.link).all()]
        list_users = [[i, link, name, link_image, description, date_begin, date_end]
                for i, link, name, link_image, description, date_begin, date_end
                in list_users  if link not in value_links_prev]

        list_id_astrologies = [[list_user, list_ids] for list_user, list_ids in
list_id_astrologies
                if list_user in [f[1] for f in list_users]]
```

```
        list_id_profession = [[list_user, list_ids] for list_user, list_ids in
list_id_professions
                if list_user in [f[1] for f in list_users]]

        objects = [User(name=name, link=link, description=description,
                        date_birth=date_birth,                    date_death=date_death,
link_image=link_image)
                   for _, link, name, link_image, description, date_birth, date_death in
list_users]
        self.make_mass_insertion(objects)

        list_astrology = [[i, name, date_begin, date_end] for i, name, date_begin,
date_end
                          in list_astrology if name
                          not in [f[0] for f in self.session.query(Astrology.name).all()]]
        objects = [Astrology(id=i, name=name, date_begin=date_begin,
date_end=date_end)
                   for i, name, date_begin, date_end in list_astrology]
        self.make_mass_insertion(objects)

        list_profession_prev = [f[0] for f in self.session.query(Profession.name).all()]
        list_profession = [[i, name] for i, name in list_profession if name not in
                           list_profession_prev]
        objects = [Profession(name=name) for _, name in list_profession]
        self.make_mass_insertion(objects)

        ids_users = [f[0] for f in self.session.query(User.id).filter(
                     User.link.in_([f[0] for f in list_id_astrologies])).all()]
        list_id_astrologies = [[id_user, id_astrology] for
                               id_user, (_, id_astrology) in zip(ids_users, list_id_astrologies)]
```

```
        objects  =  [association_table_user_astrology.insert().values(id_user=id_user,
id_astrology=int(id_astrology))
                for id_user, id_astrology in list_id_astrologies]
        self.make_basic_insertion(objects)

        ids_users = [f[0] for f in self.session.query(User.id).filter(
                    User.link.in_(f[0] for f in list_id_profession)).all()]
        list_id_professions = []
        for (_, id_professions), ids_user in zip(list_id_profession, ids_users):
            for id_profession in id_professions:
                list_id_professions.append([ids_user, id_profession])
        objects = [association_table_user_profession.insert().values(id_user=ids_user,
id_profession=id_profession)
                for ids_user, id_profession in list_id_professions]
        self.make_basic_insertion(objects)
        self.close_session()

    def get_values_user(self, value_id:int) -> set:
        """
        Method which is dedicated usage of the values
        Input:  value_id = id of the user
        Output: values for the development
        """
        if not self.check_database():
            self.session = self.return_session()
        table_user = self.session.query(User.id, User.name, User.link,
                User.link_image, User.date_birth, User.date_death,
                User.description).filter(User.id==value_id).first()

        return table_user
```

```
    def produce_basic_values_insertion(self, value_refind:bool=False) -> None:
        """
        Method which is dedicated to transform basic values of the insertion
        Input:  value_refind = boolean value which signify that
                               we need to refill the csv file from the Imdb
        Output: we prepared all possible values and removed them from the
        """
        if value_refind:
            self.parser_main.produce_dataframe()
        if not os.path.exists(self.parser_main.dataframe_storage):
            self.parser_main.produce_dataframe()
        df_value = pd.read_csv(self.parser_main.dataframe_storage)
        df_value = self.parser_main.produce_dataframe_filtration(df_value)
        df_value = pd.read_csv(self.parser_main.dataframe_storage)
        list_astrology = [[index + 1, f.get('name'),
                           f.get('begin'), f.get('end')]
                          for index, f in enumerate(dictionary_astrology)]
        list_profession =
    pd.read_csv(self.parser_main.dataframe_professions).values.tolist()
        list_users = df_value[['id', 'url', 'name', 'image',
                               'description', 'birthdate', 'deathdate']].values.tolist()
        list_id_astrology = [[value_id, value_astrology] for value_id, value_astrology
                             in zip(df_value['id'].values,
                                    df_value['astrology_index'].values)]
        list_id_astrology = [[value_id, value_astrology] for value_id, value_astrology
                             in list_id_astrology if value_astrology != 0]
        list_id_professions = [[value_id, [int(i) for i in value_id_profession.split('|')]]
                               for value_id, value_id_profession in zip(df_value['id'].values,
                                    df_value['jobs_indexes'].values)
```

```
                                    if isinstance(value_id_profession, str)]
        self.produce_insertion(list_astrology, list_profession,
                               list_users, list_id_astrology, list_id_professions)
        self.close_session()
```

Додаток Е

Модуль завантажувач моделей model_downloader.py

```
import os
import gdown
import zipfile
from config import (Models,
                    Folders,
                    link_model_gender)


class ModelDownloader:
    """
    class which is dedicated to produce the models
    to the storages in cases of self download of them
    """
    def __init__(self) -> None:
        self.check_folder = lambda x: os.path.exists(x) or os.mkdir(x)
        self.location_folder = os.path.join(Folders.folder_main,
              Folders.folder_storage, Models.folder_models)
        self.location_zip = os.path.join(Folders.folder_main,
              Folders.folder_storage, Models.gender_name_archive)

    def extract_values_models(self, value_check:bool=False) -> None:
        """
        Method which is dedicated to extract values of the
        Input:  value_check = checked value created
        Output: we outputed from the archive values
        """
        if value_check:
```

```
            return
        with zipfile.ZipFile(self.location_zip, 'r') as zip_return:
            for zip_name in zip_return.namelist():
                zip_get = os.path.join(self.location_folder, zip_name)
                if zip_name in [Models.gender_model, Models.gender_proto,
                                Models.face_model, Models.face_proto] and not
os.path.exists(zip_get):
                    zip_return.extract(zip_name, self.location_folder)

    def get_value_archive(self, value_check:bool=False) -> None:
        """
        Method which is dedicated to return value of the archives
        Input:  value_check = checked values of the created values
        Output: we developed archive with the link
        """
        if not os.path.exists(self.location_zip):
            value_check = False
        if not value_check:
            gdown.download(link_model_gender, self.location_zip)

    def download_gender_detection_models(self) -> None:
        """
        Method which is dedicated to develop values of the downloading model of it
        Input:  all presented values
        Output: we developed values of it
        """
        value_check = os.path.exists(self.location_folder)
        self.check_folder(self.location_folder)
        self.get_value_archive(value_check)
        if value_check:
```

```
        value_check = all([os.path.exists(os.path.join(self.location_folder, f)) for f in
            [Models.gender_model,    Models.gender_proto,    Models.face_model,
Models.face_proto]])
        self.extract_values_models(value_check)
        return  [os.path.join(self.location_folder, f) for f in
            [Models.gender_model,    Models.gender_proto,    Models.face_model,
Models.face_proto]]
```

Додаток Ж

Модель визначення базових характеристик користувача model_gender.py

```
import cv2
from pprint import pprint
from multiprocessing import Pool
from db.db_main import DataBaseMain
from db.db_creator import (User,
                           Gender,
                           association_table_user_gender)
from models.model_downloader import ModelDownloader
from utillities.check_all import produce_chunks
from config import dictionary_gender


class ModelGender:
    """
    class which is dedicated to produce values of the gender to selected values
    """
    def __init__(self) -> None:
        self.database_main = DataBaseMain()
        self.gender_net, self.gender_deploy, self.face_detector_pb, \
            self.face_detector_pbtxt                                    =
ModelDownloader().download_gender_detection_models()
        self.session = self.database_main.return_session()
        self.net_face                =                cv2.dnn.readNet(self.face_detector_pb,
self.face_detector_pbtxt)
        self.net_gender = cv2.dnn.readNet(self.gender_net, self.gender_deploy)
        self.model_mean = (78.4263377603, 87.7689143744, 114.895847746)
```

```
    def get_gender_value(self, value_manual:list, value_model:list,
value_priority:bool=True) -> list:
        """
        Method which is dedicated to develop answer to get them into the database
        Input:  value_list_manual = list of the gender by manually search
                value_list_model = list of the gender by modelling of the
                value_priority = priority to use one way to another
        Output: we develop values of the image to detect it from it
        """
        value_answer = []
        for (id_manual, answer_manual), (id_model, answer_model) in
zip(value_manual, value_model):
            if id_manual == id_model:
                if answer_model == 'Unknown' and answer_manual != answer_model:
                    value_answer.append([id_manual, answer_manual])
                elif answer_manual == 'Unknown' and answer_manual != answer_model:
                    value_answer.append([id_model, answer_model])
                elif answer_manual != 'Unknown' and answer_model != 'Unknown' \
                        and value_priority and answer_manual != answer_model:
                    value_answer.append([id_manual, answer_manual])
                elif answer_manual != 'Unknown' and answer_model != 'Unknown' \
                        and not value_priority and answer_manual != answer_model:
                    value_answer.append([id_model, answer_model])
                elif answer_manual == answer_model == "Unknown":
                    value_answer.append([id_model, answer_model])
                else:
                    value_answer.append([id_model, answer_model])
        return value_answer

    def produce_gender_search_manually(self, value_text:str) -> int:
```

```
        """
        Method which is dedicated to produce values from the
        Input:  value_text = text about the user
        Output: we developed value of the id to which gender of it
        """
        value_male = ['he', 'him', 'his']
        value_female = ['she', 'her', 'hers']
        value_text = [''.join(e for e in f if e.isalnum()).lower() for f in value_text.split()]
        count_male = sum([value_text.count(f) for f in value_male])
        count_female = sum([value_text.count(f) for f in value_female])
        if count_male > count_female:
            return 'Male'
        elif count_male < count_female:
            return 'Female'
        return 'Unknown'

    def highlight_image_persons(self, net, frame, threshold = 0.7) -> list:
        """
        Method which is dedicated to produce values of the faces
        Input:  net = out network for the usage
                frame = our picture
                threshold = parameter for the making sure of it
        Output: we detected the locations of the possible faces
        """
        frameOpencvDnn = frame.copy()
        height = frameOpencvDnn.shape[0]
        width = frameOpencvDnn.shape[1]
        blob = cv2.dnn.blobFromImage(frameOpencvDnn, 1.0, (300, 300), [104, 117,
123], True, False)
        net.setInput(blob)
```

```
    detections = net.forward()
    faceboxes = []

    for i in range(detections.shape[2]):
        confidence = detections[0, 0, i, 2]
        if confidence > threshold:
            faceboxes.append(
                [
                    int(detections[0, 0, i, 3] * width),
                    int(detections[0, 0, i, 4] * height),
                    int(detections[0, 0, i, 5] * width),
                    int(detections[0, 0, i, 6] * height)
                ])
    return faceboxes

def produce_gender_search_modelling(self, index, value_image_link:str):
    """
    Method which is dedicated to develop values of the image
    Input:  index = value indexing of the development
            value_image_link = image string for the development
    Output: value of the model which was previously checked
    """
    if not value_image_link:
        return index, 'Unknown'
    video = cv2.VideoCapture(value_image_link)
    padding = 20

    while cv2.waitKey(1) < 0:
        hasFrame, frame = video.read()
        if not hasFrame:
```

```
            cv2.waitKey()
            break

        faceboxes = self.highlight_image_persons(self.net_face, frame)
        if not faceboxes:
            return index, 'Unknown'
        for facebox in faceboxes:
            face = frame[
                max(0, facebox[1] - padding) : min(facebox[3] + padding,
frame.shape[0] - 1),
                max(0, facebox[0] - padding) : min(facebox[2] + padding,
frame.shape[1] - 1)]
            if not face.any():
                try:
                    blob = cv2.dnn.blobFromImage(face, 1.0, (227, 227),
self.model_mean, swapRB=False)
                    self.net_gender.setInput(blob)
                    genderPreds = self.net_gender.forward()
                    gender = ['Male', 'Female'][genderPreds[0].argmax()]
                except Exception as e:
                    # print(f'Broken; index: {index}; Error: {e}')
                    gender = 'Unknown'
            else:
                gender = 'Unknown'
        return index, gender

    def produce_values_main(self, value_refind:bool=True) -> None:
        """
        Method which is dedicated to produce values into the datbase with produce
values
```

```
        of the gender to the database
        Input:  value_refind = boolean value which signifies the work
        Output: we created values of the
        """
        if not value_refind:
            return
        user_gender_specified = self.session.query(User.id).filter(association_table_user_gender.c.id_user==User.id).all()
        user_gender_specified = [f[0] for f in user_gender_specified]
        user_desc_images_gender_without = self.session.query(
            User.id, User.description, User.link_image).filter(~User.id.in_(user_gender_specified)).all()
        value_gender_manual = [[i, self.produce_gender_search_manually(text)]
            if text else [i, 'Unknown'] for i, text, _ in user_desc_images_gender_without]
        value_image = [[i, link] for i, _, link in user_desc_images_gender_without]
        value_gender_manual = produce_chunks(value_gender_manual, 50)
        value_image = produce_chunks(value_image, 50)

        for index, images in enumerate(value_image):
            gender_models = [self.produce_gender_search_modelling(i, image) for i, image in images]
            value_gender_operated = self.get_gender_value(value_gender_manual[index], gender_models)
            value_gender_operated = [[id, dictionary_gender[name]] for id, name in value_gender_operated]
            value_gender_main = [[value, key] for key, value in dictionary_gender.items()]
            self.database_main.produce_insertion_model_gender(value_gender_main, value_gender_operated)
```

Додаток И

Модуль перевірки необхідних компонентів check_all.py

```
import os
from config import default_chunk

def check_storage(folder_path:str) -> bool:
    return os.path.exists(folder_path) \
        and os.path.isdir(folder_path)

def produce_storage(folder_path:str) -> None:
    return os.path.exists(folder_path) \
        or os.mkdir(folder_path)

def check_file_presence(file_path:str) -> bool:
    return os.path.exists(file_path) \
        and os.path.isfile(file_path)

def produce_chunks(value_list:list, value_length:int=default_chunk) -> list:
    def chunk(value_list:list, length:int):
        for i in range(0, len(value_list), length):
            yield value_list[i:i + length]
    return list(chunk(value_list, value_length))
```

Додаток К

Модуль конфігурації config.py

```
import os
from attr import dataclass
from dotenv import load_dotenv


load_dotenv()

link_webdriver = os.getenv("LINK_WEBDRIVER")
link_model_gender = os.getenv("LINK_MODEL_GENDER")

default_chunk = 10
user_numbers = 10000

@dataclass
class Db:
    sqlite_name = 'sqlite.db'
    echo = False

@dataclass
class ProductionConfig(object):
    host = '0.0.0.0'
    port = 5001
    debug = True
    # SECRET_KEY = ''

@dataclass
class Models:
```

```
    folder_models = 'models'
    face_proto = 'opencv_face_detector.pbtxt'
    face_model = 'opencv_face_detector_uint8.pb'
    gender_proto = 'gender_deploy.prototxt'
    gender_model = 'gender_net.caffemodel'
    gender_name_archive = 'gad.zip'

@dataclass
class Folders:
    folder_main = os.getcwd()
    folder_storage = 'storage'

@dataclass
class Imdb:
    link = 'https://www.opendb.com'
    link_name = 'name'
    semaphore = 10
    link_name_add = 'nm'

@dataclass
class DataFrames:
    df_name = 'names_imdb.csv'
    df_name_proffesions = 'professions.csv'
    df_name_astrology = 'astrology.csv'

dictionary_gender = {
    'Male': 1,
    'Female': 2,
    'Unknown': 3,
}
```

```
dictionary_astrology = [
    {
        'name': 'The Ram',
        'begin': '03.21',
        'end': '04.19',
    },
    {
        'name': 'The Bull',
        'begin': '04.20',
        'end': '05.20',
    },
    {
        'name': 'The Twins',
        'begin': '05.21',
        'end': '06.21',
    },
    {
        'name': 'The Crab',
        'begin': '06.22',
        'end': '07.22',
    },
    {
        'name': 'The Lion',
        'begin': '07.23',
        'end': '08.22',
    },
    {
        'name': 'The Maiden',
        'begin': '08.23',
```

```
    'end': '09.22',
  },
  {
    'name': 'The Scales',
    'begin': '09.23',
    'end': '10.22',
  },
  {
    'name': 'The Scorpion',
    'begin': '10.23',
    'end': '11.22',
  },
  {
    'name': 'The Archer',
    'begin': '11.23',
    'end': '12.21',
  },
  {
    'name': 'The Goat',
    'begin': '12.22',
    'end': '01.19',
  },
  {
    'name': 'The Water-bearer',
    'begin': '01.20',
    'end': '02.18',
  },
  {
    'name': 'The Fish',
    'begin': '02.19',
```

```
        'end': '03.20',
    },
]

# config = {
#     'dev': Website,
# }
```

Додаток Л

Модуль запуску рекомендаційної системи start.py

```
import os
import sys
import time
from db.db_main import DataBaseMain
from models.model_gender import ModelGender
from website.website import app
from config import ProductionConfig

try:
    start_database_development = time.time()
    value_add_db = True
    value_add_db = False
    DataBaseMain().produce_basic_values_insertion(value_add_db)
    print(f"Inserting new profiles took: {time.time() - start_database_development} seconds")

print('@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@')

    start_database_gender = time.time()
    ModelGender().produce_values_main(value_add_db)
    print(f"Inserting genders of the people took: {time.time() - start_database_gender} seconds")

print('@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@')
```

```
    app.run(host=ProductionConfig.host,                port=ProductionConfig.port,
debug=ProductionConfig.debug)
except Exception as e:
    exc_type, exc_obj, exc_tb = sys.exc_info()
    fname = os.path.split(exc_tb.tb_frame.f_code.co_filename)[1]
    print(exc_type, fname, exc_tb.tb_lineno)
    print('_____')
    print(e)
    print('.....')
```

Додаток М

Модуль інтеграції рекомендаційної системи з веб-сайтом website.py

```
from flask import Flask

app = Flask(__name__)
from website.routes import *
```