МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА ВЕБЗАСТОСУНКУ ДЛЯ ПУБЛИКАЦІЇ ПОВІДОМЛЕНЬ У INSTAGRAM ТА FACEBOOK»**

Виконала: студентка 2 курсу, групи 8.1212-іпз-1
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми інженерія програмного забезпечення
(назва освітньої програми)
В.Є. Шепель
(ініціали та прізвище)

Керівник доцент кафедри програмної інженерії, доцент, к.т.н. Мухін В.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент доцент кафедри комп'ютерних наук, доцент, к.т.н. Матвіїшина Н.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти магістр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ

Шепель Вікторії Євгенівні
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка вебзастосунку для публикації повідомлень у Instagram та Facebook

керівник роботи Мухін Віталій Вікторович, доцент, к.т.н
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 01 » травня 2023 року № 642-с

2. Строк подання студентом роботи 27.11.2023 р.

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3. Розробка вебзастосунку для публікації повідомлень у Instagram та Facebook.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація

6. Консультанти розділів роботи

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

7. Дата видачі завдання 08.05.2023 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	19.05.2023	
2.	Збір вихідних даних.	09.06.2023	
3.	Обробка методичних та теоретичних	03.07.2023	
	джерел.		
4.	Розробка першого та другого розділу.	15.08.2023	
5.	Розробка третього розділу.	20.10.2023	
6.	Оформлення та нормоконтроль	17.11.2023	
	кваліфікаційної роботи магістра.		
7.	Захист кваліфікаційної роботи.	15.12.2023	

Студент _____ В.Є. Шепель
(підпис) (ініціали та прізвище)

Керівник роботи _____ В.В. Мухін
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка вебзастосунку для публикації повідомлень у Instagram та Facebook»: 54 с., 33 рис., 3 табл., 9 джерел.

ВЕБСЕРВІС, КОНТЕНТМЕЙКІНГ, ФРЕЙМВОРК, AI COMPUTER VISION, API, FACEBOOK, INSTAGRAM, VUEJS.

Об’єкт дослідження – потреби контентмейкінгу та Instagram Graph API.

Мета роботи: розробити вебдодаток для публікації повідомлень у Instagram та Facebook.

Метод дослідження – методи збору та аналізу вимог до програмного забезпечення, метод проєктування програмного забезпечення.

У процесі розробки вебзастосунку було проведено дослідження сучасних потреб контентмейкерів, Instagram Graph API та MS Azure Computer Vision API, обрані необхідні функції, які сервіс повинен реалізовувати та створено додаток з використанням мови програмування PHP, фреймворків Laravel та Vue.JS. В результаті роботи отримано програмне забезпечення для планування публікацій до соціальних мереж Instagram та Facebook.

SUMMARY

Master’s qualifying paper «Development of the Web Application for message publication in Instagram and Facebook»: 54 pages, 33 figures, 3 tables, 9 references.

WEB SERVICE, CONTENTMAKING, FRAMEWORK, AI COMPUTER VISION, API, FACEBOOK, INSTAGRAM, VUEJS.

Object of the study – contentmaking needs and the Instagram Graph API.

Aim of the study: develop a web application for posting messages on Instagram and Facebook.

Method of research – methods of collecting and analyzing software requirements, software design methods.

In the process of developing the web application, a study of the current needs of content creators, Instagram Graph API and MS Azure Computer Vision API was conducted, the necessary functions that the service should implement were selected and an application was created using the PHP programming language, Laravel and Vue.JS frameworks. As a result of the work software for scheduling publications to social networks Instagram and Facebook.

ЗМІСТ

ВСТУП

В сучасному світі людина все більше часу проводить у соціальних мережах. Фото, відео, історії, текстові дописи – все це наразі основне джерело інформації про життя тієї чи іншої персони. Завдяки таким засобам можна висловити свою думку, чи навіть вплинути на думки інших, розповісти про цікаві події в житті чи попросити допомоги.

Основним з таких по праву можна вважати Instagram – соціальна мережа, що дозволяє об’єднати усі перераховані вище варіації постів, окрім цього також дозволяє просувати власні товари, бренди, послуги та багато чого іншого.

Основною з проблем Instagram є відсутність можливості запланувати публікування на певну дату. Це позбавляє людину змоги розпланувати свій контент-план завчасно та зручно керувати змінами, підлаштовуючи їх до можливих несподіваностей.

Тому все більше стає питання потреби зручних та багатофункціональних інструментів, що полегшать роботу та дозволять створювати цілі плани які пости та коли викладати.

Таким чином, головною метою кваліфікаційної роботи магістра є розробка вебзастосунку для публікації постів у Instagram та Facebook. Для виконання цієї мети у роботі поставлені наступні задачі:

– розглянути основні потреби сучасних контентмейкерів;
– розглянути API для сервісів Instagram та MS Azure Computer Vision;
– розробити додаток для спрощення та планування постингу у Instagram та Facebook.

1 ПОСТАНОВКА ЗАВДАННЯ ТА ОГЛЯД API СЕРВІСІВ

1.1 Потреби типового контентмейкера

Основними потребами сучасного користувача соціальної мережі, без сумніву, є спрощення та прискорення процесу роботи з блогом. Для цього йому необхідний певний перелік інструментів, який допоможе як і оформити пост, так і сформувати певні статистичні дані як усього профілю в цілому, так і кожної публікації юзера окремо, що допоможе в подальшому покращувати якість контенту та підвищувати охвати.

Оглянувши основні можливості Instagram та статистики того, для чого найчастіше його використовують, а також якого функціоналу ще не вистачає, можна виділити певний перелік особливостей, якими повинен володіти додаток для того, щоб стати зручним та популярним:

- можливість створювати нові пости різних типів: фото, рілз, сторіс, з можливістю додавання геолокації, тегів продуктів та відміток інших користувачів;
- можливість планувати викладення постів, з можливістю видалення та редагування запланованих;
- можливість переглядати власну статистику лайків, переглядів та коментарів, будувати графіки на їхній основі;
- можливість переглядати список постів та статистику для кожного з них, шукати необхідні.

Окрім цього, особливістю розробленого додатку є використання штучного інтелекту для обробки фото та отримання необхідних даних. У випадку з Instagram важливими є опис фото, що дозволить юзеру скоротити час створення посту. Окрім цього важливою частиною посту є теги, завдяки яким він швидше просувається та набирає охвати. І даний AI сервіс дозволяє згенерувати певну кількість можливих тегів для обраних фотографій.

Також важливою особливістю стала можливість додаткового постингу ще й у Facebook як окремого медіафайлу, так і повноцінного посту з дописом та таргетингом, що значно розширює можливості за рахунок охвату одразу декількох платформ одночасно.

Для того, щоб при авторизації сайту не потрібно було кожен раз робити нові запити до серверу, необхідно розробити структуру бази даних сайту, яка б дозволяла зберігати та отримувати як токен авторизації для кожного користувача, так і дані постів чи самого профілю, що значно покращить швидкість роботи додатку та дозволить формувати статистичні дані за різні періоди.

1.2 Microsoft Azure Computer Vision API

Microsoft Azure Computer Vision представляє собою складову хмарної платформи Microsoft Azure, яка пропонує розробникам та організаціям можливості в галузі «комп'ютерного зору». Це API забезпечує розширений набір інструментів та функціоналу для аналізу зображень та витягування інформації з них [1].

Цей сервіс може бути використаний в різних галузях, таких як автоматична обробка документів, медична діагностика, відеоспостереження, аналіз соціальних медіа та багатьох інших.

1.2.1 Авторизація

Перед початком роботи за API, необхідно створити акаунт у системі Microsoft Azure, заповнивши усі необхідні поля, такі як назва ресурсу, доменне ім'я, для подальшого його додавання до endpoint, особисті дані для верифікації та інше. Після чого система згенерує особистий endpoint, за допомогою якого

будуть виконуватись усі запити до сервісу та унікальний ключ для доступу до функціоналу (див. рис. 1.1).

Рисунок 1.1 – Приклад згенерованих параметрів доступу

Після чого можна перевірити роботу згенерованого посилання, наприклад, використовуючи потужний інструмент тестування API – Postman [2]. Для цього необхідно обрати тип запиту – GET, POST, PUT, DELETE і т.д., потім ввести необхідний url, задати параметри запиту та натиснути кнопку Send (див. рис. 1.2).

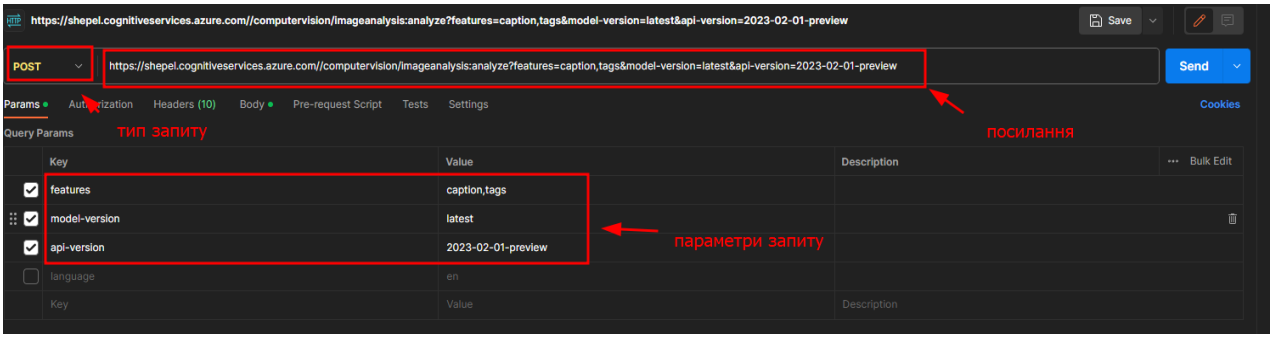

Рисунок 1.2 – Приклад написання запиту у Postman

1.2.2 Отримання даних

Для отримання даних, використовуючи даний сервіс, необхідно використати наступний запит – https://<endpoint>/computervision/imageanalysis:analyze&api-version=2023-02-01-

preview, де endpoint – унікальний, згенерований системою Microsoft Azure, персональний домен, api-version – версія даного інструменту.

Окрім основних, також є параметр features, за допомогою якого можна задати які саме дані необхідно згенерувати. Він має певну кількість значень, які можна використати у запиті для специфікації тих чи інших даних (див. табл. 1.1).

Таблиця 1.1 – Значення параметру features

Назва	Опис
caption	Описує вміст зображення повним реченням
denseCaptions	Створює детальні підписи до 10 помітних областей зображення
tags	Позначає зображення докладним списком слів, пов’язаних із його вмістом
people	Визначає людей, які з’являються на зображеннях

У розробленому проєкті було використано два з перелічених вище значення – caption, для генерації опису завантаженого медіафайлу, та tags, для генерування можливих тегів для посту Instagram .

Окрім цього для роботи API необхідно також додати параметр у саме тіло запиту, для того, щоб система розуміла, який саме файл треба обробити. Якщо використовується зовнішнє посилання, то для цього у тілі запиту необхідно у форматі JSON прописати параметр url та безпосередньо саме посилання на ресурс (див. рис. 1.3). А також у заголовках запиту додати поле Content-Type із значенням application/json. Після чого можна натиснути на клавішу Send для запуску написаного запиту.

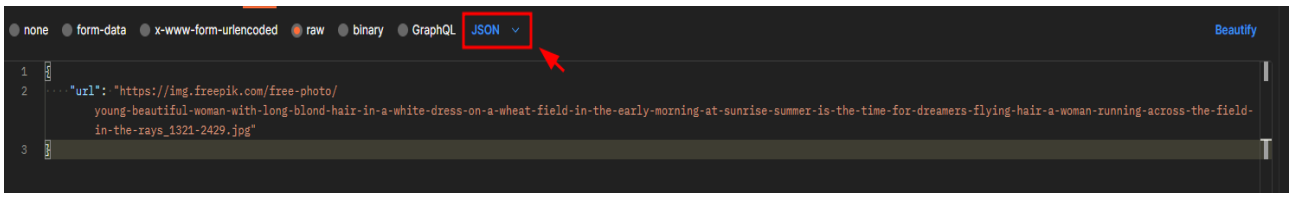

Рисунок 1.3 – Приклад написання тіла запиту

Якщо під час виконання не виникне ніяких проблем, то у якості результату сервіс поверне JSON масив з тими параметрами features, які було вказано у запиті (див. рис. 1.4).

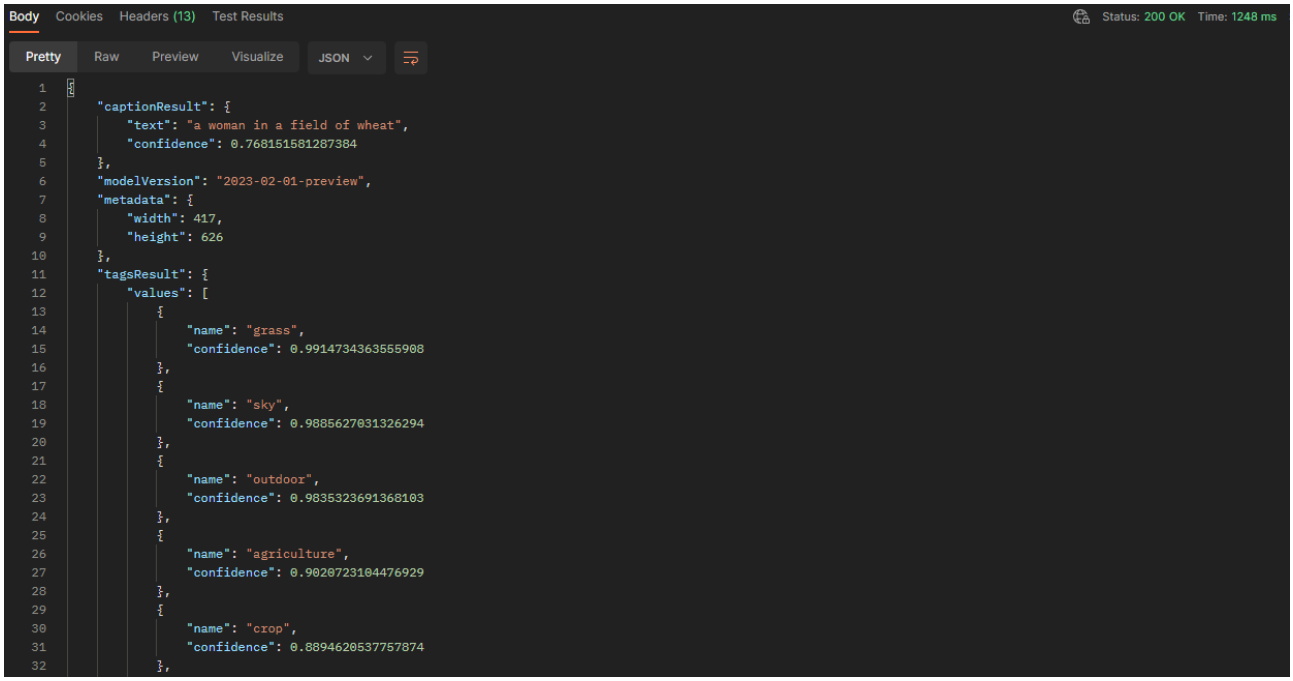

Рисунок 1.4 – Приклад результату аналізу зображення

Як видно з рисунку 1.4, окрім самого значення, система також повертає параметр confidence, який вказує на точність отриманого результату, а також повертає метадані самого зображення – висоту та ширину та версію самого сервісу.

Якщо ж під час роботи виникне якась проблема, ту якості результату API поверне масив з кодом та повідомленням цієї помилки.

1.3 Instagram Graph API

Instagram Graph API, що розроблений компанією Facebook, є потужним та різностороннім інструментом, що дозволяє користувачам отримувати доступ до функціоналу Instagram з власних додатків. З його допомогою відкривається

можливість отримувати та публікувати медіафайли, управляти коментарями, відповідати на них, знаходити контент, в якому згаданий конкретний користувач, а також використовувати хештеги для пошуку медіафайлів. Крім того, API сервіс надає можливість отримання базових метаданих і метрик для користувачів Instagram Business та авторів Instagram.

Важливо відзначити, що основою Instagram Graph API є API Graph для Facebook [3].

1.3.1 Авторизація

Перед початком роботи з даним сервісом, на платформі Meta for Developers необхідно зареєструватися як розробник та створити новий застосунок, через який с подальшому буде виконуватись робота з іншими акаунтами.

Для цього на вказаному ресурсі необхідно обрати пункт Create App та пройшовши декілька кроків реєстрації, де необхідно буде обрати мету створення застосунку та його деталі (див. рис. 1.5).

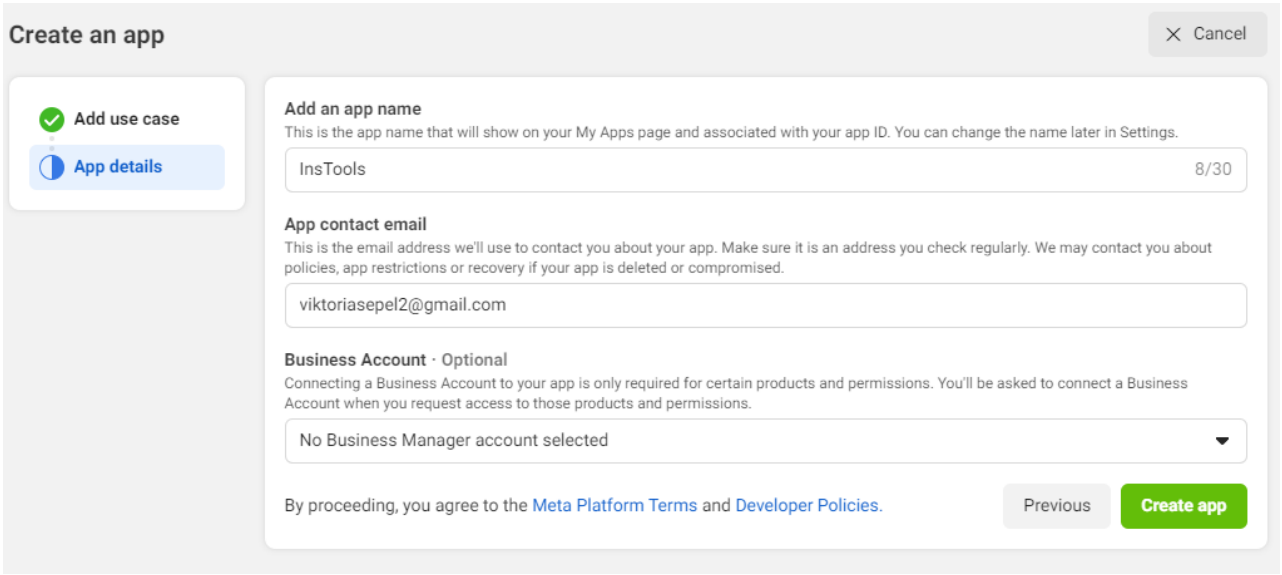

Рисунок 1.5 – Реєстрація нового застосунку у середовищі Meta

Після того, як застосунок зареєстровано, його можна кастомізувати, а саме

додати необхідні посилання, по яких користувач буде переходити для реєстрації, чи завдяки яким будуть зчитуватись введені дані, додаткові ролі юзерів, якщо це потрібно та інше. Після чого можна додавати його безпосередньо до самого проєкту. Для цього системою було згенеровано ключі доступу App ID та App secret (див. рис. 1.6), які необхідно додати до файлу конфігурацій проєкту.

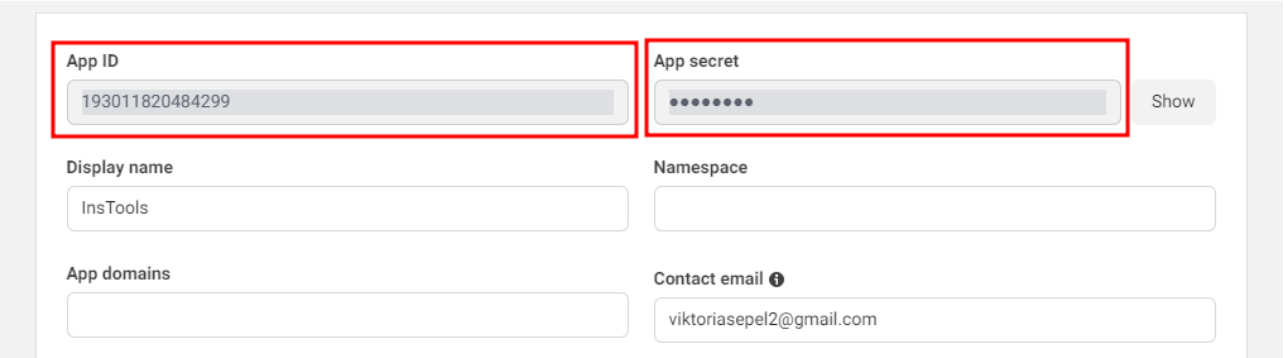

Рисунок 1.6 – Приклад згенерованих реквізитів для підключення

Після цього, коли користувач натисне на кнопку «Увійти через Facebook» у створеному застосунку, його переведе на сторінку логіна від Facebook, де необхідно буде підтвердити надання необхідних прав доступу до профілю. Потім, після успішної реєстрації, його повертає на сайт, а система в свою чергу отримає необхідні дані користувача, а найголовніше – access_token, за допомогою якого й будуть виконуватись запити до Instagram саме цього юзера.

1.3.2 Побудова запиту

Для виконання запиту до профілю користувача, необхідно побудувати певне посилання. Основою цього посилання є запит – https://graph.facebook.com/v18.0/, де вказується версія API. Надалі, після слешу додаються вже необхідні деталі.

Можна виділити декілька основних груп запитів, що було використано у роботі (див. табл. 1.2).

Таблиця 1.2 – Групи запитів у Instagram Graph API

Група	Опис
IG Comment	Відповідає за коментарі в Instagram
IG Container	Відповідає за контейнер медіафайлу для публікації в Instagram
IG Container	Відповідає за контейнер медіафайлу для публікації в Instagram
IG Hashtag	Відповідає за хештег в Instagram.
IG Media	Відповідає за фото, відео, історію чи альбом в Instagram
IG Media	Відповідає за фото, відео, історію чи альбом в Instagram
IG User	Відповідає за акаунт Instagram Business чи акаунт автора

Основною з них, звісно, є група IG User, що відповідає безпосередньо за сам акаунт та дані про нього: кількість підписників, коментарів, лайків, ім’я профілю та контактні дані. Завдяки цим даним можна будувати різноманітні графіки зі статистикою, що дозволить більш детально аналізувати якість та популярність того чи іншого профілю.

Окрім цього велику роль у додатку відіграє й група IG Media, що відповідає за контент на сторінці. Ця група дозволить отримати не тільки усі наявні дописи у користувача, а також сформувати певну статистику по кожному з них, що значно покращує можливості контролю якості контенту.

Для того, щоб створити новий пост, необхідно буде також використовувати IG Container, тому що весь процес у цьому сервісі розподіляється на 2 етапи. На першому для публікації створюється окремий контейнер, де зберігається й сам медіафайл, й уся необхідна інформація. Існує він певний проміжок часу, що дозволяє створити так звану «відкладену» публікацію. Після чого, на другому етапі відбувається безпосередньо публікація самого контейнеру на сторінці Instagram.

Також для кожного запиту є певні необхідні параметри, щоб сервіс міг отримати дані та перевірити наявні дозволи. Основний та необхідний у кожному

запиті – access_token. Це унікальний ідентифікатор юзера, який було отримано після його аунтефікації для того, щоб система розуміла для кого саме необхідно отримати дані та чи є в нього необхідні права доступу.

Другий – ідентифікатор елементу, до якого необхідно отримати доступ, наприклад, якщо запит відноситься до групи IG Media, то обов'язковим параметром буде ig-media-id – ідентифікатор медіаресурсу, якщо до IG User, то ig-user-id – ідентифікатор користувача, до даних якого будуть посилатись запити, для інших груп аналогічно.

Також у запиті необхідно вказати, які саме дані потрібно повернути у результаті. Для цього буде використаний параметр fields – розділений комою список полів. Наприклад, які саме медіафайли необхідно дістати, чи навпаки, відкинути ті поля, що наразі не є потрібними. Цей параметр вже не є обов'язковим, на відміну від попередніх двох.

1.3.3 Постинг у Facebook

Окрім роботи з Instagram, API сервіс також дозволяє зробити публікацію одразу й у Facebook, що гарно розширює функціонал та допомагає охватити більшу аудиторію. Так як для роботи з Instagram необхідна сторінка у Facebook, то у програми є також ідентифікатор цієї сторінки, що дозволяє працювати також і з даною платформою.

Процес постингу у Facebook складається з двох етапів. На першому з них необхідно додати усі медіафайли у саму систему, задля того, щоб використати їх у подальшому. На другому ж і відбувається вже сам процес створення посту з текстом, зображеннями та іншими необхідними даними, наприклад геолокацією і, після цього, викладення його на особисту сторінку. Також, яке і у випадку з публікуванням у Instagram, у Facebook є можливість «відкласти» пост на певну дату, що також є дуже зручною функцією для контентмейкера.

Для того щоб завантажити медіаресурс до свого профілю, щоб у

подальшому мати можливість додати його до допису, необхідно скористатись запитом https://graph.facebook.com/v18.0/page_id/photos, де page_id – особистий ідентифікатор користувача у Facebook. Окрім цього є також необхідний параметр, url – посилання на обране фото, яке необхідно опублікувати. Окрім цього в тілі запиту також необхідно вказати параметр published зі значенням false, що вказує системі на те, що публікувати даний файл не потрібно.

Якщо ж необхідно зробити повноцінний допис з текстом, необхідно скористатись наступним запитом: https://graph.facebook.com/v18.0/page_id/feed, де, як і у попередньому необхідно вказати ідентифікатор користувача та певний перелік параметрів самої публікації (див. табл. 1.3).

Таблиця 1.3 – Параметри запиту для публікації посту у Facebook

Параметр	Опис
message	Текст посту.
link	Посилання на певний медіа або інший ресурс.
published	Опублікувати одразу чи запланувати публікацію.
scheduled_publish_time	Дата публікації посту.

Окрім цього також можна встановити таргетинг, для обмеження перегляну публікації у певних країнах чи містах для більш конкретної та вузьконаправленої авдиторії (див. рис. 1.7).

```
-d '{
      ...
      "targeting": {
        "geo_locations": {
          "countries": [
            "CA"
          ],
          "cities": [
            {
              "key": "296875",
              "name": "Toronto"
            }
          ]
        }
      },
      ...
    }'
```

Рисунок 1.7 – Приклад використання таргетингу у запиті

Після цього при успішній публікації API поверне її ідентифікатор. Якщо ж ні, то повернеться JSON об’єкт з інформацією про помилки, які виникли під час обробки запиту.

2 ПРОЄКТУВАННЯ ВЕБДОДАТКУ «INSTOOLS»

2.1 Use case diagram

Use Case Diagram є інструментом, який дозволяє візуалізувати взаємодію різних ролей з системою. Незважаючи на те, що вона не відображає конкретний порядок виконання кроків, вона ілюструє типи ролей та їх взаємодію, наголошуючи на функціональних вимогах системи з погляду користувача. Може бути представлений як текстовий опис або у вигляді графічної діаграми [4].

Основними елементами даної діаграми є актор та прецедент. Актор може бути людиною чи іншою системою, підсистемою чи класом, які представляють щось поза сутністю. Графічно учасник зображується у вигляді «людинки». Прецедент же представляє собою поведінку сутності, описуючи взаємодію між учасниками та системою. Прецедент не показує «як» досягається певний результат, а лише «що» саме виконується. Вони зазвичай позначаються дуже простим способом – у вигляді еліпса, всередині якого вказано дію [5-6].

На діаграмі варіантів використання (див. рис. 2.1) представлені основні варіанти використання системи для користувача сайту.

2.2 Опис варіантів використання

Усі наступні перелічені прецеденти можуть бути здійснені користувачем сайту.

Прецедент «Переглянути основну інформацію публікації»

Призначення: даний варіант використання надає можливість користувачу переглянути основну інформацію кожного наявного на його сторінці поста у таблиці.

Основний потік подій: починає виконуватися, коли користувач системи

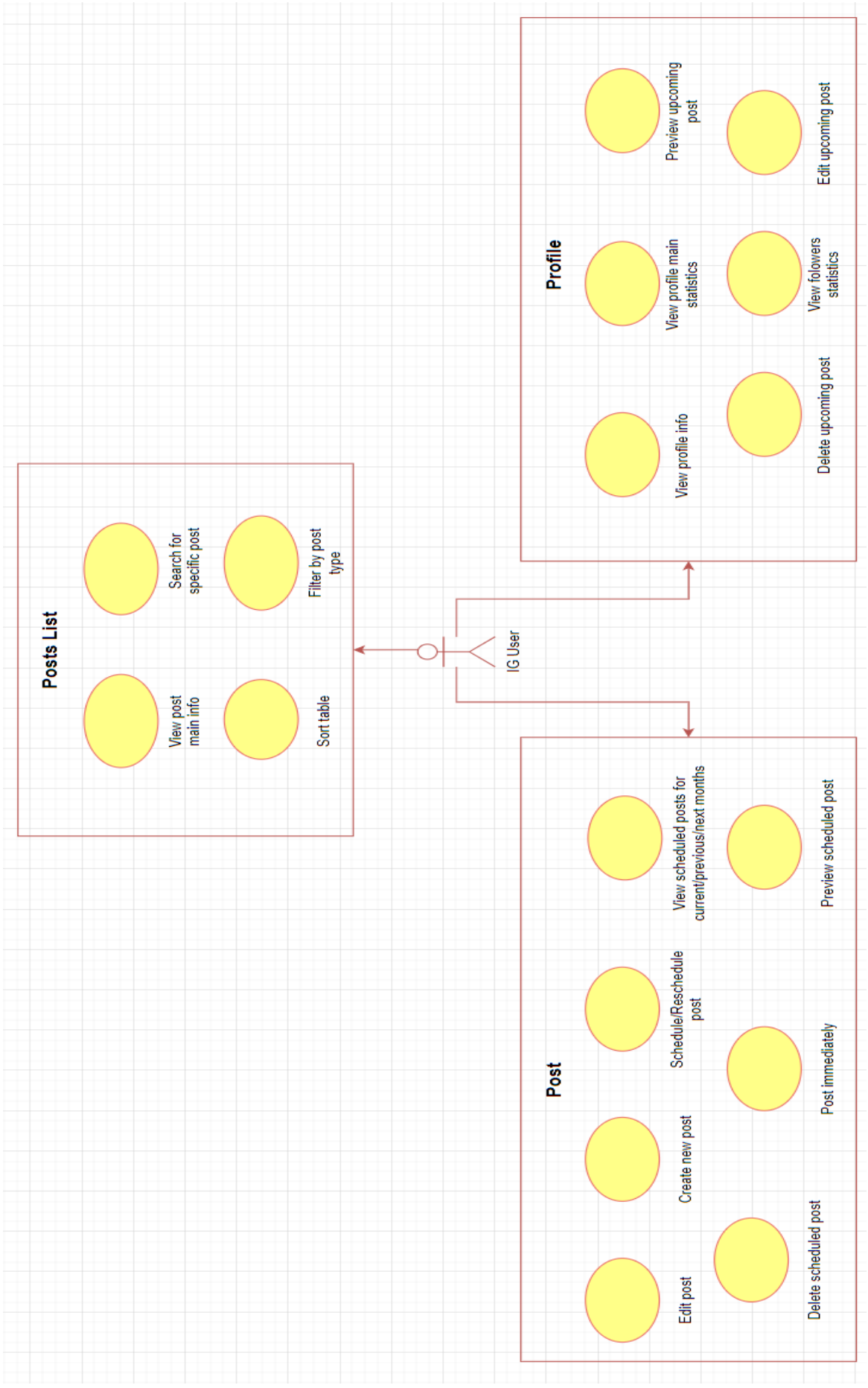

Рисунок 2.1 – Діаграма варіантів використання для користувача

заходить на сайт.

Передумова: у базі сайту є інформація про хоча б один пост користувача.

Прецедент «Знайти конкретний пост»

Призначення: користувач має змогу знайти у таблиці певні пости за описом.

Основний потік подій: користувач системи вводить текст у поле пошуку на сторінці таблиці постів.

Передумова: у базі сайту є інформація про хоча б один пост користувача, користувач ввів хоча б один символ у пошукове поле.

Прецедент «Сортувати таблицю»

Призначення: даний прецедент надає можливість користувачу відсортувати дані в таблиці за певним стовбцем.

Основний потік подій: варіант починає виконуватися, коли користувач натискає на необхідний стовбець у таблиці.

Передумова: у базі сайту є інформація про хоча б один пост користувача, користувач натиснув на необхідний стовпчик.

Прецедент «Відфільтрувати таблицю за типом контенту»

Призначення: у даному варіанті користувач може відфільтрувати дані в таблиці за типом контенту.

Основний потік подій: починає діяти, коли користувач натискає на необхідний пункт у списку типів.

Передумова: у базі сайту є інформація про хоча б один пост користувача, користувач обрав необхідний пункт для фільтрації.

Прецедент «Створити нову публікацію»

Призначення: юзер має змогу створити нову публікацію на ресурсі.

Основний потік подій: користувач переходить до сторінки «Нова Публікація», або натискає одну з наявних кнопок створення на сайті.

Передумова: даний варіант доступний у будь-який момент.

Прецедент «Редагувати створену публікацію»

Призначення: у цьому варіанті юзер може відредагувати створену на сайті

публікацію.

Основний потік подій: користувач натискає на кнопку редагування обраного посту, який він вже створив.

Передумова: у базі сайту є хоча б один створений пост поточного користувача.

Прецедент «Встановити дату постингу»

Призначення: у даному процесі користувач має змогу встановити певну дату публікації посту.

Основний потік подій: користувач обирає дату публікації у відповідно полі публікації.

Передумова: у базі даних є хоча б один створений користувачем пост.

Прецедент «Переглянути заплановані пости»

Призначення: користувач має можливість переглянути усі заплановані пости у форматі календаря помісячно.

Основний потік подій: поточний варіант починає працювати, коли юзер натискає кнопку «Calendar» у навігаційному меню.

Передумова: сторінка «Calendar» активна.

Прецедент «Запостити моментально»

Призначення: у даному прецеденті користувач може зробити публікацію посту негайно.

Основний потік подій: прецедент починає виконуватись, коли користувач натискає відповідну кнопку на запланованому пості або під час його створення.

Передумова: сторінка «Calendar» активна та у базі є запис про хоча б один запланований пост, якщо необхідно зробити публікацію запланованого посту, або прецедент працює у будь-якому випадку під час створення посту.

Прецедент «Видалити заплановай пост»

Призначення: користувач може видалити заплановану публікацію.

Основний потік подій: варіант виконується, коли користувач натискає на відповідну кнопку на запланованому пості на сторінці «Calendar» або «Dashboard».

Передумова: у базі є запис про хоча б один запланований пост.

Прецедент «Переглянути інформацію профілю»

Призначення: даний варіант використання дозволяє подивитись основну інформацію профілю Instagram.

Основний потік подій: користувач натискає на кнопку «Dashboard» у навігаційному меню.

Передумова: у базі є запис хоча б про одного користувача.

Прецедент «Переглянути основну статистику»

Призначення: прецедент дозволяє переглянути основну статистику профілю – кількість переглядів, коментарів, підписок та лайків.

Основний потік подій: починає діяти коли юзер переходить на сторінку «Dashboard».

Передумова: у базі створено запис про хоча б про одного користувача.

Прецедент «Переглянути графік підписок»

Призначення: надає змогу користувачеві переглянути статистику підписок на сторінку у форматі графіку.

Основний потік подій: виконується при переході до сторінки «Dashboard» через навігаційне меню.

Передумова: у базі існує хоча б один користувач.

Прецедент «Переглянути майбутній пост»

Призначення: користувач має змогу переглянути найближчий майбутній пост.

Основний потік подій: прецедент починає виконуватись коли користувач натискає на майбутній пост на сторінці «Dashboard».

Передумова: у базі існує хоча б один запланований пост.

Прецедент «Редагувати майбутній пост»

Призначення: прецедент дає можливість відредагувати найближчий пост, який буде опубліковано.

Основний потік подій: користувач натискає на кнопку редагування посту на домашній сторінці.

Передумова: у базі є запис про хоча б про одну заплановану публікацію.

Прецедент «Видалити майбутній пост»

Призначення: варіант використання дає змогу юзеру видалити майбутній пост до його публікації.

Основний потік подій: починає виконуватись коли юзер натискає на кнопку видалення на домашній сторінці.

Передумова: у базі існує хоча б один запланований пост.

2.3 Class diagram

Діаграми класів представляють собою конкретний вид структурних графічних зображень, спроєктованих для моделювання об’єктно-орієнтованих систем. Вони відображають класи, їхні атрибути, методи та взаємовідносини між класами.

Основна мета цих діаграм полягає в тому, щоб детально розглядати систему як компоненти та розуміти їх взаємодію.

На діаграмі класи представлені у вигляді прямокутників, які містять три відділення:

– верхня частина містить назву класу (вона надрукована напівжирним шрифтом і вирівняна по центру, починається з великої літери);
– середнє відділення містить атрибути класу (вони вирівняні по лівому краю і перша літера мала);
– нижній відсік містить операції, які може виконувати клас (вони також вирівняні по лівому краю і перша літера мала).

Можливе додання четвертого відділення, яке містить коментарі [7].

Перед тим, як розробити готовий додаток, необхідно спроєктувати усі компоненти, класи та взаємодії між ними, щоб надалі процес побудови додатку був простіший та більш зрозумілий (див. рис. 2.2).

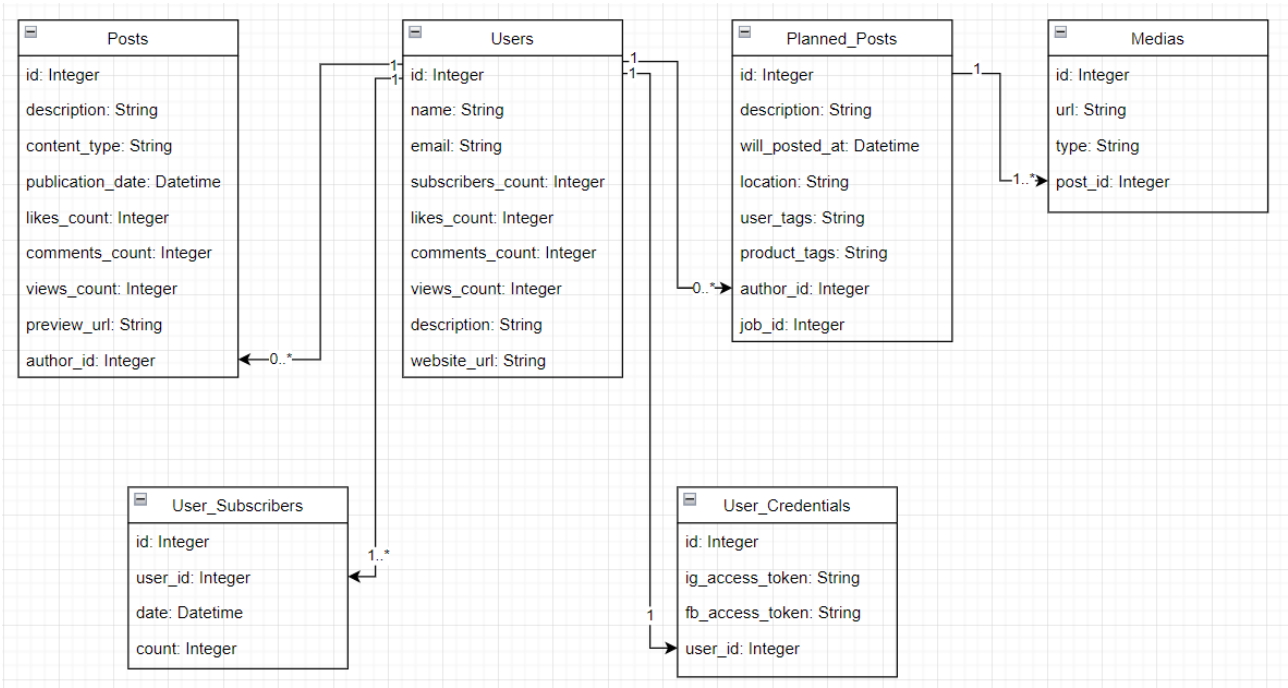

Рисунок 2.2 – Діаграма класів для розробленої програми

На даному етапі розробки було виділено 6 основних класів: Posts, Planned Posts, Users, User Subscribers, User Credentials, Medias та зв’язки між ними.

2.4 Опис класів діаграми

Для повного розуміння принципу роботи кожного класу та взаємодії з іншими, необхідно скласти загальний опис на основі створеної діаграми. Це також допомагає зрозуміти подальший певний перелік необхідних методів для роботи.

2.4.1 Клас Users

Опис: клас Users відповідає за представлення користувача у системі. Він зберігає у собі усю основну інформацію про юзера: статистику його профілю персональні дані та посилання на особистий вебсайт, якщо такий є. Окрім цього

цей клас є основною сполучною ланкою між усіма іншими, так як саме на основі його даних відбувається пошук та запис до інших сутностей.

Зв'язок з іншими класами: основними двома групами зв'язків у даного класу є: Posts та додаткові класи для даних користувача. У першому випадку це зв'язок з сутністю Posts по полю users.id-posts.author_id, для подальшого отримання списку наявних на сторінці постів для конкретного користувача, та з класом Planned_Posts по полю users.id-planned_posts.author_id за тим же самим принципом, як у попередньому. А у другому випадку це зв'язок з класом User-Credentials по полю users.id-user_credentials.user_id для того, щоб під час виконання запитів програма направляла його для необхідного юзера. А також зв'язок з сутністю User_Subscribers по полю users.id-user_subscribers.user_id, завдяки чому на основі даних зі зв'язної таблиці можна буде будувати статистичний графік підписок користувача.

2.4.2 Клас User_Subscribers

Опис: даний клас відповідає за збереження даних про кількість підписників користувача за певний проміжок часу, для подальшої побудови графіку та формування певної статистики.

Зв'язок з іншими класами: дана сутність пов'язана лише з класом Users по полю users.id-user_subscribers.user_id, як вже було описано вище для того, щоб отримувати дані лише для конкретного юзера.

2.4.3 Клас User_Credentials

Опис: ця сутність представляє собою сховище ключів доступу кожного користувача до Instagram та Facebook. Його було створено для того, щоб не ускладнювати таблицю Users та мати змогу розширювати список підключених

сервісів.

Зв'язок з іншими класами: даний клас пов'язаний лише з Users по полю users.id-user_ credentials.user_id, для того щоб програма використовувала лише коректні токени.

2.4.4 Клас Posts

Опис: клас Posts необхідний для того, щоб програма не запитувала кожний раз наявні пости у сервісу при переході на сторінку таблиці, а виконувала це лише раз у певний період. У ньому зберігається уся необхідна інформація про кожен пост юзера, що вже існує на його сторінці: кількість переглядів, коментарів, дата публікації, опис та посилання на прев'ю картинку.

Зв'язок з іншими класами: ця сутність пов'язала з класом Users через поле users.id-posts.author_id для того, щоб під час обробки бралися і використовувались лише дані для конкретного юзера, а також під час їхнього оновлення для подальшого відображення найсвіжішої статистики.

2.4.5 Клас Planned_Posts

Опис: клас Planned_Posts необхідний для того, щоб зберегти дані про запланований пост до моменту його публікації у профілі. Тут зберігається уся введена інформація: опис, відмітки людей чи товарів, геолокація (якщо вона була встановлена) та дата публікації. Так як у даної сутності та у попередньої Posts різне призначення то поля, заплановані пости й було винесено у окрему таблицю, щоб не перенавантажувати Posts та збільшити швидкість роботи самого сайту.

Зв'язок з іншими класами: цей клас пов'язаний з Users по схемі users.id-planned_posts.author_id, задля того, щоб публікувати їх на коректну сторінку та відображати у календарі для необхідного користувача. А також з класом Medias

по полю planned_posts.id-medias.post_id, задля того щоб під час відображення або публікації, відправляти на сервіс необхідні медіафайли, а також не перенавантажувати сам клас Planned_Posts.

2.4.6 Клас Medias

Опис: сутність Medias створена для того, щоб зберігати дані про кожен доданий медіаресурс під час створення публікації. В ньому зберігається посилання на сам ресурс та ідентифікатор посту, до якого цей ресурс буде додано. Завдяки йому до одного посту можна зберігати одразу декілька зображень та не перенавантажувати сам основний клас Planned_Posts, що само собою впливає на швидкість роботи сайту.

Зв'язок з іншими класами: даний клас пов'язаний лише з класом Planned_Posts по полю planned_posts.id-medias.post_id.

3 РЕАЛІЗАЦІЯ ВЕБДОДАТКУ «INSTOOLS»

3.1 Огляд UI сторінок сайту

VueJS – це фреймворк, що використовується для розробки користувацьких інтерфейсів на JavaScript. Він базується на стандартних мовах розробки, таких як HTML, CSS і JavaScript, і має можливість декларативно програмувати користувацькі інтерфейси за допомогою компонентів [8].

Цей фреймворк вирізняється своєю високою гнучкістю та легкістю використання, що дуже полегшує процес розробки вебсайтів. Більш того, завдяки вбудованій реактивності, вам не потрібно самостійно налаштовувати систему чи створювати власні методи для відстеження змін у елементах сторінки – Vue.js здійснить це автоматично.

Laravel – це фреймворк для створення вебдодатків із видатним та елегантним синтаксисом. Цей фреймворк надає не лише основу для створення програм, але і точку виходу, щоб ви могли фокусуватися на творчому процесі, поки деталі обробляються автоматично. Laravel об’єднує найкращі пакети у PHP-екосистемі, створюючи надійну та зручну структуру для розробників [9].

Так як в даній системі не має необхідності у адміністраторі, адміністративної панелі створено не було. Основний наголос було зроблено на широкий та різноманітний функціонал для самого Instagram користувача.

3.1.1 Authorization

Коли користувач потрапляє на сайт, якщо він ще не авторизований, його перекидає на сторінку логіна за адресою: /auth, де йому необхідно натиснути на відповідну кнопку (див. рис. 3.1).

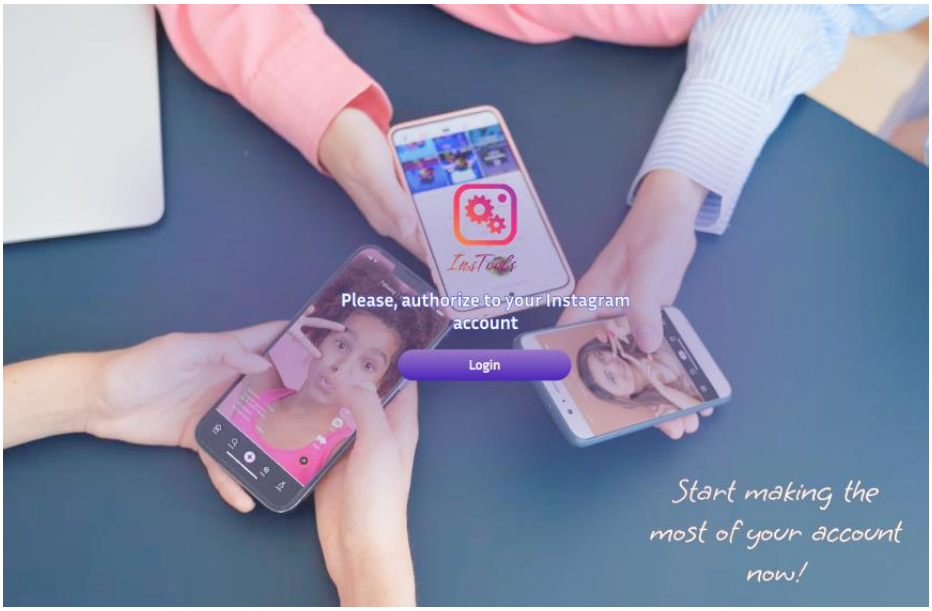

Рисунок 3.1 – Сторінка авторизації на сайті

Після натискання, юзера перекидає на сторінку самого Instagram, де користувач повинен або ввести свої необхідні дані, або ж, якщо це необхідно, скористатися логіном через Facebook (див. рис. 3.2).

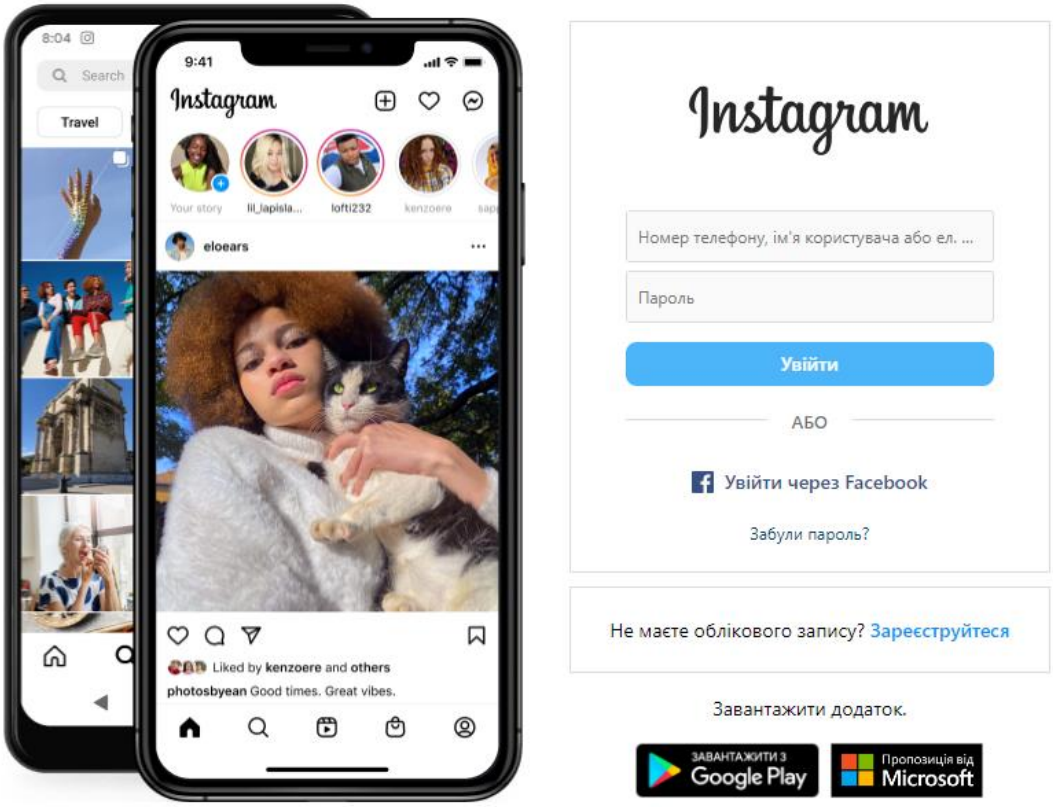

Рисунок 3.2 – Сторінка авторизації у Instagram

Якщо усі дані було введено коректно, то користувача пересилає на домашню сторінку створеного сайту з отриманою з його профілю інформацією. У протилежному випадку буде відображено сторінку з помилкою, яка виникла.

3.1.2 Dashboard

Після того, як користувач успішно авторизувався через свій профіль Instagram, він потрапляє на домашню сторінку за посиланням: /dashboard (див. рис. 3.3).

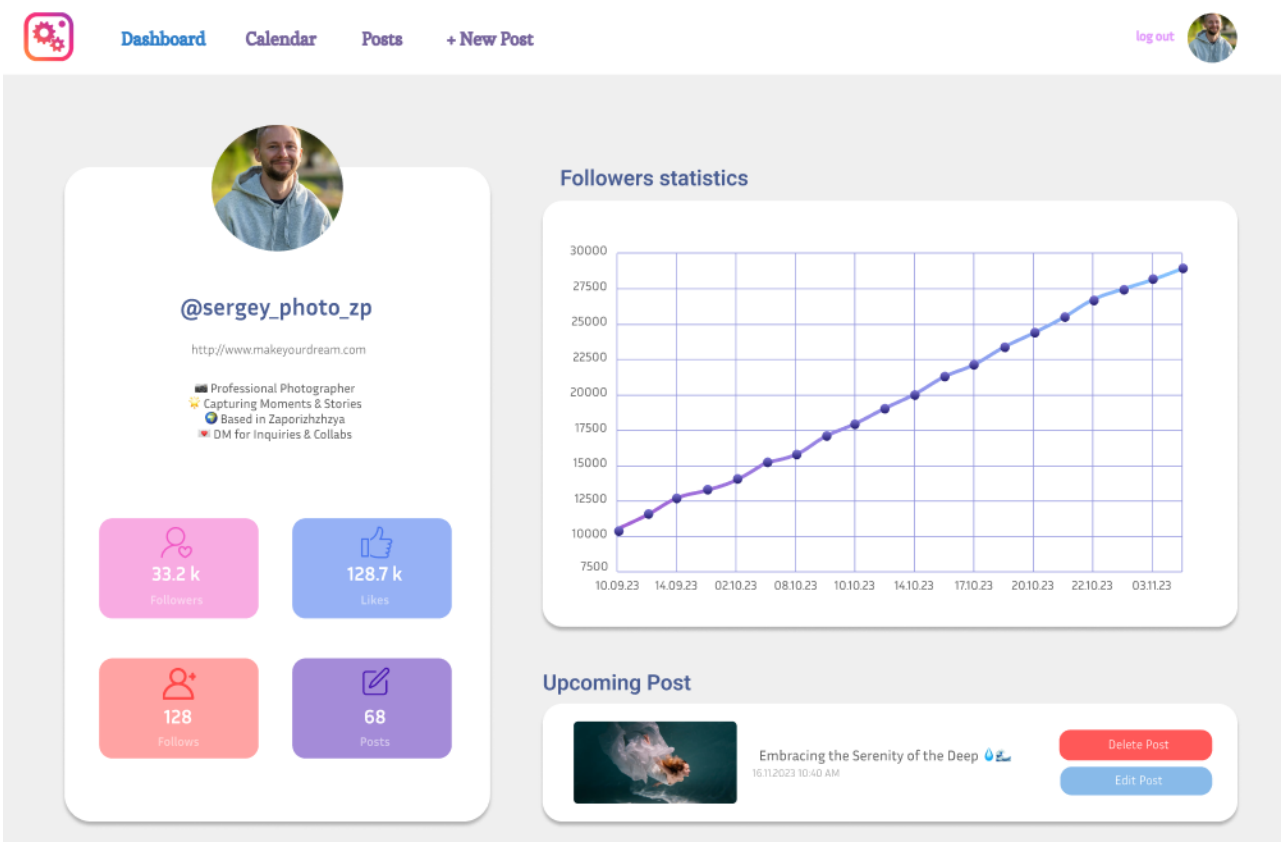

Рисунок 3.3 – Домашня сторінка користувача

Тут він може побачити усю основну інформацію зі свого профілю: ім’я користувача, опис, посилання на власний сайт, якщо таке існує, та основне фото.

Окрім цього, під цими даними також знаходиться основна статистика, що оновлюється кожен день: кількість підписників, переглядів профілю, коментарів

та відміток «сподобалось». Це допоможе йому відстежувати активність без постійної перевірки у самому Instagram.

Також у правій частині сторінки можна побачити 2 блоки – графік підписок та майбутній запланований пост, якщо такий вже існує. Перший відображає статистичний графік кількості підписників за певний проміжок часу, на даний момент це місяць, при наведені на певну відмітку можна побачити точну кількість на обрану дату (див. рис. 3.4).

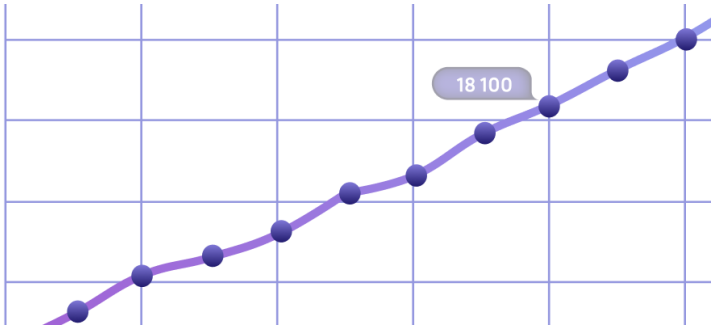

Рисунок 3.4 – Відображення конкретної кількості підписників за певною датою

Другий же блок відображає найближчу заплановану публікацію, якщо, як вже було сказано вище, хоча б одна така існує у даного юзера. У ньому відображається прев'ю медіа, яке буде відображатись при відкритті публікації у самому Instagram, короткий опис та сама дата постингу. Окрім цього одразу є можливість відредагувати, за необхідності, чи одразу видалити пост. Це значно спрощує роботу на сайті за рахунок того, що користувачу не потрібно переходити окремо на сторінку календаря та шукати найближчий запланований пост.

3.1.3 Calendar Page

На навігаційній панелі сайту також знаходяться й інші доступні сторінки. Однією з таких є календар запланованих публікацій, на який можна потрапити за URL: /calendar (див. рис. 3.5).

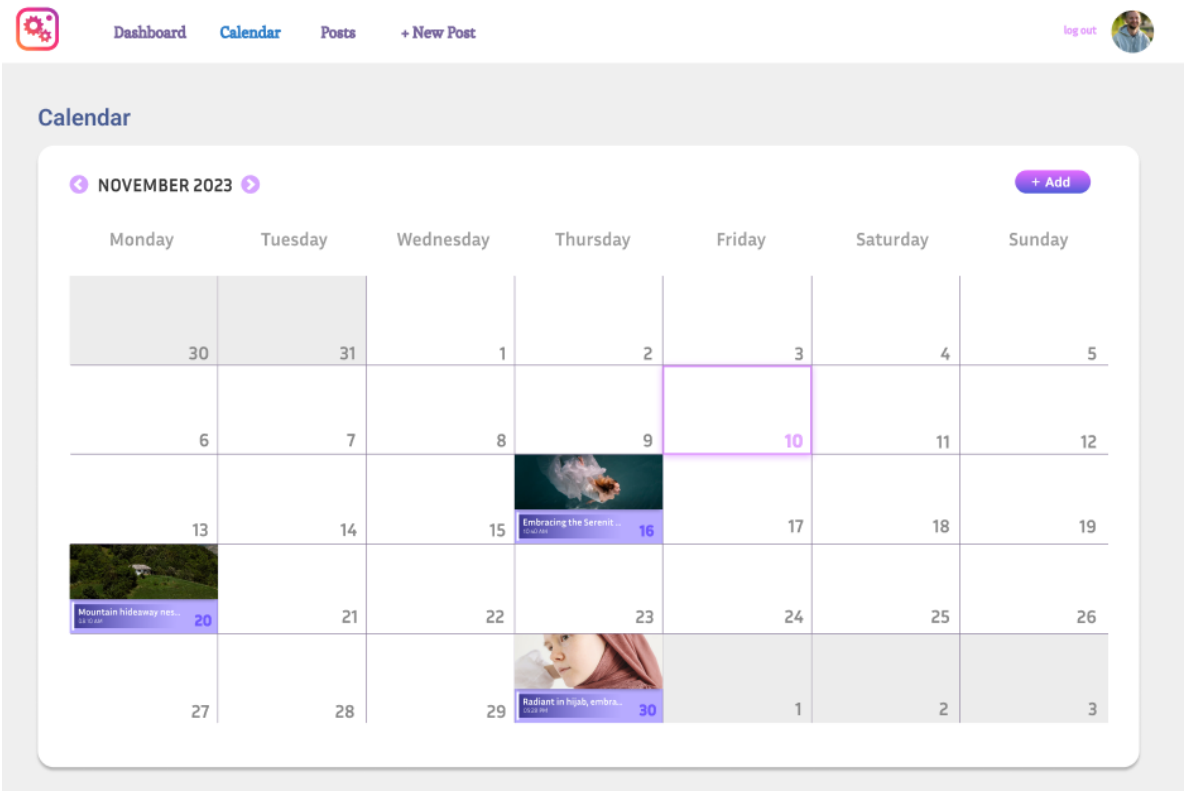

Рисунок 3.5 – Сторінка календаря запланованих постів

Тут юзер має змогу передивитись усі заплановані на певний місяць пости, з можливістю їхнього видалення або редагування. Сам пост відображається у форматі зображення-прев'ю, короткого опису та часу публікації.

Також користувач має змогу переглянути публікації й за інші місяці. Для цього необхідно скористатись стрілочками, що знаходяться по боках від назви місяця, що відображається на даний момент. А також є можливість створити новий пост, натиснувши на відповідну кнопку у правій частині календаря. При натисканні буде перейдено на окрему сторінку створення задля зручності для самого користувача.

3.1.4 Posts List

Окрім можливості перегляду запланованих постів, у користувача також є можливість переглянути вже наявні на його сторінці. Це можна зробити

перейшовши за URL: /posts-list або натиснувши відповідний пункт у навігаційній панелі (див. рис. 3.6).

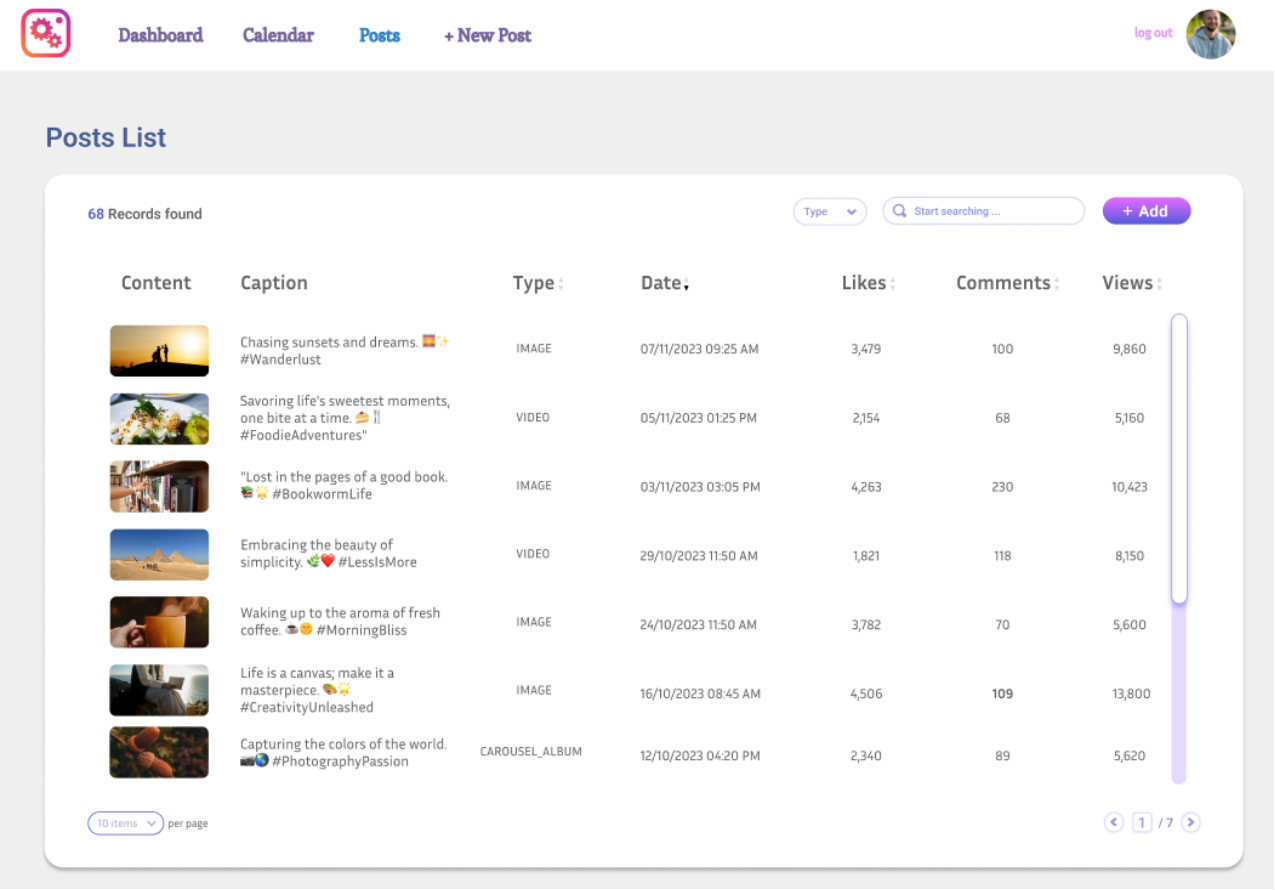

Рисунок 3.6 – Сторінка списку наявних постів

На цій сторінці користувач має змогу у форматі таблиці переглянути основну інформацію про свої пости, а саме: прев'ю самого посту та його опис, дату та тип самої публікації, а також усю основну статистичну інформацію, таку як кількість лайків, переглядів та коментарів, ці дані оновлюються кожен день задля того, щоб користувач завжди мав свіжу та актуальну інформацію.

Окрім цього також можна взаємодіяти з таблицею, наприклад обрати кількість записів, що буде відображатись на одній сторінці, за замовчуванням це 10. Також переключатись між сторінками за допомогою стрілочок, або вручну ввести номер необхідної сторінки. Є можливість відображення записів за певним порядком у деяких колонок, наприклад за замовчуванням дані відображаються в залежності від дати публікації від найсвіжішої. При натисканні на колонку, що

має відповідний значок справа від назви, можна відсортувати таблицю в порядку за іншою колонкою, чи в протилежному напрямку.

У правій частині сторінки знаходиться декілька елементів для можливої взаємодії. Перший з них – кнопка створення нового посту. При її натисканні, як і у випадку на сторінці Календаря, у новому вікні відкриється сторінка створення задля зручності для самого юзера.

Другий елемент надає можливість виконувати пошук по даним за описом публікації. Для цього у поле вводу необхідно ввести бажаний текст та натиснути клавішу Enter (див. рис. 3.7).

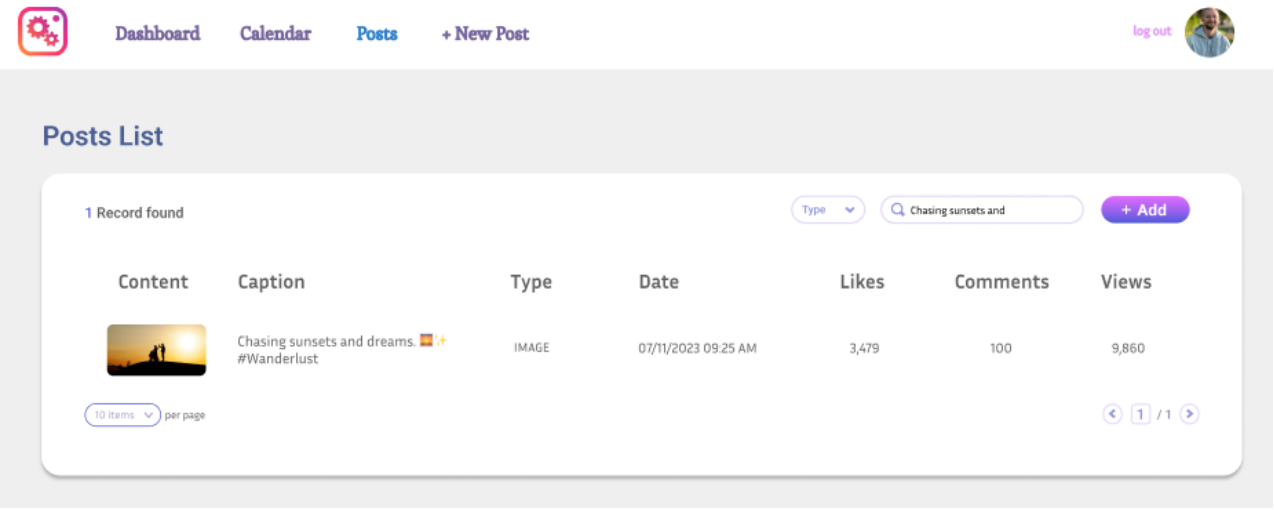

Рисунок 3.7 – Приклад пошуку запису за описом

Якщо у базі існують записи, описи яких повністю чи частково співпадають з введеними даними, то у таблиці відобразяться знайдені співпадіння. У протилежному ж випадку буде виведено інформацію про те, що таблиця пуста, що й позначає, що не було знайдено потрібних записів.

Останньою можливістю, що наявна на даній сторінці, є сортування записів за типом самої публікації. Для цього необхідно уде скористуватись селектором, що знаходиться зліва від пошукової строки, та зі списку, що з’явиться. Обрати необхідний тип з наявних, а саме: IMAGE, VIDEO, CAROUSEL_ALBUM (див. рис. 3.8).

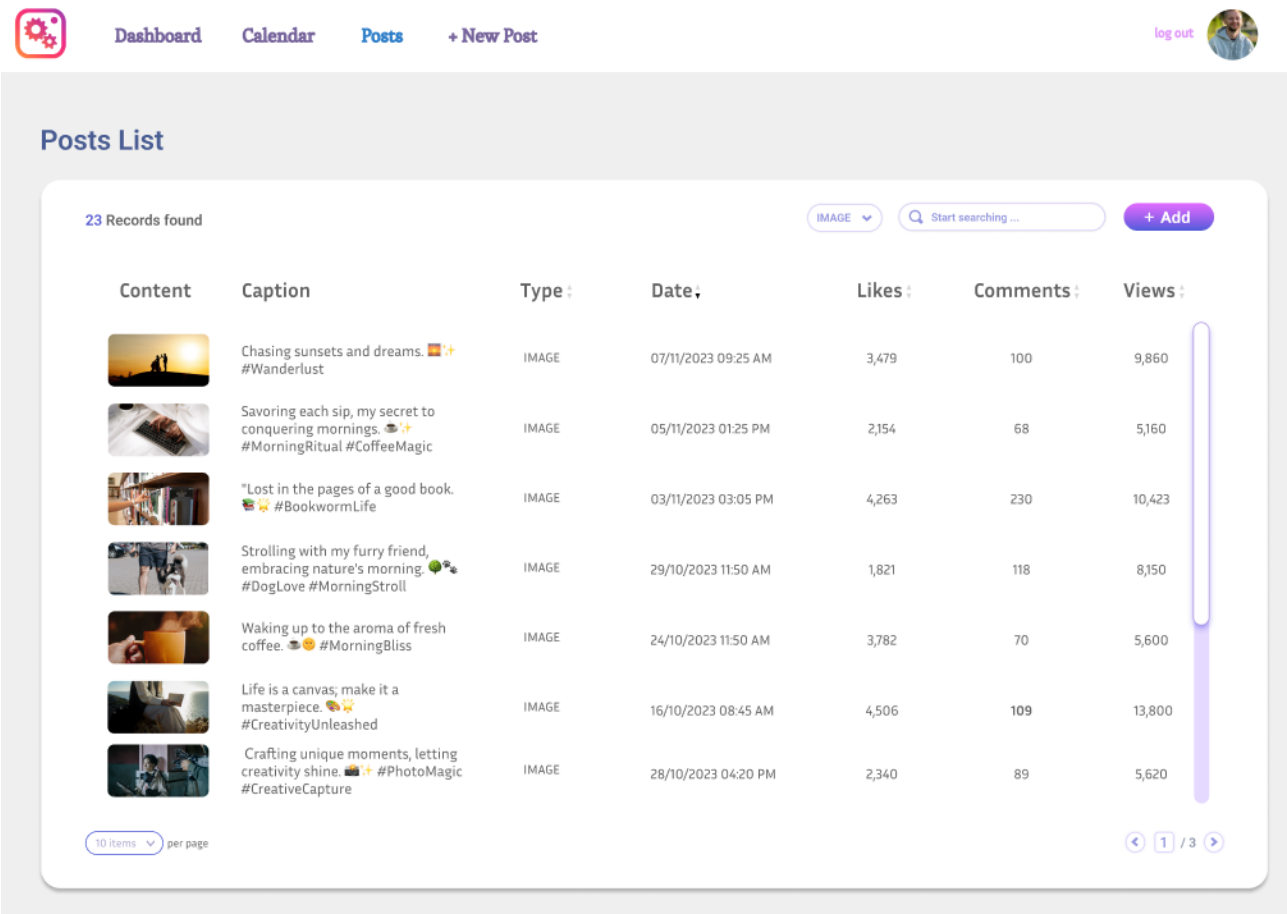

Рисунок 3.8 – Приклад сортування за типом IMAGE

Як і у випадку з пошуком, якщо у базі є записи з відповідним типом, то вони відобразяться у таблиці. У противному випадку, якщо не було знайдено жодного запису, таблиця відобразить інформацію про те, що вона пуста.

3.1.5 Create New Post

Як вже було описано вище, для того, щоб створити новий пост, користувач може скористатися відповідною кнопкою на сторінці Calendar чи Posts List, а також у навігаційній панелі є така ж кнопка, що доступна з будь-якої сторінки за умови того, що юзер авторизований.

При натисканні на будь-яку з них користувача пересилає на URL /new-post (див. рис. 3.9).

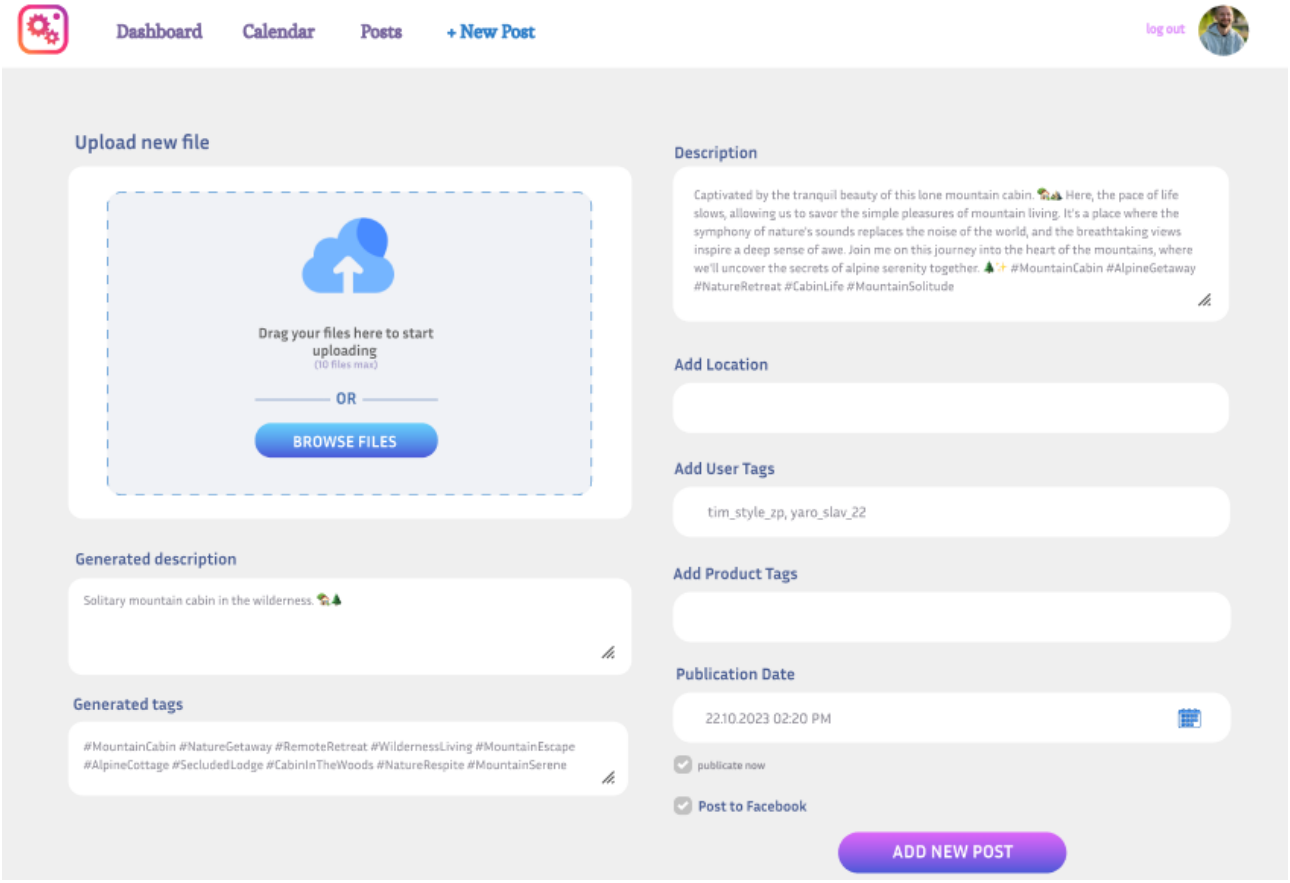

Рисунок 3.9 – Сторінка створення нової публікації

Елементи на даній сторінці розділено на дві колонки, перша з них відповідає за роботу з медіа контентом, друга – за наповнення самої публікації. У першій колонці знаходиться три постійні елементи та один, що з’являється лише коли було обрано хоча б один медіафайл. У другій завжди п’ять елементів та кнопка, при натисканні на яку й буде створено запис про сам пост у базі даних, та за необхідності, опубліковано одразу.

Перший елемент з лівої колонки – селектор фото, які будуть представлені у пості. У користувача є можливість як клікнути на кнопку BROWSE FILES, після чого відкриється провідник, де можна обрати необхідні файли. А також можна скористатись можливістю Drag-and-drop, що означає, що користувач може просто перетягнути необхідні файли у область селекту.

Після чого з’явиться описаний вище додатковий блок, де будуть показані обрані медіафайли (див. рис. 3.10).

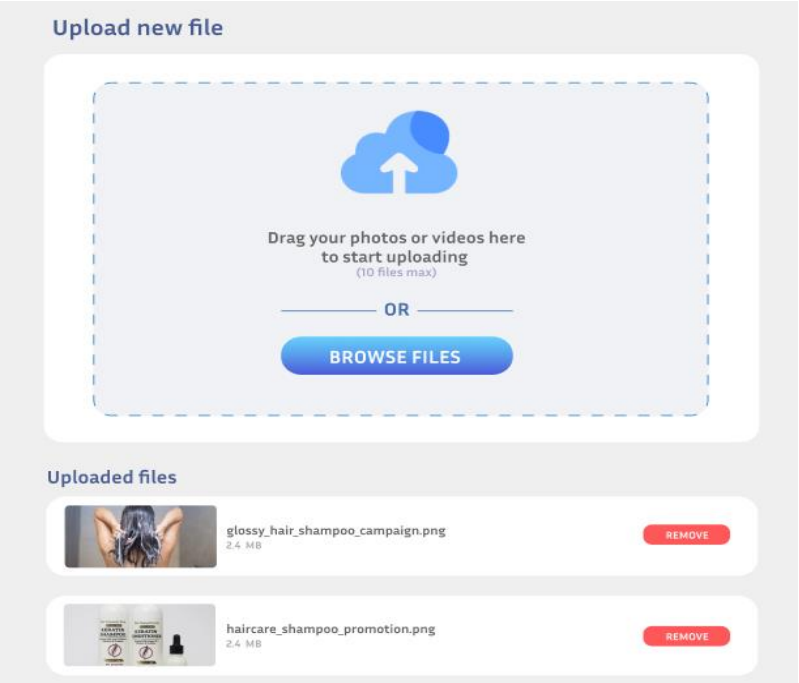

Рисунок 3.10 – Приклад відображення обраних файлів

Кожен блок файлу складається з прев'ю зображення, назви файлу та його розміру. Також у користувача у будь-який момент є можливість видалити файл, скориставшись відповідною кнопкою напроти назви та розміру.

Під селектором знаходяться ще два блоки з текстовими полями. Так як у даного ресурсу є інтеграція Microsoft Azure, це надає можливість відсканувати завантажені файли та, на основі зображеного контенту, генерувати певну інформацію. У даному проєкті було використано можливості генерації опису та тегів (див. рис. 3.11).

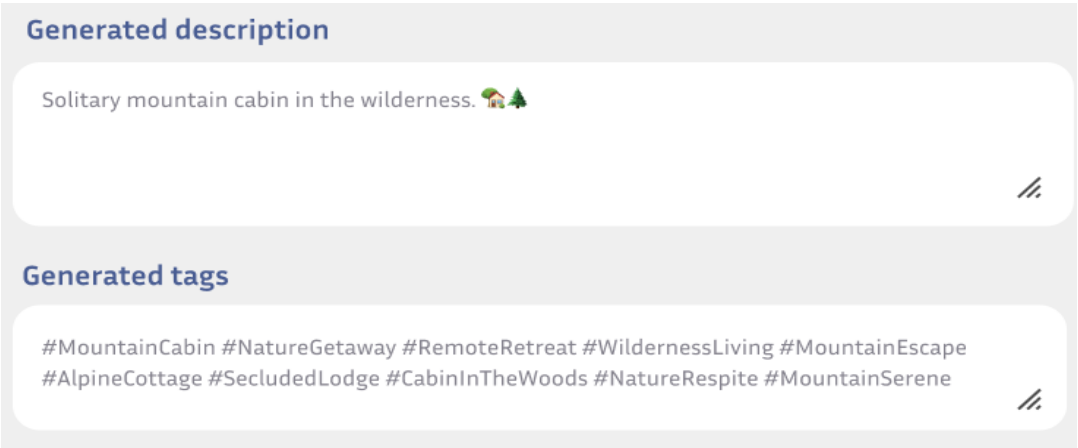

Рисунок 3.11 – Приклад згенерованого опису та тегів.

У кожному з блоків дані буде розподілено для кожного окремого файлу задля зручності співвідношення до конкретного файлу. Користувач може сам обирати, чи користуватись йому тими даними, що згенерував сервіс, чи писати власний опис до посту, який він наразі створює.

У правій колонці усі блоки відповідають за деталі самого посту. Перший з них, це опис публікації. Тут, як вже було описано вище, користувач може скористуватись згенерованими даними або ж написати власні, те ж саме стосується й тегів.

Під ними знаходиться поле вводу геолокації, якщо це необхідно. При введенні тексту, система шукає у Instagram схожі дані та виводить їх у форматі списку. Юзеру залишається лише обрати з нього необхідну адресу.

Третій блок – поле вводу імені акаунту інших користувачів, щоб потім додати відмітки про них на саму публікацію. Для цього необхідно ввести їх із знаком @ та через кому. Система потім самостійно розділить їх на окремі елементи та додасть до посту.

Наступним є селектор тегів продуктів, якщо це необхідно. У даного поля є обмеження – пошук працюватиме лише для користувачів з Америки або Канади. Принцип роботи полягає в тому, що користувач вводить назву продукту повністю чи частково, потім до Instagram надсилається пошуковий запит на співпадіння та, якщо хоча б одне було знайдено, воно відобразиться у випадаючому списку (див. рис. 3.12).

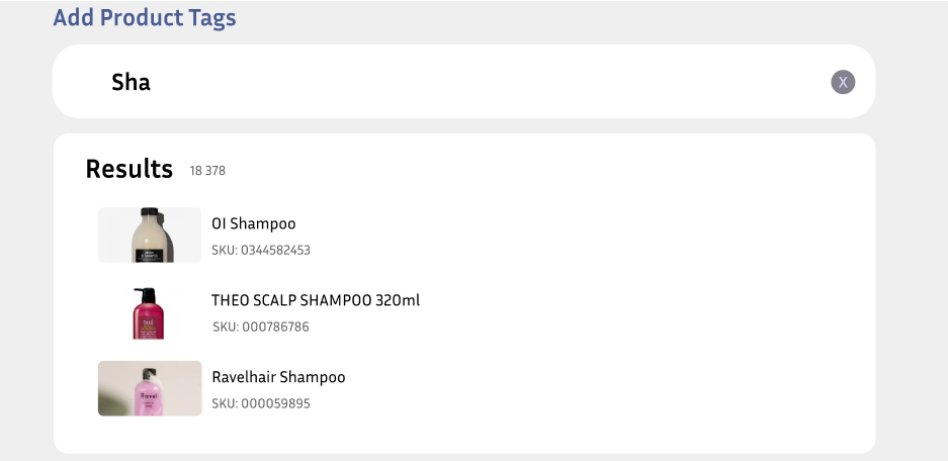

Рисунок 3.12 – Приклад результатів пошуку товарів

Кожен відображений елемент складається з фото самого товару, його назви та ідентифікатору SKU. Саме завдяки йому система у подальшому буде розуміти який саме товар необхідно відмітити. Усе, що залишається зробити – клікнути на необхідний елемент списку.

Передостаннім блоком у цій колонці є селектор дати публікації. При кліку на ньому відкривається календарик, де користувач може обрати необхідну дату, або ж є можливість ввести її вручну у поле вводу. Під самим селектором знаходиться чекбокс, який дозволяє опублікувати цей пост одразу, якщо це необхідно.

Останній елемент – чекбокс, який необхідний для того щоб повідомити системі чи необхідно додатково робити допис у Facebook. При натисканні на галочку, після того, як відбудеться публікування у основну соціальну мережу, додатково такий самий пост буде додано й до сторінки Facebook.

Після того, як усі необхідні поля були заповнені, користувач натискає на кнопку Add New Post, після чого система, в залежності від обраних параметрів виконує необхідну дію: якщо була обрана дата публікації – створює запис у таблиці planned_posts, якщо було обрано варіант «опублікувати одразу», то відсилає відповідний запит на сервіс Instagram для того, щоб він виконав відповідну дію, після чого при оновлені даних у таблиці posts буде додано запис про цей створений пост.

3.2 Огляд реалізації функціоналу

Для того, щоб реалізувати весь описаний вище функціонал, було створено необхідні контролери, репозиторії, реквести для валідації даних, та основний саме функціонал роботи з обраними API ресурсами знаходиться у сервісах, основні з яких це InstagramService, який відповідає за усю роботу з Instagram та Facebook: авторизація, отримання даних, постинг та інше, а також AzureComputerVision, в якому відбуватиметься аналіз медіа та генерація опису та тегів для нього.

3.2.1 Публікування в Instagram

Основним функціоналом на сайті, без сумнівів, є публікування постів у Instagram. Саме ця можливість є фундаментом, як і те, що цей самий пост можна запланувати на певну дату. За функціонування таких можливостей відповідає InstagramPostController, а саме метод postImages (див. рис. 3.13).

```
$connect_instagram = InstagramConnects::where('user_id', $user->id)->first();
$photos = WorkedImage::where('image_jobs_id', $image_job->id)
    ->where('status', WorkedImagesStatusEnum::$FINISHED)->get();
$date_publication = Carbon::parse($data['date_publication'])->format('d');
$userTags = [];
if(!empty($data["tags_user"])) {...}
$product_tags = [];
if(!empty($data["tags_product"])) {...}
$pinLocation = $data["pin_location"] ?? "";
$facebook = $data['post_to_facebook'] ?? false;

$publishing = PlannedPosts::create($data);
$job = (new PublishingImagesInstagramJob($photos, $publishing, $connect_instagram, $userTags,
    $product_tags, $pinLocation, $facebook))->delay(Carbon::now()->addSeconds($start_published));
$id = app(\Illuminate\Contracts\Bus\Dispatcher::class)->dispatch($job);
```

Рисунок 3.13 – Метод для публікації допису у Instagram

Перед початком роботи вхідні дані проходять перевірку через InstagramPostRequest, так як деякі є обов’язковими, деякі мають бути лише у певному форматі, наприклад теги продуктів у форматі масиву чи дата публікації у форматі DateTime. Якщо перевірка пройшла успішно, починається виконання коду.

В першу чергу необхідно отримати дані про користувача, на сторінці котрого буде відбуватись публікування. Для цього було створено окремий сервіс, який за ідентифікатором користувача знаходить його дані в базі та повертає їх до контролеру.

Після чого необхідно підготувати дані самого посту до відправки на API сервіс, так як у нього є певний перелік правил формату, наприклад усі зображення мають бути до певного розміру та роздільної здатності, опис має бути певної довжини та інші. Більшість з них вже було перевірено у реквесті, але

зображення потрібно підготувати окремо. Для цього існує метод convertImageForInstagram у сервісі ImageService (див. рис. 3.14).

```
public function convertImageForInstagram($image): string
{
    $extension = pathinfo(storage_path($image), flags: PATHINFO_EXTENSION);
    $name_img = '/temp-jpeg/image-'.mt_rand().'.jpeg';
    list($with, $height) = getimagesize(storage_path('app/public/' . $image));
    $new_with = $with;
    $new_height = $with;
    if ($height>$with){
        $new_height = intval( value: ($with/4)+$with);
    }
    $image = Storage::disk('public')->get($image);

    if ($extension != 'jpeg') {
        $interventionImage = \Image::make($image)->resizeCanvas($new_with, $new_height)->stream("jpeg", 100);
    } else {
        $interventionImage = \Image::make($image)->resizeCanvas($new_with, $new_height);
    }
    Storage::disk('public')->put($name_img, $interventionImage);

    return $name_img;
}
```

Рисунок 3.14 – Метод форматування зображення до вимог Instagram

Вхідним параметром в ньому є саме зображення, а суть цього методу полягає у тому, щоб підігнати роздільну здатність зображення до необхідних параметрів та зберегти їх у локальне сховище. У разі успішного виконання повертається назва нового, відформатованого файлу.

Якщо під час створення користувач вирішив додати відмітки інших користувачів, необхідно привести формат масиву цих даних до коректного виду (див. рис. 3.15).

```
foreach ($data["tags_user"] as $img_id => $image_user_tags) {
    $userTagsArray = explode( separator: ",", $image_user_tags);
    $y = 0.9;
    foreach ($userTagsArray as $key => $userTag) {
        if (!is_null($userTag) && !empty($userTag)) {
            $userTags[$img_id][$key]["username"] = str_replace( search: '@', replace: '', $userTag);
            $userTags[$img_id][$key]["x"] = 0;
            $userTags[$img_id][$key]["y"] = $y;
            $y -= 0.2;
        }
    }
}
```

Рисунок 3.15 – Формування масиву відміток користувачів

Система проходить по кожному тегу та формує з цих даних новий масив, де окрім самого імені акаунту додаються й координати положення цієї відмітки на самому пості. Це необхідно для того, щоб система розуміла де саме потрібно розташувати той чи інший тег. Після чого за схожим принципом відбувається формування тегів продуктів.

Коли усі підготовчі дії було зроблено, розпочинається безпосередньо формування запиту для публікації. Для цього створюється елемент класу PlannedPost, куди передаються необроблені дані, а потім цей елемент, а також ті дані, що було підігнано під вимоги Instagramy, додаються як параметри до PublishingImagesInstagramJob, яка й відповідає за те, що пост буде опубліковано саме у певну дату чи одразу через декілька секунд після створення.

Ідентифікатор цієї job у подальшому записується до таблиці planned_post задля того, щоб система потім розуміла, яку саме необхідно запустити у той чи інший час.

В залежності від того, скільки зображень було додано та чи був обраний варіант «опублікувати у Facebook», ця робота виконається за різним алгоритмом.

Варіант «Опублікувати одне зображення». Так як процес публікування в Instagram складається з двох етапів, це створення контейнеру та його публікація, то у цьому випадку буде викликана низка методів з сервісу InstagramService, а саме: publishOnePhoto, createContainer, publishContainer.

Перший з них необхідний для того, щоб викликати інші та відловити помилки, другий, як зрозуміло з назви для створення контейнеру, а останній – для його публікації.

Метод publishOnePhoto, як вже було сказано, необхідний для того, щоб згрупувати обидва етапи в один та відстежити будь-які помилки, що можуть виникнути під час його виконання (див. рис. 3.16).

Метод createContainer необхідний для того, щоб створити контейнер посту, де будуть зберігатися усі задані дані у API сервісі, щоб потім їх можна було опублікувати (див. рис. 3.17).

```php
public function publishOnePhoto($instagram_id, $image_url, $access_token,
                                 $caption = '', $user_tags = [], $product_tags = [], $pinLocation = "")
{
    try {
        $id_container = $this->createContainer($instagram_id, $image_url, $access_token,
            $caption, $user_tags, $product_tags, $pinLocation);
        $this->publishContainer($instagram_id, $id_container, $access_token);
    } catch (ApiValidationException $e) {
        throw new ApiValidationException($e);
    }

}
```

Рисунок 3.16 – Метод для публікування одного зображення

```php
public function createContainer($instagram_id, $image_url, $access_token,
                                string $caption = '', array $userTags = [],
                                array $product_tags = [], string $pinLocation = ""): ?int
{
    Log::info(json_encode([
        $instagram_id, $image_url, $access_token, $caption, $userTags, $product_tags, $pinLocation
    ]));
    try {
        $client = $this->getGuzzleClient();
        $url = "https://graph.facebook.com/v18.0/" . $instagram_id . "/media";
        $response = $client->post($url, [
            'query' => [
                'access_token' => $access_token,
                'image_url' => $image_url,
                'caption' => $caption,
                "user_tags" => json_encode($userTags),
                "product_tags" => json_encode($product_tags),
                "location" => $pinLocation,
            ],
        ]);
        $resp = $this->getDataFromResponse($response);

        return $resp['id'] ?? 0;

    } catch (\Exception $exception) {
        Log::info('createContainer: ' . $exception);
        throw new ApiValidationException( message: 'createContainer Error');

    }
}
```

Рисунок 3.17 – Метод створення контейнеру публікації

Для того, щоб виконати цю задачу, необхідно сформувати та відправити POST запит до сервісу, передавши певний перелік необхідних параметрів. Основними, звісно що, є унікальний ідентифікатор користувача, посилання на сам файл зображення, текст публікації. Також, якщо користувач додавав теги інших юзерів чи продуктів, вони будуть передані до ресурсу, як і відмітка

геолокації, хоча ці параметри вже не є обов'язковими.

Після чого, якщо процес пройшов успішно, до створеного сайту повертається результат, з якого можна отримати ідентифікатор контейнеру, щоб потім система опублікувала вірний, якщо ж ні, то буде виведено відповідну інформацію про помилку.

Коли контейнер було створено, черга переходить до методу його публікації – publishContainer (див. рис. 3.18).

```php
public function publishContainer($instagram_id, $creation_id, $access_token): ?int
{
    try {
        $client = $this->getGuzzleClient();
        $url = "https://graph.facebook.com/v18.0/" . $instagram_id . "/media_publish";
        $response = $client->post($url, [
            'query' => [
                'access_token' => $access_token,
                'creation_id' => $creation_id,
            ],
        ]);
        $resp = $this->getDataFromResponse($response);

        return $resp['id'] ?? 0;

    } catch (\Exception $exception) {
        Log::info('publishContainer: ' . $exception);
        throw new ApiValidationException( message: 'publishContainer Error');
    }
}
```

Рисунок 3.18 – Метод для публікації контейнера

Принцип методу схожий з попереднім, але звісно що відрізняється сам запит. Цього разу це /media_publish, який опублікує обраний контейнер на сторінці користувача у Instagram. Необхідними параметрами в цьому запиті є токен доступу юзера та ідентифікатор створеного контейнеру. Якщо під час постингу не виникне жодної проблеми, система поверне id створеної публікації, у протилежному випадку – інформацію про помилку. Також з таблиці planned_posts буде видалено запис про відповідний пост.

Варіант «Опублікувати декілька зображень». У цьому випадку перед тим, як зробити публікацію, спочатку необхідно створити контейнер для

кожного зображення за допомогою методу createItemContainer із того ж сервісу, що і у попередньому варіанті, потім сформувати загальний контейнер у методі createCarouselContainer і лише тоді можна публікувати його, використовуючи метод publishCarouselContainer.

Метод createItemContainer створює контейнер для кожного окремого зображення з поміткою про те, що воно відноситься до каруселі, тобто до посту з декількома зображеннями (див. рис. 3.19).

```php
public function createItemContainer($instagram_id, $image_url, $access_token,
                                     array $userTags = [], array $product_tags = [], string $pinLocation = "")
{
    try {
        $client = $this->getGuzzleClient();
        $url = "https://graph.facebook.com/v18.0/" . $instagram_id . "/media";
        $response = $client->post($url, [
            'query' => [
                'image_url' => $image_url,
                'is_carousel_item' => true,
                'access_token' => $access_token,
                'user_tags' => json_encode($userTags),
                'product_tags' => json_encode($product_tags),
                'location' => $pinLocation,
            ]
        ]);
        $resp = $this->getDataFromResponse($response);

        return $resp['id'] ?? 0;

    } catch (\Exception $exception) {
        Log::info('createItemContainer: ' . $exception);
        return 0;
    }
}
```

Рисунок 3.19 – Створення контейнеру для окремого зображення каруселі

Принцип роботи дуже схожий з описаним вище createContainer, проте є декілька важливих відмінностей у тілі запиту. Першим з них є те, що у цьому запиті не передається опис посту, так як його буде передано при формування загального контейнеру каруселі. А другий – параметр is_carouser_item зі значенням true, який й дає зрозуміти сервісу, що цей контейнер відноситься до допису з декількома зображеннями.

Якщо створення пройшло успішно, система поверне ідентифікатор, який в подальшому буде використано в загальному контейнері, якщо ж виникла

помилка – у консолі відобразиться відповідне повідомлення.

Після того, як було створено контейнери для усіх зображень, починає виконуватись метод createCarouselContainer, який згрупує усі попередні в один повноцінний (див. рис. 3.20).

```php
public function createCarouselContainer($instagram_id, array $children, $access_token, string $caption = '')
{
    try {
        $client = $this->getGuzzleClient();
        $url = "https://graph.facebook.com/v18.0/" . $instagram_id . "/media";
        $response = $client->post($url, [
            'query' => [
                'caption' => $caption,
                'media_type' => 'CAROUSEL',
                'children' => $children,
                'access_token' => $access_token
            ]
        ]);
        $resp = $this->getDataFromResponse($response);

        return $resp['id'] ?? 0;

    } catch (\Exception $exception) {
        Log::info('createCarouselContainer: ' . $exception);
        return 0;
    }
}
```

Рисунок 3.20 – Метод створення контейнера каруселі

Принцип його роботи також не відрізняється від попередьного, окрім параметрів у тілі запиту. Цього разу окрім токену доступу користувача, до нього додається опис посту, так як він є єдиним для усіх картинок. Також необхідним є параметр media_type, який вказує на те, що пост буде створено у форматі каруселі, тому він і має значення CAROUSEL. Останнім є children – масив ідентифікаторів контейнерів кожного доданого зображення.

Якщо створення пройде без помилок, до системи повернеться ідентифікатор вже цього контейнера для подальшого його публікування на сторінці.

Останнім методом, як вже зрозуміло, буде викликано publishCarouselContainer – публікування створеного контейнеру у профілі Instagram (див. рис. 3.21).

```php
public function publishCarouselContainer($instagram_id, $creation_id, $access_token)
{
    try {
        $client = $this->getGuzzleClient();
        $url = "https://graph.facebook.com/v18.0/" . $instagram_id . "/media_publish";
        $response = $client->post($url, [
            'query' => [
                'creation_id' => $creation_id,
                'access_token' => $access_token
            ]
        ]);
        $resp = $this->getDataFromResponse($response);

        return $resp['id'] ?? 0;

    } catch (\Exception $exception) {
        Log::info('createCarouselContainer: ' . $exception);
        return 0;
    }
}
```

Рисунок 3.21 – Метод публікації контейнеру у форматі карусель

Як і у попередніх випадка, принцип його роботи схожий з методом publishContainer, проте для того, щоб не використовувати один метод для різних типів посту, було створено окремий. В якості параметрів також передаються ідентифікатор-токен користувача та id створеного контейнеру. Якщо весь процес виконається успішно та публікація з’явиться на сторінці, до сайту повернеться ідентифікатор посту, після чого з таблиці planned_posts буде видалено запис про відповідний запланований пост та додано запис у таблицю posts.

3.2.2 Публікування в Facebook

Окрім основних можливостей постингу в Instagram, як вже було описано раніше, користувач також має змогу зробити пост у Facebook. Так як публікацію потрібно робити з медіа, тому процес як і у випадку з Instagram розбито на два етапи: завантаження медіафайлу та безпосередньо публікування посту в Facebook. Для цього у сервісі InstagramService є два методи: createFacebookImageUpload та postFacebookImageUpload.

Перший з них, як вже було сказано раніше, відповідає за завантаження зображень на Facebook задля подальшого їх використання у пості, який буде створено (див. рис. 3.22).

```php
public function createFacebookImageUpload($access_token, $image_url)
{
    try {
        $client = $this->getGuzzleClient();
        $url = "https://graph.facebook.com/v18.0/me/photos";
        $response = $client->post($url, [
            'query' => [
                'access_token' => $access_token,
                'published' => false,
                'url' => $image_url,
            ]
        ]);
        $resp = $this->getDataFromResponse($response);

        return $resp['id'] ?? false;

    } catch (\Exception $exception) {
        Log::info('createFacebookImageUpload: ' . $exception);
        return false;
    }
}
```

Рисунок 3.22 – Метод завантаження файлів на Facebook

Вхідними параметрами у ньому є унікальний токен ідентифікації користувача та посилання на медіафайл, який необхідно завантажити. Далі формується сам запит, основними елементами в ньому є посилання, в якому вказана версія ресурсу, та необхідні параметри. Вони складаються власне з ідентифікатору користувача, відмітки published, яка відповідає за те, щоб цей медіафайл не було опубліковано, та посилання на сам файл. Після чого запит відправляється на сервіс, і якщо процес був пройдений успішно, повертається ідентифікатор обраного файлу.

Другий метод відповідає за публікування допису з текстом. Саме він є основним, що дозволяє разом з публікацією у Instagram, зробити відповідну у профіль Facebook (див. рис. 3.23).

У сам метод передаються, як і у попередньому, токен доступу користувача,

посилання на завантажені медіафайли та власне текст допису. Починається метод з формування посилання запиту, в якому як і у попередньому випадку вказано версію сервісу та вказівник /feed, що повідомляє які саме дії необхідно буде виконувати. Після цього формується масив медіа на основі ідентифікаторів доданих до цього файлів.

```php
public function postFacebookImageUpload($access_token, $uploadIds, $message)
{
    try {
        $client = $this->getGuzzleClient();
        $url = "https://graph.facebook.com/v18.0/me/feed";
        $attached_media = [];
        foreach ($uploadIds as $uploadId) {
            array_push( &array: $attached_media, [
                'media_fbid' => $uploadId
            ]);
        }
        $response = $client->post($url, [
            'query' => [
                'access_token' => $access_token,
                'published' => true,
                'attached_media' => $attached_media,
                'message' => $message,
            ]
        ]);
        $resp = $this->getDataFromResponse($response);

        return $resp['id'] ?? false;

    } catch (\Exception $exception) {
        Log::info('createFacebookImageUpload: ' . $exception);
        return false;
    }
}
```

Рисунок 3.23 – Метод публікації допису у Facebook

Далі формується сам запит, де вказується посилання та необхідні параметри, серед яких ідентифікатор доступу користувача, відмітка published, але наразі зі значенням true ,що вказує системі на те, що необхідно опублікувати пост одразу, список медіа, які буде відображено та відповідно сам текст допису.

Надалі вже використаний сервіс виконує ряд необхідних дій та, якщо все пройшло успішно, повертає ідентифікатор створеної публікації, надалі ці дані буде записано до бази даних.

3.2.3 Аналіз медіафайлу з використанням Microsoft Azure

Для того, щоб згенерувати опис та теги для тих медіафайлів, що завантажує користувач, було створено сервіс AzureComputerVisionService. В ньому наявний лише один відповідний метод, який викликається при кожному додаванні файлу під час створення посту (див. рис. 3.24).

```
public function getImageAnalyze($imageUrl)
{
    $params = 'features=tags,caption';
    $response = Http::withHeaders([
        'Ocp-Apim-Subscription-Key' => env('AZURE_KEY'),
        'Content-Type' => 'application/json'
    ])->post(env('AZURE_ENDPOINT') .
        '/computervision/imageanalysis:analyze??api-version=2023-02-01-preview&' . $params, [
        'url' => $imageUrl
    ]);
    return $response->body();
}
```

Рисунок 3.24 – Метод аналізу зображення

Вхіднім параметром в ньому є посилання на сам файл, а повертає при успішному виконанні тіло відповіді.

В першу чергу створюється змінна, в якій зберігаються параметри для основного запиту. В цьому випадку це елементи, які потрібно згенерувати: Tags, Description. У подальшому можна розширювати можливості та додавати нові дані.

Далі формується сам запит за схемою, що була описала раніше. Основними елементами є версія сервісу, унікальний згенерований для сайту ідентифікатор, обрані раніше параметри та безпосередньо посилання на сам файл, для якого необхідно зробити аналіз.

У результаті посилається POST запит на сервер, де вже він перехвачує запит та виконує усі необхідні дії. Після чого повертає на основний ресурс результаті у форматі JSON, приклад якого вже було описано у попередньому розділі. Далі система повертає результат на фронт та обробляє його, відображаючи у текстовому полі у форматі: «назва_файлу-згенеровані дані».

ВИСНОВКИ

Сьогодні ведення блогу у соціальних мережах нерідко прирівнюється до повноцінної професії. З кожним роком все більше людей бажають вести власні блоги, сторінки у соціальних мережах, тощо. Саме через це кожен контентмейкер прагне спростити процес роботи та зробити його більш комфортним для себе.

Тому, у цій ніші з’являється потреба у створенні додаткових сервісів та послуг. Одним з таких сервісів може бути можливість планувати публікування постів та у будь-який час змінити чи видалити їх.

Instagram Graph API дозволяє розробникам інтегрувати соціальну мережу Instagram у будь-які додатки, зокрема мобільні та веб. Цей API надає можливість створювати пости та збирати інформацію для статистики профілів, власного чи інших користувачів. Наприклад, користувач зможе переглядати основну інформацію про пости, що вже існують на його сторінці, без потрібності використання офіційних клієнтів Instagram.

Одним з інноваційних підходів у сучасному інтернеті є використання штучного інтелекту. Зокрема, користувач може використовувати його для створення опису зображень, що будуть опубліковані. Microsoft Azure Computer Vision вміє аналізувати картинку та генерувати для неї художній опис, набір хештегів та альтернативний підпис до HTML зображення.

Створений вебдодаток використовує в повному обсязі модель MVC, має простий та зручний інтерфейс, який реалізує усі необхідні для постингу механізми. Зокрема, генерування підпису та тегів до зображення, додання відміток користувачів та продуктів, зручний та функціональний календар запланованих постів.

Таким чином, результатом кваліфікаційної роботи магістра є створений вебдодаток з інтегрованими сервісами Instagram Graph API та MS Azure Computer Vision.

ПЕРЕЛІК ПОСИЛАНЬ

1. Azure AI Vision. URL: https://azure.microsoft.com/en-us/products/ai-services/ai-vision (дата звернення: 09.10.2023).
2. Використання postman в тестуванні. URL: https://training.qatestlab.com/blog/technical-articles/use-postman-in-testing/ (дата звернення: 11.10.2023).
3. API Graph Instagram. URL: https://developers.facebook.com/docs/instagram-api/ (дата звернення: 11.10.2023).
4. Як будувати UML-діаграми. Розбираємо три найпопулярніші варіанти. URL: https://dou.ua/forums/topic/40575/ (дата звернення 14.09.2023).
5. Diagram of use cases (UseCase diagram). URL: https://flexberry.github.io/en/fd_use-case-diagram.html (дата звернення 14.09.2023).
6. What is Use Case Diagram? URL: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/ (дата звернення: 15.09.2023).
7. Діаграми класів. URL: https://ua5.org/oop/392-diagrami-klasiv.html (дата звернення 19.09.2023).
8. Що таке Vue? URL: https://ua.vuejs.org/guide/introduction.html (дата звернення 24.09.2023).
9. Laravel Documentation. URL: https://laravel.com/docs/9.x (дата звернення 28.09.2023).