НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Навчально-науковий інститут атомної та теплової енергетики

Кафедра цифрових технологій в енергетиці

«До захисту допущено»
Завідувач кафедри
_____ Наталія АУШЕВА
“___”_____2025 р.

Дипломна робота
на здобуття ступеня бакалавр

За освітньою програмою “Цифрові технології в енергетиці”

Спеціальності 122 “Комп’ютерні науки”

на тему: “Система обміну даними між ERP Odoo та платформою Instagram”

Виконав: студент 4 курсу, групи ТР-11
Бугайченко Владислав Романович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник доцент, к.т.н, доцент Вячеслав ЛІСКІН _____
(посада, науковий ступінь, вчене звання, ім’я, ПРІЗВИЩЕ) (підпис)

Рецензент доцент каф. ПМА, к.т.н, доцент Сергій СИРОТА _____
(посада, науковий ступінь, вчене звання, ім’я, ПРІЗВИЩЕ) (підпис)

Н.контроль ст.викладач Ольга БЕСПАЛА _____
(посада, ім’я, ПРІЗВИЩЕ) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ - 2025

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ АТОМНОЇ ТА ТЕПЛОВОЇ ЕНЕРГЕТИКИ

Кафедра ЦИФРОВИХ ТЕХНОЛОГІЙ В ЕНЕРГЕТИЦІ

Рівень вищої освіти – перший (бакалаврський)
спеціальність 122 “Комп’ютерні науки”
Освітньо-професійна програма “Цифрові технології в енергетиці”

ЗАТВЕРДЖУЮ

Завідувач кафедри ЦТЕ

_____ Наталія АУШЕВА
(підпис)

“___” _____2025 р.

ЗАВДАННЯ
на дипломну роботу студенту

_____Бугайченку Владиславу Романовичу_____
(прізвище, ім’я, по батькові)

1. Тема роботи “Система обміну даними між ERP Odoo та платформою Instagram”
Науковий керівник Ліскін Вячеслав Олегович, к.т.н, доцент
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від “ 2 ” червня 2025 року № 1875-с

2. Термін подання студентом роботи 09.06.2025

3. Вихідні дані до роботи мови програмування Python та JavaScript, фреймворк Owl, бібліотеки Requests, PyAV та PIL, СКБД PostgreSQL, інструмент для тестування ngrok, середовище розробки Visual Studio Code

4. Перелік питань, які потрібно розробити _____

1) проаналізувати існуючі рішення для обміну даними між системами управління ресурсами підприємств та соціальними мережами;

2) обґрунтувати вибір інструментів, технологій та алгоритмів, використаних для реалізації програмного забезпечення;
3) спроектувати архітектуру програмного забезепечення та структуру бази даних
4) розробити алгоритми взаємодії між системою та соціальною мережею
5) представити сценарії використання програмного продукту
5. Орієнтовний перелік графічного ілюстративного матеріалу діаграми, що демонструють роботу алгоритмів обміну даними, архітектуру системи, бази даних, взаємодію користувача з системою, класи серверної частини програмного продукту; приклади роботи з системою.
6. Дата видачі завдання 13.09.2024

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Вибір теми роботи	13.09.2024	виконано
2.	Аналіз методів та засобів розв'язання задачі	14-20.04.25	виконано
3.	Розробка архітектури та загальної структури системи	21-27.04.25	виконано
4.	Розробка окремих підсистем	28.04-04.05.25	виконано
5.	Програмна реалізація системи	05-11.05.25	виконано
6.	Оформлення пояснювальної записки	15-17.05.25	виконано
7.	Захист програмного забезпечення	12-16.05.25	виконано
8.	Передзахист	26-28.05.25	виконано
9.	Захист	16-20.06.25	виконано

Студент _____ Владислав БУГАЙЧЕНКО
(підпис) (прізвище та ініціали)

Керівник _____ Вячеслав ЛІСКІН
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Дипломну роботу виконано на 55 сторінках. Робота містить 30 ілюстрацій, 2 таблиці, 1 додаток, 44 джерел у переліку посилань.

Мета роботи — створення програмного забезпечення обміну даними між системою управління ресурсами підприємства Odoo та соціальною мережею Instagram для підвищення ефективності маркетингових кампаній та взаємодії з цільовою аудиторією через соціальну мережу Instagram.

Методи та засоби: мови програмування Python та JavaScript, фреймворк Owl для розробки користувацького інтерфейсу, бібліотеки Requests, PyAV та PIL, система керування базами даних PostgreSQL, інструмент для тестування ngrok, середовище розробки Visual Studio Code.

У результаті було створено програмні модулі, що забезпечують двосторонню синхронізацію даних між системою Odoo та платформою Instagram й підтримують управління публікаціями акаунту, обмін текстовими та медіа повідомленнями й моніторинг зулученості клієнтів, що дозволяє значно скоротити час на адміністрування соціальних мереж та забезпечити більш ефективне використання ресурсів підприємства для досягнення бізнес-цілей у сфері цифрового маркетингу.

Ключові слова: Odoo, Instagram, Python, JavaScript, обмін даними, інтеграція, Instagram API, ERP.

ABSTRACT

The thesis work is completed on 55 pages. The work contains 30 illustrations, 2 tables, 1 appendix, and 44 sources in the reference list.

The objective of the work is to create software for data exchange between the enterprise resource planning system Odoo and the social network Instagram to improve the efficiency of marketing campaigns and interaction with the target audience through the Instagram social network.

Methods and tools include Python and JavaScript programming languages, Owl framework for user interface development, Requests, PyAV and PIL libraries, PostgreSQL database management system, ngrok testing tool, and the Visual Studio Code development environment.

As a result, software modules were created that provide bidirectional data synchronization between the Odoo system and the Instagram platform and support account publication management, exchange of text and media messages, and monitoring of customer engagement. This allows for significant reduction in time spent on social media administration and ensures more efficient use of enterprise resources to achieve business objectives in the field of digital marketing.

Keywords**:** Odoo, Instagram, Python, JavaScript, data exchange, integration, Instagram API, ERP.

ЗМІСТ

ВСТУП

У сучасному цифровому світі інтеграція різних інформаційних систем та соціальних платформ стає критично важливою для ефективного ведення бізнесу [1]. Особливої актуальності набуває питання взаємодії корпоративних систем управління ресурсами підприємства (ERP) з соціальними мережами, зокрема з Instagram, який є одним з найпопулярніших каналів комунікації з цільовою аудиторією [2].

Актуальність розробки системи обміну даними між ERP Odoo та платформою Instagram обумовлена зростаючою потребою бізнесу в автоматизації процесів управління соціальними мережами та інтеграції даних процесів у загальну систему управління підприємством [3]. Сучасні компанії потребують ефективних інструментів для керування контентом, аналізу взаємодії з аудиторією та оптимізації маркетингових стратегій [4].

Практична значущість роботи полягає у створенні інструменту, який дозволить підприємствам ефективніше керувати своєю присутністю в соціальних мережах, автоматизувати рутинні процеси та приймати обґрунтовані рішення на основі аналітичних даних [5]. Це сприятиме підвищенню ефективності маркетингових кампаній та покращенню взаємодії з цільовою аудиторією [6].

Результати дослідження можуть бути використані як основа для подальшого розвитку систем інтеграції корпоративних ERP рішень з соціальними мережами та іншими цифровими платформами [7]. Розроблений модуль стане важливим інструментом для компаній, які прагнуть оптимізувати свою діяльність у цифровому просторі та підвищити ефективність взаємодії з клієнтами через соціальні мережі [8].

1 ПОСТАНОВКА ЗАДАЧІ РОЗРОБКИ СИСТЕМИ ОБМІНУ ДАНИМИ МІЖ ERP ODOO ТА ПЛАТФОРМОЮ INSTAGRAM

Метою дипломної роботи є створення програмного модуля, який забезпечить безперебійний двосторонній обмін даними між системою Odoo та соціальною мережею Instagram. Це дозволить автоматизувати процеси публікації контенту, управління коментарями та отримання аналітичних даних щодо взаємодії користувачів з контентом бренду.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

— провести аналіз можливостей ERP системи Odoo

— дослідити способи взаємодії з платформою Instagram

— сформувати вимоги до модуля інтеграції

— спроектувати архітектуру та механізми взаємодії Odoo та платформи Instagram

— розробити прогрмане забезпечення та протестувати його

Реалізація такої системи дозволить підприємствам ефективніше керувати своєю присутністю в соціальних мережах, автоматизувати процеси взаємодії з клієнтами та покращити аналітичні можливості для прийняття бізнес рішень. При цьому особлива увага має бути приділена забезпеченню стабільності роботи системи та мінімізації можливих збоїв у процесі обміну даними.

1.1 Система планування ресурсів підприємства Odoo

Odoo є потужною системою планування ресурсів підприємства (ERP), яка вирізняється своєю гнучкістю та всеохоплюючим підходом до управління бізнес процесами. ERP рішення відіграють важливу роль у покращенні та спрощенні

операційної діяльності компаній через систему, що базується на добре організованих даних та програмних модулях [9].

Архітектура Odoo побудована на основі модульної структури, що дозволяє підприємствам впроваджувати лише ті компоненти, які відповідають їхнім конкретним потребам. Система включає широкий спектр функціональних можливостей, серед яких управління продажами, закупівлями, складським обліком, виробництвом, бухгалтерією, персоналом та взаємовідносинами з клієнтами [10]. Особливістю Odoo є можливість поступового розширення функціоналу через додавання нових модулів без необхідності повної реконфігурації системи [11].

Важливою перевагою Odoo є її універсальність щодо масштабів бізнесу. Система однаково ефективно працює як для малих підприємств, так і для середніх та великих компаній. Це досягається завдяки можливості масштабування та налаштування відповідно до специфічних вимог кожного підприємства [12]. Особливо актуальною є можливість інтеграції з різними зовнішніми системами та сервісами, що розширює функціональні можливості платформи.

Система демонструє високу ефективність у різних галузях бізнесу, включаючи виробництво, роздрібну торгівлю, послуги, дистрибуцію та електронну комерцію. Архітектура Odoo базується на сучасних технологіях та програмних рішеннях, що забезпечує надійність та продуктивність системи [10]. Важливою особливістю є відкритий код системи, що дозволяє розробникам створювати власні модулі та адаптувати функціонал під специфічні потреби бізнесу [11].

Odoo надає потужні інструменти для автоматизації бізнес процесів, що включають можливості для створення автоматизованих робочих процесів, системи сповіщень та звітності. Система також забезпечує високий рівень безпеки даних та можливість детального налаштування прав доступу для різних категорій користувачів [12]. Це робить її особливо привабливою для підприємств, які приділяють значну увагу захисту корпоративної інформації та потребують гнучкого управління доступом до різних модулів системи.

1.2 Використання Instagram API для керування присутністю в медіа просторі

У сучасному цифровому середовищі присутність бізнесу в соціальних мережах, особливо в Instagram, стала критично важливим елементом маркетингової стратегії. Дослідження показують, що ефективне використання соціальних медіа дозволяє підприємствам встановлювати більш тісний зв'язок із цільовою аудиторією та підвищувати рівень залученості користувачів [13]. При цьому особливого значення набуває можливість автоматизації та оптимізації взаємодії з платформою через програмний інтерфейс Instagram API.

Instagram API надає бізнес акаунтам широкий спектр можливостей для керування своєю присутністю в медіа просторі. Функціональність включає можливість програмного розміщення контенту, отримання детальної аналітики щодо взаємодії користувачів з публікаціями, а також моніторинг згадувань бренду в мережі [14]. Важливо відзначити, що використання API дозволяє автоматизувати процеси збору та аналізу даних, що суттєво підвищує ефективність маркетингових кампаній.

Дослідження взаємодії користувачів з контентом у соціальних мережах демонструє, що якість публікацій має більше значення, ніж кількість підписників [15]. Це підкреслює важливість використання API для аналізу ефективності різних типів контенту та оптимізації стратегії публікацій. При цьому необхідно враховувати, що Instagram встановлює певні обмеження на частоту запитів до API та обсяг даних, які можна отримати, що вимагає ретельного планування автоматизованих процесів.

Особливу увагу варто приділити можливостям API щодо роботи з різними форматами медіа контенту. Згідно з дослідженнями, різні типи публікацій (фотографії, відео, альбоми) викликають різний рівень залученості аудиторії [16]. API дозволяє не лише публікувати різноманітний контент, але й отримувати детальну статистику щодо його ефективності, що є визначальним для оптимізації контент стратегії.

Важливим аспектом використання Instagram API є можливість автоматизованого моніторингу та управління взаємодією з користувачами. Дослідження показують, що швидкість та якість відповідей на коментарі та згадування бренду суттєво впливають на рівень залученості аудиторії [17]. API надає інструменти для оперативного відстеження та реагування на активність користувачів, що дозволяє підтримувати високий рівень залученості аудиторії.

При використанні Instagram API необхідно враховувати технічні особливості платформи, зокрема латентність передачі даних та обмеження на кількість запитів. Дослідження показують, що Instagram може забезпечити достатньо низьку затримку при передачі даних, особливо для мультимедійних повідомлень [14]. Це дозволяє створювати ефективні системи автоматизації, які можуть працювати практично в режимі реального часу.

1.3 Вимоги до модуля інтеграції

Розробка модуля інтеграції між системою ERP Odoo та платформою Instagram вимагає чіткого визначення функціональних та нефункціональних вимог для забезпечення ефективної роботи системи.

У контексті функціональних вимог модуль повинен забезпечувати повноцінну двосторонню комунікацію між платформою Instagram та системою Odoo. Базова функціональність включає передачу текстових повідомлень в обох напрямках із збереженням контексту діалогу, відображенням реакцій на повідомлення отриманих з Instagram та цитувань шляхом прив’язки до конкретного повідомлення. Система має підтримувати обмін медіафайлами різних форматів, включаючи зображення, відео та аудіофайли. Важливою функціональною вимогою є забезпечення можливості автоматизованого створення лідів на основі взаємодій з користувачами Instagram, а також реалізація функціоналу масової розсилки повідомлень для одночасної комунікації з множиною контактів.

Система повинна надавати інструменти для ефективного управління контентом Instagram акаунту, включаючи можливість перегляду існуючих публікацій та їхніх метрик й метаданих, таких як кількість вподобань й коментарів, а також час публікації, тип контенту, опис та хештеги.

Взаємодія з коментарями під публікаціями представляє окремий блок функціональних вимог. Модуль має забезпечувати створення нових коментарів безпосередньо з інтерфейсу Odoo, підтримку відповідей на існуючі коментарі із збереженням ієрархічної структури обговорення, а також можливість модерації контенту через видалення неприйнятних коментарів.

Система повинна підтримувати публікацію різних типів контенту, включаючи фотографії й відео, а також пости, що одночасно підтримують обидва типи. Додатково має забезпечуватися створення сторіс з фото та відеоматеріалами. Необхідною є функція попередньої валідації контенту перед публікацією та можливість створення запланованих постів.

Щодо нефункціональних вимог, система повинна забезпечувати високу продуктивність та стабільність роботи при обробці великих обсягів даних та медіафайлів. Важливими є вимоги до безпеки, включаючи захист облікових даних користувачів та безпечну передачу інформації між системами. Інтерфейс модуля має бути інтуїтивно зрозумілим та відповідати загальному дизайну системи Odoo, забезпечуючи швидке освоєння функціоналу користувачами.

Сценарії використання модуля передбачають різноманітні варіанти взаємодії, починаючи від базового моніторингу активності в Instagram до маркетингових кампаній з використанням запланованих публікацій та масових розсилок. Типовий сценарій включає створення та публікацію контенту, моніторинг взаємодій користувачів, обробку вхідних повідомлень та коментарів, а також автоматизоване створення лідів на основі активності потенційних клієнтів.

2 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ІНТЕГРАЦІЇ ERP СИСТЕМ ІЗ СОЦІАЛЬНИМИ МЕРЕЖАМИ

У сучасному бізнес середовищі інтеграція ERP систем із соціальними мережами стає все більш критичною для успішної діяльності підприємств. Аналіз існуючих рішень демонструє різноманітність підходів до такої інтеграції, кожен з яких має свої особливості та обмеження [18].

Офіційне рішення Odoo — модуль Social Marketing — пропонує базову функціональність для роботи з соціальними мережами, включаючи Instagram через Facebook Graph API. Однак, даний модуль має обмежений функціонал, особливо щодо специфічних можливостей Instagram, що знижує його ефективність для бізнесів, які потребують глибокої інтеграції саме з цією платформою [19].

На ринку представлені різноманітні сторонні модулі для Odoo, розроблені компаніями спільноти. Серед них варто відзначити "CRM Instagram Lead Integration" від Pragmatic Techsoft, "Facebook Catalog Integration & Instagram Shopping" від Garazd, та "Omnichannel Communication Odoo" від TechUltra Solutions. Проте дані рішення характеризуються вузькою спеціалізацією та обмеженою функціональністю [20].

Розглянемо існуючі підходи до інтеграції ERP систем із соціальними мережами. Нижче наведено порівняльну таблицю 2.1 основних моделей інтеграції.

Таблиця 2.1 — Порівняння моделей інтеграції ERP із соціальними мережами

Характеристика	**Пряма інтеграція через API**	**Інтеграція через iPaaS**
Гнучкість	Висока	Обмежена
Швидкість впровадження	Низька	Висока
Вартість розробки	Висока	Низька
Можливості кастомізації	Необмежені	Обмежені конекторами

Окремої уваги заслуговують iPaaS рішення, такі як Zapier, Make та n8n, які пропонують готові конектори для інтеграції ERP систем із соціальними мережами [21]. Прикладом існуючого рішення для автоматизації комунікації є платформа для управління чат ботами, інтерфейс якої представлено на рисунку 2.1.

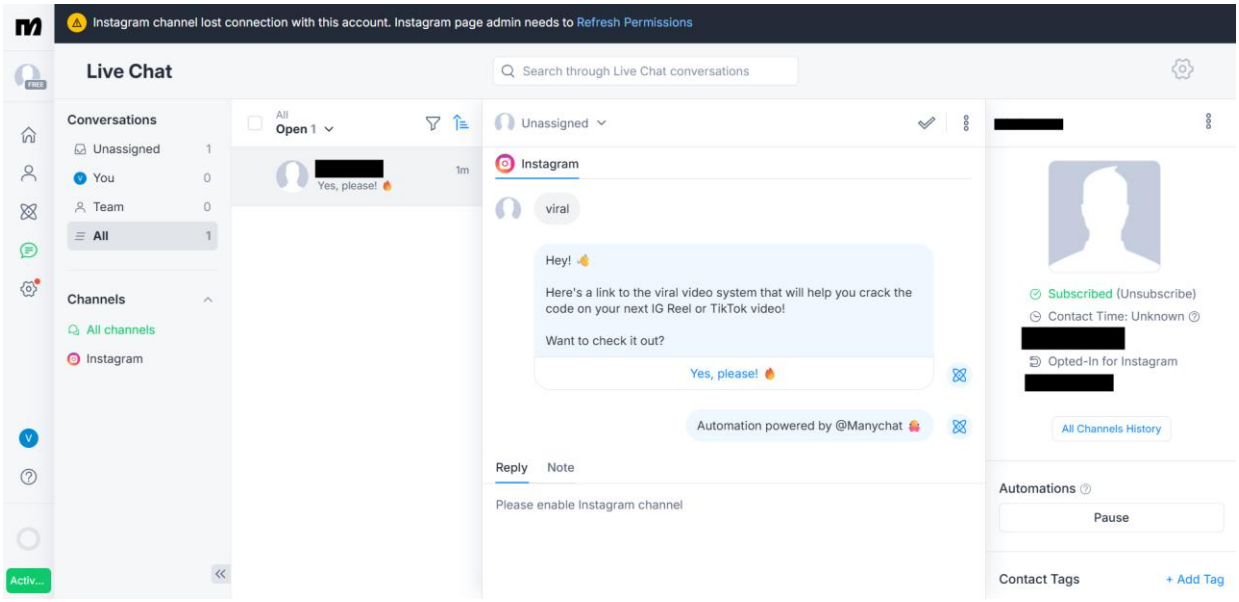

Рисунок 2.1 — Інтерфейс платформи для управління чат ботами

Аналіз існуючих рішень виявляє суттєві обмеження як офіційних, так і сторонніх інтеграційних модулів. Більшість з них не забезпечують повноцінної функціональності для роботи з Instagram API, мають складну процедуру налаштування та обмежені можливості кастомізації [18]. Це обґрунтовує необхідність розробки власного рішення, яке б забезпечило повноцінну інтеграцію між ERP Odoo та платформою Instagram, враховуючи специфічні потреби бізнесу та можливості сучасних технологій обміну даними.

3 ВИБІР ЗАСОБІВ РОЗРОБКИ СИСТЕМИ ОБМІНУ ДАНИМИ МІЖ ERP ODOO ТА ПЛАТФОРМОЮ INSTAGRAM

У даному розділі розглядаються ключові технології та інструменти, які використовуються для розробки системи обміну даними між ERP Odoo та соціальною платформою Instagram. Вибір оптимальних засобів розробки є критично важливим етапом, оскільки від нього залежить як ефективність реалізації проекту, так і подальша підтримка та масштабування системи.

Особлива увага приділяється аналізу програмних компонентів та технологічних рішень, які забезпечують надійну інтеграцію між двома різними за архітектурою та призначенням платформами. Розглядаються як базові технології, що лежать в основі ERP системи Odoo, так і специфічні інструменти для роботи з API Instagram.

В процесі вибору засобів розробки враховуються такі критичні фактори як продуктивність, надійність, масштабованість та зручність подальшої підтримки системи. Значна увага приділяється також питанням сумісності обраних технологій та їх відповідності сучасним стандартам розробки програмного забезпечення. Детальний аналіз кожного компоненту системи дозволяє обґрунтувати вибір конкретних інструментів та технологій для реалізації поставлених завдань.

3.1 Середовище розробки Visual Studio Code

Visual Studio Code є ефективним та універсальним редактором коду, який надає розробникам широкий спектр можливостей для ефективної розробки програмного забезпечення. Це середовище розробки вирізняється своєю легкістю, швидкодією та модульною архітектурою, що дозволяє налаштувати робочий простір відповідно до індивідуальних потреб розробника.

Одною з ключових переваг Visual Studio Code є вбудована підтримка системи контролю версій Git, що забезпечує зручну роботу з репозиторіями безпосередньо з інтерфейсу редактора [22]. Розробники можуть здійснювати комміти, переглядати історію змін та вирішувати конфлікти без необхідності використання додаткових інструментів. Інтегрований термінал дозволяє виконувати команди безпосередньньо в середовищі розробки, що значно підвищує продуктивність роботи [23].

Система розширень є однією з найсильніших сторін Visual Studio Code. Розробники мають доступ до тисяч плагінів, які додають підтримку різних мов програмування, фреймворків та інструментів розробки. Це дозволяє створити індивідуальне середовище розробки, оптимізоване під конкретні завдання та технології. Вбудований інтелектуальний помічник IntelliSense забезпечує потужні можливості автодоповнення коду, підказки та рефакторинг [24].

Важливою особливістю є підтримка користувацьких налаштувань та тем оформлення. Розробники можуть налаштувати не лише візуальний вигляд середовища, але й комбінації клавіш, правила форматування коду та інші параметри редактора. Вбудована підтримка Markdown дозволяє зручно працювати з документацією проекту безпосередньо в середовищі розробки [25].

Visual Studio Code забезпечує високу продуктивність завдяки оптимізованій роботі з великими проектами та файлами. Система пошуку та заміни з підтримкою регулярних виразів, можливість швидкої навігації по коду та вбудовані інструменти рефакторингу значно прискорюють процес розробки. Підтримка віддалених робочих просторів дозволяє ефективно працювати з кодом, розміщеним на віддалених серверах або в контейнерах [26].

3.2 Обґрунтування вибору мов програмування

Для реалізації системи обміну даними між ERP Odoo та платформою Instagram необхідно обрати оптимальні мови програмування, які забезпечать

ефективну розробку та подальшу підтримку системи. У контексті даного дослідження особливу увагу приділено Python та JavaScript як основним інструментам розробки.

Python є високорівневою об'єктно орієнтованою мовою програмування, яка демонструє значні переваги при розробці серверної частини системи. Python характеризується динамічною системою типізації та автоматичним керуванням пам'яттю, що суттєво спрощує процес розробки [27]. Важливою перевагою Python є наявність великої стандартної бібліотеки та підтримка різних парадигм програмування, включаючи об'єктно орієнтовану та функціональну.

Особливо важливим аспектом використання Python у контексті даної системи є його ефективність при роботі з ERP системами. Python демонструє високу ефективність при інтеграції з ERP системами, забезпечуючи можливість створення надійних рішень для синхронізації даних у реальному часі [28]. Це особливо актуально для розробки API інтерфейсів та обробки даних між Odoo та Instagram.

JavaScript, як мова програмування для фронтенд частини системи, забезпечує створення інтерактивного та чутливого користувацького інтерфейсу. Використання JavaScript дозволяє реалізувати асинхронну взаємодію з серверною частиною без перезавантаження сторінки, що критично важливо для систем обміну даними в реальному часі. Сучасні фреймворки JavaScript надають потужні інструменти для створення динамічних веб інтерфейсів, які забезпечують зручну взаємодію користувача з системою [29].

Комбінація Python та JavaScript створює міцну технологічну базу для реалізації системи. Python забезпечує надійну обробку даних та інтеграцію з API платформ, тоді як JavaScript відповідає за презентаційний рівень та взаємодію з користувачем. Така архітектура дозволяє досягти оптимального балансу між продуктивністю серверної частини та інтерактивністю користувацького інтерфейсу [30].

Важливо відзначити, що обидві мови мають розвинуті екосистеми з великою кількістю бібліотек та фреймворків, що значно прискорює процес розробки та розширює можливості системи. Згідно з дослідженням, Python демонструє

стабільне зростання популярності серед розробників, що гарантує довгострокову підтримку та розвиток технології [27].

3.3 Фреймворк Odoo

Odoo є всеохоплюючим фреймворком для створення бізнес додатків, який базується на модульній архітектурі. Система побудована за принципами трирівневої архітектури, що забезпечує гнучкість та масштабованість рішень [31]. Основними компонентами архітектури Odoo є сервер бази даних, сервер додатків та веб клієнт.

Архітектура Odoo характеризується чітким розподілом на функціональні модулі, кожен з яких відповідає за певний аспект бізнес логіки. На рисунку 3.1 зображена архітектура модулю в системі.

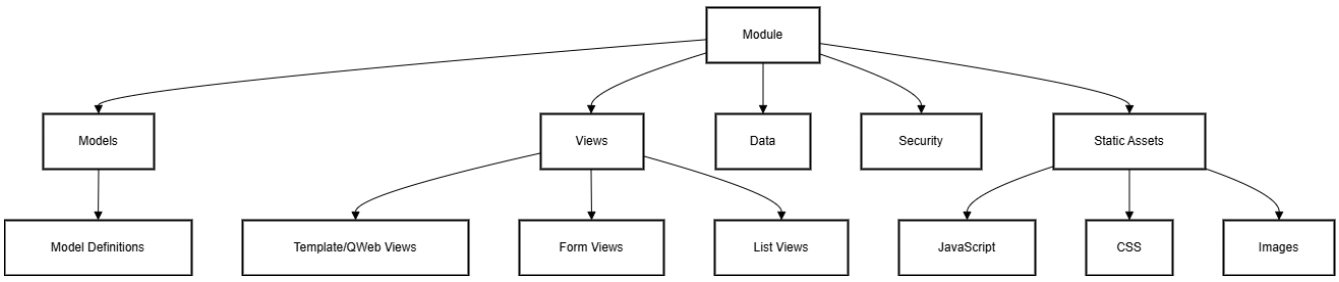

Рисунок 3.1 — Типова структура модулю Odoo

Важливою особливістю фреймворку є використання об'єктно—реляційного відображення (ORM), що дозволяє розробникам працювати з базою даних на рівні Python об'єктів, абстрагуючись від безпосередніх SQL запитів. Це значно спрощує розробку та підтримку додатків [31]. На рисунку 3.2, відображений потік даних, характерний для даної ERP системи.

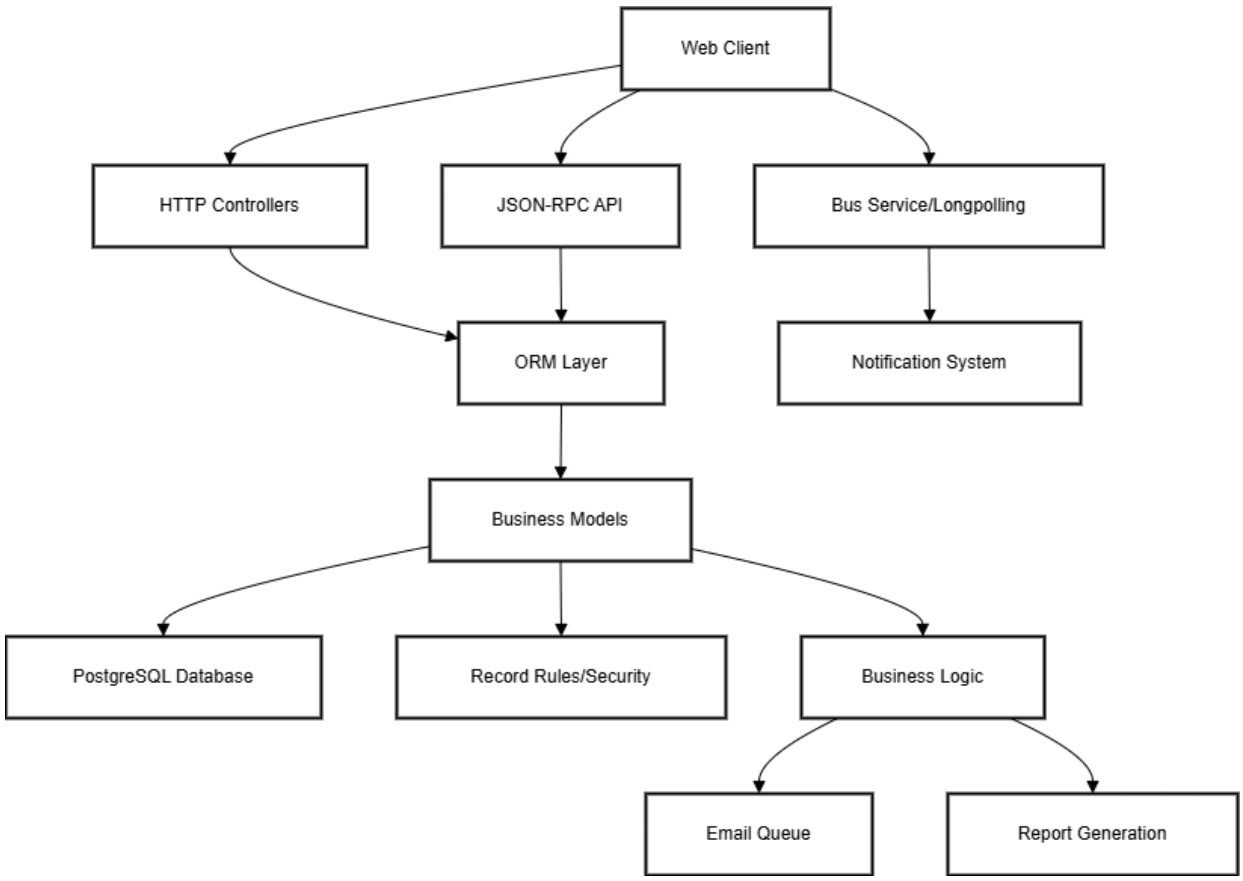

Рисунок 3.2 — Архітектура потоку даних Odoo

Веб інтерфейс Odoo реалізований з використанням сучасних веб технологій та забезпечує інтерактивний дизайн. Система використовує власний JavaScript фреймворк OWL (Odoo Web Library) для створення динамічних інтерфейсів користувача [32]. Архітектура передбачає можливість кастомізації як серверної, так і клієнтської частини додатку.

Система подій в Odoo дозволяє реалізувати складну бізнес логіку через механізми тригерів та обробників подій [33]. Це забезпечує гнучкість при розробці нових модулів та їх інтеграції з існуючими компонентами системи. Важливою особливістю є також вбудована система управління правами доступу, яка забезпечує детальний контроль над діями користувачів на рівні окремих полів та записів [34].

Архітектура Odoo підтримує можливість горизонтального масштабування через механізми реплікації баз даних та балансування навантаження. Це дозволяє

системі ефективно працювати як у невеликих організаціях, так і в великих підприємствах з високим навантаженням на систему [35].

3.4 Обґрунтування вибору сторонніх бібліотек

Для роботи з Instagram API обрано бібліотеку Requests, яка є фундаментальним інструментом для здійснення HTTP запитів у середовищі Python. Ця високорівнева бібліотека забезпечує уніфікований інтерфейс для взаємодії з веб сервісами, що робить її оптимальним вибором при інтеграції з Instagram API.

Бібліотека Requests відзначається абстрагуванням складних механізмів низькорівневої мережевої взаємодії, пропонуючи розробникам інтуїтивно зрозумілий API, що суттєво спрощує процеси автентифікації, передачі параметрів та обробки відповідей від серверів Instagram [36]. Важливою перевагою Requests є вбудована підтримка різноманітних методів HTTP (GET, POST, PUT, DELETE), що повністю відповідає архітектурним принципам RESTful API, на яких базується Instagram API.

З точки зору продуктивності, Requests демонструє оптимальний баланс між швидкодією та зручністю використання. Бібліотека підтримує як синхронні, так і асинхронні режими роботи, що дозволяє масштабувати систему відповідно до навантаження. Крім того, Requests забезпечує автоматичне стиснення даних, переспрямування запитів та підтримку проксі серверів, що підвищує ефективність мережевої взаємодії [36].

При розробці системи інтеграції з платформою Instagram критично важливим є забезпечення коректної обробки та валідації медіаконтенту перед його публікацією. Instagram встановлює жорсткі технічні вимоги до форматів зображень та відео, що можуть бути опубліковані через API платформи. Саме тому використання спеціалізованих бібліотек для попередньої обробки медіафайлів є необхідною складовою розроблюваної системи.

Бібліотека PyAV відіграє ключову роль у обробці відеоконтенту. Вона надає потужний інструментарій для аналізу та валідації відеофайлів, дозволяючи перевіряти їх відповідність вимогам Instagram щодо тривалості, роздільної здатності, частоти кадрів та бітрейту. PyAV забезпечує високу продуктивність завдяки використанню нативних bindings до бібліотеки FFmpeg, що є промисловим стандартом для обробки відео [37]. Важливою перевагою PyAV є можливість отримання метаданих відео без необхідності його повного декодування, що суттєво оптимізує процес валідації.

Для роботи із зображеннями обрано бібліотеку PIL (Python Imaging Library), яка є стандартом для обробки растрової графіки в Python. PIL забезпечує широкий функціонал для аналізу та модифікації зображень, включаючи перевірку розмірів, пропорцій, колірного простору та формату файлу. Бібліотека дозволяє ефективно виконувати необхідні перетворення для приведення зображень у відповідність до вимог Instagram API, такі як зміна розміру, обрізка та оптимізація якості [38].

Використання даних бібліотек надає суттєві переваги при розробці системи. По перше, вони забезпечують надійну валідацію медіаконтенту ще до спроби його завантаження, що запобігає виникненню помилок при взаємодії з API Instagram. По друге, обидві бібліотеки мають відмінну документацію та активну спільноту розробників, що спрощує їх інтеграцію та подальшу підтримку. По третє, вони надають оптимізований програмний інтерфейс, що дозволяє мінімізувати використання системних ресурсів при обробці медіафайлів.

Важливо відзначити, що обрані бібліотеки забезпечують крос платформну сумісність та не створюють додаткових залежностей, які могли б ускладнити розгортання системи. Їх стабільність та зрілість гарантують надійну роботу системи валідації медіаконтенту, що є критичним для безперебійного функціонування інтеграції між ERP Odoo та Instagram.

3.5 База даних PostgreSQL

PostgreSQL є багатофункціональною об'єктно—реляційною системою управління базами даних, яка має відкритий вихідний код та активно розвивається протягом більше 30 років [39]. Ця система забезпечує надійне зберігання даних та підтримує складні операції з ними, що робить її особливо привабливою для використання в корпоративних середовищах та веб додатках.

Однією з ключових переваг PostgreSQL є її архітектура, яка забезпечує високу надійність та масштабованість. Система підтримує паралельне виконання запитів, що дозволяє ефективно обробляти великі обсяги даних та обслуговувати численні одночасні підключення. Важливою особливістю є також підтримка транзакцій, що відповідають принципам ACID (атомарність, узгодженість, ізольованість, довговічність) [39].

PostgreSQL надає розширені можливості для реплікації даних, що є критично важливим для забезпечення високої доступності систем. Механізми реплікації дозволяють створювати розподілені системи з балансуванням навантаження, що значно підвищує продуктивність та надійність роботи бази даних. Особливо варто відзначити підтримку асинхронної реплікації, яка забезпечує ефективну роботу системи навіть при високих навантаженнях [40].

Важливою перевагою PostgreSQL є вбудована система резервного копіювання та відновлення даних. Ця функціональність дозволяє створювати надійні рішення для збереження даних та швидкого відновлення роботи системи у випадку збоїв. Система також підтримує point in time recovery, що дає можливість відновлювати стан бази даних на будь який момент часу [41].

PostgreSQL відрізняється від інших систем управління базами даних своєю розширюваністю. Користувачі можуть створювати власні типи даних, функції та процедури, використовуючи різні мови програмування. Це надає значну гнучкість при розробці складних систем та дозволяє оптимізувати роботу бази даних під конкретні потреби [39].

Система також забезпечує високий рівень безпеки даних завдяки підтримці різних методів автентифікації та шифрування. PostgreSQL надає гнучку систему управління правами доступу, що дозволяє точно контролювати, які користувачі мають доступ до певних даних та операцій. Це робить її особливо привабливою для використання в проектах, де безпека даних є критично важливою [42].

3.6 Організація захищеного тунелювання для обміну даними з використанням ngrok

При розробці системи інтеграції з Instagram API критично важливим є забезпечення надійного та безпечного каналу комунікації між локальним середовищем розробки та зовнішніми сервісами. Сучасні веб додатки, особливо ті, що взаємодіють з соціальними мережами, потребують обміну даними в реальному часі через механізм вебхуків. Instagram API, як і більшість сучасних API соціальних платформ, висуває жорсткі вимоги безпеки до кінцевих точок, які отримують сповіщення про події.

Основними технічними викликами на етапі розробки є необхідність підтримки протоколу HTTPS з валідним SSL сертифікатом для всіх кінцевих точок, що приймають вебхуки. Instagram API категорично відмовляється надсилати сповіщення на незахищені HTTP адреси або адреси з самопідписаними сертифікатами [43].

Для вирішення цієї задачі оптимальним рішенням є використання сервісу ngrok, що представляє собою спеціалізоване рішення для створення захищених тунелів між локальними серверами та публічним інтернетом. Архітектурно ngrok функціонує як reverse proxy сервер, який встановлює захищене з'єднання між клієнтським додатком, запущеним на локальній машині розробника, та серверами ngrok, розташованими в хмарній інфраструктурі [44].

Принцип роботи ngrok базується на створенні постійного WebSocket з'єднання між локальним клієнтом та хмарними серверами. Коли зовнішній сервіс,

такий як Instagram API, надсилає HTTP запит на публічну ngrok адресу, сервер ngrok отримує цей запит, шифрує його та передає через встановлене з'єднання на локальний сервер. Відповідь від локального сервера проходить зворотний шлях через той же захищений канал [44]. Процес налаштування та використання ngrok показано на рисунку 3.3.

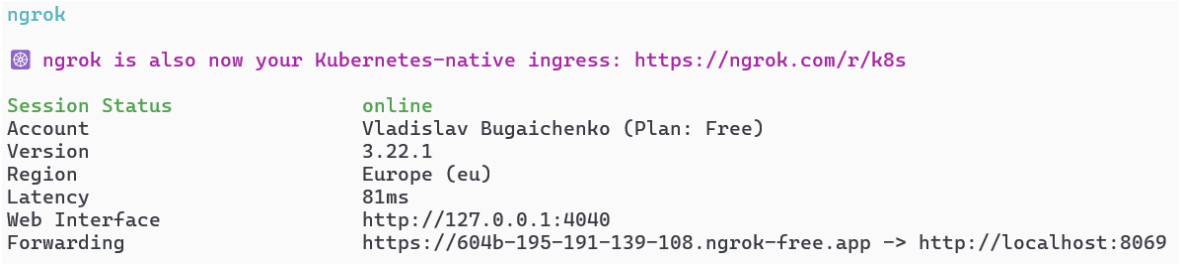

Рисунок 3.3 — Приклад роботи ngrok тунелю для перенаправлення HTTPS трафіку на локальний порт

Використання ngrok також надає додаткові переваги при розробці, такі як можливість інспектування всіх HTTP запитів в реальному часі, що значно спрощує процес налагодження взаємодії з API Instagram. Сервіс автоматично генерує унікальну публічну URL адресу, яку можна використовувати для налаштування вебхуків в Instagram API, забезпечуючи при цьому всі необхідні вимоги безпеки.

Важливо відзначити, що ngrok забезпечує не лише технічну можливість отримання вебхуків, але й відповідає всім вимогам безпеки Instagram API, включаючи підтримку валідних SSL сертифікатів та захищеного HTTPS з'єднання, що дозволяє зосередитись на реалізації бізнес логіки замість вирішення інфраструктурних завдань на етапі розробки.

4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ СИСТЕМИ ОБМІНУ ДАНИМИ МІЖ ERP ODOO ТА ПЛАТФОРМОЮ INSTAGRAM

У даному розділі детально розглядається програмна реалізація системи обміну даними між ERP Odoo та соціальною платформою Instagram. Основна увага приділяється технічним аспектам розробки, архітектурним рішенням та особливостям імплементації ключових компонентів системи.

Розроблена система представляє собою комплексне програмне рішення, що забезпечує безперебійний двосторонній обмін даними між корпоративною системою управління ресурсами підприємства Odoo та соціальною мережею Instagram. Реалізація включає створення надійних механізмів автентифікації, обробки даних та синхронізації інформації між платформами. Важливим аспектом розробки є забезпечення масштабованості та надійності системи, що досягається через використання сучасних технологій.

В процесі реалізації було враховано специфіку роботи з API обох платформ, особливості форматів даних та вимоги до безпеки інформації. Система розроблена з урахуванням можливості подальшого розширення функціоналу та адаптації до змін у вимогах користувачів чи оновлень платформ.

4.1 Архітектура розробленої системи

Розроблена система інтеграції між ERP Odoo та платформою Instagram складається з двох основних модулів: модуля управління контентом (us_instagram) та модуля обміну повідомленнями (us_instagram_messenger). Розглянемо детальніше архітектуру, модуля керування публікаціями акаунту, діаграма класів якого зображена на рисунку 4.1.

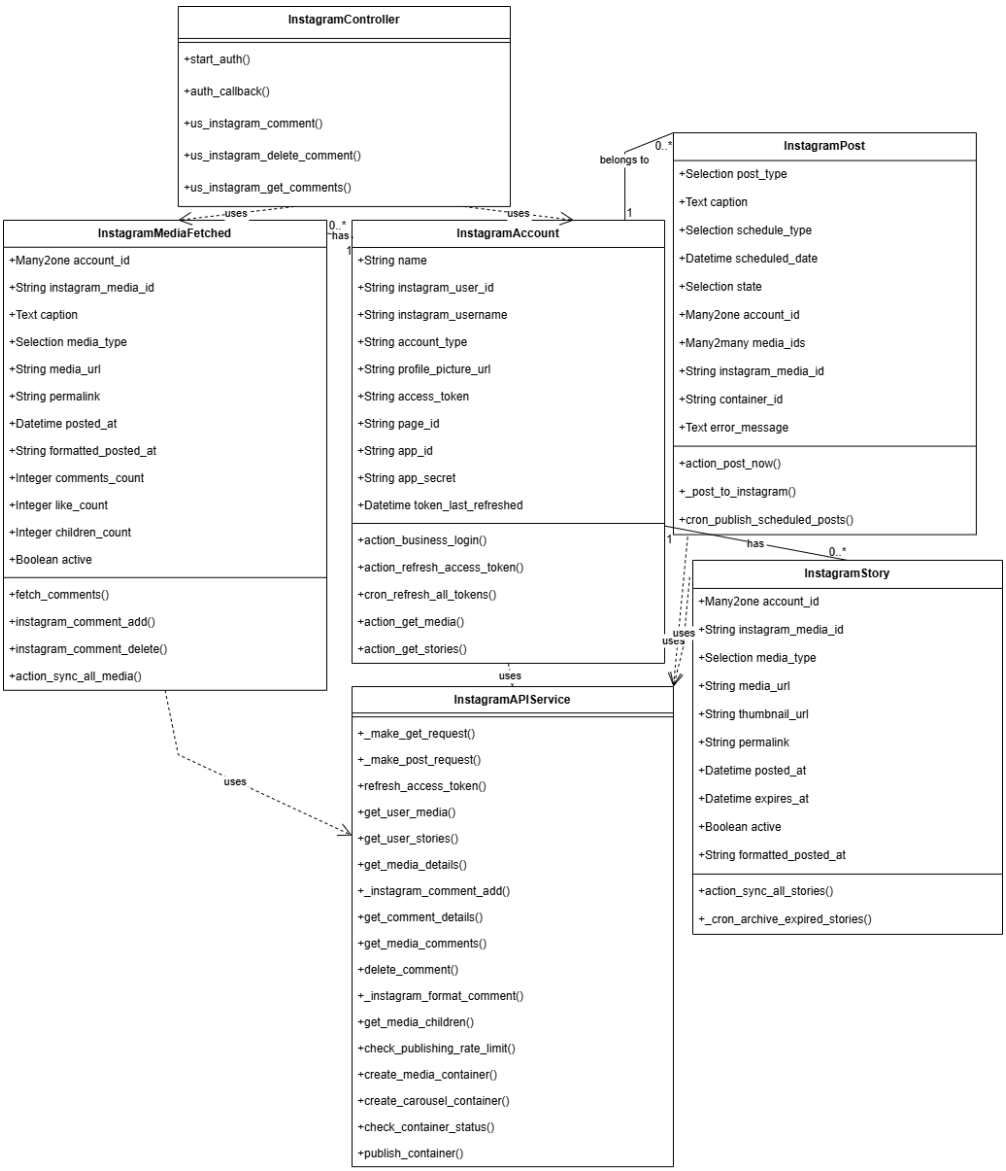

Рисунок 4.1 — Діаграма класів модуля управління контентом Instagram

Модуль управління контентом (us_instagram) забезпечує основну функціональність для роботи з Instagram акаунтом. Центральним компонентом є клас InstagramAccount, який представляє обліковий запис Instagram та містить всю необхідну інформацію для автентифікації та взаємодії з платформою. InstagramMediaFetched відповідає за завантаження та зберігання медіа контенту з Instagram, тоді як InstagramPost та InstagramStory представляють відповідно дописи та історії, які можуть бути опубліковані в соціальній мережі.

Взаємодія з Instagram Graph API реалізована через абстрактний клас InstagramAPIService, який надає необхідні методи для виконання операцій з API.

InstagramController виступає як контролер Odoo, що обробляє HTTP запити, пов'язані з автентифікацією та взаємодією з Instagram. Для зберігання медіафайлів використовується стандартна модель Odoo IrAttachment.

Наступним розглянемо архітектуру модулю обміну повідомленнями (us_instagram_messenger). InstagramWebhook виступає основним контролером для обробки вхідних повідомлень з Instagram API. Модуль розширює функціональність стандартних каналів Odoo через клас DiscussChannel, додаючи підтримку повідомлень Instagram. Модель ResPartner доповнена можливістю зберігання даних Instagram контактів. На рисунку 4.2 наведена діаграма класів даного модулю.

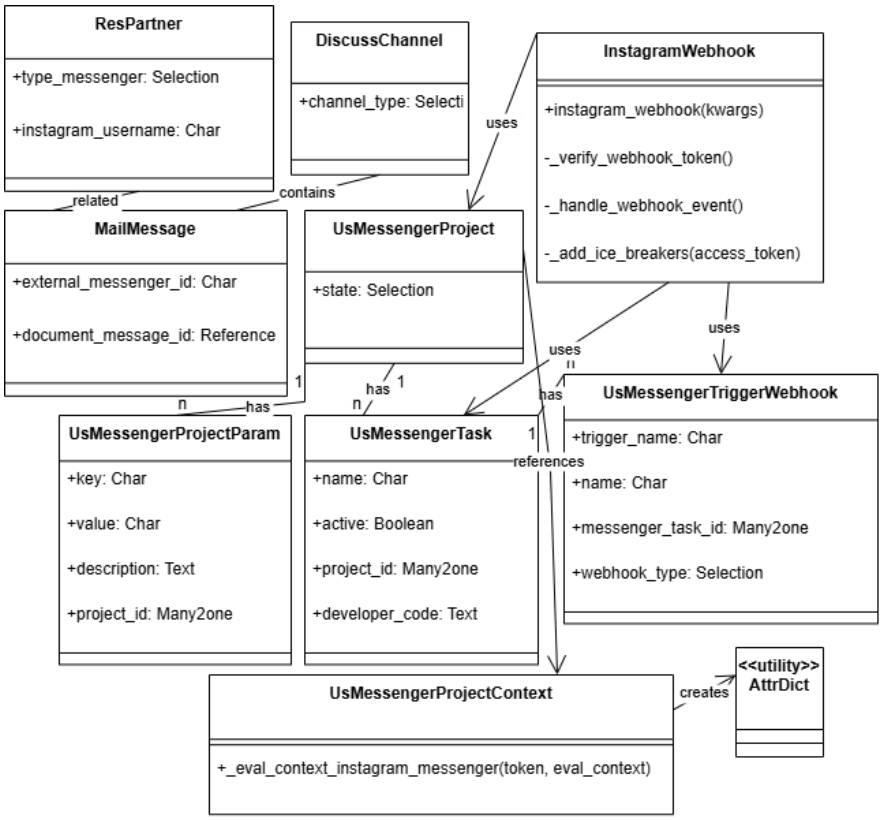

Рисунок 4.2 — Діаграма класів модуля обміну повідомленнями Instagram

Архітектура системи побудована за модульним принципом, де UsMessengerProject виступає центральним конфігураційним об'єктом, який об'єднує всі компоненти системи. UsMessengerProjectContext забезпечує необхідну API функціональність, а UsMessengerTask визначає задачі для обробки

повідомлень. Система підтримує різні типи тригерів (Webhook, Automation, Button), які визначають умови та способи запуску задач.

Така архітектурна організація забезпечує гнучкість системи, можливість її розширення та масштабування, а також чітке розділення відповідальності між компонентами. Модульний підхід дозволяє незалежно розвивати функціональність управління контентом та обміну повідомленнями, зберігаючи при цьому цілісність системи та зручність її використання.

4.2 Взаємодія системи Odoo з платформою Instagram

Перший модуль, що забезпечує обмін повідомленнями між системами, базується на використанні webhook технології. У цій схемі взаємодії присутні три основні компоненти: база даних Odoo, база даних Instagram та користувачі системи (оператор та клієнт). Механізм обміну повідомленнями працює в обох напрямках, забезпечуючи двосторонню комунікацію між оператором системи Odoo та клієнтом у Instagram, схему взаємодії наведено на рисунку 4.3.

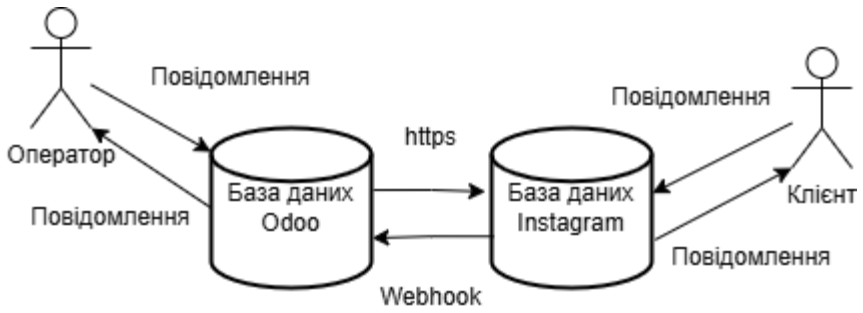

Рисунок 4.3 — Схема взаємодії модуля обміну повідомленнями

Другий модуль відповідає за управління публікаціями в Instagram акаунті та використовує протокол HTTP для обміну даними. У цьому випадку взаємодія відбувається безпосередньо між базами даних обох систем, де оператор може керувати контентом через інтерфейс Odoo, що показано на рисунку 4.4.

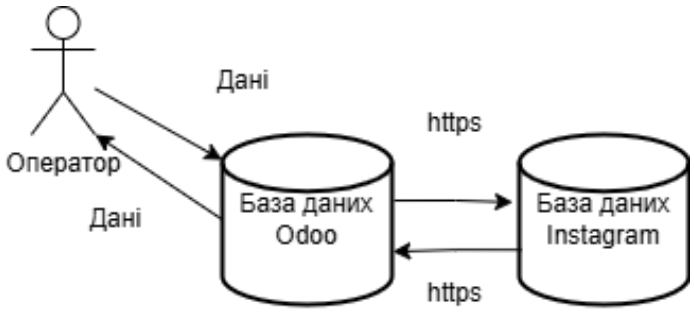

Рисунок 4.4 — Схема взаємодії модуля управління публікаціями

Обидва модулі працюють незалежно один від одного, але разом формують комплексне рішення для інтеграції Odoo з Instagram. Важливою особливістю такої архітектури є забезпечення безпеки даних через використання захищених протоколів передачі інформації та автентифікації користувачів. Система також передбачає можливість масштабування та додавання нових функціональних можливостей без порушення роботи існуючих компонентів.

Така архітектура дозволяє ефективно організувати робочий процес, де оператори можуть керувати взаємодією з клієнтами та контентом Instagram безпосередньо з інтерфейсу Odoo, що значно підвищує ефективність роботи та зменшує ймовірність помилок при передачі даних між системами.

4.3 Розробка структури бази даних

Розглянемо детально структуру бази даних першого модулю, us_instagram. Центральною сутністю є таблиця instagram.account, яка зберігає всі необхідні облікові дані. Ця таблиця має зв'язки типу "один до багатьох" з іншими основними сутностями системи: instagram.media.fetched та instagram.post. Детальну ієрархію даної бази даних зображено на рисунку 4.5.

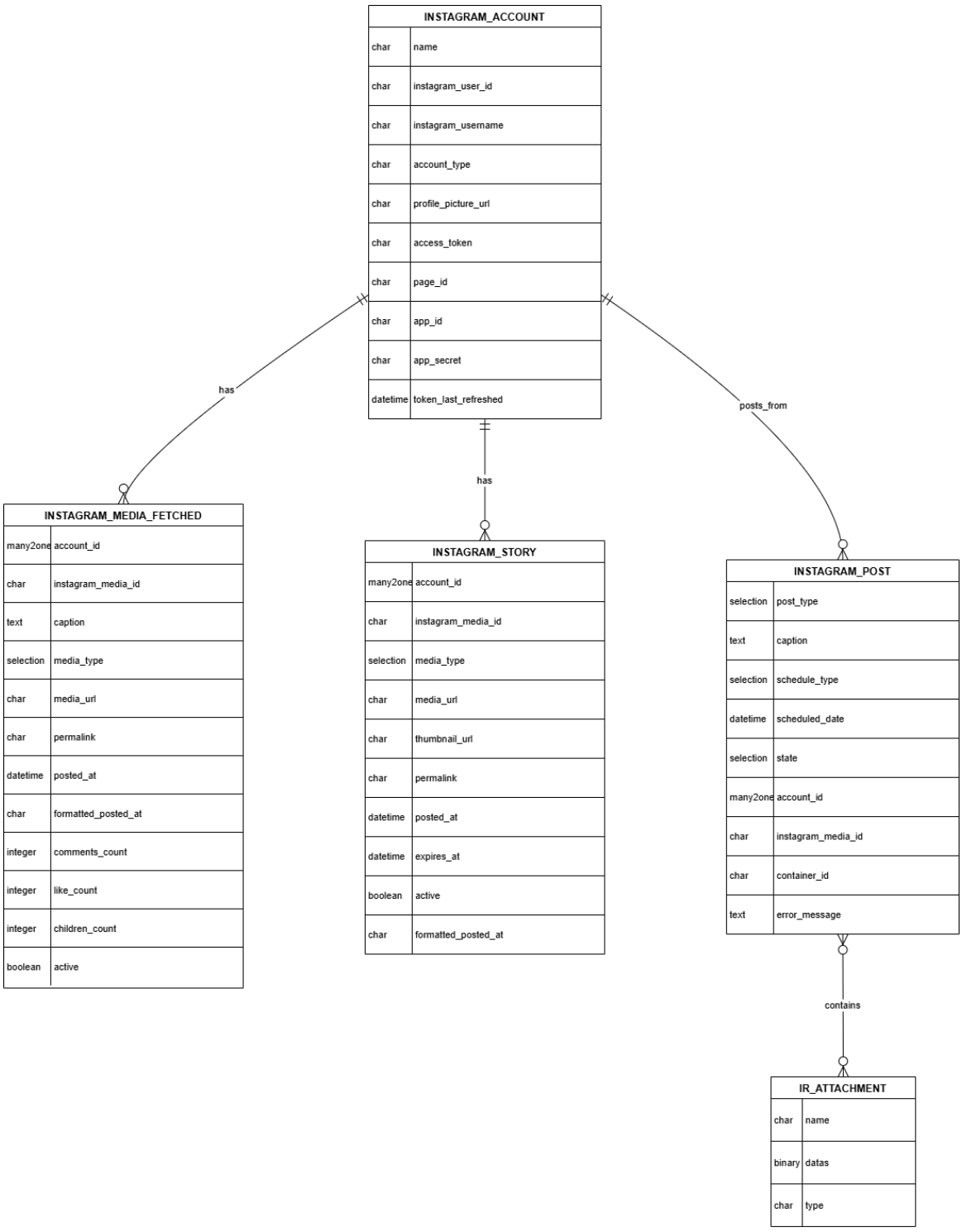

Рисунок 4.5 — Структура бази даних модуля управління публікаціями Instagram

Таблиця instagram.media.fetched призначена для зберігання контенту, отриманого з Instagram, включаючи метадані, такі як тип медіа, опис та часові мітки публікації. Таблиця instagram.post використовується для керування новими

публікаціями, які готуються до розміщення в Instagram, та підтримує різні типи контенту від звичайних зображень до каруселей та рілз.

Другий модуль системи, us_instagram_messenger, забезпечує функціональність обміну повідомленнями. Структура бази даних даного модулю побудована навколо таблиці us.messenger.project, яка містить конфігурацію ботів (акаунтів інстаграм) та їх контексти виконання. Детальна ієрархія бази даних, представлена на рисунку 4.6

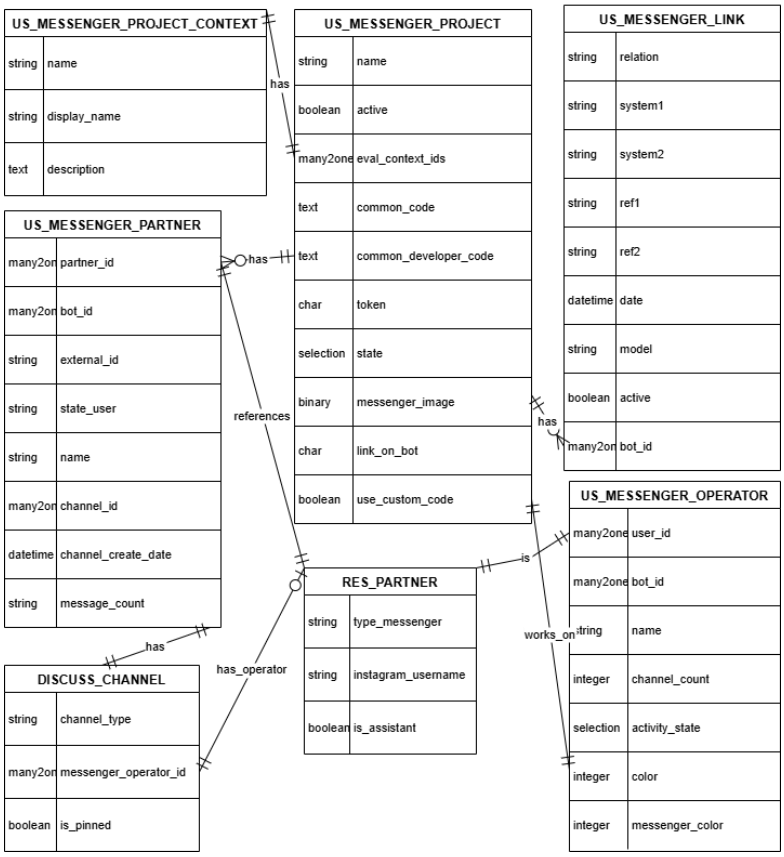

Рисунок 4.6 — Структура бази даних модуля обміну повідомленнями

Важливим елементом структури є таблиця us.messenger.partner, яка створює зв'язок між контактами системи (res.partner) та ботами, зберігаючи ідентифікатори користувачів у зовнішніх системах. Таблиця discuss.channel забезпечує підтримку різних каналів комунікації, включаючи Instagram, та містить зв'язки з операторами системи.

Для забезпечення цілісності даних та коректної синхронізації між системами використовується таблиця us.messenger.link, яка зберігає відповідності між об'єктами в Odoo та Instagram. Додатково, таблиця us.messenger.project.context визначає середовище виконання для ботів та їх налаштування.

Така структура бази даних забезпечує ефективне зберігання та обробку даних, необхідних для повноцінної інтеграції між ERP Odoo та Instagram, підтримуючи як публікацію контенту, так і двосторонню комунікацію з користувачами соціальної мережі.

4.4 Реалізація основного функціоналу

У процесі розробки системи обміну даними між ERP Odoo та платформою Instagram ключовим етапом є реалізація основного функціоналу, який забезпечує безперебійну та ефективну взаємодію між двома платформами. Основний функціонал системи спрямований на автоматизацію процесів синхронізації даних, керування контентом та забезпечення надійного обміну інформацією між корпоративною системою управління ресурсами та соціальною мережею.

Реалізація передбачає створення комплексного програмного рішення, яке враховує особливості архітектури обох платформ та забезпечує їх дієву інтеграцію. Важливим аспектом є також забезпечення масштабованості рішення та його здатності обробляти значні обсяги даних без втрати продуктивності.

4.4.1 Алгоритми обробки та відправлення повідомлень

Система обробки та відправлення повідомлень між ERP Odoo та платформою Instagram реалізована через комплексний алгоритмічний підхід, який забезпечує безперебійну двосторонню комунікацію та ефективну обробку різноманітних медіа даних.

Процес обробки вхідних повідомлень починається з фази декодування та валідації: вхідні дані перетворюються з JSON формату та валідуються для підтвердження джерела (Instagram). Далі відбувається обробка реакцій на повідомлення, при їх наявності, алгоритм ідентифікує події реакцій, здійснює пошук відповідного повідомлення за зовнішнім ідентифікатором, та виконує додавання або видалення реакцій у внутрішній системі. Наступним кроком відбувається диференціація повідомлень, що реалізована розмежуванням між повідомленнями надісланими оператором системи (echo messages) та новими вхідними повідомленнями від клієнта. Далі відбувається ідентифікація користувача: для нових повідомлень здійснюється запит до API Instagram для отримання інформації про профіль користувача та створення/оновлення відповідного запису картки партнера (клієнта) в системі, що продемонстровано на рисунку 4.7.

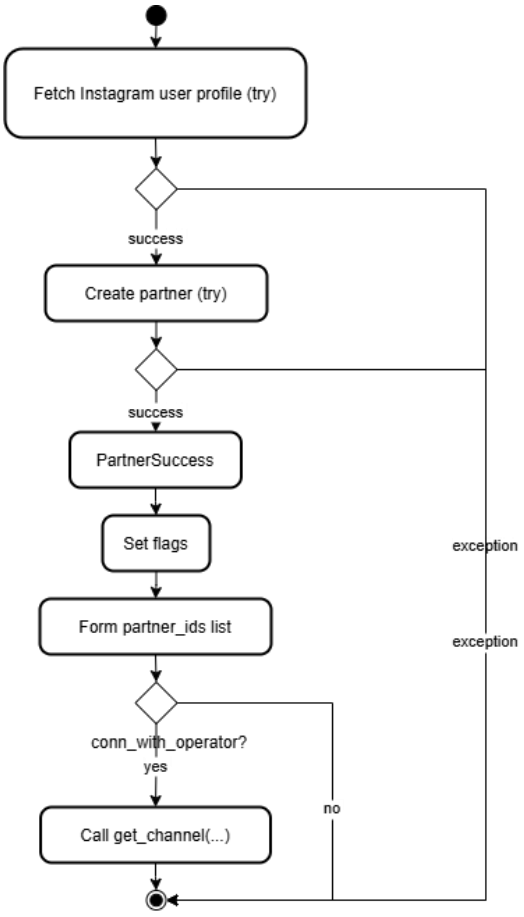

Рисунок 4.7 — Діаграма активності створення картки клієнта та чату

Наступним кроком створюються відповідні канали для комунікації, із можливим створенням супутніх бізнес об'єктів (потенційні клієнти у CRM). Далі відбувається диференційована обробка різних типів медіа вкладень (аудіо, відео, зображення). Для кожного медіа елемента визначається його тип та URL адреса, після чого генерується унікальний ідентифікатор для забезпечення цілісності даних. Система диференціює обробку аудіо та відео контенту від інших типів медіа, застосовуючи відповідні методи для отримання бінарних даних.

Для аудіо та відео файлів застосовується спеціалізований метод instagram_service_api.get_media_data(), що забезпечує коректне отримання медіа контенту. Залежно від типу медіа даних система динамічно формує відповідні розширення файлів MP3 для аудіо та MP4 для відео, створюючи уніфіковану структуру іменування файлів з використанням унікальних ідентифікаторів, блок-схема алгоритму наведена на рисунку 4.8.

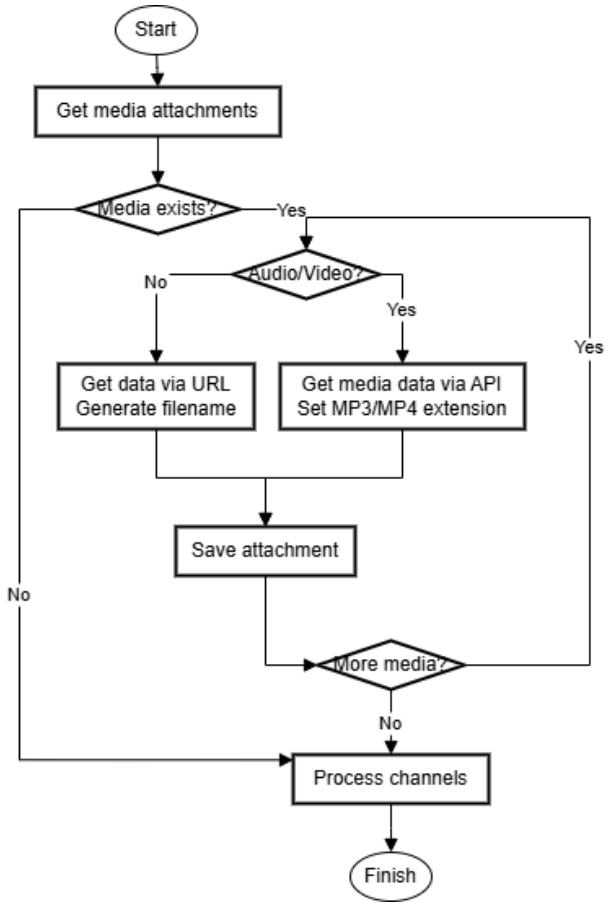

Рисунок 4.8 — Блок-схема алгоритму обробки медіа вкладень

Подальша трансмісія повідомлень до відповідних каналів комунікації в Odoo реалізується через функцію message_post(), що інкапсулює всі необхідні метадані та вкладення. Важливими аспектами є встановлення правильних ідентифікаторів автора, зовнішніх посилань та атрибутів повідомлення для забезпечення коректної асоціації з відповідними бізнес об'єктами в ERP системі.

Даний алгоритм демонструє ефективність інтеграції зовнішньої системи обміну повідомленнями з корпоративною ERP системою, забезпечуючи безперервність комунікації та автоматизацію бізнес процесів.

Алгоритм відправлення повідомлень з Odoo до Instagram, реалізований через спеціалізований обробник, який контролює відповідність повідомлень вимогам платформи Instagram. Система перевіряє наявність зовнішніх ідентифікаторів, типу каналу та зв'язку між системами. Особлива увага приділяється обробці медіа файлів, де контролюються формати та розміри вкладень згідно з обмеженнями Instagram. Діаграма послідовності алгоритму, що відповідає за відправлення повідомлень, зображена на рисунку 4.9.

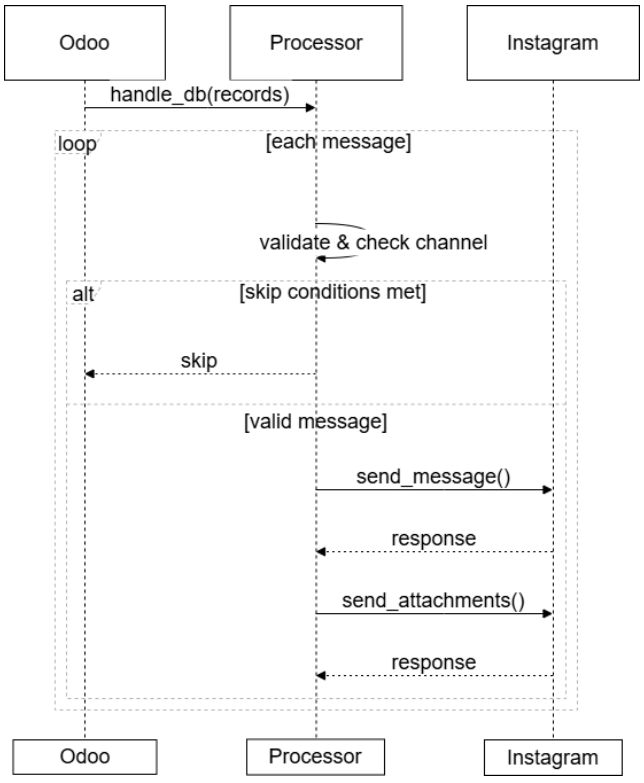

Рисунок 4.9 — Діаграма послідовності відправлення повідомлень

Система забезпечує коректне форматування текстових повідомлень, включаючи можливість додавання підписів та спеціальних міток, що покращує користувацький досвід та полегшує ідентифікацію джерела повідомлень.

4.4.2 Методи синхронізації та публікації контенту

У даному пункті розглянемо методи синхронізації та публікації контенту між системою ERP Odoo та соціальною мережею Instagram. Основними компонентами реалізації є два ключові процеси: синхронізація існуючих публікацій та створення нових дописів.

Алгоритм синхронізації працює за чітко визначеною послідовністю дій. Процес включає перевірку наявності нових публікацій, оновлення існуючих записів та видалення застарілих даних з системи, що відображено на рисунку 4.10.

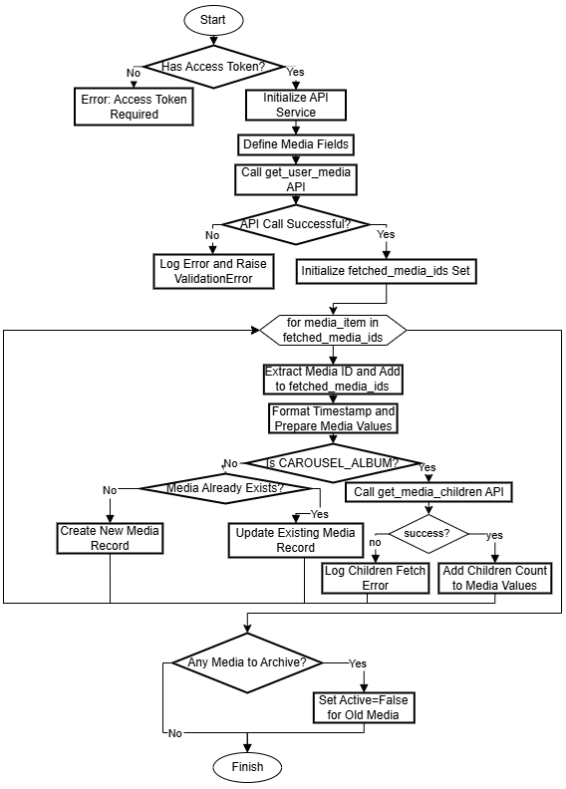

Рисунок 4.10 — Блок-схема алгоритму синхронізації даних

Процес синхронізації медіа контенту з Instagram реалізовано через метод action_get_media(), який забезпечує комплексний підхід до отримання, обробки та збереження даних.

На початковому етапі відбувається перевірка наявності та валідності токена доступу. Ця перевірка гарантує, що процес синхронізації виконуватиметься лише за наявності коректних авторизаційних даних, що забезпечує належний рівень безпеки та відповідність вимогам Instagram Graph API. Далі виконується запит до API здійснюється через спеціалізований сервісний клас, що інкапсулює логіку взаємодії з API, забезпечуючи абстракцію низькорівневих деталей HTTP комунікації та обробки помилок. Після отримання відповіді від API система виконує ітеративну обробку кожного елемента медіа контенту. На цьому етапі формується множина ідентифікаторів отриманих публікацій, яка в подальшому використовуватиметься для виявлення неактуальних записів у базі даних. Далі для кожної публікації система формує структурований набір даних. Структура даних відповідає моделі instagram.media.fetched в системі Odoo та забезпечує систематизоване збереження інформації. Для публікацій типу CAROUSEL_ALBUM реалізовано додаткову логіку отримання та обробки дочірніх елементів. Система забезпечує коректне визначення кількості елементів у каруселі з відповідною обробкою потенційних помилок, що підвищує стійкість алгоритму до нестандартних ситуацій. Наступним кроком на основі ідентифікаторів публікацій система визначає необхідність оновлення існуючих записів або створення нових. Такий підхід реалізує ідемпотентність операцій та забезпечує цілісність даних при повторних синхронізаціях. Завершальним етапом є виявлення та архівація публікацій, які були видалені з Instagram або стали недоступними через API. Даний підхід забезпечує збереження історичних даних з одночасним відображенням актуального стану контенту в Instagram. Діаграма послідовності алгоритму, що відповідає за синхронізацію даних наведено на рисунку 4.11.

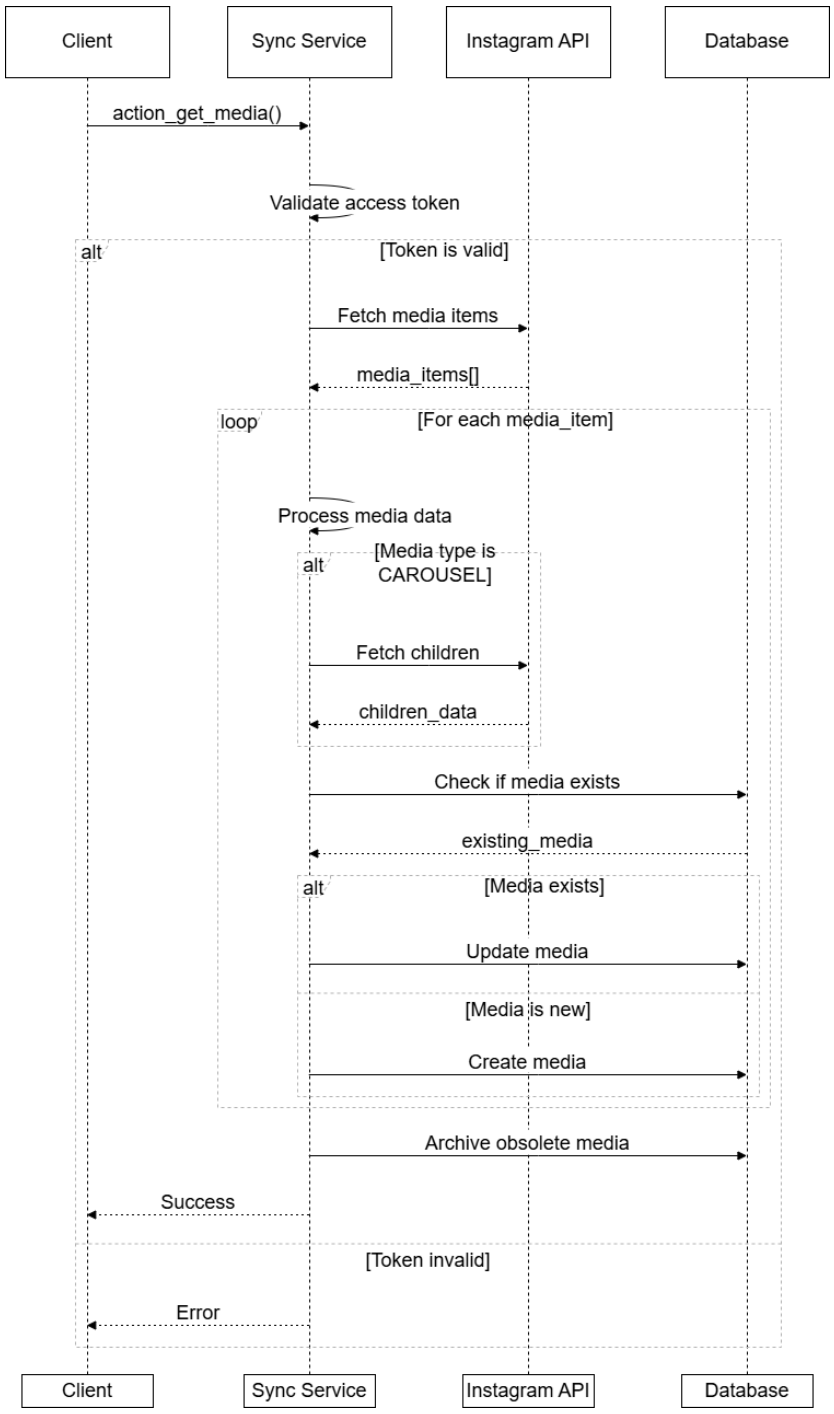

Рисунок 4.11 — Діаграма активності синхронізації даних публікацій

Для забезпечення надійної публікації контенту розроблено комплексний механізм валідації медіафайлів. Особлива увага приділяється перевірці зображень перед їх публікацією, що включає валідацію формату, розміру та інших технічних параметрів. Блок-схему алгоритму, що відповідає за валідацію медіа наведено на рисунку 4.12.

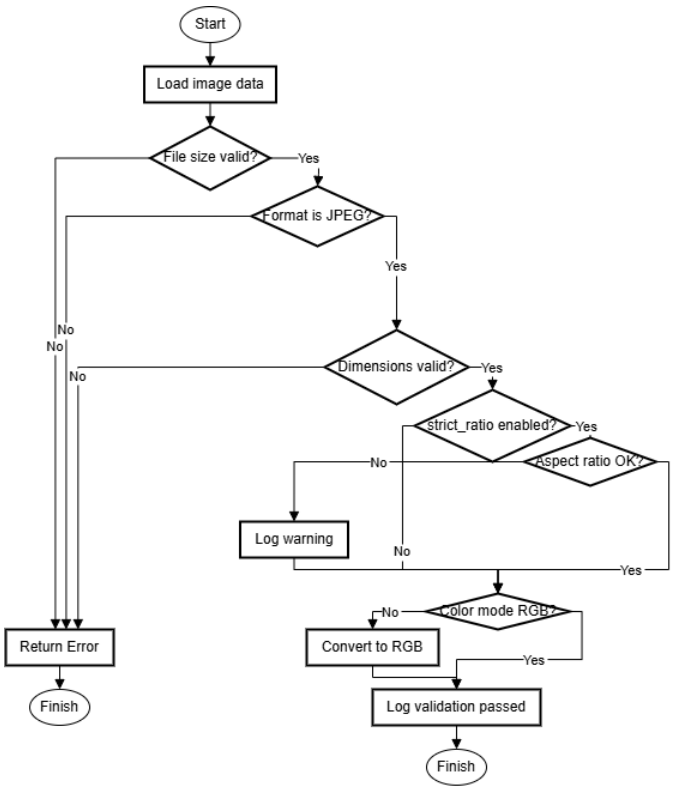

Рисунок 4.12 — Блок-схема алгоритму валідації зображень

Процес публікації контенту реалізовано як послідовність взаємопов'язаних етапів. Спочатку відбувається підготовка, під час якої користувач обирає Instagram акаунт, завантажує необхідні медіафайли та додає опис публікації. Наступним кроком є ініціювання публікації через метод action_post_now(), який запускає процес валідації та перевірки всіх необхідних параметрів.

Важливим аспектом є валідація медіафайлів, яка включає перевірку формату файлу (обмеження до JPEG/JPG), контроль розміру файлу відповідно до встановлених лімітів платформи Instagram, перевірку співвідношення сторін зображення та відповідність кольорового режиму вимогам RGB. Після успішного проходження всіх перевірок система отримує необхідні токени доступу та ідентифікатори користувача для взаємодії з API Instagram.

Фінальний етап включає створення медіа контейнера через Instagram API, перевірку його статусу та безпосередню публікацію контенту. Після успішного завершення публікації система оновлює статус допису на 'posted' та зберігає всі необхідні ідентифікатори для подальшого відстеження та керування публікацією.

4.4.3 Алгоритм взаємодії з коментарями

Алгоритм взаємодії з коментарями в системі інтеграції ERP Odoo з Instagram реалізує комплексний процес додавання нових коментарів до медіа постів. Розглянемо детально його реалізацію та основні етапи роботи.

Процес починається з моменту, взаємодії з коментарями через канбан запис в інтерфейсі Odoo. Як показано на рисунку 4.13, обробка користувацької взаємодії реалізується через спеціальний метод _onInstagramCommentsClick, який відповідає за перехоплення кліку та подальшу обробку події.

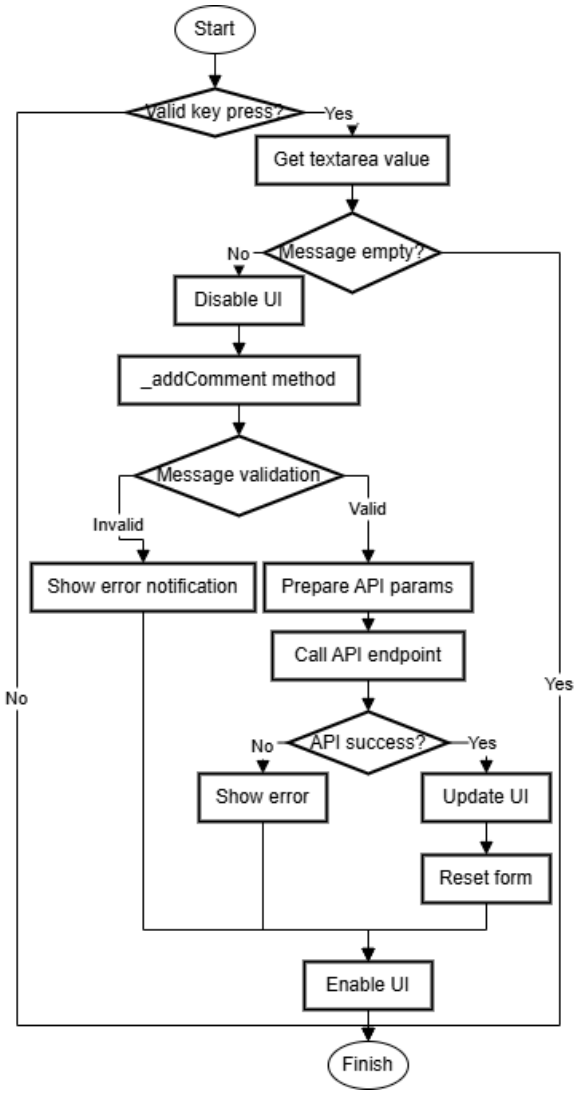

Рисунок 4.13 — Блок-схема алгоритму обробки користувацької взаємодії при додаванні коментаря

Після ініціації процесу система виконує RPC запит для отримання існуючих коментарів через endpoint /us_instagram/get_comments. Користувачу надається можливість введення нового коментаря через спеціальний компонент UsInstagramCommentsReply, який також підтримує функціонал додавання емодзі завдяки інтеграції з useEmojiPicker.

Особливу увагу приділено процесу відправки коментаря на серверну частину, діаграма послідовності якого демонструється на рисунку 4.14. При натисканні клавіші Enter активується метод _onAddComment, який здійснює валідацію введеного тексту та передає керування методу _addComment для подальшої обробки.

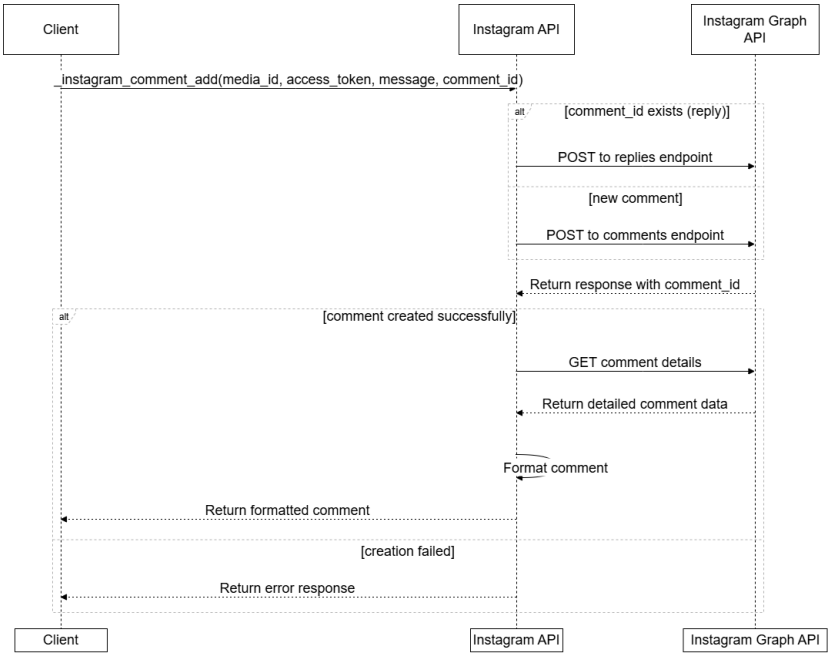

Рисунок 4.14 — Діаграма послідовності алгоритму створення коментаря

Взаємодія з бекендом відбувається через RPC запит до endpoint /us_instagram/comment, де формується структурований об'єкт з необхідними параметрами, включаючи media_id, текст коментаря та, за потреби, ідентифікатор батьківського коментаря для реалізації функціоналу відповідей.

На серверній стороні контролер us_instagram_comment здійснює пошук відповідного запису instagram.media.fetched та викликає метод

instagram_comment_add моделі. Цей метод, у свою чергу, взаємодіє з Instagram Graph API через спеціалізований сервіс, виконуючи POST запит для публікації коментаря.

Після успішного додавання коментаря система форматує отримані дані через метод _instagram_format_comment, створюючи структурований об'єкт, що містить всю необхідну інформацію про новий коментар. У випадку успішного виконання операції, інтерфейс користувача оновлюється: новий коментар додається на початок списку, оновлюється лічильник коментарів, а форма введення очищується для подальшого використання.

Важливим аспектом реалізації є обробка помилок та зворотний зв'язок з користувачем. У разі виникнення помилок під час додавання коментаря, система відображає відповідне повідомлення через механізм notification.add, забезпечуючи належне інформування користувача про статус операції.

4.4.4 Алгоритм автентифікації та авторизації

Алгоритм автентифікації та авторизації користувача Instagram в системі Odoo реалізовано як комплексний процес, що складається з кількох послідовних етапів.

Процес починається з ініціювання автентифікації через метод start_auth, який виконує первинну перевірку та підготовку даних. На цьому етапі система отримує ідентифікатор облікового запису Instagram та перевіряє наявність необхідних параметрів конфігурації. Після успішної валідації параметрів, ідентифікатор account_id зберігається в сесії для подальшого використання. Далі формується URL адреса для OAuth авторизації з необхідними параметрами, і користувач перенаправляється на сторінку авторизації Instagram.

Після надання дозволу користувачем система через метод auth_callback отримує код авторизації, який обмінюється на короткостроковий токен доступу, а потім конвертується в довгостроковий. Блок-схему даного алгоритму зображено на рисунку 4.15.

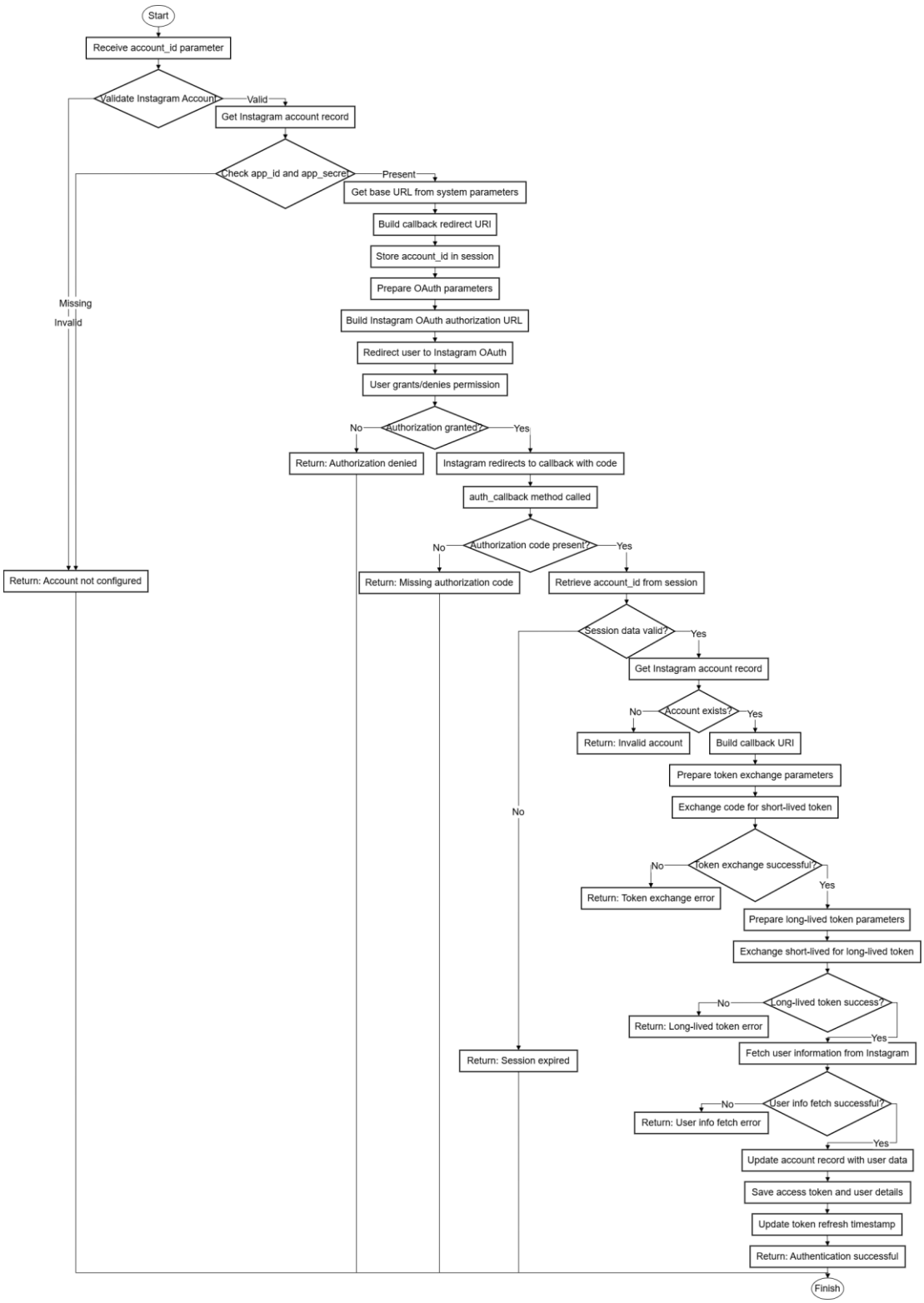

Рисунок 4.15 — Блок-схема алгоритму автентифікації та авторизації в системі

Заключним етапом є збереження отриманого довгострокового токену в моделі instagram.account. Цей токен надалі використовується для здійснення API запитів до Instagram, забезпечуючи безперервну роботу інтеграції між системами.

5 РОБОТА КОРИСТУВАЧА З СИСТЕМОЮ

У даному розділі розглядається використання системи обміну даними між ERP Odoo та соціальною платформою Instagram з точки зору кінцевого користувача. Значну увагу приділено огляду користувацького інтерфейсу, що забезпечує оптимальну взаємодію із системою, та ключових функціональних можливостей програмного продукту.

5.1 Системні вимоги

Системні вимоги для розгортання та експлуатації ERP системи Odoo суттєво залежать від масштабу впровадження та кількості одночасних користувачів системи, як показано в таблиці 5.1. Для забезпечення стабільної роботи серверної частини необхідно враховувати як апаратні характеристики, так і програмне забезпечення.

Таблиця 5.1 — Системні вимоги серверної частини Odoo

Кількість користувачів	**CPU**	**RAM**	**Дисковий простір**	**Додатково**
До 10	2-4 ядра	2-4 ГБ	10-20 ГБ	
До 50	4 ядра	8-16 ГБ	20-50 ГБ	
50-100	4-8 ядер	16-32 ГБ	50 ГБ+	Можливо розділення серверів
150+	8+ ядер	64 ГБ+	50 ГБ+	Балансування навантаження

Для малих підприємств з кількістю користувачів до десяти осіб достатнім є використання серверної конфігурації початкового рівня. При цьому мінімальні вимоги передбачають наявність двоядерного процесора та 2-4 гігабайти оперативної пам'яті. Така конфігурація забезпечує комфортну роботу з базовими модулями системи.

При збільшенні кількості користувачів до середнього рівня (20-50 осіб) зростають вимоги до апаратного забезпечення. У такому випадку рекомендується використовувати сервер з чотириядерним процесором та обсягом оперативної пам'яті від 8 до 16 гігабайт. Значне підвищення продуктивності досягається за рахунок використання твердотільних накопичувачів, особливо при роботі з великими масивами даних.

Для великих підприємств з кількістю користувачів понад сто осіб рекомендується застосовувати розподілену архітектуру з окремими серверами для додатку та бази даних. Кожен сервер повинен мати щонайменше восьмиядерний процесор та 32-64 гігабайти оперативної пам'яті. При перевищенні межі у 150 користувачів доцільним стає впровадження системи балансування навантаження та виділення окремого потужного сервера для системи керування базами даних.

Важливо зазначити, що наведені вимоги є рекомендованими та можуть варіюватися залежно від специфіки використання системи, кількості одночасно працюючих користувачів та інтенсивності операцій з базою даних. Операційна система Linux є рекомендованою платформою для розгортання серверної частини Odoo, а PostgreSQL використовується як основна система керування базами даних.

5.2 Встановлення та налаштування системи

Процес встановлення та налаштування системи обміну даними між ERP Odoo та платформою Instagram складається з декількох послідовних етапів. Насамперед необхідно здійснити інсталяцію відповідного модуля в системі Odoo. Як показано

на рисунку 5.1, встановлення модуля Instagram Integration відбувається через стандартний інтерфейс управління застосунками Odoo.

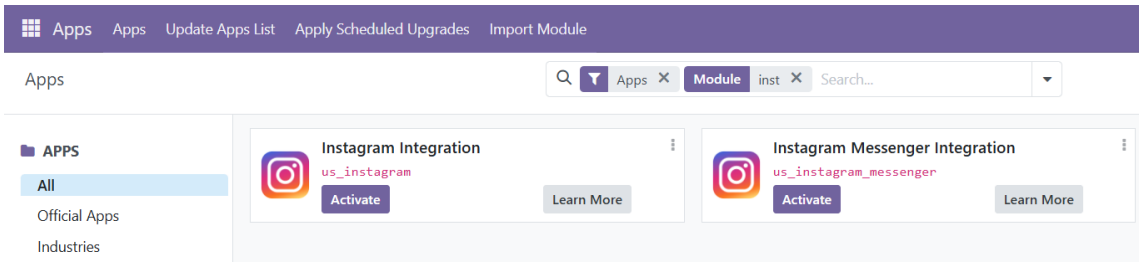

Рисунок 5.1 — Встановлення модуля Instagram Integration в Odoo

Наступним важливим кроком є отримання необхідних облікових даних та ключів доступу від платформи Meta для взаємодії з Instagram API. На рисунку 5.2 представлено інтерфейс налаштування Instagram API, де необхідно отримати відповідні ключі доступу для подальшого застосування при автентифікації.

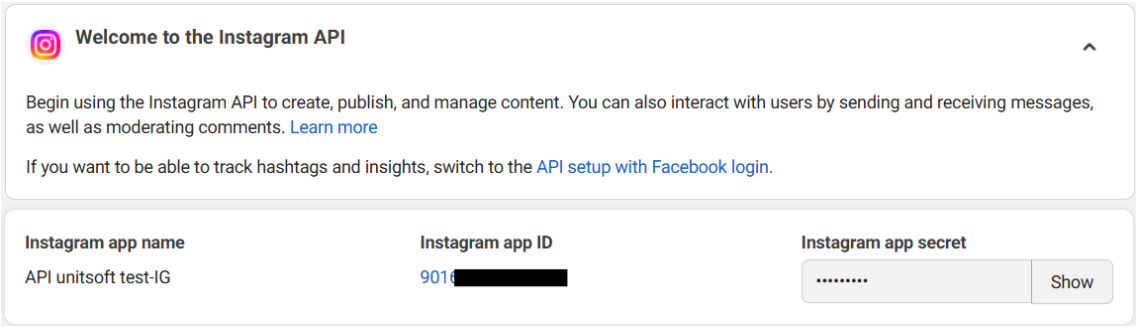

Рисунок 5.2 — Інтерфейс налаштування Instagram API

Після отримання необхідних даних здійснюється їх введення в систему Odoo. У конфігураційному меню модуля вказуються основні параметри з'єднання: назва акаунту в системі, ідентифікатор застосунку (Instagram App ID) та секретний ключ (Instagram App Secret). Наступним кроком є автентифікація бізнес акаунту шляхом настискання відповідної кнопки «Authenticate Business Account», що зображено на рисунку 5.3.

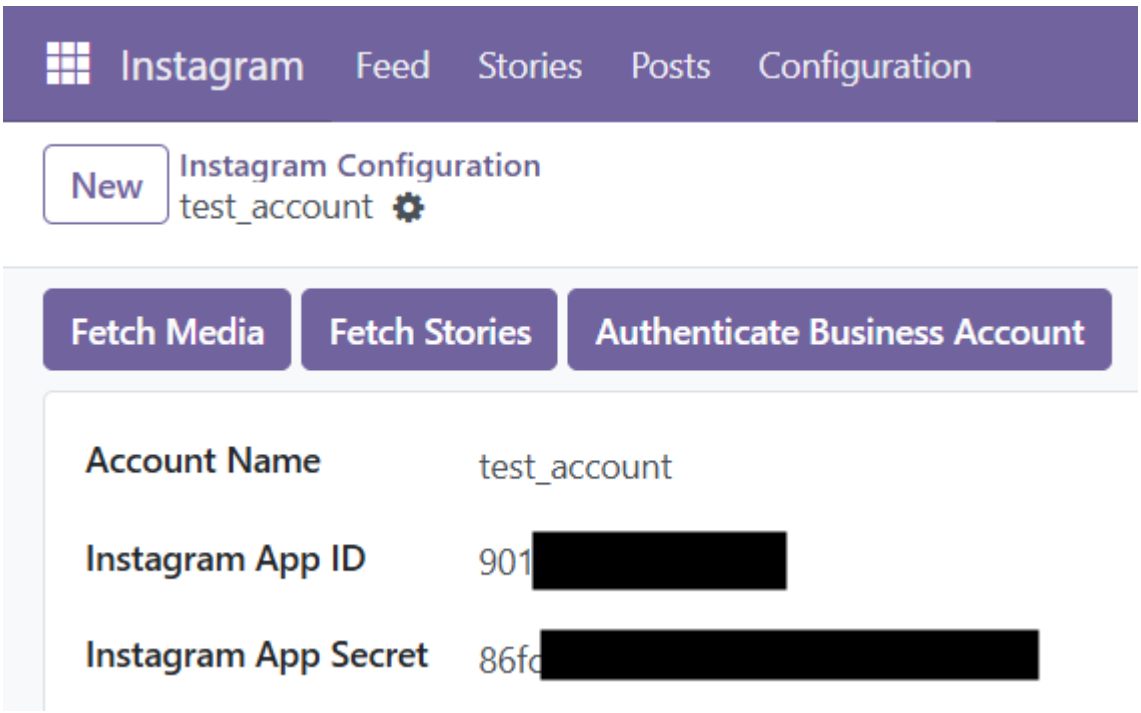

Рисунок 5.3 — Конфігурація облікового запису Instagram в Odoo

Варто зазначити, що коректне налаштування всіх параметрів є критично важливим для забезпечення надійного та безпечного обміну даними між системами. Після завершення налаштування рекомендується провести тестування з'єднання для перевірки правильності введених даних та функціональності системи в цілому.

5.3 Огляд й сценарії взаємодії з системою

У даному підрозділі розглянемо основні сценарії взаємодії користувачів із розробленою системою обміну даними між ERP Odoo та платформою Instagram. Особлива увага приділяється інтерфейсу користувача, який забезпечує зручну та ефективну взаємодію з системою, а також огляду інструментальниих засобів для оптимального керування контентом та підвищення ефективності комерційних операцій.

5.3.1 Огляд сценарію управління публікацією контенту

Система управління публікацією контенту в розробленому інтеграційному рішенні забезпечує повний цикл роботи з публікаціями в Instagram безпосередньо з інтерфейсу ERP Odoo. Розглянемо основні сценарії взаємодії користувача з системою при управлінні контентом.

Процес створення нової публікації реалізований через інтуїтивно зрозумілий інтерфейс. Користувач має можливість вибрати тип публікації (зображення, карусель, історія або коротке відео), додати необхідні медіафайли та написати текстовий опис. Система підтримує завантаження файлів різних форматів, включаючи зображення JPG та відео MP4. Приклад створення нової публікації зображено на рисунку 5.4.

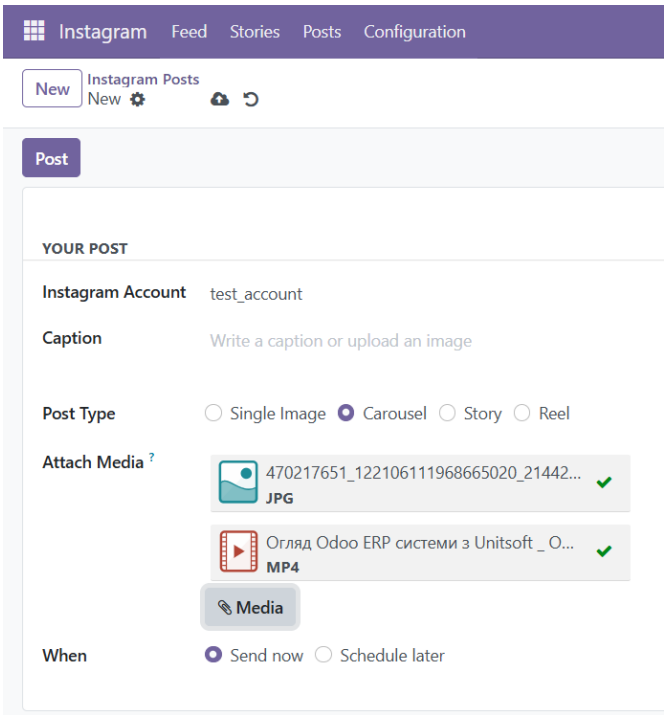

Рисунок 5.4 — Інтерфейс створення нової публікації в системі

Важливим аспектом функціонування системи є можливість перегляду та моніторингу вже опублікованого контенту. На рисунку 5.5 представлено інтерфейс

відображення стрічки публікацій акаунту, де користувач може переглядати історію розміщених постів та їх метрики.

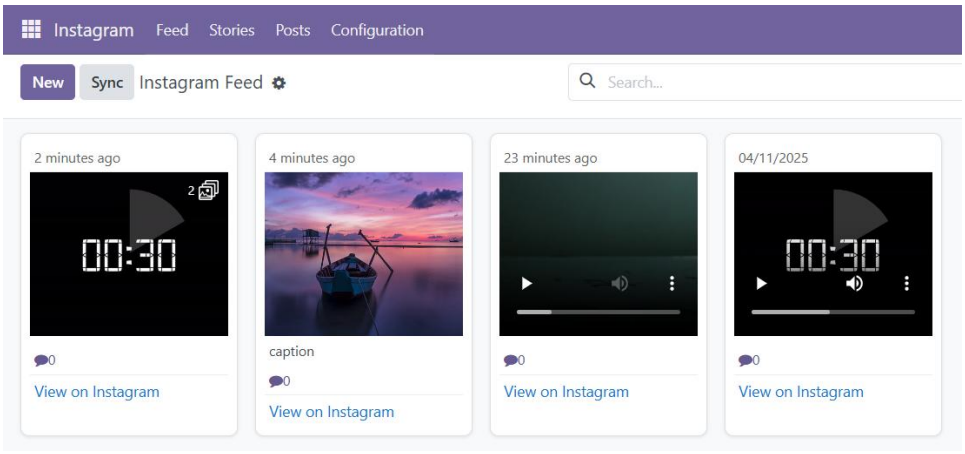

Рисунок 5.5 — Інтерфейс перегляду опублікованого контенту

Система також надає функціонал планування публікацій на майбутній час. Як показано на рисунку 5.6, користувач може встановити дату та час автоматичної публікації контенту.

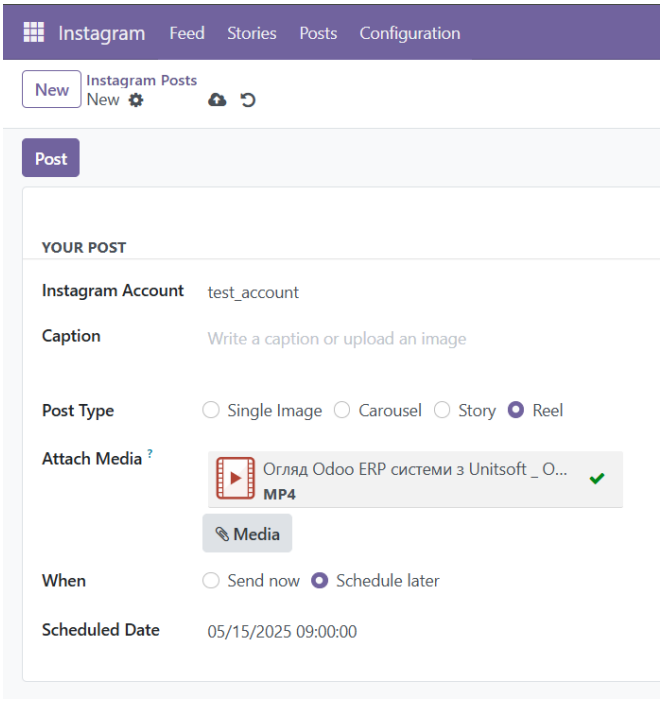

Рисунок 5.6 — Інтерфейс планування публікації

Розроблений функціонал дозволяє ефективно керувати контентом Instagram безпосередньо з системи Odoo, забезпечуючи централізоване управління маркетинговими активностями компанії. Реалізований підхід суттєво оптимізує робочі процеси та мінімізує необхідність перемикання між різними платформами при роботі з соціальними мережами.

5.3.2 Огляд сценарію модерації коментарів

Процес модерації коментарів є важливою складовою управління контентом у соціальних мережах, особливо при інтеграції з корпоративними системами. В рамках розробленої системи обміну даними між ERP Odoo та Instagram реалізовано комплексний підхід до модерації користувацьких коментарів, що дозволяє ефективно керувати взаємодією з аудиторією.

На рисунку 5.7 представлено інтерфейс публікації в Instagram із коментарями користувачів, які потребують модерації.

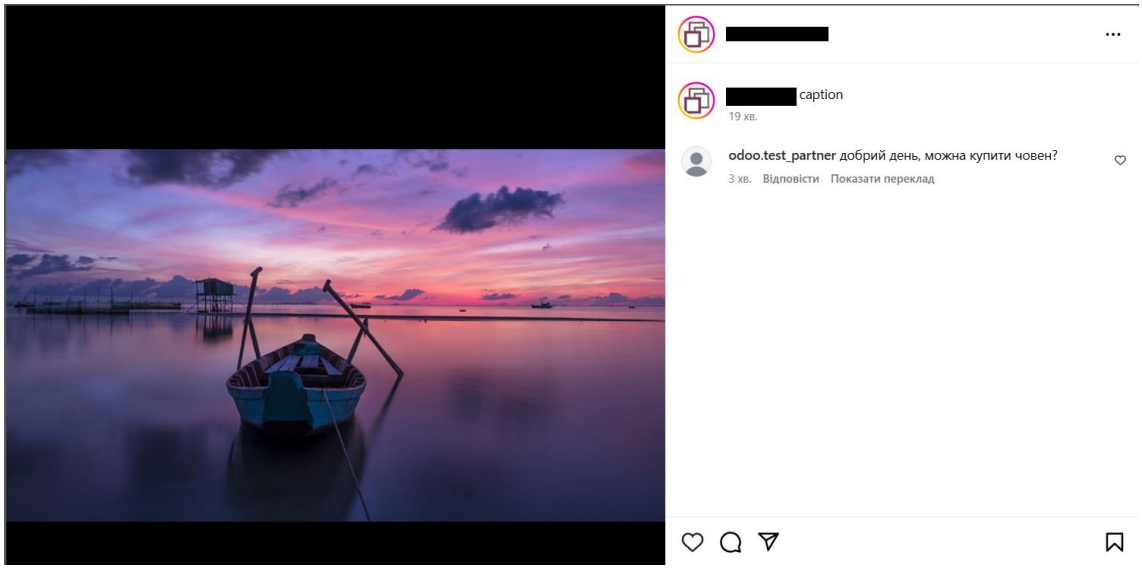

Рисунок 5.7 — Інтерфейс публікації в Instagram з коментарями користувачів

Система автоматично агрегує всі коментарі до публікації та надає можливість їх перегляду безпосередньо в інтерфейсі Odoo. У даному інтерфейсі, що зображено на рисунку 5.8, представлено контекст відповідної публікації разом із пов'язаними коментарями, що дозволяє оперативно приймати рішення щодо модерації контенту.

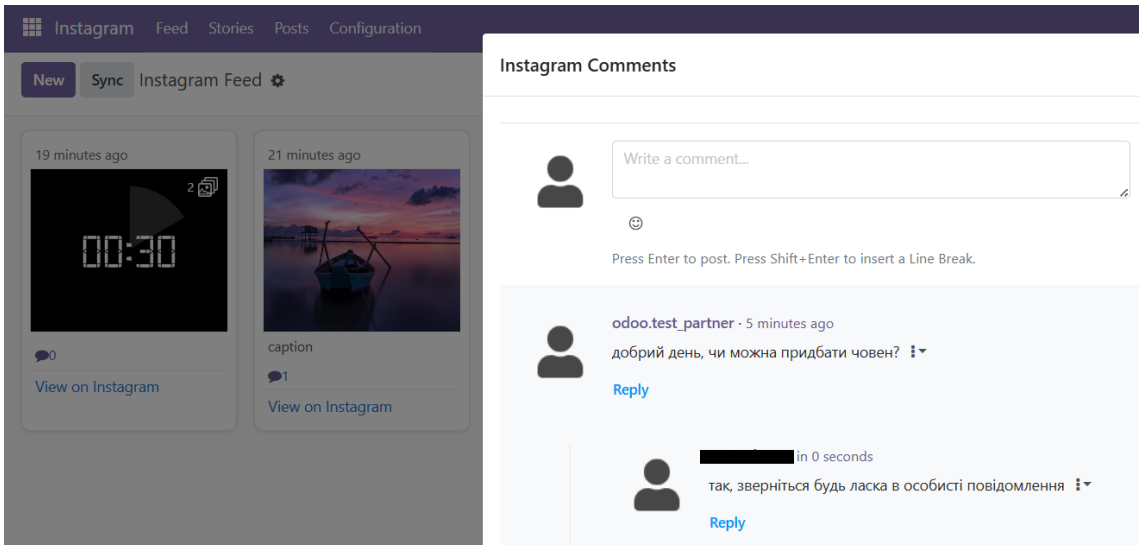

Рисунок 5.8 — Інтерфейс модерації коментарів в системі Odoo

Процес модерації включає можливість перегляду часу публікації коментаря, його змісту та автора. Адміністратор може здійснювати різні дії з коментарями: відповідати на них або видаляти небажаний контент. Така функціональність забезпечує ефективний контроль над користувацьким контентом та дозволяє підтримувати належний рівень комунікації з аудиторією.

5.3.3 Огляд сценарію обміну повідомленнями

Розроблена система обміну даними між ERP Odoo та платформою Instagram передбачає зручний механізм комунікації через повідомлення. Розглянемо

детальний сценарій такої взаємодії на прикладі типового використання системи. На рисунку 5.9 зображено приклад комунікації з боку клієнта.

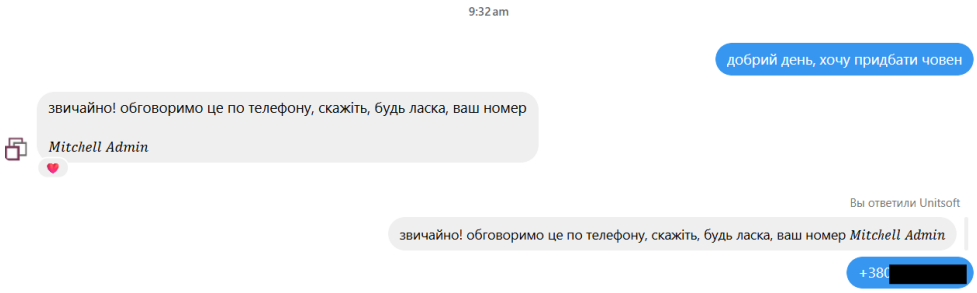

Рисунок 5.9 — Вигляд взаємодії з системої з боку клієнта

При надходженні нового повідомлення від клієнта через Instagram, воно автоматично відображається в інтерфейсі Odoo у відповідному каналі комунікації. Система створює окремий тред для кожного нового діалогу, що дозволяє зберігати історію спілкування та контекст взаємодії з клієнтом. Користувач системи має можливість переглядати повідомлення в режимі реального часу та надавати відповіді безпосередньо через інтерфейс Odoo. Приклад комунікації з боку користувача системи зображено на рисунку 5.10.

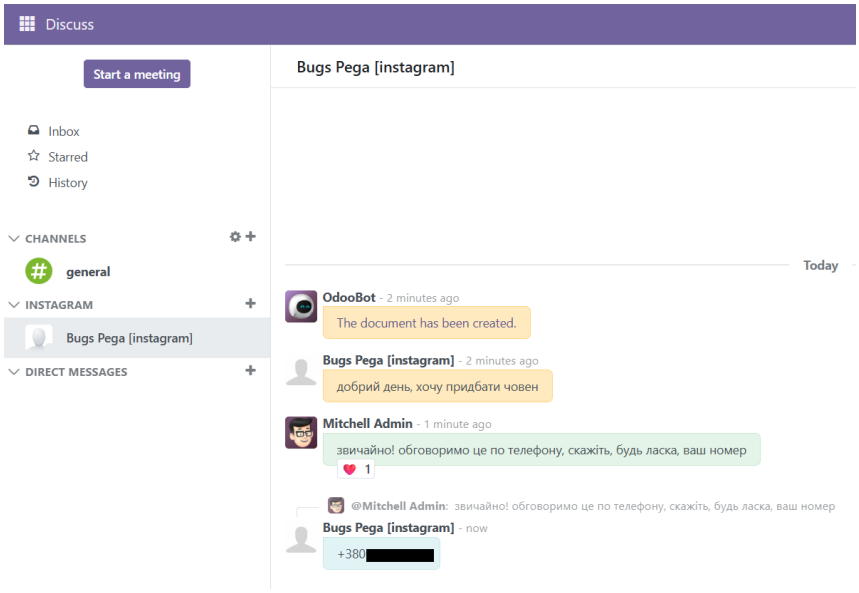

Рисунок 5.10 — Інтерфейс обміну повідомленнями в системі Odoo

Важливою особливістю реалізованого рішення є збереження цілісності комунікації незалежно від платформи відправлення повідомлень. Усі повідомлення синхронізуються між Instagram та Odoo, забезпечуючи безперервність діалогу та актуальність інформації в обох системах. Це дозволяє операторам ефективно керувати комунікацією з клієнтами, не переключаючись між різними інтерфейсами.

Також для кожного нового клієнта в системі, який ініціював діалог з відповідним акаунтом Instagram, створюється картка клієнта та лід, що в подільшому будуть заповнені необхідною інформацією в процесі взаємодії для забезпечення належного клієнтського досвіду. Приклад створених контактів в системі зображено на рисунку 5.11.

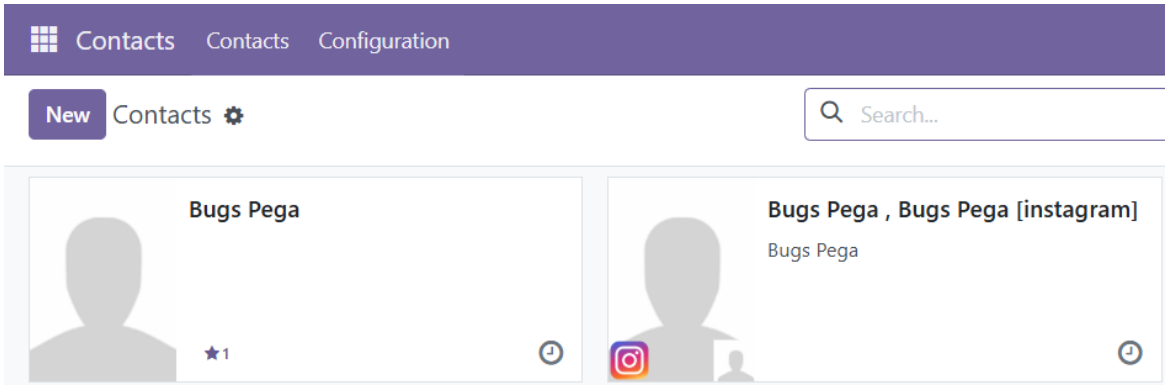

Рисунок 5.11 — Відображення карток клієнта в системі Odoo

Інтерфейс системи також надає додаткові можливості для категоризації та пріоритезації повідомлень, що особливо корисно при роботі з великою кількістю клієнтських звернень. Оператори можуть позначати статуси повідомлень, призначати відповідальних співробітників та встановлювати нагадування для подальшого опрацювання запитів.

ВИСНОВКИ

В результаті виконання дипломної роботи було досягнуто поставленої мети щодо створення системи обміну даними між ERP Odoo та платформою Instagram. На основі проведеного аналізу існуючих підходів, методів та алгоритмів розв'язання поставленої задачі було обґрунтовано використання REST API для взаємодії з Instagram, асинхронної обробки даних та механізму webhook для оновлення інформації в реальному часі.

Розроблена програмна система забезпечує двосторонній обмін даними між ERP Odoo та соціальною мережею Instagram. Реалізовано функціонал автоматичної синхронізації публікацій, коментарів та повідомлень, що дозволяє ефективно керувати маркетинговою діяльністю підприємства в соціальних мережах безпосередньо з інтерфейсу ERP системи. Впроваджено механізми автентифікації та авторизації, що забезпечують безпечну взаємодію між системами.

Проведене всебічне тестування розробленого програмного забезпечення підтвердило його надійність та відповідність функціональним вимогам. Перевірка охопила всі ключові компоненти системи, включаючи процеси автентифікації, синхронізації даних, обробки медіаконтенту та взаємодії через webhook. Результати тестування довели коректність роботи системи та її готовність до практичного використання.

Подальший розвиток системи може бути спрямований на розширення функціональності для впровадження механізмів аналітики та автоматизації маркетингових кампаній, а також оптимізацію продуктивності при обробці великих обсягів даних. Розроблена архітектура системи забезпечує можливість такого масштабування та вдосконалення функціоналу відповідно до потреб користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. The potential of social media in conjunction with an ERP system - System CRM and ERP Firmao. *System CRM and ERP Firmao*. URL: https://firmao.net/blog_net/wms/the-potential-of-social-media-in-conjunction-with-an-erp-system (дата звернення: 14.04.2025).

2. Instagram for business odoo CRM integration - quick connect - zapier. Zapier. URL: https://zapier.com/apps/instagram-for-business/integrations/odoo (дата звернення: 16.04.2025).

3. Instagram odoo ERP self hosted integration | appy pie automate. Appy Pie Automate. URL: https://www.appypieautomate.ai/integrate/apps/instagram/integrations/odoo-erp-self-hosted (дата звернення: 13.05.2025).

4. Why your business needs ERP to leverage social media for growth. BusinessTech. URL: https://www.hashmicro.com/blog/erp-social-media-growth/ (дата звернення: 22.04.2025).

5. Deskera. How is social media influencing ERP?. Deskera Blog. URL: https://www.deskera.com/blog/how-is-social-media-influencing-erp/ (дата звернення: 02.05.2025).

6. Make. Odoo and Instagram for Business (Facebook login) Integration. Make. URL: https://www.make.com/en/integrations/odoo/instagram-business (дата звернення: 03.05.2025).

7. ERP integration: definition, types, advantages, and best practices. AI-Driven Legacy Modernization Platform | OpenLegacy. URL: https://www.openlegacy.com/blog/erp-integration (дата звернення: 18.04.2025).

8. Integrating ERP with social media | metasfresh ERP. metasfresh ERP. URL: https://metasfresh.com/en/2018/02/19/integrating-erp-with-social-media/ (дата звернення: 18.04.2025).

9. Marinesko P. O. Optimization of the workflow in the company Odoo using the CRM-system module. *Scientific notes of the state university of telecommunications*.

2021. Т. 1, № 1. URL: https://doi.org/10.31673/2518-7678.2021.014350 (дата звернення: 23.04.2025).

10. Автоматизація за допомогою ERP Odoo: виклики та можливості. *DOU.ua*. URL: https://dou.ua/forums/topic/42444/ (дата звернення 26.04.2025).

11. 10 модулів Odoo ERP, які повинна використовувати кожна компанія. *Simple ERP*. URL: https://simpleerp.com.ua/top-moduliv-odoo-erp (дата звернення: 06.05.2025).

12. Основні модулі Odoo та їх функціональність. *ERP Consulting.* URL: https://erp-consulting.pro/osnovni-moduli-odoo-ta-ih-funkcionalnist/ (дата звернення: 04.05.2025).

13. Perea D., Bonsón E., Bednárová M. Citizen reactions to municipalities' Instagram communication. *Government information quarterly*. 2021. С. 101579. URL: https://doi.org/10.1016/j.giq.2021.101579 (дата звернення: 05.05.2025).

14. Himawan A., Priadana A., Murdiyanto A. Implementation of web scraping to build a web-based instagram account data downloader application. *IJID (international journal on informatics for development)*. 2020. Т. 9, № 2. С. 59–65. URL: https://doi.org/10.14421/ijid.2020.09201 (дата звернення: 19.04.2025).

15. Consumer engagement through corporate social responsibility communication on social media: Evidence from Facebook and Instagram Bank Accounts / L. S. Macca et al. *Journal of business research*. 2024. Vol. 172. P. 114433. URL: https://doi.org/10.1016/j.jbusres.2023.114433 (дата звернення: 03.05.2025).

16. Ayora V., Horita F., Kamienski C. Profiling online social network platforms: twitter vs. instagram. *Hawaii international conference on system sciences*. 2021. URL: https://doi.org/10.24251/hicss.2021.341 (дата звернення: 09.05.2025).

17. Cuevas-Molano E., Sánchez-Cid M., Gordo-Molina V. Brand strategy and content management on Instagram: scheduling and message length as factors to improve engagement. *Communication & society*. 2022. P. 71–87. URL: https://doi.org/10.15581/003.35.2.71-87 (дата звернення: 15.05.2025).

18. Shankararaman V., Lum E. K. Integration of social media technologies with ERP:a prototype implementation. 2013. URL: https://ink.library.smu.edu.sg/sis_research/2043/. (дата звернення: 13.05.2025).

19. Social Marketing – Odoo 17.0 documentation. *Open Source ERP and CRM | Odoo*. URL: https://www.odoo.com/documentation/17.0/applications/marketing/social_marketing.html (дата звернення: 14.05.2025).

20. Omni_channel_communication_odoo | odoo apps store. *Odoo Apps Store*. URL: https://apps.odoo.com/apps/modules/17.0/omni_channel_communication_odoo (дата звернення: 15.05.2025).

21. Herman H. Zapier social media automation | Zapier. *Zapier: Automate AI Workflows, Agents, and Apps*. URL: https://zapier.com/blog/social-media-automation/ (дата звернення: 17.05.2025).

22. Microsoft. Using Git source control in VS Code. *Visual Studio Code - Code Editing. Redefined*. URL: https://code.visualstudio.com/docs/sourcecontrol/overview (дата звернення: 19.05.2025).

23. Microsoft. Terminal basics. *Visual Studio Code - Code Editing. Redefined*. URL: https://code.visualstudio.com/docs/editor/integrated-terminal (дата звернення: 07.05.2025).

24. Microsoft. IntelliSense. *Visual Studio Code - Code Editing. Redefined*. URL: https://code.visualstudio.com/docs/editor/intellisense (дата звернення: 07.05.2025).

25. Microsoft. Personalize VS code. *Visual Studio Code - Code Editing. Redefined*. URL: https://code.visualstudio.com/docs/getstarted/personalize-vscode (дата звернення: 07.05.2025).

26. Microsoft. VS code remote development. *Visual Studio Code - Code Editing. Redefined*. URL: https://code.visualstudio.com/docs/remote/remote-overview (дата звернення: 07.05.2025).

27. Reis J. Exploring applications and practical examples by streamlining material requirements planning (MRP) with python. *Logistics*. 2023. Vol. 7, no. 4. P. 91. URL: https://doi.org/10.3390/logistics7040091 (дата звернення: 10.05.2025).

28. A A. Y. Head first python. *International journal of engineering in computer science*. 2021. Vol. 3, no. 1. P. 12–15. URL: https://doi.org/10.33545/26633582.2021.v3.i1a.41 (дата звернення: 11.05.2025).

29. 27 best javascript frameworks for 2025 | lambdatest. *LambdaTest*. URL: https://www.lambdatest.com/blog/best-javascript-frameworks/ (дата звернення: 12.05.2025).

30. Rise of Web Programming using Python and JavaScript. *GO-Globe*. URL: https://www.go-globe.com/web-programming-using-python-and-javascript/ (дата звернення: 12.05.2025).

31. Gómez-Llanez C. Y., Diaz-Leal N. R., Angarita-Sanguino C. R. A comparative analysis of the ERP tools, Odoo and Openbravo, for business management. *Aibi revista de investigación, administración e ingeniería*. 2020. P. 145–153. URL: https://doi.org/10.15649/2346030x.789 (дата звернення: 29.04.2025).

32. Framework Overview – Odoo 18.0 documentation. *Open Source ERP and CRM | Odoo*. URL: https://www.odoo.com/documentation/18.0/developer/reference/frontend/framework_overview.html (дата звернення: 14.05.2025).

33. Automation rules – Odoo 18.0 documentation. *Open Source ERP and CRM | Odoo*. URL: https://www.odoo.com/documentation/18.0/applications/studio/automated_actions.html (дата звернення: 14.05.2025).

34. Access rights – Odoo 18.0 documentation. *Open Source ERP and CRM | Odoo*. URL: https://www.odoo.com/documentation/18.0/applications/general/users/access_rights.html (дата звернення: 14.05.2025).

35. Deploying & scaling your odoo server. *SlideShare*. URL: https://www.slideshare.net/openobject/deploying-scaling-your-odoo-server (дата звернення: 15.05.2025).

36. Requests: HTTP for Humans™ – Requests 2.32.3 documentation. *Requests: HTTP for Humans™ – Requests 2.32.3 documentation*. URL: https://requests.readthedocs.io/en/latest/ (дата звернення: 16.05.2025).

37. PyAV Documentation – PyAV 9.0.2 documentation. *PyAV Documentation – PyAV 9.0.2 documentation*. URL: https://pyav.org/docs/stable/ (дата звернення: 16.05.2025).

38. Overview. *Pillow (PIL Fork)*. URL: https://pillow.readthedocs.io/en/stable/handbook/overview.html (дата звернення: 16.05.2025).

39. 1. What Is PostgreSQL?. *PostgreSQL Documentation*. URL: https://www.postgresql.org/docs/current/intro-whatis.html (дата звернення: 17.05.2025).

40. Chapter 26. High Availability, Load Balancing, and Replication. *PostgreSQL Documentation*. URL: https://www.postgresql.org/docs/current/high-availability.html (дата звернення: 17.05.2025).

41. Chapter 25. Backup and Restore. *PostgreSQL Documentation*. URL: https://www.postgresql.org/docs/current/backup.html (дата звернення: 18.05.2025).

42. Chapter 20. Client Authentication. *PostgreSQL Documentation*. URL: https://www.postgresql.org/docs/current/client-authentication.html (дата звернення: 18.05.2025).

43. Webhooks - Instagram-Plattform - Dokumentation - Meta for Developers. *Social Technologies | Meta for Developers*. URL: https://developers.facebook.com/docs/instagram-platform/instagram-api-with-instagram-login/webhooks (дата звернення: 19.05.2025).

44. How does ngrok work? | ngrok documentation. *ngrok | API Gateway, Kubernetes Ingress, Webhook Gateway*. URL: https://ngrok.com/docs/how-ngrok-works/ (дата звернення: 20.05.2025).

ДОДАТОК А

Система обміну даними між ERP Odoo та платформою Instagram

Текст програми

УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ІАТЕ_ЦТЕ_ТР-11

Аркушів 6

Київ – 2025

Програмні засоби:

— мова програмування — Python;

— виконання HTTP-запитів — Requests;

instagram_account.py — модель зберігання даних облікових записів

```
class InstagramAccount(models.Model):
    _name = 'instagram.account'
    _description = 'Instagram Account'

    name = fields.Char('Account Name', required=True)
    instagram_user_id = fields.Char('Instagram User ID')
    instagram_username = fields.Char('Instagram Username')
    account_type = fields.Char('Account Type')
    profile_picture_url = fields.Char('Profile Picture URL')
    access_token = fields.Char('Access Token')
    page_id = fields.Char('Facebook Page ID')
    app_id = fields.Char('Instagram App ID', required=True)
    app_secret = fields.Char('Instagram App Secret', required=True, password=True)
    token_last_refreshed = fields.Datetime("Token Last Refreshed")
    media_ids = fields.One2many('instagram.media.fetched', 'account_id', string='Media Posts')
    story_ids = fields.One2many('instagram.story', 'account_id', string='Stories')

    def action_business_login(self):
        """Returns a URL that starts the OAuth process for business login."""
        base_url = self.env['ir.config_parameter'].sudo().get_param('web.base.url')
        redirect_url = f"{base_url}/us_instagram/auth/start?account_id={self.id}"
        return {
            'type': 'ir.actions.act_url',
            'url': redirect_url,
            'target': 'new',
        }

    def action_refresh_access_token(self):
        """
        Refresh the current long-lived access token.
        """
        self.ensure_one()
        if not self.access_token:
            raise ValidationError("No access token available to refresh.")

        api_service = self.env['instagram.api.service']
        try:
            result = api_service.refresh_access_token(self.access_token)

            self.write({
                'access_token': result['new_token'],
                'token_last_refreshed': fields.Datetime.now(),
            })

            _logger.info(
                "Access token for account %s refreshed successfully. New token expires in %s seconds.",
                self.name, result['expires_in']
            )
            return True
        except Exception as e:
            _logger.error("Error refreshing token for account %s: %s", self.name, str(e))
            raise ValidationError(f"Error refreshing the long-lived access token: {str(e)}")

    @api.model
```

```
def cron_refresh_all_tokens(self):
    """
    Cron job method to refresh access tokens for all Instagram accounts automatically.
    """
    accounts = self.search([('access_token', '!=', False)])
    now = datetime.now(timezone.utc)
    for account in accounts:
        if account.token_last_refreshed:
            last_refreshed = account.token_last_refreshed.replace(tzinfo=timezone.utc)
            if (now - last_refreshed).total_seconds() < 86400:
                _logger.info("Skipping refresh for account %s; refreshed less than 24 hours ago.", account.name)
                continue
        try:
            account.action_refresh_access_token()
        except Exception as e:
            _logger.error("Error refreshing token for account %s: %s", account.name, e)
    return True

def action_get_media(self):
    """
    Fetch media posts from Instagram for this business account and archive
    any previously fetched media that is no longer returned by the API.
    """
    self.ensure_one()
    if not self.access_token:
        raise ValidationError("Access token is required to fetch media.")

    api_service = self.env['instagram.api.service']
    media_model = self.env['instagram.media.fetched']
    fetched_media_ids = set()

    try:
        media_fields = ['id', 'caption', 'media_type', 'media_url', 'permalink', 'timestamp',
                'like_count', 'comments_count']

        media_items = api_service.get_user_media(self.access_token, media_fields)
        _logger.info(f"Fetched {len(media_items)} media items for account {self.name}.")

        for media_item in media_items:
            media_id = media_item.get('id')

            fetched_media_ids.add(media_id)

            posted_at = datetime.fromisoformat(
                media_item.get('timestamp').replace('+0000', '+00:00')
            ).replace(tzinfo=None)

            media_values = {
                'account_id': self.id,
                'instagram_media_id': media_id,
                'caption': media_item.get('caption'),
                'media_type': media_item.get('media_type'),
                'media_url': media_item.get('media_url'),
                'permalink': media_item.get('permalink'),
                'posted_at': posted_at,
                'like_count': media_item.get('like_count', 0),
                'comments_count': media_item.get('comments_count', 0),
                'children_count': 0,
                'active': True,
            }

            if media_item.get('media_type') == 'CAROUSEL_ALBUM':
                try:
```

```
                    children_list = api_service.get_media_children(media_id, self.access_token)
                    media_values['children_count'] = len(children_list)
                except Exception as ce:
                    _logger.error(f"Error fetching children for carousel {media_id}: {ce}")
                    # Continue processing the main media item even if children fetch fails

            existing_media = media_model.search([
                ('account_id', '=', self.id),
                ('instagram_media_id', '=', media_id)
            ], limit=1)

            if existing_media:
                existing_media.with_context(from_instagram_sync=True).write(media_values)
            else:
                media_model.create(media_values)

        # Archive media not found in the latest fetch
        media_to_archive = media_model.search([
            ('account_id', '=', self.id),
            ('instagram_media_id', 'not in', list(fetched_media_ids)),
            ('active', '=', True)
        ])

        if media_to_archive:
            _logger.info(f"Archiving {len(media_to_archive)} media items for account {self.name} that are no longer available via API.")
            media_to_archive.write({'active': False})

        return True

    except Exception as e:
        _logger.error("Error fetching media for account %s: %s", self.name, str(e))
        raise ValidationError(f"Error fetching media from Instagram: {str(e)}")

def action_get_stories(self):
    """
    Fetch active stories from Instagram for this business account.
    """
    self.ensure_one()
    if not self.access_token:
        raise ValidationError("Access token is required to fetch stories.")

    api_service = self.env['instagram.api.service']
    story_model = self.env['instagram.story']

    try:
        story_ids_data = api_service.get_user_stories(self.access_token)
        _logger.info(f"Found {len(story_ids_data)} active stories for account {self.name}.")

        fetched_story_ids = set()
        existing_story_ids = set(story_model.search([
            ('account_id', '=', self.id),
            ('instagram_media_id', '!=', False)
        ]).mapped('instagram_media_id'))

        for story_ref in story_ids_data:
            story_id = story_ref.get('id')

            fetched_story_ids.add(story_id)

            if story_id in existing_story_ids:
                _logger.info(f"Story {story_id} already exists for account {self.name}. Skipping details fetch.")
                continue
```

```
                story_details = api_service.get_media_details(story_id, self.access_token)

                posted_at = datetime.fromisoformat(
                    story_details.get('timestamp').replace('+0000', '+00:00')
                ).replace(tzinfo=None)

                story_values = {
                    'account_id': self.id,
                    'instagram_media_id': story_id,
                    'media_type': story_details.get('media_type'),
                    'media_url': story_details.get('media_url'),
                    'thumbnail_url': story_details.get('thumbnail_url'),
                    'permalink': story_details.get('permalink'),
                    'posted_at': posted_at,
                }

                story_model.create(story_values)
                _logger.info(f"Created new story record {story_id} for account {self.name}.")

            expired_stories = story_model.search([
                ('account_id', '=', self.id),
                ('instagram_media_id', 'not in', list(fetched_story_ids))
            ])
            if expired_stories:
                _logger.info(f"Archiving {len(expired_stories)} expired stories for account {self.name}.")
                expired_stories.action_archive()

            return {
                'type': 'ir.actions.client',
                'tag': 'display_notification',
                'params': {
                    'title': _('Success'),
                    'message': _('Successfully fetched Instagram stories.'),
                    'sticky': False,
                }
            }

        except Exception as e:
            _logger.error("Error fetching stories for account %s: %s", self.name, str(e))
            raise ValidationError(f"Error fetching stories from Instagram: {str(e)}")
```

instagram_media_fetched.py — модель зберігання публікацій отриманих з Instagram

```
class InstagramMediaFetched(models.Model):
    _name = 'instagram.media.fetched'
    _description = 'Instagram Media fetched from Instagram (immutable once published)'
    _order = 'posted_at desc'

    account_id = fields.Many2one('instagram.account', string='Instagram Account', ondelete='cascade')
    instagram_media_id = fields.Char('Instagram Media ID')
    caption = fields.Text('Caption')
    media_type = fields.Selection([
        ('IMAGE', 'Image'),
        ('VIDEO', 'Video'),
        ('CAROUSEL_ALBUM', 'Carousel Album'),
    ], string='Media Type', required=True)
    media_url = fields.Char('Media URL')
    permalink = fields.Char('Permalink')
    posted_at = fields.Datetime('Posted At')
    formatted_posted_at = fields.Char('Formatted Posted At', compute='_compute_formatted_posted_at')
    comments_count = fields.Integer('Comments Count', default=0)
```

```
like_count = fields.Integer('Likes Count', default=0)
children_count = fields.Integer('Children Count', default=0, help="Number of media items in a carousel album.")
active = fields.Boolean('Active', default=True, index=True)
instagram_username = fields.Char(related='account_id.instagram_username', string='Instagram Username', store=True)

@api.depends('posted_at')
def _compute_formatted_posted_at(self):
    for media in self:
        media.formatted_posted_at = self._format_posted_at_date(media.posted_at) if media.posted_at else False

@api.model
def _format_posted_at_date(self, posted_at_date):
    """ Formats to '5 minutes ago' instead of date if not older than 12 hours. """
    if not posted_at_date:
        return ''
    if isinstance(posted_at_date, str):
         posted_at_date = fields.Datetime.from_string(posted_at_date)

    if (datetime.now() - posted_at_date) < timedelta(hours=12):
        return _format_time_ago(self.env, (datetime.now() - posted_at_date), add_direction=True)
    else:
        return format_date(self.env, posted_at_date)

@api.model
def action_sync_all_media(self, *args):
    """Sync all Instagram media by calling action_get_media for each Instagram account."""
    accounts = self.env['instagram.account'].search([])
    for account in accounts:
        try:
            account.action_get_media()
        except Exception as e:
            _logger.error("Error syncing media for account %s: %s", account.name, e)
    return True

@api.model
def fetch_comments(self):
    self.ensure_one()
    api_service = self.env['instagram.api.service']
    try:
        # Pass the pagination parameters received from the controller
        result = api_service.get_media_comments(
            self.instagram_media_id,
            self.account_id.access_token
        )
        return result
    except UserError as e:
         _logger.error("Failed to fetch comments for media %s: %s", self.instagram_media_id, e)
         raise
    except Exception as e:
        _logger.error("An unexpected error occurred fetching comments for media %s: %s", self.instagram_media_id, e)
        raise UserError(_("Could not fetch comments. Please try again later."))

def instagram_comment_add(self, message: str, comment_id: str = None):
    """Adds a comment or reply to this Instagram media item."""
    self.ensure_one()
    if not self.account_id or not self.account_id.access_token:
        raise UserError(_("Instagram account is not configured or lacks an access token."))
    if not message:
        raise UserError(_("Comment message cannot be empty."))

    api_service = self.env['instagram.api.service']
    try:
        formatted_comment = api_service._instagram_comment_add(
```

```
                media_id=self.instagram_media_id,
                access_token=self.account_id.access_token,
                message=message,
                comment_id=comment_id
            )
            if formatted_comment.get('error'):
                raise UserError(_("Failed to add comment: %s") % formatted_comment.get('error'))

            return formatted_comment
        except UserError as e:
            _logger.error("Failed to add Instagram comment for media %s: %s", self.instagram_media_id, e)
            raise
        except Exception as e:
            _logger.error("Unexpected error adding Instagram comment for media %s: %s", self.instagram_media_id, e)
            raise UserError(_("An unexpected error occurred while adding the comment."))

    def instagram_comment_delete(self, comment_id: str):
        """Deletes a specific comment by its ID."""
        self.ensure_one()
        if not self.account_id or not self.account_id.access_token:
            raise UserError(_("Instagram account is not configured or lacks an access token."))
        if not comment_id:
            raise UserError(_("Comment ID is required for deletion."))

        api_service = self.env['instagram.api.service']
        try:
            success = api_service.delete_comment(
                comment_id=comment_id,
                access_token=self.account_id.access_token
            )
            if not success:
                raise UserError(_("Failed to delete the comment on Instagram. It might have already been deleted or there was an API issue."))

            return {'success': True}
        except UserError as e:
            _logger.error("Failed to delete Instagram comment %s for media %s: %s", comment_id, self.instagram_media_id, e)
            raise
        except Exception as e:
                _logger.error("Unexpected error deleting Instagram comment %s for media %s: %s", comment_id, self.instagram_media_id, e)
            raise UserError(_("An unexpected error occurred while deleting the comment."))
```