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 ЦТЕ
_____ Наталія АУШЕВА

“___”_____202__р.

Дипломна робота
на здобуття ступеня бакалавра

За освітньою програмою “Цифрові технології в енергетиці”

Спеціальності 122 “Комп’ютерні науки”

на тему: Обмін інформацією системи Odoo з платформою OLX

Виконала:
студентка IV курсу, групи ТР-23
БУХАЛО Євгенія Юріївна _____
(прізвище, ім’я, по батькові) (підпис)

Керівник: *професор каф. цифрових технологій в енергетиці*
д.т.н., проф. ШУШУРА Олексій Миколайович _____
(посада, науковий ступінь, вчене звання, прізвище, ім’я, по батькові) (підпис)

Рецензент: *зав. каф. теплової та альтернативної енергетики*
к.т.н., доц., ПЕШКО Віталій Анатолійович _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище, ім’я, по батькові) (підпис)

Н. контроль: ас. каф. МЕЛЬНИЧЕНКО Артем Васильович _____
(посада, прізвище, ім’я, по батькові) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів відповідних посилань.
Студентка _____
(підпис)

Київ – 2026

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)
“___” _____2026 р.

ЗАВДАННЯ

на дипломну роботу студенту

_____Бухало Євгенії Юріївні_____
(прізвище, ім’я, по батькові)

1. Тема роботи “Обмін інформацією системи ODOO з платформою OLX”
Науковий керівник Шушура Олексій Миколайович, д.т.н., професор
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від “_28_” __травня___ 2026 року № 1992_

2. Термін подання студентом роботи 08.06.2026

3. Вихідні дані до роботи мова програмування Python, ERP-система Odoo, API платформи OLX, СКБД PostgreSQL, середовище розробки Visual Studio Code.

4. Перелік питань, які потрібно розробити _____

1) провести аналіз рішень інтеграції ERP-систем з торговими платформами;
2) дослідити архітектуру Odoo та можливості інтеграції з маркетплейсами;
3) проаналізувати API платформи OLX та обґрунтувати вибір інструментів реалізації;
4) розробити архітектуру інтеграційного модуля та структуру зберігання даних;
5) реалізувати програмний модуль для обміну даними між Odoo та OLX;

6) описати процес налаштування модуля і сценарії використання користувачем.

5. Орієнтований перелік ілюстративного матеріалу діаграми, що демонструють роботу алгоритмів інтеграційного модуля, архітектуру системи, бази даних та структури обміну даними, взаємодію користувачів із системою, класи програмного забезпечення; приклади роботи з програмним продуктом.

6. Дата видачі завдання 13.09.2025

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Вибір теми роботи	13.09.25	
2.	Аналіз методів та засобів розв'язання задачі	17.03.26-21.03.26	
3.	Розробка архітектури та загальної структури системи	21.03.26-31.03.26	
4.	Розробка окремих підсистем	01.04.26-14.04.26	
5.	Програмна реалізація системи	14.04.26-18.05.26	
6.	Оформлення пояснювальної записки	18.05.26-25.05.26	
7.	Захист програмного забезпечення	12.05.26	
8.	Передзахист	03.06.2026	
9.	Захист	16.06.26-21.06.26	

Студент _____ Євгенія БУХАЛО
(підпис) (прізвище та ініціали)

Керівник _____ Олексій ШУШУРА
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська кваліфікаційна робота виконана на 61 сторінках, містить 14 ілюстрацій, 2 додатки, 24 джерел в переліку посилань.

Мета роботи – розробка програмного модуля інтеграції ERP-системи Odoo з маркетплейсом OLX із використанням REST API.

Методи та засоби: архітектура REST API, протокол авторизації OAuth 2.0, формати обміну даними JSON, технології синхронізації даних, фреймворк Odoo, Python, бібліотеки requests та odoorpc.

Результат – розроблений програмний модуль, який забезпечує автоматизовану двосторонню інтеграцію між ERP-системою Odoo та маркетплейсом OLX, дозволяє створювати, редагувати та синхронізувати оголошення про товари, а також підвищує ефективність процесів онлайн-торгівлі.

Ключові слова: ODOO, OLX, REST API, ERP, ІНТЕГРАЦІЯ СИСТЕМ, OAUTH 2.0, ЕЛЕКТРОННА КОМЕРЦІЯ, АВТОМАТИЗАЦІЯ.

ANOTATION

The bachelor's qualification thesis is performed on 61 pages, contains 14 illustrations, 2 appendices, and 24 sources in the list of references.

The aim of the work is the development of a software module for integrating the Odoo ERP system with the OLX marketplace using REST API.

Methods and tools: REST API architecture, OAuth 2.0 authorization protocol, JSON data exchange format, data synchronization technologies, Odoo framework, Python programming language, requests and odoorpc libraries.

The result of the work is a developed software module that provides automated bidirectional integration between the Odoo ERP system and the OLX marketplace. It enables the creation, editing, and synchronization of product listings, as well as increases the efficiency of online trading processes.

Keywords: ODOO, OLX, REST API, ERP, SYSTEM INTEGRATION, OAUTH 2.0, E-COMMERCE, AUTOMATION.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

ERP – Enterprise Resource Planning, система планування ресурсів підприємства, що забезпечує автоматизацію бізнес-процесів організації.

API – Application Programming Interface, програмний інтерфейс застосунків, який використовується для взаємодії між програмними системами.

REST – Representational State Transfer, архітектурний стиль побудови вебсервісів, що використовує HTTP-протокол для обміну даними.

JSON – JavaScript Object Notation, формат обміну даними у вигляді тексту, який легко читається та обробляється програмними системами.

OAuth 2.0 – протокол авторизації, що використовується для надання обмеженого доступу до ресурсів без передачі пароля користувача.

Odoo – відкрита ERP-система для управління бізнес-процесами підприємства.

OLX – онлайн-платформа оголошень для купівлі та продажу товарів і послуг.

ВСТУП

Інтеграція ERP-систем із торговими платформами є одним із ключових напрямів автоматизації бізнес-процесів у сфері електронної комерції. Стрімкий розвиток онлайн-торгівлі та зростання кількості цифрових каналів продажу зумовлюють необхідність ефективного управління товарами, оголошеннями та супутньою інформацією. За таких умов особливого значення набувають програмні рішення, які забезпечують автоматизований обмін даними між внутрішніми інформаційними системами підприємств і зовнішніми торговими платформами.

Однією з найпопулярніших ERP-систем з відкритим вихідним кодом є Odoo, яка надає широкий набір інструментів для автоматизації управління підприємством. Водночас платформа OLX є одним із найбільших онлайн-майданчиків для розміщення оголошень та продажу товарів в Україні. Інтеграція між цими системами дозволяє автоматизувати процеси публікації та оновлення оголошень, зменшити кількість помилок, пов’язаних із ручним введенням даних, а також підвищити оперативність керування інформацією про товари.

Попри наявність окремих інструментів для роботи з торговими платформами, більшість існуючих рішень не забезпечують повноцінної інтеграції Odoo з OLX або потребують значних витрат часу на налаштування та супровід. Це обумовлює необхідність розробки спеціалізованого інтеграційного модуля, який враховуватиме особливості архітектури Odoo та механізми взаємодії, що надаються API платформи OLX.

Мета дипломної роботи полягає у розробці модуля інтеграції між ERP-системою Odoo та платформою OLX, який забезпечуватиме автоматизований обмін інформацією між системами, створення та оновлення оголошень, а також централізоване керування даними через єдиний інтерфейс Odoo.

Для досягнення поставленої мети необхідно виконати такі завдання:

– провести аналіз сучасних підходів до інтеграції ERP-систем із торговими платформами;

– дослідити функціональні можливості платформи OLX та особливості її API;

– визначити вимоги до інтеграційного рішення;

– спроєктувати архітектуру програмного модуля для взаємодії Odoo та OLX;

– реалізувати механізм API-взаємодії та авторизації користувачів;

– розробити засоби синхронізації даних між системами;

– реалізувати механізми обробки помилок та ведення журналу подій;

– провести тестування розробленого програмного забезпечення та оцінити результати його роботи.

Об’єктом дослідження є процеси обміну інформацією між ERP-системами та платформами електронної комерції.

Предметом дослідження є методи, засоби та технології інтеграції ERP-системи Odoo з платформою OLX із використанням API.

Практичне значення роботи полягає у створенні програмного модуля, який автоматизує взаємодію між Odoo та OLX, забезпечує централізоване керування оголошеннями та підвищує ефективність виконання бізнес-процесів у сфері онлайн-торгівлі.

Дипломна робота містить рисунки, додатки та список використаних джерел. У першому розділі розглянуто особливості ERP-системи Odoo та платформи OLX, а також обґрунтовано задачу розробки системи обміну інформацією. Другий розділ присвячено аналізу існуючих підходів до інтеграції інформаційних систем і проєктуванню інтеграційного рішення. У третьому розділі обґрунтовано вибір засобів реалізації, розглянуто архітектуру Odoo, можливості API платформи OLX та використані програмні засоби. Четвертий розділ містить опис програмної реалізації інтеграційного модуля, його архітектури, структури даних, механізмів взаємодії з API, авторизації та синхронізації даних. У п’ятому розділі наведено опис роботи користувача із системою, системні вимоги та огляд розробленого програмного забезпечення.

1 ТЕОРЕТИЧНІ ЗАСАДИ ІНТЕГРАЦІЇ ІНФОРМАЦІЙНИХ СИСТЕМ У ЕЛЕКТРОННІЙ КОМЕРЦІЇ

Динамічний розвиток електронної комерції вимагає від бізнесу максимальної автоматизації та швидкої синхронізації даних. Ефективним рішенням у цій сфері є інтеграція внутрішніх управляючих систем підприємства із зовнішніми торговельними майданчиками, що дозволяє уникнути ручної рутини та мінімізувати помилки.

У даному розділі досліджено теоретичні підходи до побудови інтеграційних рішень, методи обміну даними та архітектурні принципи взаємодії інформаційних систем. Особливу увагу приділено теоретичному обґрунтуванню синхронізації процесів між ERP-системою Odoo та платформою оголошень OLX, що є основою для подальшої практичної розробки проєкту.

1.1 ERP-система ODOO як засіб автоматизації бізнес-процесів

Швидке масштабування бізнесу, збільшення обсягів продажів та розширення каналів комунікації з клієнтами вимагають від підприємств відмови від рутинних ручних операцій на користь комплексних систем автоматизації. Центральне місце у вирішенні цих завдань посідають системи планування ресурсів підприємства (ERP), які дозволяють об'єднати всі ключові підрозділи компанії в єдине інформаційне середовище [1]. Впровадження ERP-систем є фундаментом цифрової трансформації бізнесу, забезпечуючи оптимізацію використання матеріальних, фінансових та людських ресурсів [2]. Завдяки централізованому збереженню даних керівництво та менеджери отримують можливість приймати обґрунтовані управлінські рішення в

режимі реального часу, знижуючи вплив «людського фактора» та мінімізуючи ризики виникнення помилок у рутинній діяльності [3].

Серед розмаїття сучасного програмного забезпечення для автоматизації бізнесу особливе місце посідає ERP-система Odoo. Її популярність зумовлена насамперед модульною архітектурою та відкритим вихідним кодом (open-source), що дозволяє гнучко адаптувати систему під специфіку конкретного підприємства без надмірних капіталовкладень. На мою думку, саме компонентний підхід Odoo забезпечує бізнесу масштабованість: підприємство може почати з автоматизації базового складського обліку, а згодом нарощувати потужності, підключаючи модулі CRM, бухгалтерського обліку чи інтернет-торгівлі, при цьому всі модулі функціонують на базі єдиного ядра та спільної бази даних.

Для збереження та маніпулювання даними система використовує реляційну систему керування базами даних PostgreSQL [4]. Важливою архітектурною перевагою Odoo є наявність потужного шару ORM (Object-Relational Mapping), який виступає посередником між об'єктами Python та таблицями бази даних SQL [5]. Це суттєво спрощує розробку, оскільки програміст оперує високорівневими об'єктами мови Python, а ORM автоматично транслює ці дії в оптимізовані SQL-запити до бази даних [5, 6].

Для розв'язання задачі інтеграції із зовнішніми платформами Odoo надає багатий інструментарій розробника (Developer Documentation) та потужний зовнішній API (External API) [7, 8]. Завдяки вбудованому механізму RPC (Remote Procedure Call), Odoo дозволяє стороннім сервісам безпечно зчитувати, створювати та модифікувати внутрішні дані системи (наприклад, інформацію про товари, залишки на складах чи замовлення від клієнтів) [8].

Таким чином, використання Odoo як базової платформи для автоматизації бізнес-процесів є технологічно обґрунтованим рішенням завдяки її модульній структурі [7], гнучкому ORM-прошарку [9] та розвиненим можливостям інтеграції.

1.2 Платформа OLX як інструмент електронної комерції

OLX є однією з найбільших платформ онлайн-оголошень у світі, що функціонує у понад 30 країнах та охоплює мільйони активних користувачів щомісяця. В Україні OLX займає лідируючі позиції серед майданчиків для купівлі-продажу товарів і послуг у категоріях нерухомості, транспорту, електроніки, одягу та багатьох інших. Платформа орієнтована як на приватних осіб, так і на бізнес-користувачів, пропонуючи останнім спеціальні інструменти для ведення комерційної діяльності.

З точки зору електронної комерції, OLX функціонує за моделлю C2C (consumer-to-consumer) та B2C (business-to-consumer), що відрізняє її від класичних маркетплейсів на кшталт Amazon чи Rozetka. Тобто платформа надає майданчик для розміщення оголошень, але не бере безпосередньої участі у транзакціях між продавцем і покупцем. Проте для бізнес-акаунтів доступний ширший функціонал: пакети оголошень, просування товарів, аналітика переглядів та статистика активності [10].

Для розробників та бізнесу, що прагне автоматизувати роботу з платформою, OLX надає публічний API — програмний інтерфейс, що дозволяє керувати оголошеннями, категоріями, локаціями та атрибутами товарів програмним способом [10]. API побудований за архітектурним стилем REST, що робить його сумісним із переважною більшістю сучасних технологічних стеків та спрощує процес інтеграції. Завдяки цьому підприємства можуть автоматично публікувати, оновлювати та деактивувати оголошення без ручного втручання, що є особливо важливим при великому асортименті товарів.

Серед ключових можливостей OLX API варто виділити: отримання списку категорій та атрибутів, створення і редагування оголошень, завантаження зображень, управління статусом публікацій, а також отримання статистики по оголошеннях [10]. Авторизація у API здійснюється за протоколом OAuth 2.0, що забезпечує безпечний доступ до акаунту без передачі облікових даних третім сторонам [11].

Отже, аналіз платформи OLX доводить, що вона є зрілим та високоефективним інструментом електронної комерції [12], а наявність розвиненого Public API та безпекових стандартів OAuth 2.0 / JWT робить її повністю готовою до інтеграції із сучасними обліковими системами, відкриваючи можливість для створення автоматизованого комплексу обміну інформацією.

1.3 Особливості інтеграції ERP-систем з маркетплейсами

Інтеграція інформаційних систем (ІС) — це процес об'єднання різнорідних інформаційних систем, додатків та баз даних в єдине інформаційне середовище для забезпечення безперешкодного обміну даними та автоматизації наскрізних бізнес-процесів підприємства [12].

З архітектурної точки зору, інтеграція ERP-систем з маркетплейсами реалізується переважно через програмні інтерфейси — API. Найпоширенішим підходом є використання REST (Representational State Transfer) — архітектурного стилю, що описує принципи побудови розподілених мережевих систем [13]. REST-архітектура базується на використанні стандартних HTTP-методів (GET, POST, PUT, DELETE), що робить інтеграційні рішення платформонезалежними та відносно простими у реалізації [14]. Саме REST є основою як зовнішнього API Odoo, так і публічного API платформи OLX, що суттєво спрощує завдання побудови інтеграційного шару між ними.

Окремої уваги заслуговує питання безпеки інтеграційних рішень. Оскільки обмін даними відбувається між системами через мережу, критично важливим є забезпечення надійної автентифікації та авторизації. Стандартом де-факто у сучасних API-інтеграціях є протокол OAuth 2.0, що дозволяє делегувати доступ до ресурсів без передачі облікових даних [11]. Додатково для передачі автентифікаційної інформації між системами широко застосовуються JSON Web Tokens (JWT) — компактний

формат токенів, що містить верифіковані твердження про користувача або систему [10]. Обидва стандарти використовуються в API OLX, що необхідно враховувати при проектуванні інтеграційного модуля.

ERP-система Odoo має всі необхідні внутрішні та зовнішні програмні інтерфейси для автоматизації, а платформа OLX надає структуроване Public API для керування контентом.

2 АНАЛІЗ ІНТЕГРАЦІЙНОГО РІШЕННЯ

Інтеграційні рішення в сучасних інформаційних системах відіграють важливу роль у забезпеченні ефективного обміну даними між різними програмними платформами, особливо у сфері електронної комерції. Взаємодія ERP-систем із зовнішніми сервісами потребує чіткого визначення вимог, аналізу існуючих підходів та формування оптимальної архітектури програмного забезпечення.

Особливу увагу приділено дослідженню процесів обміну інформацією між системами, визначенню ключових сценаріїв взаємодії та принципів побудови інтеграційних модулів. Також розглядаються підходи до проєктування структури даних і організації інформаційних потоків між Odoo та OLX.

2.1 Аналіз бізнес-процесів онлайн-торгівлі

Перш ніж проектувати будь-яке інтеграційне рішення, необхідно детально розуміти бізнес-процеси, які воно має автоматизувати. Аналіз процесів онлайн-торгівлі дозволяє визначити, які саме дані потребують синхронізації між системами, з якою частотою та в якому напрямку відбувається їх обмін.

Сучасна онлайн-торгівля на базі маркетплейсів охоплює низку взаємопов’язаних бізнес-процесів, які можна розподілити на три ключові блоки [15]: управління контентом та каталогом товарів, облік складських залишків (інвентаризація) та обробка й виконання замовлень (Fulfillment). Перший процес включає формування карток товарів, опис характеристик, завантаження медіафайлів і встановлення актуальних цін. При роботі з OLX цей етап ускладнюється регіональними правилами, лімітами та модерацією оголошень [10]. Другий процес пов’язаний із необхідністю своєчасного оновлення інформації про кількість доступного товару, оскільки невідповідність реальних складських залишків даним на

маркетплейсі може призводити до скасування замовлень та зниження рейтингу продавця [16]. Третій блок охоплює фіксацію замовлень, комунікацію з покупцями, резервування товарів, формування транспортних накладних і контроль оплат.

З точки зору системного підходу, ERP-система має виступати центральним елементом управління бізнес-процесами підприємства [1]. Odoo забезпечує широкі можливості для автоматизації обліку, продажів і складських операцій, однак без інтеграції із зовнішніми торговими платформами її функціональність у сфері електронної комерції є обмеженою. Тому впровадження інтеграційного рішення між Odoo та OLX є необхідною умовою для забезпечення автоматизованого, узгодженого та ефективного обміну даними між системами.

2.2 Аналіз існуючих підходів до інтеграції інформаційних систем

Перш ніж переходити до безпосереднього проєктування інтеграції між Odoo та OLX, необхідно дослідити сучасні підходи до інтеграції інформаційних систем взагалі та специфічні особливості кожного середовища зокрема. Правильно обраний архітектурний підхід на старті визначає до 80% успіху проєкту, тому до цього аналізу варто поставитися особливо уважно.

2.2.1 API-орієнтовані інтеграційні рішення

Найбільш поширеним та технологічно зрілим підходом до інтеграції сучасних вебзастосунків є використання інтерфейсів програмування додатків (API), побудованих на основі архітектурного стилю REST (Representational State Transfer) [13]. REST-орієнтована взаємодія ґрунтується на концепції ресурсів, кожен з яких

ідентифікується унікальним URI та оперується стандартними методами протоколу HTTP [14].

Основними перевагами API-орієнтованого підходу є слабке зв’язування систем, стандартизація обміну даними та прогнозована масштабованість. Слабке зв’язування забезпечує незалежність систем одна від одної, що дозволяє змінювати внутрішню реалізацію без порушення інтеграції за умови збереження контракту API [17]. Стандартизація даних досягається за рахунок використання легковагих форматів обміну, насамперед JSON, що забезпечує зручну серіалізацію та десеріалізацію об’єктів у різних мовах програмування, зокрема Python. Прогнозована масштабованість реалізується завдяки використанню REST-патернів і можливості розширення функціоналу шляхом додавання нових кінцевих точок (endpoints) без суттєвого впливу на існуючу логіку [18].

Загалом, API-орієнтований підхід є найбільш доцільним для інтеграції Odoo з OLX, оскільки обидві платформи надають відповідні REST API і не потребують використання більш складних інтеграційних механізмів, таких як брокери повідомлень або ESB (Enterprise Service Bus).

2.2.2 Інтеграції в середовищі Odoo

Odoo як платформа надає кілька механізмів для реалізації інтеграцій із зовнішніми системами, кожен з яких має свою область застосування та технічні особливості.

Внутрішня інтеграція базується на розширенні ORM (Object-Relational Mapping) API та розробці кастомних модулів безпосередньо в екосистемі Odoo [7]. Використовуючи мову програмування Python та вбудований ORM-механізм Odoo, розробник може створювати нові бізнес-моделі, розширювати існуючі сутності та додавати додаткові поля до стандартних об’єктів, наприклад ідентифікатори

оголошень OLX. Крім того, існує можливість перевизначення стандартної бізнес-логіки, що дозволяє автоматизувати запуск процесів синхронізації при зміні даних, таких як залишки товарів або статуси замовлень [5]. Основною перевагою цього підходу є прямий доступ до ядра системи та висока продуктивність завдяки відсутності проміжних мережевих шарів.

Зовнішня інтеграція реалізується через External API Odoo, який надає можливість віддаленої взаємодії з системою за допомогою протоколів XML-RPC або JSON-RPC. Такий підхід дозволяє зовнішнім застосункам виконувати операції читання, створення та оновлення даних у базі Odoo через ORM-рівень. Він є доцільним у випадках, коли інтеграційний компонент реалізується як окремий сервіс або мікросервіс, що функціонує незалежно від ядра ERP-системи.

2.2.3 Особливості інтеграції торгових платформ

Інтеграція з торговими платформами та маркетплейсами має ряд специфічних особливостей, що відрізняють її від загальних сценаріїв API-інтеграцій. Ці особливості зумовлені природою торгових даних, вимогами платформ та очікуваннями кінцевих користувачів.

Суворі ліміти на кількість запитів (Rate Limits) є одним із найважливіших технічних обмежень. Вони запроваджуються для захисту API від перевантаження та запобігання зловживанням, що унеможливлює часті або масові запити до системи [18]. У результаті цього інтеграційні рішення повинні враховувати необхідність пакетної обробки даних, використання черг задач та оптимізацію кількості звернень до API.

Динамічна структура категорій і атрибутів також є характерною особливістю маркетплейсів. На відміну від фіксованих довідників у ERP-системах, структура даних на стороні OLX може змінюватися, а набір атрибутів залежить від конкретної категорії

товару. Це потребує реалізації механізмів динамічного отримання та обробки метаданих через API.

Окремо слід виділити особливості життєвого циклу оголошень. Процес їх створення та публікації включає етапи модерації, можливого відхилення, активації або зміни статусу. Інтеграційна система повинна забезпечувати коректну обробку цих станів та їх відображення в ERP-системі для подальшого управління бізнес-процесами [10].

Отже, аналіз існуючих підходів показує, що оптимальним для інтеграції Odoo та OLX буде гібридне рішення: використання RESTful API з автентифікацією OAuth 2.0/JWT як транспортного протоколу, ORM API Odoo для внутрішньої роботи з даними, доповнене планувальниками (cron) для періодичної синхронізації у разі відсутності вебхуків з боку OLX. Такий підхід враховує обмеження обох платформ і забезпечує необхідний рівень надійності.

2.3 Визначення вимог до інтеграційного рішення

Якість кінцевого продукту безпосередньо залежить від повноти та коректності визначених вимог — помилки, допущені на цьому етапі, є найбільш дорогими у виправленні [19]. Тому перед проектуванням архітектури інтеграційного модуля необхідно чітко сформулювати як функціональні, так і нефункціональні вимоги до системи.

Функціональні вимоги визначають набір можливостей, які система повинна забезпечувати для користувачів, зокрема менеджерів з продажу, операторів та адміністраторів [19]. На основі аналізу бізнес-процесів онлайн-торгівлі сформовано такі основні функціональні вимоги до інтеграційного рішення. Система повинна забезпечувати автентифікацію та управління токенами доступу, включаючи проходження процедури авторизації за протоколом OAuth 2.0 із використанням

authorization_code та подальше автоматичне оновлення access_token і refresh_token без участі користувача. Необхідною функцією є синхронізація товарного каталогу та категорій, що передбачає отримання актуальної структури категорій і набору атрибутів із OLX API та їх зіставлення з внутрішніми даними Odoo. Це забезпечує коректне формування оголошень відповідно до вимог маркетплейсу. Інтеграційний модуль також повинен підтримувати створення, оновлення та видалення оголошень на OLX безпосередньо з системи Odoo. Передбачено можливість масового оновлення даних, включаючи ціни та описи товарів, а також автоматичне деактивування оголошень у разі відсутності товару на складі. Усі операції обміну даними між Odoo та OLX мають фіксуватись у журналі з відображенням статусу виконання, часу операції та, у разі помилки, детального опису проблеми. Це забезпечує можливість діагностики та усунення несправностей [19].

Нефункціональні вимоги описують якісні характеристики системи, які визначають її стабільність, продуктивність та безпеку [19]. До основних нефункціональних вимог належать наступні. Система повинна забезпечувати високу продуктивність та масштабованість шляхом використання пакетної обробки запитів і асинхронного виконання операцій. Це дозволяє уникати перевищення лімітів API (rate limiting) та забезпечує стабільну роботу при обробці великих обсягів даних. Облікові дані для доступу до OLX API мають зберігатись у захищеному вигляді та не передаватись у відкритому вигляді. Авторизація реалізується виключно через стандартні протоколи OAuth 2.0 та JWT, що унеможливлює компрометацію облікового запису OLX. Модуль має бути сумісним із актуальними версіями Odoo Community та Enterprise, а також відповідати стандартам розробки модулів Odoo, визначеним у офіційній документації [7].

3 ВИБІР ЗАСОБІВ РЕАЛІЗАЦІЇ

Вибір технологій і засобів реалізації є важливим етапом розробки програмного забезпечення, оскільки визначає інструментальну базу майбутньої системи та безпосередньо впливає на її продуктивність, масштабованість і зручність супроводу. На цьому етапі здійснюється обґрунтування вибору платформи розробки, мов програмування, засобів взаємодії з API та додаткових бібліотек.

3.1 Реалізація архітектури програмного рішення

Розуміння внутрішньої архітектури Odoo є необхідною передумовою для розробки якісного інтеграційного модуля. Odoo побудований за класичною трирівневою архітектурою, що включає рівень представлення, рівень бізнес-логіки та рівень даних [7]. Такий поділ забезпечує чіткий розподіл відповідальностей між компонентами системи та спрощує її розширення і супровід.

3.1.1 Рівень представлення

Для розроблюваного модуля інтеграції з OLX на рівні представлення необхідно реалізувати: форму налаштувань підключення до OLX API, додаткову вкладку або поля на картці товару для управління оголошенням, а також перегляд журналу синхронізації. Усі ці елементи описуються стандартними засобами Odoo без необхідності написання власного JavaScript-коду, що спрощує розробку та забезпечує візуальну узгодженість із рештою системи.

Інтерфейс користувача в Odoo описується декларативно за допомогою XML-файлів — так званих views. Існує кілька типів представлень: форма (form view) для

перегляду та редагування окремого запису, список (list view) для відображення набору записів, канбан (kanban view) для візуалізації процесів, а також графіки, зведені таблиці та інші спеціалізовані представлення [7]. Розробник інтеграційного модуля може розширювати або перевизначати існуючі представлення, додаючи нові поля, кнопки та елементи керування.

Рівень представлення забезпечує відображення результатів роботи API OLX у зрозумілому для користувача вигляді. Це включає отримані відповіді про створення або оновлення оголошень, а також інформацію про помилки, які можуть виникати під час взаємодії систем.

3.1.2 Рівень бізнес-логіки

Рівень бізнес-логіки є центральним компонентом архітектури Odoo. Саме тут описуються всі правила функціонування бізнесу, алгоритми обробки інформації та зв'язки між сутностями [1]. Вся логіка ядра та модулів Odoo пишеться виключно мовою програмування Python.

Кожна модель Odoo успадковується від базового класу models.Model або його похідних і декларативно описує свої поля за допомогою спеціальних типів: fields.Char, fields.Integer, fields.Many2one тощо [5]. Фреймворк автоматично транслює ці описи у відповідні структури бази даних та генерує базові CRUD-операції. Це суттєво прискорює розробку та зменшує обсяг шаблонного коду.

Ключовим механізмом рівня бізнес-логіки є система декораторів та методів ORM API [5]. Декоратор @api.depends дозволяє визначати обчислювані поля, що автоматично перераховуються при зміні залежних значень. Декоратор @api.onchange реагує на зміни полів в інтерфейсі користувача. Методи create, write та unlink можуть бути перевизначені для додавання власної логіки при операціях зі записами. Саме перевизначення методів write та create моделі product.template є основним механізмом

для відстеження змін товарів і ініціювання синхронізації з OLX у розроблюваному модулі.

Крім того, для забезпечення асинхронної синхронізації списку оголошень користувача використовується механізм «серверних дій» (server actions) та планувальників (cron).

3.1.3 Рівень даних

Рівень даних Odoo побудований на основі реляційної СУБД PostgreSQL. Усі дані системи зберігаються у реляційних таблицях, схема яких автоматично генерується та оновлюється фреймворком Odoo відповідно до описів моделей [5]. При встановленні або оновленні модуля Odoo автоматично виконує необхідні міграції схеми бази даних, що суттєво спрощує розгортання та оновлення системи.

PostgreSQL є потужною об'єктно-реляційною СУБД, що підтримує широкий спектр типів даних, індексів та механізмів забезпечення цілісності даних [4]. Odoo активно використовує можливості PostgreSQL: зовнішні ключі для забезпечення реляційної цілісності, індекси для прискорення пошуку, транзакції для забезпечення атомарності операцій.

Проте, ключова особливість Odoo полягає в тому, що розробник практично ніколи не взаємодіє з СКБД напряму за допомогою сирого SQL-коду [6]. Замість цього платформа надає високорівневий інструмент — Odoo ORM (Object-Relational Mapping) API. Патерн ORM автоматично транслює Python-класи в таблиці бази даних, а об'єкти класів — у рядки цих таблиць.

Важливим аспектом роботи з даними є також механізм транзакцій Odoo. Кожен HTTP-запит до сервера Odoo виконується в рамках окремої транзакції бази даних, що автоматично підтверджується при успішному завершенні або відкочується при виникненні помилки [7].

3.2 API платформи OLX

Інтеграція системи Odoo з платформою OLX у межах даного проєкту базується на використанні публічного API OLX, яке забезпечує програмний доступ до функцій створення, оновлення та керування оголошеннями.

Для початку роботи з API OLX необхідно зареєструвати додаток у порталі розробника та отримати облікові дані: client_id та client_secret [10]. Це стандартна практика для всіх сучасних API-платформ, яка дозволяє ідентифікувати, який саме додаток звертається до системи.

API OLX використовує протокол OAuth 2.0 для автентифікації та авторизації. Процес отримання токену доступу виглядає наступним чином:

1. Користувач Odoo надає дозвіл на доступ до свого облікового запису OLX;
2. Сервер OLX видає тимчасовий код авторизації;
3. Odoo обмінює цей код на access_token (зазвичай діє 1-2 години) та refresh_token (діє довго, до відкликання доступу);
4. Access_token передається в кожному запиті до API в заголовку Authorization: Bearer <token>;
5. Коли access_token закінчується, Odoo використовує refresh_token для отримання нового токену без повторної автентифікації користувача.

Створення оголошення здійснюється через POST-запит до відповідного ендпоінту з передачею JSON-об'єкту, що містить усі необхідні параметри: назву, опис, ціну, категорію, локацію, атрибути та ідентифікатори раніше завантажених зображень [10]. Важливою особливістю є те, що набір обов'язкових та необов'язкових атрибутів різниться залежно від категорії оголошення. Наприклад, для категорії електроніки обов'язковим може бути стан товару (новий/вживаний), тоді як для одягу — розмір та колір. Перед створенням оголошення необхідно отримати актуальний список атрибутів для конкретної категорії через відповідний ендпоінт API [10].

Оскільки маркетплейс використовує сувору типізацію та валідацію для кожної категорії (наприклад, не можна опублікувати оголошення в категорії «Телефони» без обов'язкового вказання атрибуту «Стан» або «Марка»), модуль повинен спочатку динамічно завантажувати вимоги до полів через API, порівнювати їх із карткою товару в ERP і лише після успішної внутрішньої перевірки виконувати відправку [10],[16]. Такий підхід мінімізує кількість помилкових мережевих запитів та забезпечить високу надійність роботи системи.

Таким чином, використання офіційного RESTful-інтерфейсу OLX Public API разом із протоколом безпеки OAuth 2.0 відкриває всі необхідні технічні можливості для повної автоматизації бізнес-процесів електронної комерції та синхронізації даних між системами

3.3 Мова програмування Python та використані бібліотеки

Оскільки ядро та модулі платформи Odoo розроблені на мові Python, саме вона є основним інструментом для написання кастомного модуля обміну інформацією.

Python — це високорівнева мова програмування загального призначення з динамічною типізацією, яка відрізняється лаконічним синтаксисом, високою швидкістю розробки та потужною стандартною бібліотекою [20].

Для реалізації інтеграційного модуля Odoo–OLX було використано як стандартні бібліотеки Python, так і вбудовані можливості самої платформи Odoo.

Модуль base64 є частиною стандартної бібліотеки Python і забезпечує функції кодування та декодування даних у форматі Base64 [21]. У контексті розроблюваного модуля він використовується для обробки зображень товарів: Odoo зберігає зображення у вигляді рядків, закодованих у Base64, і перед завантаженням на OLX необхідно декодувати їх у бінарний формат, прийнятний для multipart/form-data запиту.

Без коректної обробки зображень неможлива повноцінна публікація оголошень на платформі.

Бібліотека logging застосовується для ведення журналів роботи системи, зокрема для фіксації помилок синхронізації, запитів до API OLX та результатів виконання операцій. Використання логування є важливою практикою при розробці розподілених систем, оскільки дозволяє відслідковувати стан інтеграції та спрощує діагностику помилок [19].

Бібліотека requests є найпопулярнішою сторонньою бібліотекою Python для виконання HTTP-запитів [20]. Вона надає зручний високорівневий інтерфейс для взаємодії з REST API: методи requests.get(), requests.post(), requests.put(), requests.delete() відповідають стандартним HTTP-методам [14].

Модуль datetime є частиною стандартної бібліотеки Python і надає класи для роботи з датами та часом [21]. У модулі він використовується для кількох цілей: фіксації часу останньої синхронізації у журналі операцій, перевірки терміну дії access_token перед виконанням запиту до OLX API.

Модуль json входить до стандартної бібліотеки Python і забезпечує серіалізацію та десеріалізацію даних у форматі JSON. Оскільки OLX API використовує JSON як основний формат обміну даними, цей модуль є необхідним для формування тіла запитів та розбору відповідей API.

3.4 Засоби розробки та підтримки програмного коду

Якість програмного забезпечення залежить не лише від обраних технологій реалізації, а й від інструментів, що використовуються в процесі розробки. Правильний вибір засобів розробки суттєво впливає на продуктивність програміста, зручність відлагодження та якість кінцевого продукту [22].

Основною платформою реалізації інтеграційного модуля є Odoo. Як зазначалось у попередніх розділах, Odoo надає потужний фреймворк для розробки модулів, що включає ORM для роботи з базою даних, систему представлень на основі XML, механізм подій та планувальник задач [7].

Програмна реалізація виконана мовою Python, яка є основною мовою розробки в середовищі Odoo. Використання Python дозволяє ефективно реалізовувати бізнес-логіку, працювати з API зовнішніх сервісів та використовувати вбудовані механізми ORM для доступу до даних.

Для опису елементів користувацького інтерфейсу та конфігурацій модуля використовується мова XML. За допомогою XML у Odoo визначаються форми, списки, меню, дії та інші компоненти інтерфейсу, з якими взаємодіє користувач під час роботи із системою [7]. Такий підхід дозволяє відокремити логіку роботи модуля від його візуального представлення.

Для зберігання даних використовується PostgreSQL — реляційна СУБД, що є стандартною для Odoo [5]. PostgreSQL забезпечує надійне зберігання як основних даних системи, так і специфічних даних інтеграційного модуля: параметрів підключення до OLX API, зіставлення категорій, ідентифікаторів оголошень та журналу синхронізації.

Для написання та редагування програмного коду використовувалося середовище розробки Visual Studio Code. Даний редактор підтримує роботу з Python, XML та іншими технологіями, які застосовуються під час створення модулів Odoo.

Для тестування API-запитів до OLX використовувався Postman. Це спеціалізований інструмент для тестування RESTful API, який дозволяє: надсилати HTTP-запити різних типів (GET, POST, PUT, PATCH, DELETE), переглядати відповіді сервера у зручному форматі, зберігати історію запитів, створювати колекції запитів для повторного використання.

4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Програмна реалізація є завершальним етапом розробки інтеграційного рішення, під час якого розроблені архітектурні та проєктні рішення втілюються у вигляді програмного коду. Реалізація інтеграційного модуля передбачає створення компонентів, що забезпечують взаємодію між ERP-системою Odoo та платформою OLX, обробку даних, виконання запитів до API та автоматизацію процесів синхронізації.

4.1 Основні функціональні можливості системи

Для формалізації функціональних можливостей системи розроблено UML-діаграму варіантів використання (рисунок 4.1), що дозволяє чітко представити взаємодію між акторами та функціями модуля.

Базовим варіантом використання є авторизація через OAuth 2.0, яка забезпечує отримання та автоматичне оновлення токенів доступу для роботи з OLX API [11]. Усі подальші операції виконуються лише після успішної автентифікації.

Створення оголошення здійснюється з картки товару в Odoo шляхом формування та надсилання структурованого запиту до OLX API. Після успішного виконання система отримує ідентифікатор оголошення та зберігає його в базі даних Odoo [23].

Модуль також підтримує оновлення опису, ціни та інших параметрів товару, а також деактивацію або видалення оголошень. Після виконання таких операцій статуси оголошень синхронізуються між системами та фіксуються в журналі обміну даними.

Для забезпечення актуальності інформації реалізовано механізм автоматичної синхронізації, який відстежує зміни в Odoo та періодично перевіряє стан оголошень

через OLX API. Крім того, користувач може переглядати список оголошень і їх поточні статуси безпосередньо з інтерфейсу Odoo.

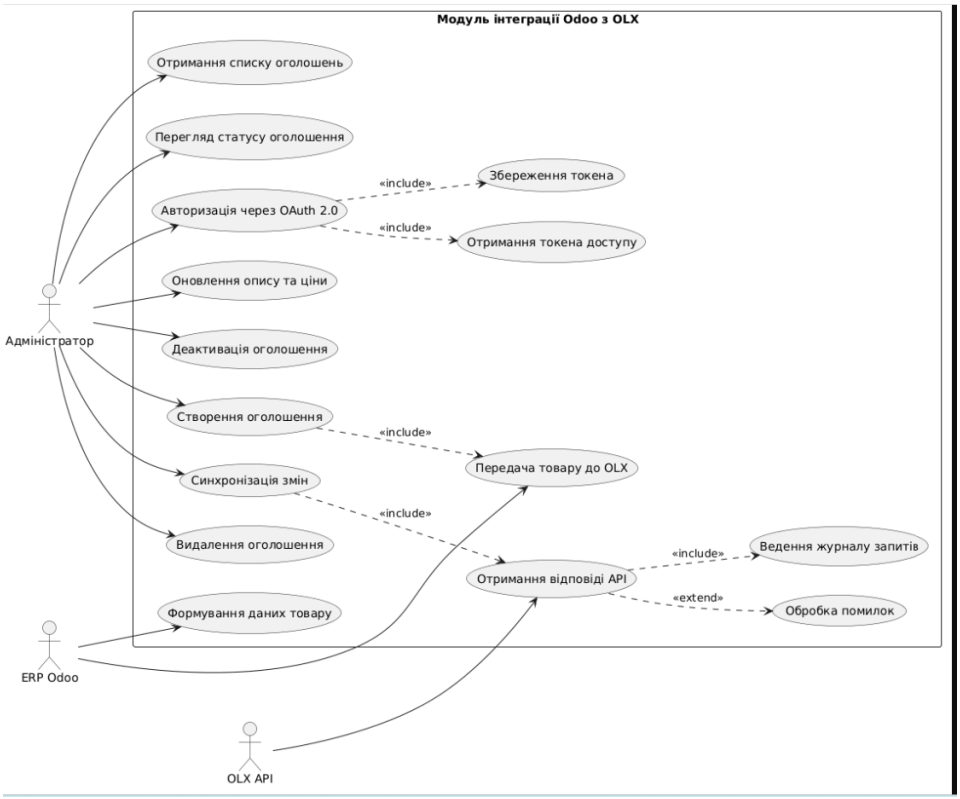


Рис. 4.1 – Діаграма прецедентів

Окремими функціями є формування даних товару для передачі на платформу, ведення журналу запитів та обробка помилок. Усі звернення до API реєструються в системі, а у випадку виникнення помилок виконується їх фіксація та, за потреби, повторне надсилання запиту .

Таким чином, діаграма варіантів використання охоплює повний функціональний цикл інтеграційного модуля — від авторизації до журналювання результатів кожної операції. Усі варіанти використання пов'язані між собою через механізми включення та розширення, що відображає реальну залежність операцій одна від одної і унеможливлює виконання будь-якої дії з OLX API без попередньої авторизації.

4.2 Архітектура інтеграційного модуля Odoo–OLX

4.2.1 Загальна архітектура рішення

Інтеграційний модуль спроектований відповідно до принципів модульності та розділення відповідальностей, що є фундаментальними вимогами до якісного програмного забезпечення [11]. Загальна архітектура рішення базується на трирівневій моделі, де кожен рівень виконує чітко визначену роль і взаємодіє з іншими через визначені інтерфейси.

Перший рівень — рівень інтерфейсу користувача — реалізований засобами XML-представлень Odoo і забезпечує відображення елементів управління інтеграцією в стандартному інтерфейсі системи [7]. На цьому рівні розташовані розширення форми товару, сторінка налаштувань підключення до OLX, інтерфейс зіставлення категорій та представлення журналу синхронізації. Користувач взаємодіє з модулем виключно через цей рівень, не маючи прямого доступу до нижчих рівнів.

Другий рівень — рівень бізнес-логіки — реалізований на Python і є центральним компонентом архітектури. Він містить логіку синхронізації даних, обробники подій Odoo, механізм управління токенами авторизації, логіку трансформації даних між форматами Odoo та OLX, а також обробку помилок [17]. Саме цей рівень координує взаємодію між іншими компонентами системи.

Третій рівень — рівень взаємодії з OLX API — інкапсулює всі деталі роботи з зовнішнім REST API платформи OLX. Він відповідає за формування HTTP-запитів, передачу заголовків авторизації, обробку HTTP-відповідей та перетворення їх у формат, зрозумілий бізнес-логіці модуля. Ізоляція цього рівня є принципово важливою: у разі змін у OLX API достатньо оновити лише компоненти цього рівня, не торкаючись бізнес-логіки.

Взаємодія між рівнями є суворо однонаправленою: рівень інтерфейсу звертається до рівня бізнес-логіки, який у свою чергу використовує рівень взаємодії з

API. Зворотні залежності відсутні, що відповідає принципу інверсії залежностей та забезпечує низьку зв'язність компонентів [24].

Таким чином, трирівнева архітектура інтеграційного модуля забезпечує чіткий розподіл відповідальностей між компонентами системи, мінімізує зв'язність між рівнями та створює надійну основу для подальшого розширення функціональності без порушення існуючої логіки.

4.2.2 Структура програмних модулів

Структура файлів інтеграційного модуля відповідає стандартним конвенціям розробки модулів Odoo і організована у вигляді Python-пакету з чітким розподілом файлів за їх функціональним призначенням [7]. ієрархічну структуру розробленого програмного модуля наведено на рисунку 4.2.

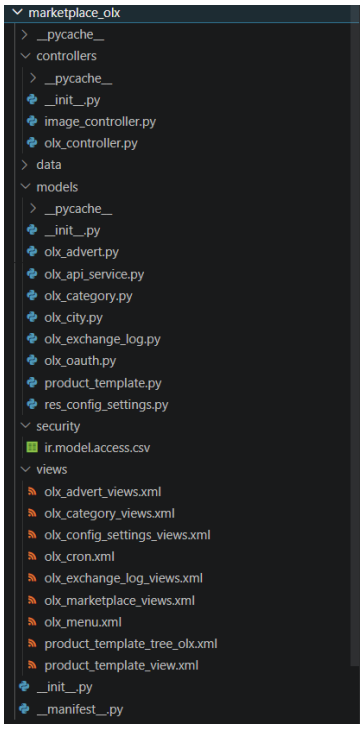


Рис. 4.2 – Структура програмного модуля

Головний конфігураційний файл додатка, __manifest__.py, написаний у вигляді Python-словника. Він містить метадані модуля (назву, версію, автора), а також визначає список залежностей та черговість завантаження XML-файлів інтерфейсу в систему [7].

Підкаталог controllers містить два файли з HTTP-контролерами модуля. Файл olx_controller.py реалізує основні веб-ендпоінти для службових операцій інтеграції. Файл image_controller.py є окремим контролером, що відповідає за обробку та передачу зображень товарів до OLX API — винесення логіки роботи із зображеннями в окремий контролер забезпечує чіткий розподіл відповідальностей [17].

Підкаталог models є центральним компонентом модуля і містить такі файли. Файл olx_advert.py описує модель оголошення, що зберігає інформацію про пов'язані з товарами оголошення на платформі OLX та їх поточний статус. Файл olx_api_service.py інкапсулює логіку взаємодії з OLX API — формування HTTP-запитів, обробку відповідей та управління помилками; розміщення сервісного класу у підкаталозі моделей є характерною практикою для модулів Odoo [9]. Файл olx_category.py реалізує модель зіставлення категорій між системами Odoo та OLX. Файл olx_city.py містить модель для зберігання та зіставлення локацій, що є обов'язковим параметром при створенні оголошення на OLX [10]. Файл olx_exchange_log.py описує модель журналу обміну даними, що фіксує всі операції синхронізації із зазначенням статусу та деталей виконання. Файл olx_oauth.py реалізує логіку авторизації за протоколом OAuth 2.0 — отримання, зберігання та автоматичне оновлення токенів доступу; винесення цієї логіки в окремий файл підкреслює її самостійну відповідальність. Файл product_template.py розширює стандартну модель товару Odoo додатковими полями та методами для управління оголошеннями на OLX. Файл res_config_settings.py розширює стандартну модель налаштувань Odoo параметрами конфігурації підключення до OLX API, що дозволяє зберігати client_id, client_secret та інші параметри через стандартний інтерфейс налаштувань системи [7].

Підкаталог security містить файл ir.model.access.csv з визначенням прав доступу до всіх моделей модуля, що регулює, які групи користувачів Odoo мають право читати, створювати, редагувати або видаляти записи кожної моделі [7].

Підкаталог views містить XML-файли представлень інтерфейсу користувача. Файл olx_advert_views.xml описує представлення для перегляду та управління оголошеннями. Файл olx_category_views.xml містить інтерфейс управління зіставленням категорій. Файл olx_config_settings_views.xml описує розширення сторінки налаштувань Odoo параметрами підключення до OLX. Файл olx_cron.xml містить визначення запланованих задач автоматичної синхронізації. Файл olx_exchange_log_views.xml описує представлення журналу обміну даними. Файл olx_marketplace_views.xml містить загальні представлення модуля. Файл olx_menu.xml визначає структуру меню модуля в інтерфейсі Odoo. Файли product_template_tree_olx.xml та product_template_view.xml реалізують розширення стандартних представлень товарів — списку та форми — додатковими елементами управління інтеграцією з OLX [7].

4.3 Структура даних і взаємодії

4.3.1 Ключові сутності та зв’язки

Інформаційна модель інтеграційного модуля включає сім ключових сутностей, що разом забезпечують повний цикл управління оголошеннями на OLX із системи Odoo. Структура сутностей та зв'язки між ними відображені на діаграмі сутність-зв'язок (рисунок 4.3).

Сутність ProductTemplate є основною моделлю товару в Odoo. В рамках інтеграційного модуля вона була розширена додатковими полями та методами.

OlxAdvert є центральною сутністю модуля і представляє оголошення на платформі OLX. Сутність пов'язана відношенням «багато до одного» з ProductTemplate, OlxCategory та OlxCity — тобто кожне оголошення належить одному товару, має одну категорію та одну локацію.

OlxCategory зберігає зіставлення категорій між Odoo та OLX. OlxCity є аналогічною довідниковою сутністю для локацій. Вона зберігає назву міста та його ідентифікатор у системі OLX (olx_id), оскільки платформа вимагає обов'язкової прив'язки оголошення до конкретної локації.

OlxExchangeLog є сутністю журналу обміну даними і фіксує кожну операцію взаємодії з OLX API.

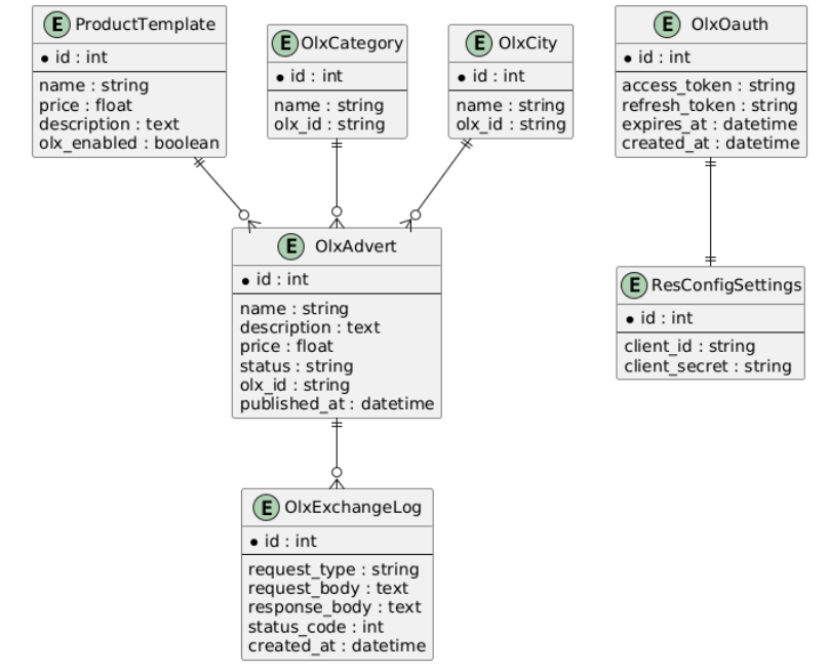


Рис. 4.3 – ER-діаграма основних сутнестей модуля

OlxOauth зберігає токени авторизації для доступу до OLX API відповідно до протоколу OAuth 2.0 [11].

ResConfigSettings є розширенням стандартної моделі налаштувань Odoo і зберігає облікові дані для підключення до OLX API: client_id та client_secret. Ці параметри вводяться адміністратором одноразово при первинному налаштуванні модуля і використовуються компонентом авторизації для отримання токенів доступу.

Зв'язки між сутностями відображають реальну бізнес-логіку процесу — оголошення є похідним від товару, прив'язане до конкретної категорії та локації, а всі операції з ним фіксуються у журналі. Така структура забезпечує цілісність даних на рівні бази даних та спрощує навігацію між пов'язаними записами в інтерфейсі Odoo.

4.3.2 Діаграма класів

Об'єктно-орієнтована структура програмного коду представлена розробленою діаграмою класів UML, яка визначає архітектурні межі, розподіл обов'язків (SRP) та інтерфейси методів взаємодії між компонентами додатка [24]. Діаграма класів інтеграційного модуля відображає програмну реалізацію описаних сутностей у вигляді класів Python та зв'язки між ними (рисунок 4.4).

Згідно з діаграмою класів, програмне рішення складається з кількох ключових компонентів, які відповідають за різні рівні обробки даних та взаємодії між Odoo і OLX.

ProductTemplate (рівень моделі ERP) містить логічний атрибут olx_enabled та метод olx_sync(), який запускає процес синхронізації при зміні стану товару. OlxAdvert (рівень бізнес-моделі шлюзу) акумулює дані оголошення та реалізує метод publish(), що формує об’єкт для передачі та передає його на сервісний рівень.

OlxApiService є центральним сервісним класом, який забезпечує взаємодію з OLX API через HTTP-запити за допомогою бібліотеки requests [20]. Він інкапсулює логіку REST-викликів [7] та містить методи authorize() для автентифікації, create_ad(),

update_ad(), delete_ad() для CRUD-операцій, а також get_ads() для отримання списку оголошень.

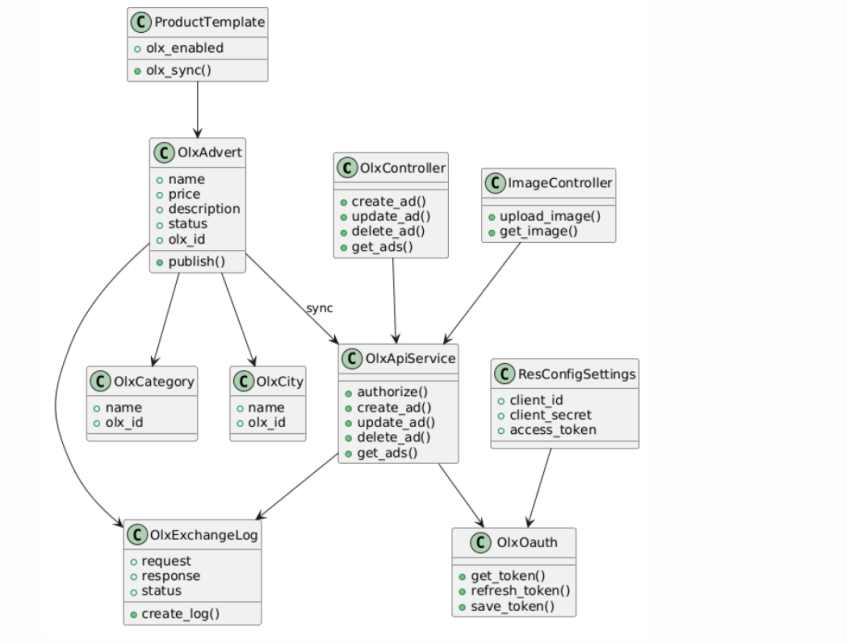


Рис. 4.4 – Діаграма класів механізму обміну

OlxController та ImageController відповідають за обробку запитів і керування асинхронними процесами. OlxController реалізує управління оголошеннями, тоді як ImageController виконує кодування та завантаження зображень через метод upload_image() сервісного рівня.

OlxOauth відповідає за безпеку та управління токенами доступу, реалізуючи методи get_token(), refresh_token() та save_token(), включаючи автоматичне оновлення токенів і їх збереження в базі даних.

OlxExchangeLog забезпечує аудит взаємодії із зовнішнім API через метод create_log(), який фіксує всі запити та відповіді системи незалежно від їх результату, формуючи історію обміну даними.

4.4 Реалізація механізму API-взаємодії

Ключовою складовою інтеграційного модуля є механізм безпосередньої взаємодії з REST API платформи OLX.

Діаграма послідовності відображає взаємодію п'яти учасників: користувача (User), інтерфейсу Odoo (Odoo Interface), інтеграційного модуля (Integration Module), сервера авторизації OLX (OLX OAuth Server) та безпосередньо OLX API(рисунок 4.5).

Процес починається з натискання користувачем кнопки публікації оголошення в інтерфейсі Odoo (Publish advert), після чого викликається метод sendAdvert() інтеграційного модуля.

Далі модуль виконує авторизацію, надсилаючи POST-запит до /oauth/token та отримуючи access_token, який зберігається для подальших запитів через клас OlxOauth.

Після цього формується JSON-об’єкт оголошення з даних товару Odoo (name, description, price, category), при цьому категорії зіставляються між Odoo та OLX, а серіалізація виконується через модуль json.

Сформовані дані надсилаються POST-запитом до /adverts OLX API з передачею токена в заголовку Authorization: Bearer <token>. У відповідь OLX повертає advert_id та статус active, що підтверджує успішну публікацію.

Отримані дані зберігаються в моделі OlxAdvert у Odoo, а також фіксуються в журналі OlxExchangeLog разом із параметрами запиту та відповіді. Після цього користувач отримує повідомлення про успішне створення оголошення в інтерфейсі системи.

Логіка взаємодії інкапсульована в класі OlxApiService, який реалізує основні методи роботи з API (create_ad, update_ad, delete_ad, get_ads). Для запитів із кодом 429 реалізовано механізм повторних спроб із затримкою, а всі операції додатково логуються для діагностики та контролю роботи модуля [18], [19].

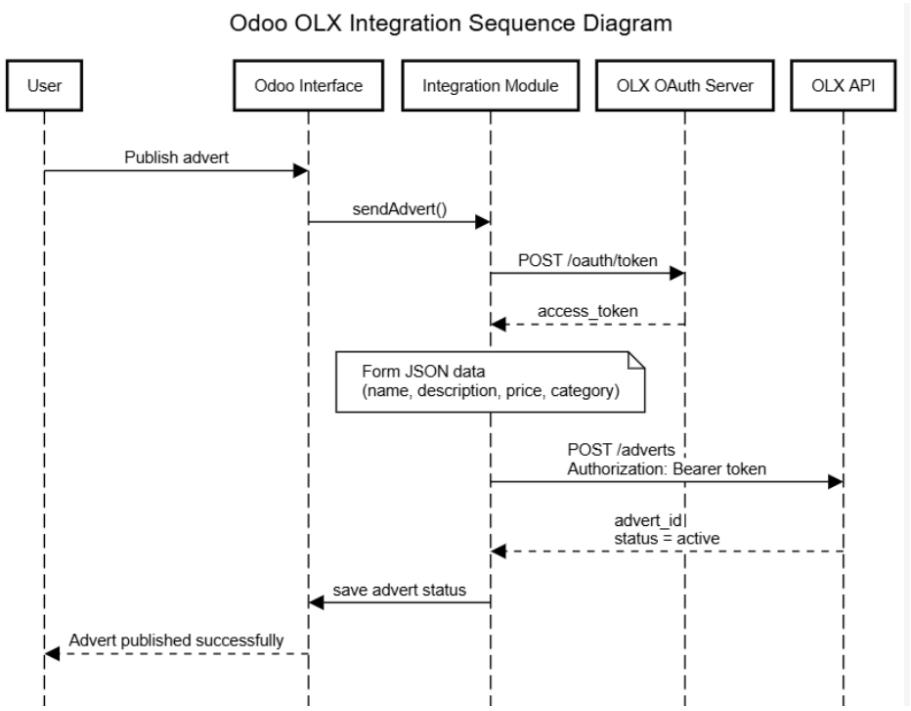


Рис. 4.5 – Діграма послідовності взаємодії Odoo з OLX

Використання OAuth 2.0 авторизації, структурованого журналювання операцій та обробки помилок із механізмом повторних спроб гарантує стабільну роботу модуля в умовах реального навантаження. Інкапсуляція логіки в класі OlxApiService спрощує підтримку та подальше розширення функціональності інтеграції.

4.5 Реалізація авторизації та захисту даних

Логіка авторизації реалізована у класі OlxOauth, який відповідає за отримання, збереження, перевірку актуальності та оновлення токенів доступу, відокремлюючи цей функціонал від основної бізнес-логіки модуля [22].

Первинна авторизація виконується методом get_token(), який отримує client_id та client_secret із ResConfigSettings і надсилає POST-запит до /oauth/token сервера

OLX. У відповідь повертаються access_token для доступу до API та refresh_token для його оновлення.

Метод save_token() зберігає токени в базі даних Odoo разом із часом створення та строком дії (expires_at), що дозволяє перевіряти їх актуальність без додаткових запитів до API. Зберігання в базі забезпечує збереження авторизаційного стану між перезапусками системи [5], [7].

Оновлення токена реалізується методом refresh_token(), який викликається при завершенні строку дії access_token. Запит до /oauth/token виконується з використанням refresh_token, після чого нові дані автоматично зберігаються через save_token().

Доступ до облікових даних обмежено адміністраторами через механізм ResConfigSettings, а передача даних здійснюється виключно через HTTPS із використанням Bearer-токена в заголовках запитів [7]. Для підвищення безпеки токени не зберігаються у відкритому вигляді в логах.

Обробка помилок передбачає автоматичне оновлення токена при відповіді 401, повторну авторизацію при 400 та логування всіх критичних подій через систему logging Odoo [21].

Реалізована система забезпечує безпечне отримання, збереження та автоматичне оновлення токенів доступу, а також захищену взаємодію з API через HTTPS. Використання механізмів обробки помилок і логування дозволяє підвищити надійність роботи модуля та стабільність обміну даними між системами.

4.6 Синхронізація даних між Odoo та OLX

Модуль реалізує два взаємодоповнюючих механізми: подієво-орієнтований та плановий.

Подієво-орієнтована синхронізація базується на перевизначенні методу write() у моделі товару ProductTemplate. При зміні важливих полів (ціна, назва, опис) і

активованій інтеграції (olx_enabled = True) автоматично викликається метод olx_sync(), який визначає потрібну дію: створення, оновлення або деактивацію оголошення залежно від стану товару [10]. Додатково відстежуються зміни залишків складу, що дозволяє автоматично приховувати або відновлювати оголошення при зміні доступності товару.

Планова синхронізація реалізується через Scheduled Actions Odoo і періодично перевіряє відповідність даних між системами. Вона використовується як резервний механізм для виправлення можливих розбіжностей, а також для оновлення статусів оголошень на OLX та первинної масової синхронізації [23].

Стан кожного оголошення зберігається в моделі OlxAdvert (status, olx_id, published_at) і відображається в інтерфейсі Odoo. У разі помилок інформація фіксується в журналі OlxExchangeLog, що дозволяє виконувати діагностику та повторну синхронізацію.

Таким чином, реалізована система синхронізації забезпечує надійний та автоматизований обмін даними між Odoo та OLX за будь-яких умов.

4.7 Обробка помилок та логування

У класі OlxApiService всі HTTP-запити реалізовані через бібліотеку requests із використанням конструкцій try-except, що дозволяє перехоплювати мережеві помилки, тайм-аути та інші виключення без зупинки роботи Odoo. Усі критичні події додатково фіксуються через стандартний модуль logging, що забезпечує технічний аудит на рівні сервера.

Другий рівень логування реалізований через модель OlxExchangeLog, яка зберігає історію всіх API-взаємодій у базі даних PostgreSQL [4]. Для кожної операції фіксуються тип запиту, тіло запиту, відповідь сервера, HTTP-статус та час виконання,

що дозволяє повністю відтворити будь-яку транзакцію та провести діагностику [14],[21].

Окремо реалізована обробка специфічних HTTP-кодів. Помилка 401 автоматично ініціює оновлення токена через refresh_token(), 429 призводить до тимчасової паузи та повторної спроби запиту, а 400 фіксується як помилка валідації з оновленням статусу оголошення на failed і передачею повідомлення користувачу в інтерфейс Odoo.

5 РОБОТА КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

Взаємодія користувача з програмною системою є завершальним етапом реалізації інтеграційного рішення, який визначає практичну зручність використання розробленого модуля та ефективність виконання основних бізнес-операцій. Саме на цьому рівні користувач отримує доступ до функціоналу синхронізації, керування оголошеннями та контролю стану обміну даними між Odoo та OLX.

Інтерфейс системи побудований таким чином, щоб мінімізувати кількість ручних операцій та забезпечити інтуїтивну роботу з інтеграційними функціями без необхідності глибоких технічних знань. Основні сценарії взаємодії охоплюють налаштування підключення до OLX, управління товарами, запуск синхронізації, а також перегляд результатів виконаних операцій і журналу обміну даними.

5.1 Системні вимоги

Для коректної роботи інтеграційного модуля необхідне виконання ряду системних вимог, що стосуються як серверного середовища, так і програмного забезпечення.

Модуль інтеграції розроблений як розширення Odoo, тому основною вимогою є встановлена платформа Odoo 18. Мінімальні апаратні ресурси для малого бізнесу становлять приблизно 2 CPU ядра та 4 ГБ оперативної пам’яті.

Серверна частина повинна містити Python 3.10+ та PostgreSQL 13+, які є базовими компонентами Odoo [4],[21]. Для HTTP-взаємодії з OLX API використовується бібліотека requests, яка зазвичай входить до стандартного середовища Odoo [20].

Обов’язковою умовою є стабільне інтернет-з’єднання з доступом до HTTPS (порт 443), оскільки модуль взаємодіє з OLX API та OAuth-серверами. Блокування

мережевих запитів або нестабільне з’єднання може призводити до помилок синхронізації, які обробляються системою журналювання та повторних спроб [11].

Також необхідний активний бізнес-акаунт OLX із доступом до Partner API, включаючи client_id та client_secret [10]. Обмеження платформи щодо кількості запитів і оголошень залежать від тарифу і не контролюються модулем.

Клієнтська частина працює через веббраузер, тому достатньо сучасного браузера (Chrome, Firefox, Edge, Safari), без встановлення додаткового ПЗ [7]. Додатково потрібні базові навички адміністрування Odoo для первинного налаштування, тоді як щоденна робота користувача не вимагає технічної підготовки.

Загалом вимоги є типовими для веборієнтованих ERP-рішень і не створюють значного навантаження на інфраструктуру підприємства.

5.2 Огляд програмної системи

Робота користувача з системою починається із встановлення модуля, що виконується стандартним способом для всіх розширень Odoo. У списку застосунків системи з'являється відповідний модуль. Адміністратор знаходить модуль за назвою або ключовим словом «OLX» та натискає кнопку встановлення. Після завершення встановлення система автоматично створює необхідні таблиці у базі даних PostgreSQL, реєструє заплановані задачі та додає нові пункти меню до інтерфейсу Odoo. Успішно встановлений модуль можна оновити або видалити (рисунок 5.1).

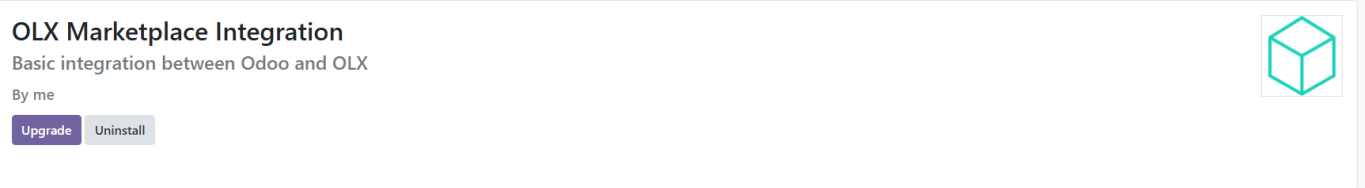


Рис. 5.1 – Успішно встановлений модуль

Першим кроком після встановлення є налаштування підключення до OLX API. Адміністратор переходить до розділу «Налаштування» Odoo, де модуль додає окрему секцію з параметрами інтеграції з OLX. У відповідні поля вводяться client_id та client_secret, отримані при реєстрації застосунку на платформі OLX (рисунок 5.2).

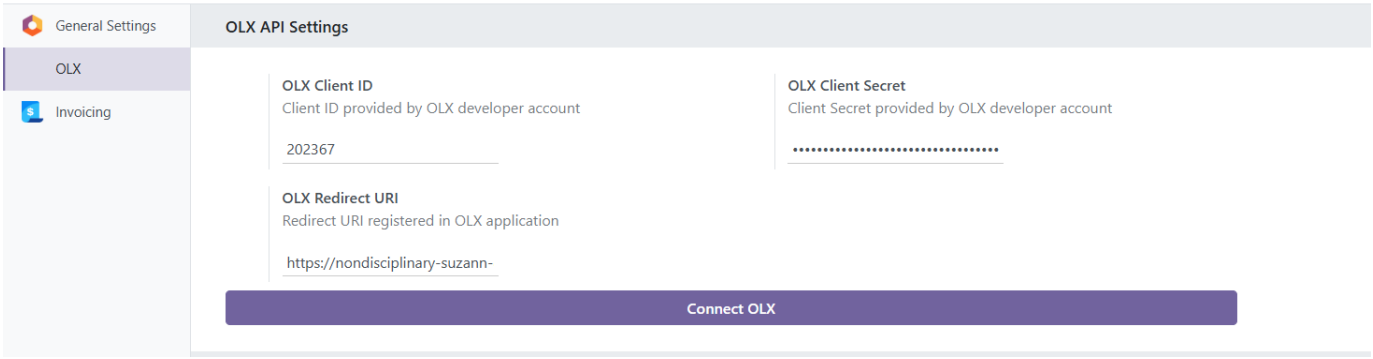


Рис. 5.2 – Налаштування підключення до OLX API

Після збереження налаштувань адміністратор ініціює процес авторизації, натискаючи відповідну кнопку. Модуль автоматично виконує OAuth 2.0 авторизацію [9], отримує токени доступу та зберігає їх у системі. Успішна авторизація підтверджується відповідним повідомленням в інтерфейсі, після чого модуль готовий до роботи (рисунок 5.3). У разі помилки авторизації система відображає детальне повідомлення про причину проблеми для її швидкого усунення.

Рис. 5.3 – Успішна авторизація

Наступним кроком користувач переходить у встановлений модуль. Вкладка OLX Products містить список усіх товарів користувача, який він може редагувати. Тут відображається назва товару, помітка чи опубліковано даний товар на OLX, кількість оголошень, а також внутрішні посилання від Odoo (рисунок 5.4).

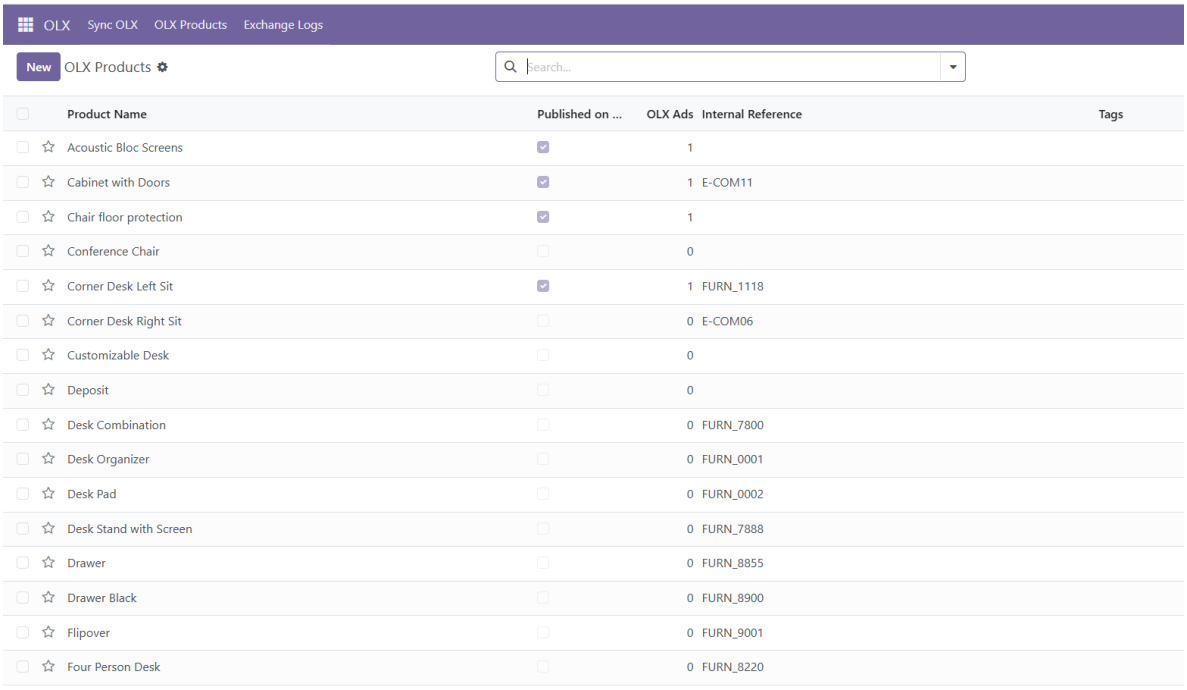


Рис. 5.4 – Вкладка OLX Products

Користувач має можливість відкрити картку товару щоб дізнатися більше інформації або створити оголошення(рисунок 5.5). Тут знаходиться фото товару, кнопка «Create OLX Advert», яка створює оголошення та переводить користувача на вкладку для публікації оголошення. Додатково було реалізовано смарт кнопку, яка має посилання на список всіх оголошень даного товару – OlX Adverts. Такий підхід дозволяє користувачу швидко отримати повну картину щодо активності конкретного товару на платформі, не виходячи з його картки. Завдяки цьому зручно відстежувати кількість створених оголошень, їх поточний статус та за потреби оперативно переходити до редагування або деактивації окремих із них.

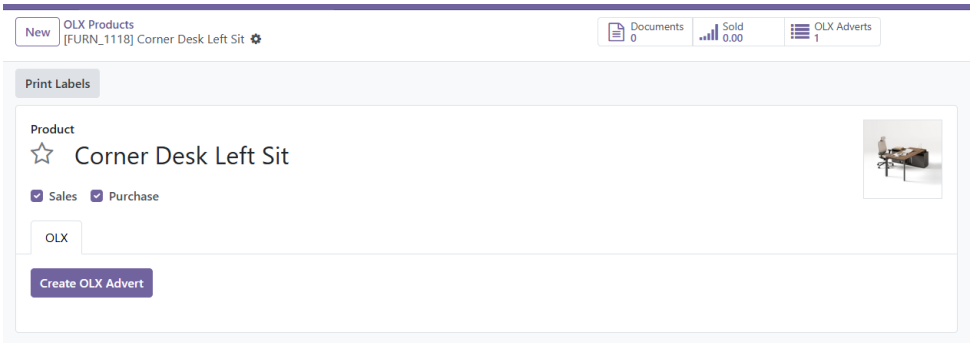


Рис. 5.5 – Картка товару

Форма створення оголошення відкривається з картки товару і поєднує автоматично заповнені поля з полями, що потребують введення користувачем (рисунок 5.6).

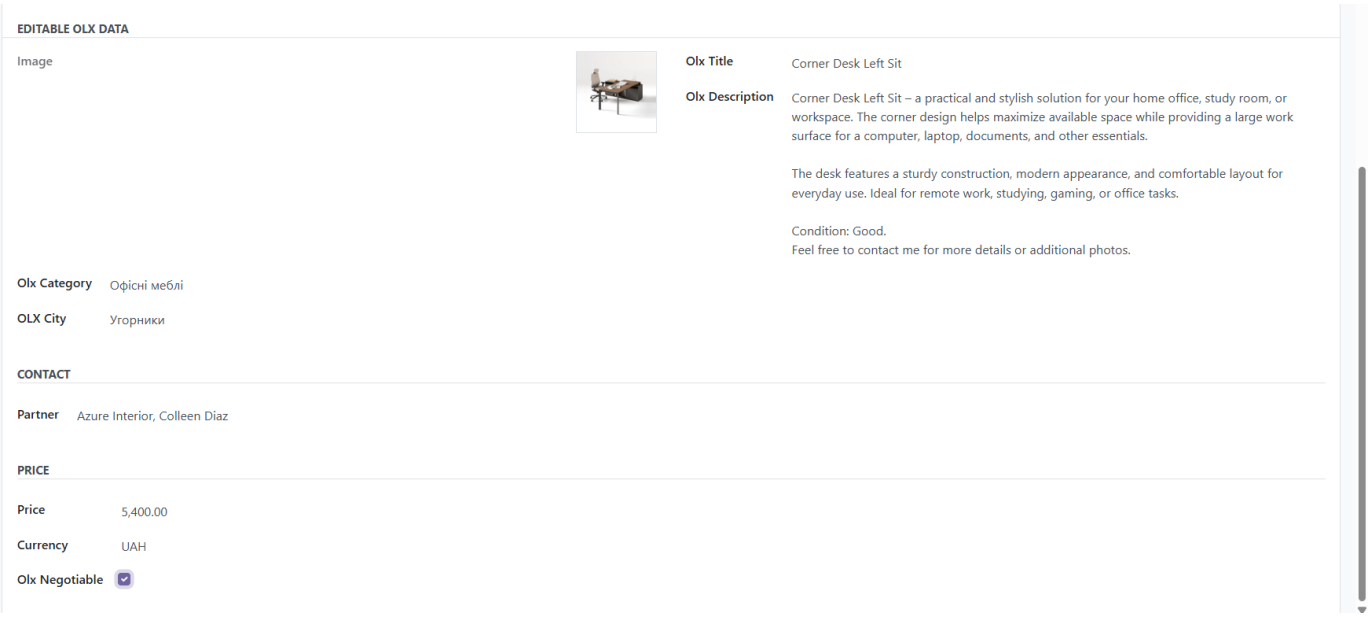
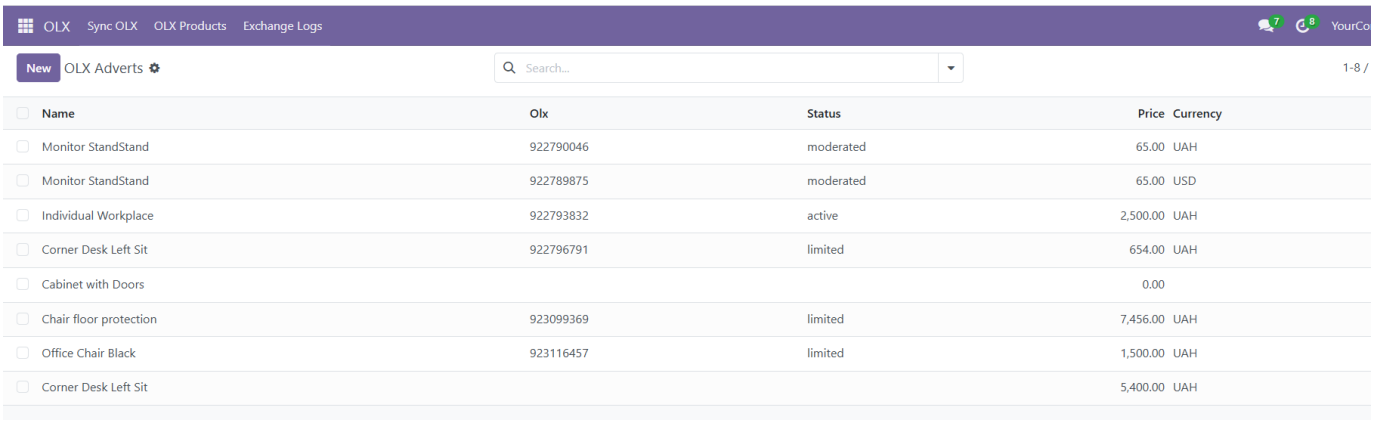


Рис. 5.6 – Створення оголошення

Назва товару та фотографії підтягуються з картки товару Odoo. Опис потрібно вводити вручну з Odoo, а модуль перевіряє його відповідність мінімальній довжині, встановленій OLX. Категорії та міста завантажуються через OLX API, що забезпечує

актуальність довідників та можливість автоматичного вибору категорії за попередньо налаштованими відповідностями. Контактна інформація заповнюється на основі даних зі стандартного модуля Odoo Contacts. Водночас комерційні параметри, такі як ціна, валюта та можливість торгу, вводяться користувачем вручну, оскільки можуть відрізнятися від даних, що зберігаються в системі Odoo.

Робота користувача з оголошеннями реалізована у вкладці OLX Sync в системі Odoo. У цій вкладці відображаються всі оголошення, що були створені або синхронізовані з платформою OLX. Для кожного оголошення доступна основна інформація: назва товару, OLX ID — унікальний ідентифікатор оголошення на платформі, поточний статус, а також ціна та валюта(рисунок 5.7).



Name	Olx	Status	Price	Currency
Monitor StandStand	922790046	moderated	65.00	UAH
Monitor StandStand	922789875	moderated	65.00	USD
Individual Workplace	922793832	active	2,500.00	UAH
Corner Desk Left Sit	922796791	limited	654.00	UAH
Cabinet with Doors			0.00	
Chair floor protection	923099369	limited	7,456.00	UAH
Office Chair Black	923116457	limited	1,500.00	UAH
Corner Desk Left Sit			5,400.00	UAH

Рис. 5.7 – Вкладка Sync OLX

Система відображає як опубліковані оголошення, так і чернетки, що дозволяє користувачу контролювати повний життєвий цикл оголошень — від підготовки до публікації та подальшого управління аж до деактивації або видалення. Наявність чернеток є особливо зручною для попередньої підготовки оголошень перед їх фактичною публікацією на платформі.

Синхронізація даних між Odoo та OLX відбувається автоматично кожні 10 хвилин через механізм запланованої задачі Odoo, що забезпечує постійну актуальність

інформації про статуси оголошень без ручного втручання. Водночас усі активні дії користувача — створення, оновлення або видалення оголошення — виконуються миттєво через API без очікування наступного циклу планової синхронізації [26].

Для моніторингу стану інтеграції модуль надає окремий розділ журналу обміну даними (рисунок 5.8). Журнал відображається у вигляді списку записів з інформацією про тип операції, час виконання, HTTP-статус відповіді та пов'язане оголошення. При відкритті окремого запису адміністратор бачить повну деталізацію: тіло запиту, що був надісланий до OLX API, та відповідь сервера у форматі JSON — що є критично важливим для діагностики проблем .

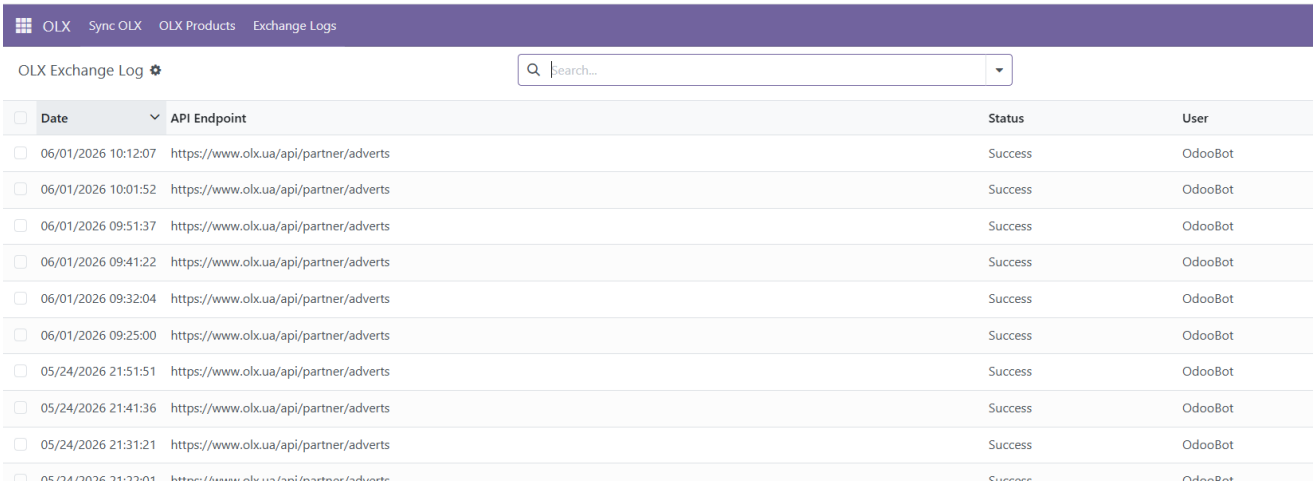


Рис.5.8 – Вкладка Exchange Logs

Після успішної публікації оголошення користувач може переглянути його безпосередньо на платформі OLX (рисунок 5.9). Після успішної публікації оголошення користувач може переглянути його безпосередньо на платформі OLX (рисунок 5.9). Опубліковане оголошення містить усі раніше введені дані — назву товару, опис, фотографії, ціну та контактну інформацію — і відображається у відповідній категорії платформи у стандартному для OLX вигляді, доступному для потенційних покупців.

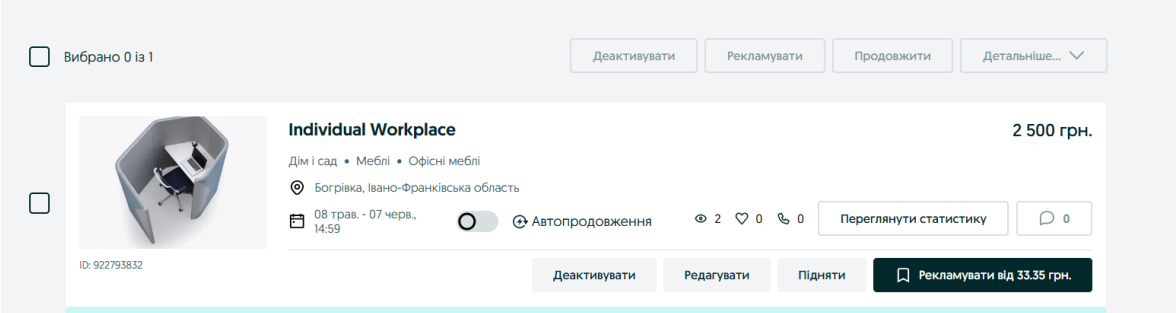


Рис. 5.9 – Активне оголошення OLX

Таким чином, інтуїтивний інтерфейс дозволяє користувачам ефективно керувати товарами та оголошеннями безпосередньо в середовищі Odoo, не перемикаючись між системами.

ВИСНОВКИ

Виконана бакалаврська робота присвячена вирішенню актуальної задачі автоматизації процесів електронної комерції шляхом організації обміну інформацією між ERP-системою Odoo та платформою OLX. Актуальність дослідження обумовлена зростанням ролі маркетплейсів у сучасній онлайн-торгівлі та необхідністю забезпечення оперативного й надійного обміну даними між внутрішніми інформаційними системами підприємства та зовнішніми торговими платформами.

У процесі виконання роботи було проведено аналіз сучасних підходів до інтеграції інформаційних систем, досліджено особливості функціонування ERP-систем і маркетплейсів, а також розглянуто можливості використання REST API та протоколу OAuth 2.0 для реалізації безпечної взаємодії між програмними платформами. Аналіз бізнес-процесів онлайн-торгівлі показав, що ручне керування оголошеннями на маркетплейсах призводить до дублювання даних, виникнення помилок під час оновлення інформації, збільшення навантаження на персонал та зниження ефективності роботи підприємства. Особливо це проявляється при роботі з великим асортиментом товарів, коли підтримання актуальності інформації потребує значних часових витрат.

На основі проведеного аналізу було сформовано вимоги до інтеграційного рішення, виконано моделювання системи за допомогою UML-діаграм та спроєктовано архітектуру програмного модуля. Формалізація процесів взаємодії між користувачем, ERP-системою та платформою OLX дозволила визначити основні інформаційні потоки, структуру даних та послідовність виконання операцій обміну інформацією. Застосування UML-моделювання дало можливість систематизувати вимоги до системи та створити основу для подальшої реалізації програмного рішення.

Практичним результатом роботи стала розробка інтеграційного модуля для Odoo, який забезпечує автоматизований обмін інформацією з платформою OLX. Реалізоване рішення підтримує авторизацію через OAuth 2.0, передачу даних про

товари, створення та оновлення оголошень, контроль їх статусів, а також механізми синхронізації та журналювання операцій. Використання автоматизованої синхронізації дозволяє підтримувати актуальність інформації на маркетплейсі та зменшує ризик виникнення розбіжностей між даними в Odoo та OLX.

Проведений аналіз результатів показав, що впровадження розробленого рішення сприяє скороченню кількості ручних операцій, зменшенню впливу людського фактора та підвищенню ефективності управління товарами. Централізація процесів у межах ERP-системи дозволяє користувачам працювати в єдиному інформаційному середовищі без необхідності постійного перемикання між різними платформами. Крім того, модуль спроєктовано з урахуванням можливості подальшого розширення функціональності та адаптації до інших торгових майданчиків.

Таким чином, поставлену мету роботи досягнуто, а всі визначені завдання виконано. Розроблене інтеграційне рішення забезпечує надійний обмін інформацією між ERP-системою Odoo та платформою OLX, має практичну цінність для підприємств малого та середнього бізнесу та може бути використане як основа для подальшого розвитку багатоканальних систем електронної комерції.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Monk E., Wagner B. Concepts in Enterprise Resource Planning. 5th ed. Boston Cengage Learning, 2022. 320 p.

2. Velosio. Role of ERP in Digital Transformation. 2024. URL: https://www.velosio.com (дата звернення: 27.05.2026)

3. Gestisoft. The Critical Role of ERP in Digital Transformation. 2024. URL https://www.gestisoft.com (дата звернення: 27.05.2026)

4. Elmasri R., Navathe S. Fundamentals of Database Systems. 7th ed. Boston : Pearson, 2017. 1272 p.

5. Odoo Documentation. ORM API. URL: https://www.odoo.com/documentation (дата звернення: 28.05.2026)

6. Celko J. SQL for Smarties: Advanced SQL Programming. 5th ed. Burlington : Morgan Kaufmann, 2014. 920 p.

7. Odoo Documentation. Developer Documentation. URL: https://www.odoo.com/documentation (дата звернення: 28.05.2026)

8. Odoo Documentation. External API. URL: https://www.odoo.com/documentation (дата звернення: 30.05.2026)

9. Odoo Documentation. Web Controllers (HTTP). URL: https://www.odoo.com/documentation/17.0/developer/reference/backend/http.html (дата звернення: 29.05.2026)

10. OLX Group. Public API Documentation. URL: https://developer.olx.com (дата звернення: 29.05.2026)

11. Hardt D. The OAuth 2.0 Authorization Framework. RFC 6749. URL: https://datatracker.ietf.org/doc/html/rfc6749 (дата звернення: 29.05.2026)

12. Laudon K., Traver C. E-Commerce: Business, Technology, Society. 18th ed. London Pearson, 2024. 912 p.

13. Fielding R. Architectural Styles and the Design of Network-based Software Architectures : Doctoral Dissertation. University of California, Irvine, 2000. 162 p.;

14. Richardson L., Amundsen M. RESTful Web APIs. Sebastopol : O'Reilly Media, 2013. 408 p.

15. Shop-Express. Бізнес-процеси інтернет-магазину. URL: https://shop express.ua/ukr/blog/online-store-business-processes/ (дата звернення: 30.05.2026).;

16. API2Cart. Ecommerce API Integration Guide. 2025. URL: https://api2cart.com (дата звернення: 30.05.2026)

17. Fowler M. Patterns of Enterprise Application Architecture. Boston : Addison-Wesley, 2002. 560 p.

18. Allamaraju S. RESTful Web Services Cookbook. Sebastopol : O'Reilly Media, 2010. 400 p.

19. Sommerville I. Software Engineering. 10th ed. Boston : Pearson, 2016. 816 p.

20. Martelli A. Python in a Nutshell. 4th ed. Sebastopol : O'Reilly Media, 2023. 800 p.

21. Python Software Foundation. Python Documentation. URL: https://docs.python.org/3 (дата звернення: 31.05.2026)

22. Pressman R., Maxim B. Software Engineering: A Practitioner's Approach. 9th ed. New York : McGraw-Hill Education, 2019. 880 p.

23. Бухало Є. Ю., Шушура О. М. Моделювання інтеграції ERP-системи Odoo з маркетплейсом OLX на основі UML. Матеріали XIII Всеукраїнської науково-практичної конференції здобувачів вищої освіти та молодих вчених з автоматичного управління. Херсон-Хмельницький : ФОП Вишемирський В. С., 2026. С. 136

24. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Boston : Addison-Wesley, 1994. 395 p.

Додаток А

Реалізація обміну інформацією системи ODOO з OLX

Програмні засоби:

– мова програмування – Python;
– база даних – PostgreSQL;
– фреймворк – Odoo;
– OLX API;
– надсилання HTTP-запитів – бібліотека requests

Код методів публікації, оновлення, деактивації та видалення оголошень(olx_api_service.py):

```
def _prepare_payload(self, advert):

    self._validate_advert(advert)

    base_url = self.env['ir.config_parameter'].sudo().get_param('web.base.url')

    images = []

    if advert.product_id.image_1920:
        images.append({
            "url": f"{base_url}/olx/public_image/{advert.product_id.id}"
        })

    return {
        "title": (advert.olx_title or advert.name)[:70],
        "description": advert.olx_description[:9000],
```

```
"category_id": advert.olx_category_id.olx_id,
"advertiser_type": advert.olx_advertiser_type or "private",
"external_id": f"ODOO_ADVERT_{advert.id}",
"images": images,

"contact": {
   "name": advert.partner_id.name,
   "phone": advert.partner_id.phone or advert.partner_id.mobile,
},

"location": {
   "city_id": advert.olx_city_id.olx_id,
},

"price": {
   "value": advert.price,
   "currency": advert.currency or "UAH",
   "negotiable": advert.olx_negotiable or False,
   "trade": False,
   "budget": False,
},

"attributes": [
   {
      "code": "state",
      "value": "used"
   }
],
```

```
    }

def _log_call(self, endpoint, request_data, response=None, error=None):
    self.env['olx.exchange.log'].sudo().create_log(
        endpoint=endpoint,
        request_data=request_data,
        response_data=response,
        status='error' if error else 'success',
        error_message=str(error) if error else None
    )

def publish_ad(self, advert):

    payload = self._prepare_payload(advert)

    url = f"{self.BASE_URL}/adverts"

    try:
        response = requests.post(
            url,
            json=payload,
            headers=self._headers(),
            timeout=20
        )

        self._log_call(url, payload, response.text)

        if response.status_code not in (200, 201):
```

```
            raise Exception(response.text)

        raw_data = response.json()

        _logger.info("OLX RESPONSE: %s", raw_data)

        data = raw_data.get("data", {})

        advert.write({
            "olx_id": str(data.get("id")),
            "status": data.get("status"),
            "url": data.get("url"),
        })
        try:
            self.upload_images_from_product(advert)

        except Exception as e:
            _logger.exception("OLX image upload failed: %s", e)

        return data

    except Exception as e:
        self._log_call(url, payload, error=e)
        raise

def update_ad(self, advert):

    if not advert.olx_id:
```

```
            raise Exception("Missing olx_id")

        payload = self._prepare_payload(advert)

        url = f"{self.BASE_URL}/adverts/{advert.olx_id}"

        try:
            response = requests.put(
                url,
                json=payload,
                headers=self._headers(),
                timeout=20
            )

            self._log_call(url, payload, response.text)

            if response.status_code not in (200, 204):
                raise Exception(response.text)

            return True

        except Exception as e:
            self._log_call(url, payload, error=e)
            raise

    def delete_ad(self, advert):

        if not advert.olx_id:
```

```
            raise Exception("Missing olx_id")

        url = f"{self.BASE_URL}/adverts/{advert.olx_id}"

        try:
            response = requests.delete(
                url,
                headers=self._headers(),
                timeout=20
            )

            self._log_call(url, None, response.text)

            if response.status_code not in (200, 204):
                raise Exception(response.text)

            advert.unlink()

            return True

        except Exception as e:
            self._log_call(url, None, error=e)
            raise

    def deactivate_ad(self, advert):

        if not advert.olx_id:
            raise Exception("Missing olx_id")
```

```
url = f"{self.BASE_URL}/adverts/{advert.olx_id}/commands"

payload = {
    "command": "deactivate",
    "is_success": True
}

response = requests.post(
    url,
    json=payload,
    headers=self._headers(),
    timeout=20
)

_logger.info("DEACTIVATE RESPONSE: %s", response.text)

if response.status_code not in (200, 204):
    raise Exception(response.text)

advert.write({
    "status": "removed_by_user"
})

return True
```

Додаток Б

Апробація роботи

УДК 004.738.5

[1]Є.Ю. Бухало, [2]О.М.Шушура

[1] Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»

Bukhalo.yevheniia@lll.kpi.ua

МОДЕЛЮВАННЯ ІНТЕГРАЦІЇ ERP-СИСТЕМИ ODOO З МАРКЕТПЛЕЙСОМ OLX НА ОСНОВІ UML

Постановка проблеми та її актуальність

На сьогодні підприємства активно переходять на використання ERP-систем для організації обліку ресурсів, складських операцій та продажів. Одним із популярних рішень є Odoo завдяки її модульній структурі та відносно простій можливості розширення функціоналу[1].

Водночас значна частка продажів проходить через онлайн-майданчики, зокрема OLX. При ручному перенесенні даних про товари (найменування, опис, ціна, фото тощо) між ERP та маркетплейсом виникають суттєві витрати часу, зростає ймовірність помилок, порушується актуальність цін і статусів, а оновлення інформації відбувається із затримками.

Тому розробка моделі автоматичної інтеграції між ERP-системою та зовнішнім маркетплейсом виглядає доцільною задачею, яка дозволяє оптимізувати процеси створення, редагування та синхронізації оголошень.

Аналіз останніх досліджень

Проблематика інтеграції корпоративних інформаційних систем з зовнішніми сервісами вже достатньо широко висвітлена в літературі. Сучасні ERP-системи, включно з Odoo, підтримують взаємодію через RESTful API та стандарти авторизації OAuth 2.0. У наукових публікаціях часто зустрічаються описи сервісно-орієнтованої архітектури (SOA), мікросервісних підходів, а також використання UML для моделювання бізнес-процесів та точок інтеграції.

Водночас конкретні кейси інтеграції Odoo саме з OLX (або подібними локальними класифайдами) у відкритих джерелах представлені фрагментарно. Існують окремі модулі та комерційні конектори, але формалізовані моделі взаємодії з використанням UML для цієї пари систем практично не описані. Це визначає необхідність самостійної побудови такої моделі.

Формулювання мети

Метою роботи є створення функціональної моделі інтеграції ERP-системи Odoo з маркетплейсом OLX з використанням UML-діаграми варіантів використання (Use Case Diagram) та описом основних механізмів обміну даними між внутрішніми модулями Odoo та зовнішнім API OLX.

Основна частина

Для опису бізнес-процесу інтеграції ERP-системи Odoo з маркетплейсом OLX розроблено UML-діаграму варіантів використання(рис.1). Вона дозволяє чітко представити, як система обробляє товари від моменту їх створення в Odoo до повноцінної публікації, оновлення та управління статусами на стороні OLX. [2]

Робота модуля починається зі створення товару в інтерфейсі Odoo: адміністратор (менеджер продажів) заповнює основні поля - назву, опис, ціну, категорію, характеристики та додає зображення. Цей крок є базовим і необхідним для всіх подальших дій з публікацією.

Після створення товару система може ініціювати публікацію оголошення на OLX. Перед будь-яким зверненням до зовнішнього API обов'язково виконується авторизація за протоколом OAuth 2.0 - отримується токен доступу, який використовується для всіх наступних запитів[3]. Далі формується структурований JSON-запит на основі даних товару з Odoo і надсилається на сервер OLX методом POST. У разі успішної відповіді система отримує унікальний ідентифікатор оголошення, який відразу зберігається в базі даних Odoo. Після цього проводиться синхронізація статусу: модуль перевіряє, чи оголошення дійсно опубліковане, і оновлює відповідні поля в картці товару (наприклад, посилання на оголошення, статус, дата публікації).

Аналогічна логіка застосовується при оновленні оголошення. Якщо адміністратор змінює ціну, опис чи інші параметри в Odoo, модуль знову проходить авторизацію (якщо токен протермінований), а потім надсилає запит на оновлення (PUT або PATCH) з актуальними даними. Після успішного виконання обов'язково виконується синхронізація статусу, щоб переконатися, що зміни застосовано і оголошення не заблоковане чи не відхилене модерацією OLX.

Для деактивації оголошення (зняття з публікації) модуль використовує відповідний запит до API (наприклад, зміна статусу або DELETE-ендпоінт, залежно від специфікації OLX), знову з обов'язковою авторизацією та подальшою синхронізацією статусу в Odoo - щоб картка товару відображала актуальний стан (наприклад, «деактивовано на OLX»).

Окремо передбачено механізм періодичної синхронізації статусів: за розкладом (наприклад, раз на годину чи день) модуль надсилає GET-запити до OLX API за збереженими ідентифікаторами оголошень, перевіряє їх поточний стан (активне, заблоковане, видалене тощо) і автоматично оновлює дані в Odoo. Це дозволяє своєчасно реагувати на зовнішні зміни без ручного втручання[4].

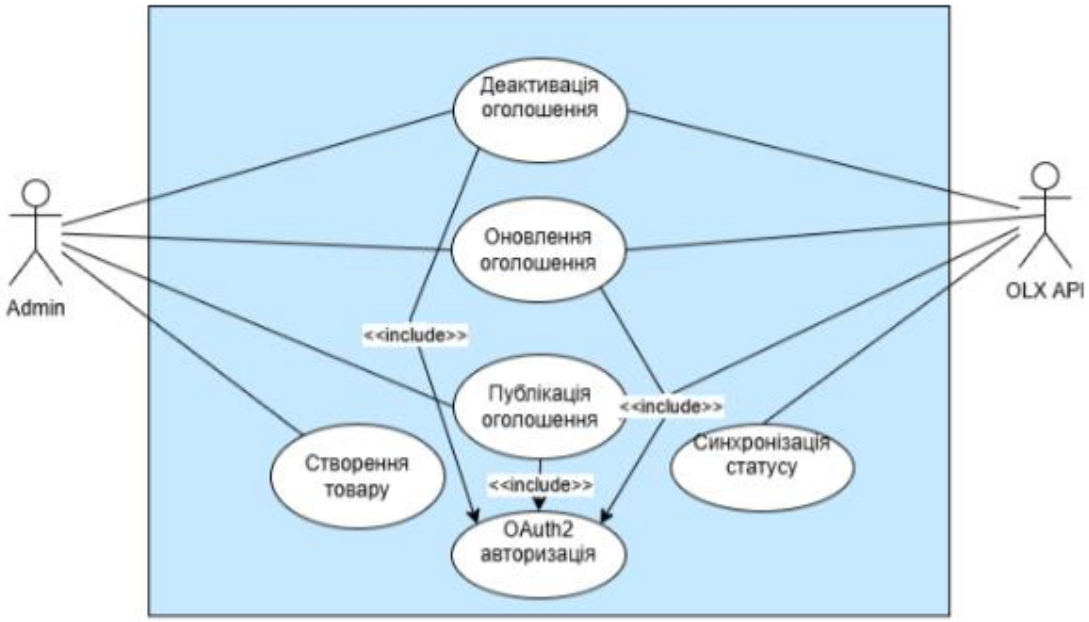


Рис.1 – UML-діаграма варіантів використання інтеграції Odoo з OLX

Вся взаємодія побудована на клієнт-серверній моделі з використанням стандартних HTTP-методів REST API. Повторне використання авторизації та синхронізації статусу як допоміжних процесів (через механізм включення) дозволяє уникнути дублювання коду та забезпечити стабільність інтеграції[3].

Запропонована модель чітко окреслює послідовність операцій, ключові точки інтеграції та залежності між процесами, що робить її зручною основою для програмної реалізації кастомного модуля в Odoo (наприклад, на Python з використанням requests або odoo rpc для внутрішньої логіки).

Побудована UML-діаграма варіантів використання також дозволяє чітко визначити ролі учасників системи та їх взаємодію з основними функціональними модулями. Основним актором виступає адміністратор системи (менеджер продажів), який ініціює ключові бізнес-процеси, пов'язані з управлінням товарами та оголошеннями. Водночас зовнішнім актором є API маркетплейсу OLX, який забезпечує обробку запитів, модерацію та публікацію оголошень.

Діаграма демонструє, що більшість сценаріїв використання пов'язані між собою через відношення <<include>>, що свідчить про наявність спільних підпроцесів. Зокрема, авторизація та синхронізація статусу є обов'язковими складовими практично кожної операції взаємодії з OLX.

Крім того, модель враховує можливі виняткові ситуації, які можуть виникати під час інтеграції. Наприклад, у разі помилки авторизації або втрати чинності токена доступу система повинна автоматично ініціювати повторне отримання токена. У випадку невдалої публікації або оновлення оголошення передбачається обробка помилок, логування та інформування користувача про причини відмови (наприклад, невідповідність вимогам OLX або помилки у форматі даних).

Висновки

У рамках роботи розроблено функціональну модель інтеграції Odoo з OLX на основі UML-діаграми варіантів використання. Запропонований підхід дозволяє описати та формалізувати процес автоматичного розміщення, оновлення та синхронізації оголошень, що зменшує ручну працю, знижує кількість помилок і забезпечує своєчасне оновлення даних на маркетплейсі.

Наукова новизна полягає в застосуванні UML-моделювання саме до задачі інтеграції Odoo з OLX Partner API, оскільки подібні конкретні моделі у відкритих джерелах раніше не фіксувалися.

Практична цінність полягає у можливості використання моделі як технічного завдання для розробки модуля інтеграції, що буде корисним для малих та середніх підприємств, які активно продають через OLX і вже використовують (або планують впровадити) Odoo.