

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

**«Створення стороннього модуля для платформи
електронної комерції Magento 2»**

Виконав студент групи ІПЗ-19-1
МЕЛЬНИК Олександр Олександрович

Керівник роботи:
ТОЛСТОЙ Ігор Анатолійович

Рецензент:
ЧИЖМОТРЯ Олена Геннадіївна

Житомир – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

В.о. зав. кафедри інженерії
програмного забезпечення
Андрій МОРОЗОВ
«14» лютого 2023 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **МЕЛЬНИК Олександр Олександрович**

Керівник роботи: **ТОЛСТОЙ Ігор Анатолійович**

Тема роботи: **«Створення стороннього модуля для платформи електронної комерції Magento 2»**,

затверджена Наказом закладу вищої освіти від «14» лютого 2023 р., №70с

Вихідні дані для роботи: **створити модуль для платформи електронної комерції Magento 2**

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Толстой І.А.	16.02.2023	16.02.2023
2	Толстой І.А.	16.02.2023	16.02.2023
3	Толстой І.А.	16.02.2023	16.02.2023

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка задачі	16 лютого 2023 – 20 лютого 2023	Виконано
2	Аналіз існуючих розробок	22 лютого 2023 – 26 лютого 2023	Виконано
3	Визначення вимог до додатку	28 лютого 2023 – 06 березня 2023	Виконано
4	Обирання технологій для розробки веб-проекту	06 березня 2023 – 10 березня 2023	Виконано
5	Розробка архітектури веб-додатка	13 березня 2023 – 15 березня 2023	Виконано
7	Реалізація архітектури веб-додатка	16 березня 2023 – 28 березня 2023	Виконано
8	Розробка моделі	29 березня 2023 – 04 квітня 2023	Виконано
9	Написання програмного коду	05 квітня 2023 – 05 травня 2023	Виконано
10	Тестування роботи програмного продукту	10 травня 2023 – 12 травня 2023	Виконано
11	Оформлення пояснювальної записки	15 травня 2023 – 24 травня 2023	Виконано
12	Попередній захист	05 травня 2023	Виконано

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з програмного комплексу стороннього модулю для платформи електронної комерції Magento 2 та пояснювальної записки. Пояснювальна записка до випускної кваліфікаційної роботи містить текстову частину, викладену на 55 сторінках друкованого тексту, має 9 сторінок додатків. Список використаних джерел містить 9 найменувань і займає 1 сторінку. В роботі наведено 47 рисунків та 2 таблиці. Загальний обсяг роботи – 66 сторінки.

Метою роботи є розробка стороннього модулю для імпорту медіа-контенту на базі однієї з самих популярних платформ електронної комерції Magento 2 засобами PHP, HTML, CSS, XML та JavaScript.

У роботі було проаналізовано теоретичні та функціональні потреби проектування для реалізації стороннього модулю, проаналізовано та порівняно вже існуючі методи імпортування медіа-файлів, сформоване технічне завдання, а також було реалізовано сторонній модуль.

КЛЮЧОВІ СЛОВА: ЕЛЕКТРОННА КОМЕРЦІЯ, МЕДІА-ФАЙЛ, ІМПОРТ, MAGENTO 2

					ІПЗ.КР.Б – 121 – 23 – ІПЗ			
Зм.	Лист	№ документа	Підпис	Дата	Створення стороннього модуля для платформи електронної комерції Magento 2 Пояснювальна записка	Літера	Лист	Листів
Розроб.	Мельник О.О.						4	
Керівник	Толстой І.А.							
Рецензент	Чижмотря О.Г.							
Н.Контр.								
Зав. каф.	Морозов А.В.					Житомирська політехніка ФІКТ, гр. ІПЗ-19-1		

ABSTRACT

The bachelor's thesis consists of a third-party module software package for the Magento 2 e-commerce platform and an explanatory note. The explanatory note to the final qualification work contains the text part, laid out on 55 pages of printed text, has 9 pages of appendices. The list of used sources contains 9 names and occupies 1 page. The work contains 47 figures and 2 tables. The total volume of work is 66 pages.

The purpose of the work is to develop a third-party module for importing media content based on one of the most popular e-commerce platforms, Magento 2, using PHP, HTML, CSS, XML and JavaScript.

The paper analyzed the theoretical and functional design needs for the implementation of a third-party module, analyzed and compared the already existing methods of importing media files, formed a technical task, and also implemented a third-party module.

KEY WORDS: ELECTRONIC COMMERCE, MEDIA FILE, IMPORT, MAGENTO 2

					ПЗ.КР.Б – 121 – 23 – ПЗ	Лист
Зм.	Лист	№ документа	Підпис	Дата		5

ЗМІСТ

ВСТУП.....	8
РОЗДІЛ 1. АНАЛІЗ ТЕОРЕТИЧНИХ ТА ФУНКЦІОНАЛЬНИХ ПОТРЕБ ПРОЕКТУВАННЯ ДЛЯ РЕАЛІЗАЦІЇ СТОРОННЬОГО МОДУЛЮ.....	11
1.1. Опис проекту та постановка задачі	11
1.2. Аналіз вже представлених методів імпортування медіа-контенту, визначення унікальності.....	14
1.3. Формування технічного завдання	28
Висновки до першого розділу.....	29
РОЗДІЛ 2. ВИЗНАЧЕННЯ ТЕХНОЛОГІЙ ТА ОБ'ЄКТНОЇ СТРУКТУРИ СТОРОННЬОГО МОДУЛЮ	30
2.1. Визначення варіантів використання та об'єктної структури стороннього модулю	30
2.2. Розмірковування і проектування реалізації.....	34
Висновки до другого розділу	37
РОЗДІЛ 3. РОЗРОБКА СТОРОННЬОГО МОДУЛЮ	38
3.1. Підготовка до створення проекту на базі платформи Magento 2	38
3.2. Створення проекту на базі платформи Magento 2	41
3.3. Написання стороннього модулю	45
3.4. Тестування стороннього модулю	47
Висновки до третього розділу.....	54
ВИСНОВКИ	55
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	56
ДОДАТКИ	57

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

PDP (product detailed page) – сторінка з інформацією про продукт

PLP (product listing page) – сторінка з інформацією про декілька сторінок, зазвичай використовується для сторінок результатів пошуку, категорій і вибірок

Cross-sell – техніка продажів для заохочення користувачів до додаткових покупок

Up-sell – техніка продажів для заохочення користувачів до більш дорогих покупок

SSH (Secure Shell Protocol) – протокол передачі даних

FTP (File Transfer Protocol) – протокол передачі даних

SFTP (SSH File Transfer Protocol) – протокол передачі даних

SKU (Stock Keeping Unit) – унікальний ідентифікатор продукту

CSV (comma-separated values) – розширення файлів

CLI (command-line interface) – інтерфейс командного рядка

ВСТУП

Веб-технологій в сучасному світі є невід'ємною частиною нашого життя. Сьогодні майже всі сервіси є цифровізованими для зручного користування за допомогою комп'ютера або телефона. Починаючи з соціальних мереж, в яких люди проводять левову частину свого життя, продовжуючи інтернет-магазинами і сайтами з корисною інформацією, і закінчуючи інтернет банкінгами і електронним документообігом, які значно полегшують наше життя.

Тема інтернет-магазинів є дуже важкою і в той же час дуже цікавою. Беручи до уваги відношення звичайних людей до інтернет-магазинів ще 10 років назад, спокійно можна було зробити висновок, що в цьому немає майбутнього, так як люди просто не довіряють таким сайтам, тим паче коли оплата відбувається не після отримання товару, а до його відправлення. Але час йде, побутові звички змінюються і відношення до інтернет-магазинів змінилось з «я їм не довіряю» до «навіщо їм мене дурити». Саме після зміни цієї парадигми в головах звичайних користувачів відбувся «бум» інтернет-магазинів і інтернет-комерції в цілому. На даний момент інтернет-магазин є зручним місцем для купівлі товару, який тебе цікавить, не покидаючи своєї зони комфорту.

Беручи до уваги все вище сказане, ні для кого не стало відкриттям прагнення людей-ентузіастів до спрощення процесу створення сайтів для електронної комерції (продажів в інтернеті) і як результат створення безмежної кількості платформ для їх створення. Як показала практика, дані сотень досліджень і тестів на фокус групах всі вони мають однакову особливість, основним фактором утримання користувача на сайті та спонуканням його до покупки є гарний дизайн, швидка робота сайту і якісний фото та медіа контент, який використовується на сайті. Без якісних фото товару, жоден користувач не зверне на нього уваги.

Всі популярні платформи електронної комерції мають можливість додавання фото і відео продукту, але дуже мало з них мають зручний спосіб для масового завантаження контенту, в результаті, це стає великою проблемою для організацій, які стикаються з ситуацією, коли потрібно одночасно завантажити велику кількість

					ІІЗ.КР.Б – 121 – 23 – ІІЗ	Лист
Зм.	Лист	№ документу	Підпис	Дата		8

фото і розмістити їх в правильному місці, в правильному порядку і з правильними ролями для кожної одиниці контенту.

Мета даної роботи полягає в розробці стороннього модулю для імпорту медіа-контенту на базі однієї з самих популярних платформ електронної комерції Magento 2 засобами PHP, HTML, CSS, XML та JS. Сторонній модуль має забезпечувати можливість імпортування медіа-контенту в зручний спосіб, як для програміста або системного адміністратора, який обслуговує сайт і має розширений доступ до файлів платформи, так і для замовника, який має обмежений доступ до файлів платформи. Також метою є поглиблення теоретичних і практичних знань та формування вмінь і навиків, необхідних для розроблення стороннього модулю для платформи електронної комерції Magento 2.

Поставлена мета обумовлює необхідність виконання наступних завдань:

- аналіз теоретичних та функціональних потреб проектування для реалізації стороннього модулю;
- аналіз існуючих та вже представлених методів імпортування медіа-контенту, визначення унікальності;
- формування технічного завдання;
- визначення технологій та об'єктної структури стороннього модулю;
- розробка стороннього модулю.

Об'єктом дослідження є методи та засоби розробки стороннього модулю для платформи електронної комерції Magento 2.

Предметом дослідження є технології, структури, методи та засоби розробки та взаємодії сторонніх модулів та платформи електронної комерції Magento 2, а також використанні необхідні веб-орієнтовані технології.

В роботі було використано методи об'єктно орієнтованого програмування та веб програмування.

Розроблений сторонній модуль може бути використаний як практична частина будь-якого інтернет магазину побудованого на платформі Magento 2 з можливою подальшою модернізацією за допомогою, вбудованих в саму платформу, методів розширення та заміни функціональних частин.

					ІІЗ.КР.Б – 121 – 23 – ІІЗ	Лист
						9
Зм.	Лист	№ документа	Підпис	Дата		

Записка випускної кваліфікаційної роботи складається з вступу, 3-х розділів, висновків, списку використаних джерел із 9 найменувань (1 сторінка), 1 додатку (9 сторінок). Повний обсяг роботи становить 66 сторінок, з них 55 сторінки основного тексту.

					ПЗ.КР.Б – 121 – 23 – ПЗ	Лист
Зм.	Лист	№ документу	Підпис	Дата		10

РОЗДІЛ 1. АНАЛІЗ ТЕОРЕТИЧНИХ ТА ФУНКЦІОНАЛЬНИХ ПОТРЕБ ПРОЕКТУВАННЯ ДЛЯ РЕАЛІЗАЦІЇ СТОРОННЬОГО МОДУЛЮ

1.1. Опис проекту та постановка задачі

На сьогоднішній день практично кожна комерційна організація має власний інтернет-магазин. В умовах сучасного світу це є невід'ємною частиною здорового і конкурентоспроможного бізнесу, який дозволяє розширити клієнтську базу та поле для маркетингової діяльності. Інтернет-магазини можуть бути різних типів, приклад деяких з них:

- сторінкою в соціальній мережі;
- частиною загальновідомого маркетплейсу;
- окремим і незалежний сайтом, побудованим на власноруч написаному або готову фреймворку, що дозволяє зекономити кошти на розробку, та вийти за функціональні межі готових платформ;
- окремим і незалежним сайтом, побудованим на популярній платформі електронної комерції, що дозволяє отримувати оновлення безпеки та функціональні нововведення по системі релізів.

В незалежності від типу інтернет-магазину одним із найважливіших критеріїв для приваблення нових клієнтів, та утримання їх в магазині є якісний медіа-контент, на який користувачу приємно дивитися, і з яким користувач може асоціювати себе. Завдяки якісному медіа-контенту рівень комерційного успіху бізнесу зростає, що в свою чергу дозволяє зміцнювати позиції на ринку і розвивати та масштабувати сам бізнес. Практика показує, що магазин, який має гарне оформлення, якісний медіа-контент та у якого правильно налаштована таргет-реклама, живий трафік з пошукових систем збільшується, а при правильній і лояльній роботі маркетингового відділу, збільшується в рази. Бізнеси витрачаються величезні кошти на якісний та правильний, з естетичної точки зору, медіа-контент, який використовується в усіх галузях пов'язаних з інтернет-магазином, вони наймають кращих фотографів та 3D художників, заключають договори з лідерами думок, моделями та відомими людьми на амбасадорство бренду і тому подібне. В

					ІІЗ.КР.Б – 121 – 23 – ІІЗ	Лист
Зм.	Лист	№ документа	Підпис	Дата		11

результаті цієї титанічної роботи, людина, або компанія людей, відповідальна за медіа складову бізнесу, мають велику кількість матеріалу, який потрібно обробити і використати в максимальній кількості місць, для виправдання витрачених ресурсів і часу.

Нижче приведено декілька прикладів інтернет-магазинів, з якісним медіа-контентом (рис. 1.1).

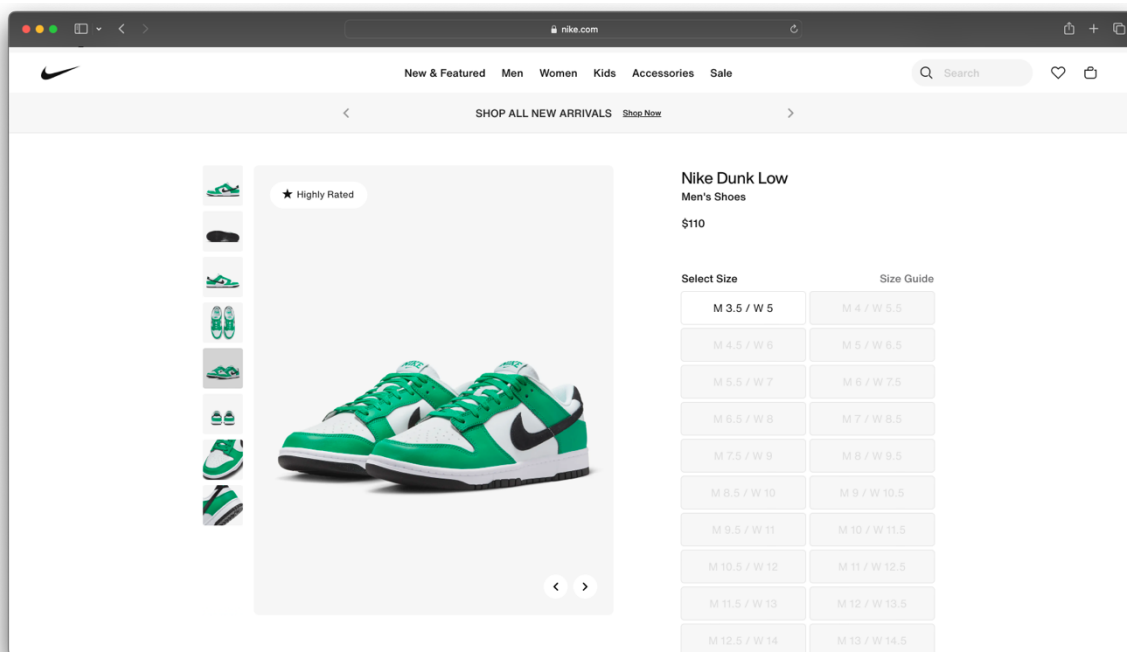


Рис 1.1. Приклад магазину з якісним медіа-контентом nike.com

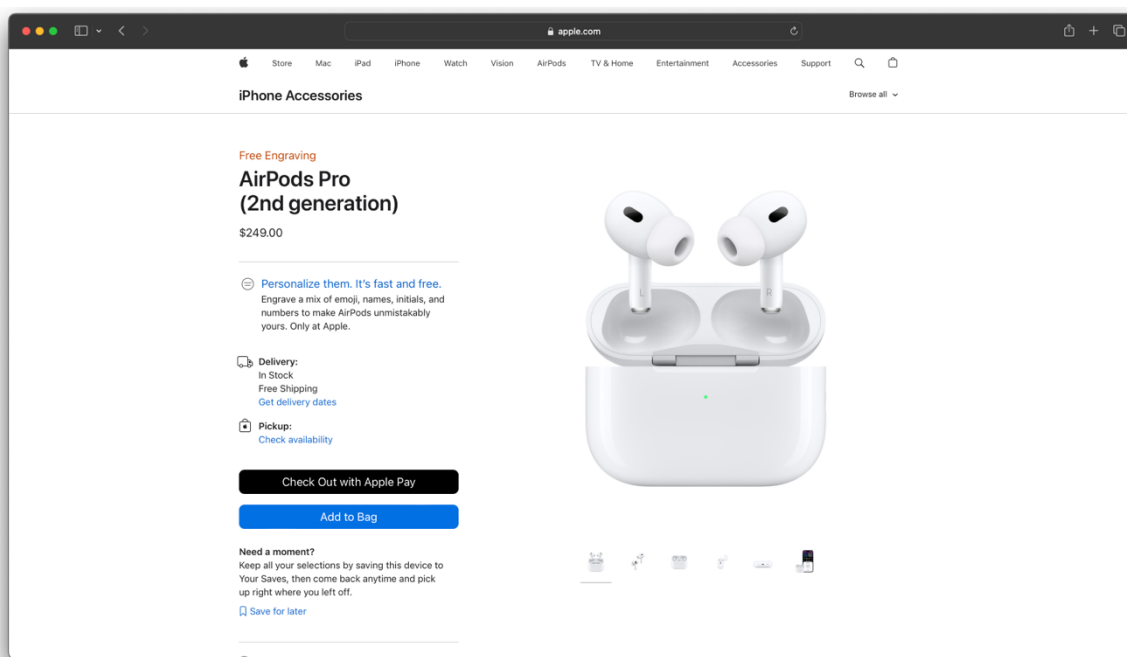


Рис 1.2. Приклад магазину з якісним медіа-контентом apple.com

Для порівняння знизу наведено приклад інтернет-магазину зі спірним медіа-контентом (рис. 1.3).

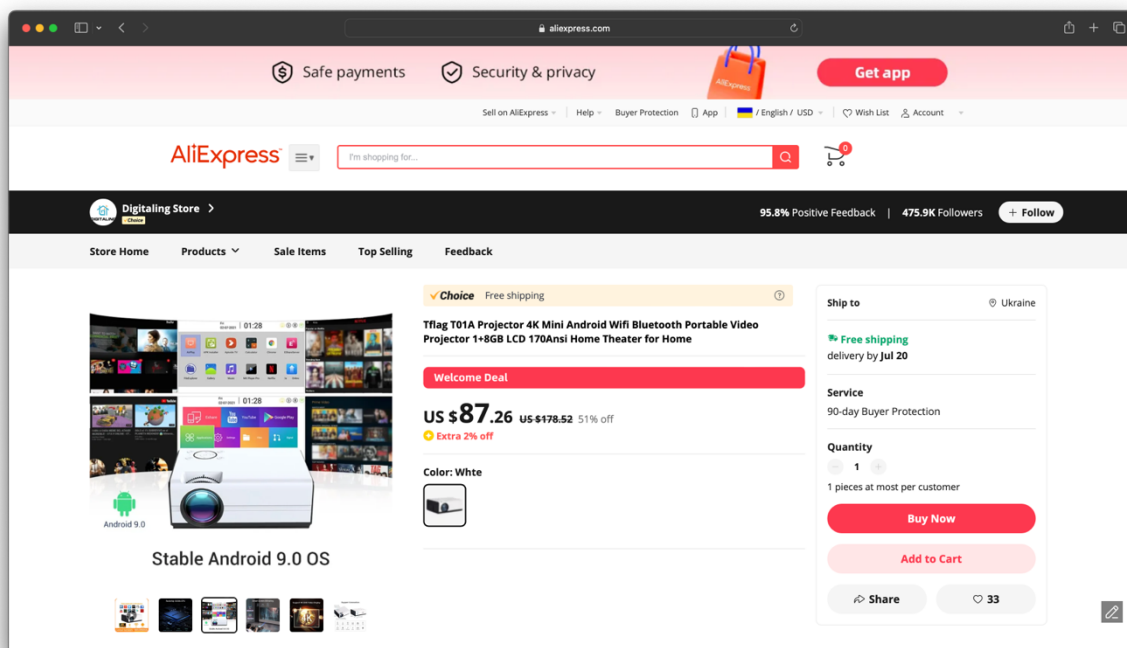


Рис 1.3. Приклад магазину зі спірним медіа-контентом aliexpress.com

Аналізуючи і порівнюючи відношення користувачів до інтернет-магазинів з різним по якості контентом, майже всі бізнеси дійшли до висновку, що якісні ілюстрації та фотографії продукту позитивно впливають на продажі продуктів і на репутацію бренду в цілому.

Саме тому виникає ситуація, коли бренду або бізнесу, випускаючи нову лінійку продуктів, або оновлюючи чи замінюючи контент для вже існуючих продуктів, потрібно завантажити велику кількість нового медіа-контенту. Тут і з'являється питання зручності і швидкості його завантаження на платформу для подальших дій і ознайомлення майбутніх клієнтів з продуктом.

1.2. Аналіз вже представлених методів імпортування медіа-контенту, визначення унікальності

Magento 2, як одна з найпопулярніших платформ для створення сайтів електронної комерції в своєму арсеналі має два методи завантаження зображень та ілюстрацій.

Першим і основним з них є завантаження зображень відразу на рівні продукту. Це дозволяє візуально і зручно додати зображення, розмістити їх в правильному порядку, приховати і назначити ролі для зображення. Ролі, які можуть бути використані для зображення, в базовій версії наступні:

- Thumbnail – маленька картинка, яка відображається на сторінці кошику та сторінці оформлення замовлення в правій частині, де знаходиться підсумок замовлення;
- Small Image – маленька картинка, яка відображається на PDP, на сторінці результатів пошуку, а також в Cross-sell та Up-sell блоках;
- Base Image – базова, або основна картинка, яка відображається на PDP;
- Swatch – може бути використано для ілюстрації візерунку, кольору або текстури.

Використати даний метод завантаження медіа-контенту може людина, яка має обліковий запис до адміністраторської частини сайту (Admin panel) з відповідними правами і виглядає це наступним чином.

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Лист
Зм.	Лист	№ документа	Підпис	Дата		14

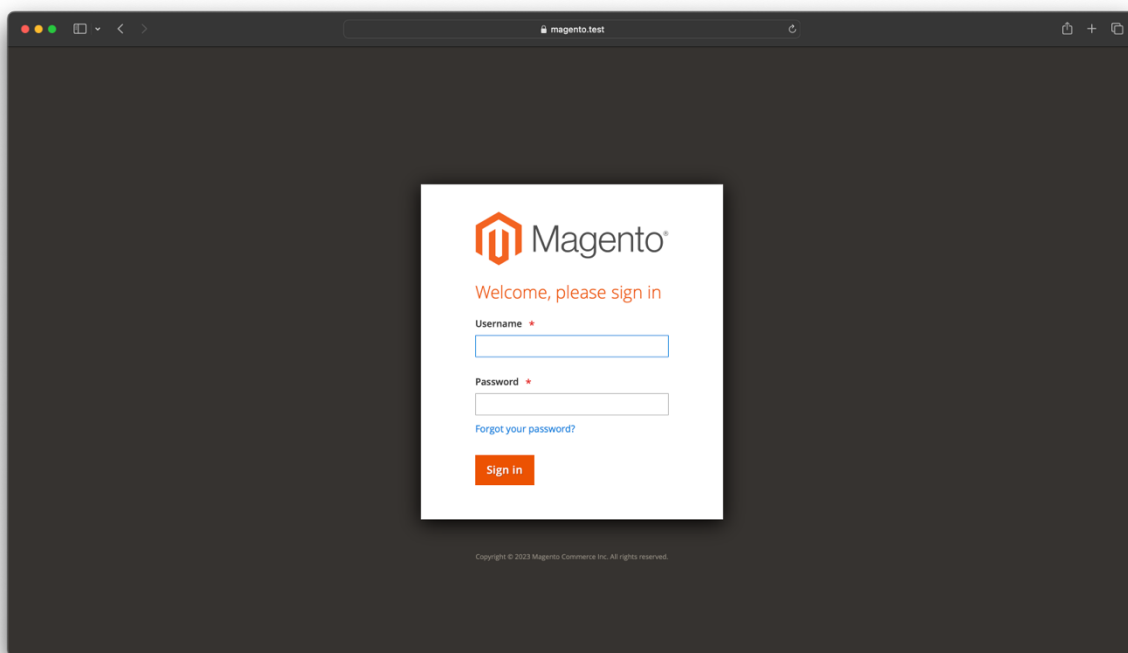


Рис 1.4. Крок 1: Сторінка входу до адміністраторської частини сайту

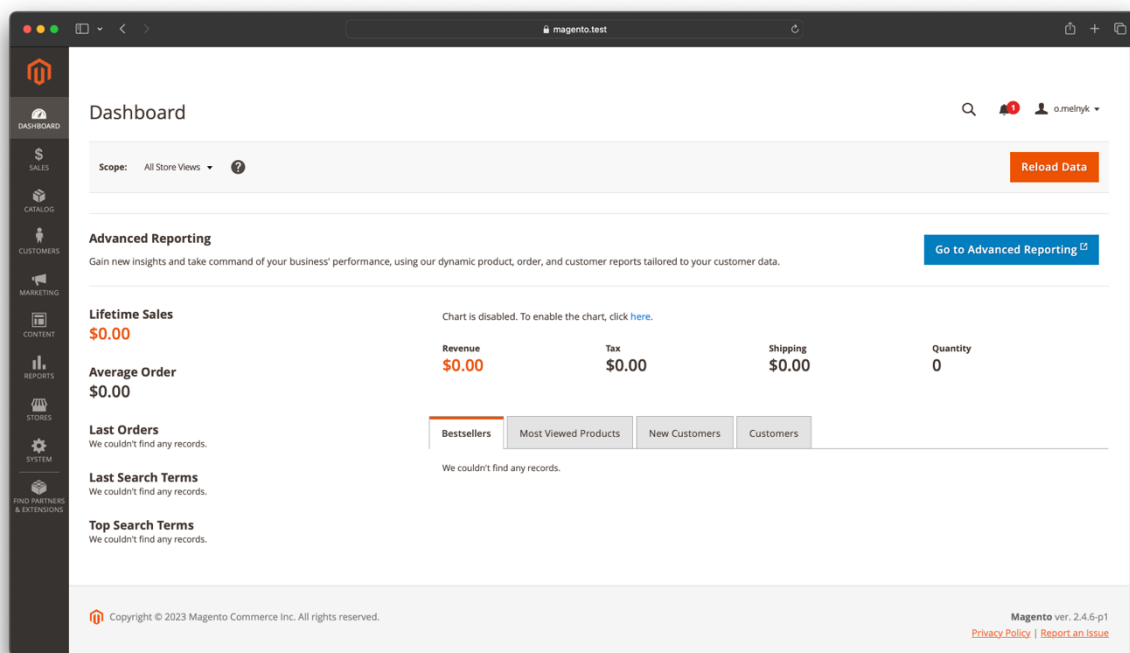


Рис 1.5. Крок 2: Відображення головної сторінки адміністраторської частини сайту після успішної авторизації

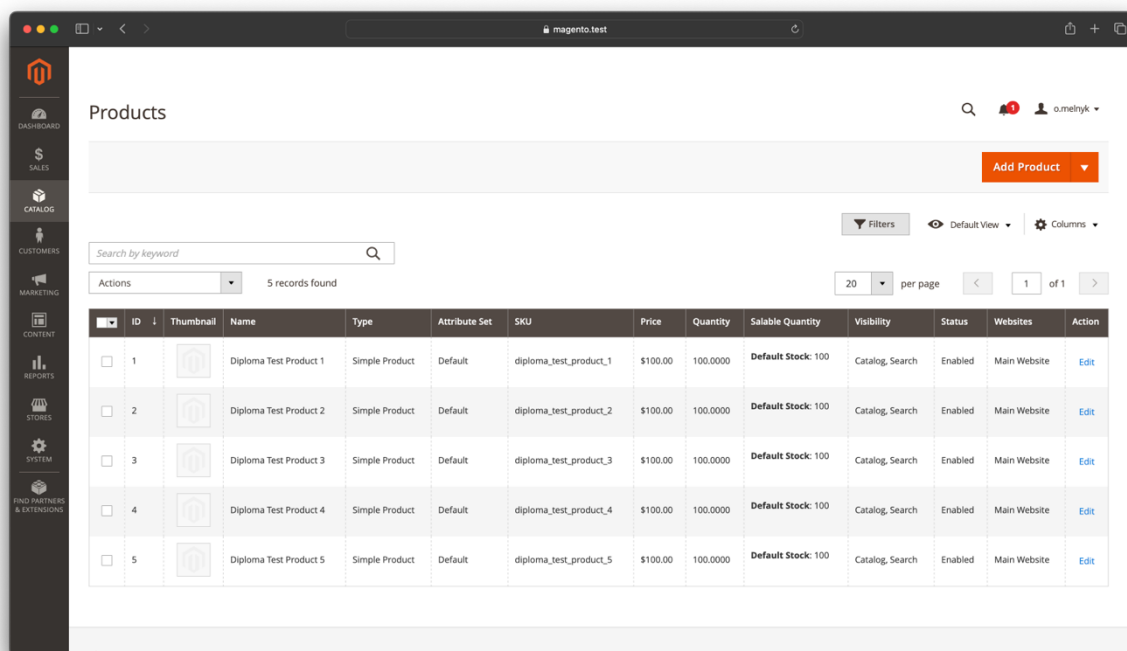


Рис 1.6. Крок 3: Відкриваємо сторінку управління всіма продуктами

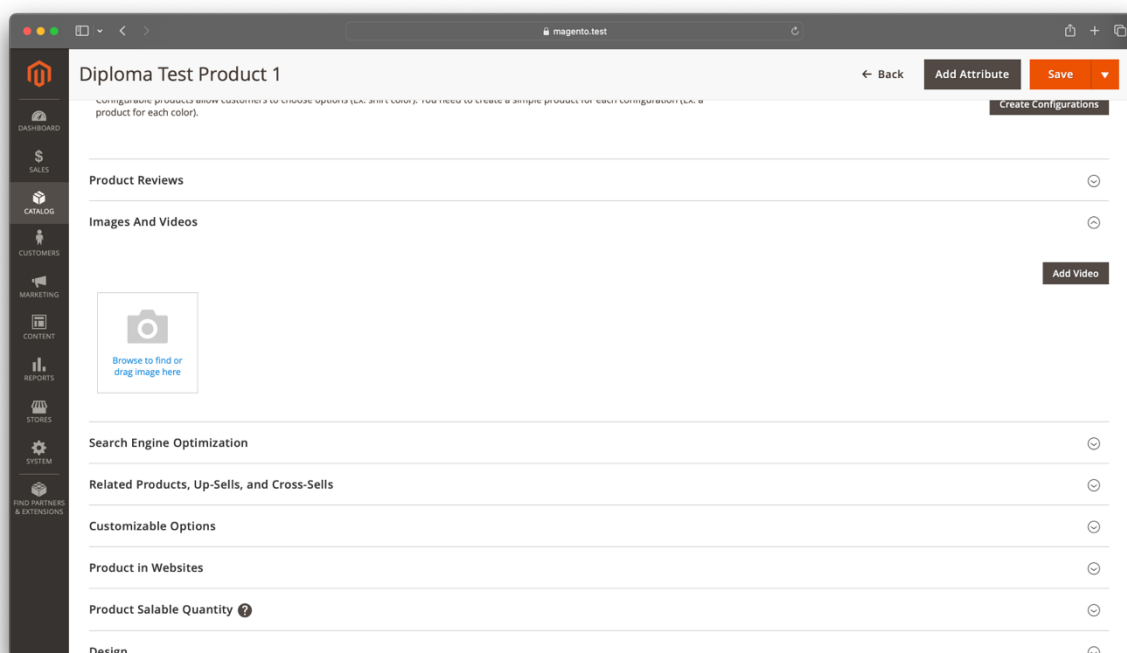


Рис 1.7. Крок 4: Відкриваємо сторінку управління продуктом, секція
Зображень та Відео

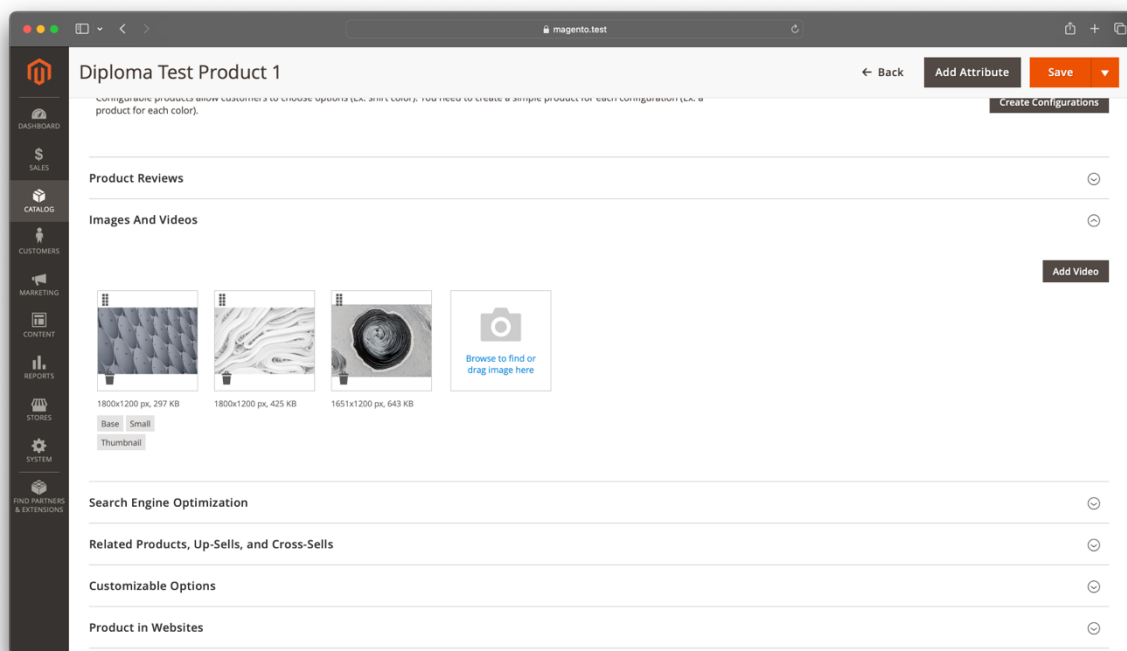


Рис 1.8. Крок 5: Завантажуємо зображення

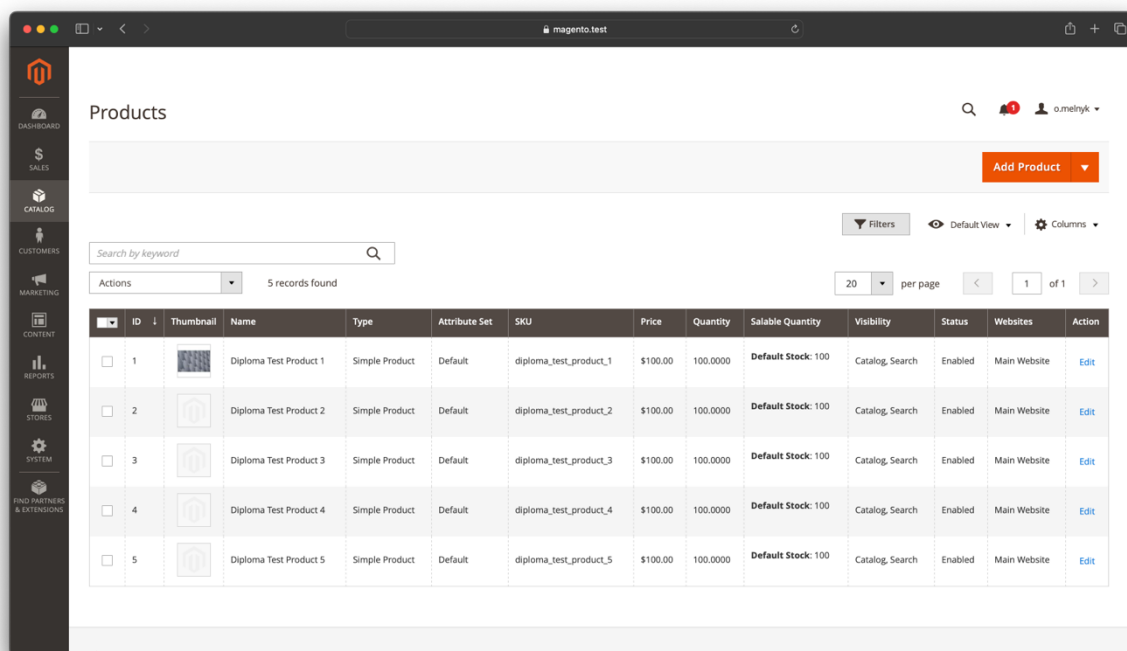


Рис 1.9. Крок 6: Перевіряємо результат після збереження продукту на сторінці управління всіма продуктами

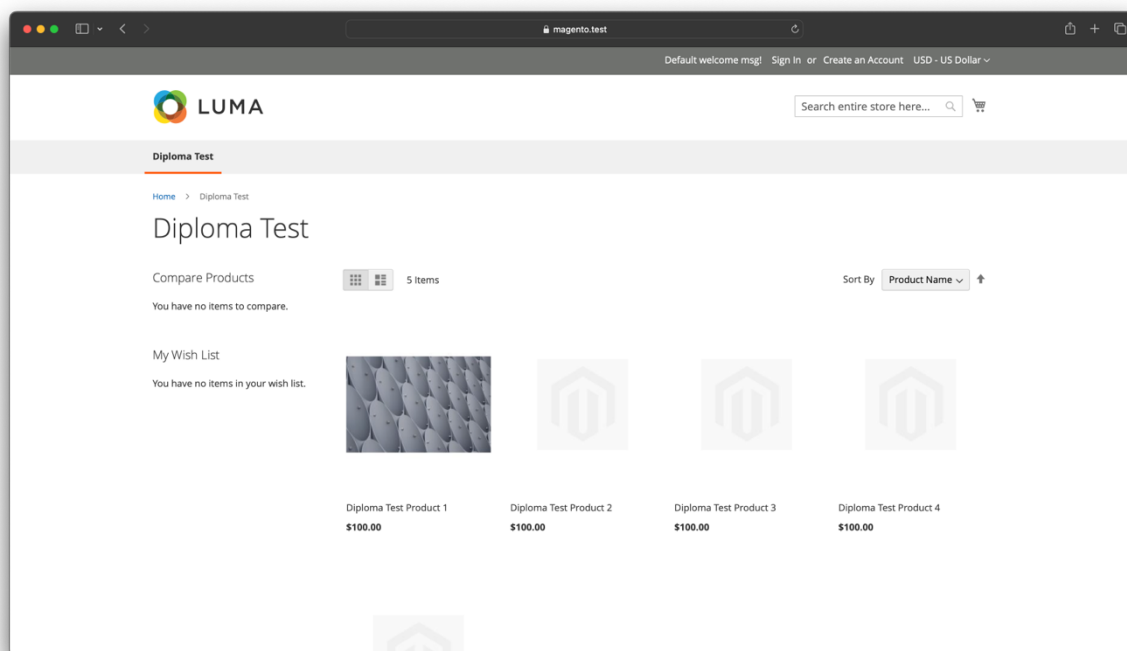


Рис 1.10. Крок 7: Перевіряємо результат після збереження продукту на PLP

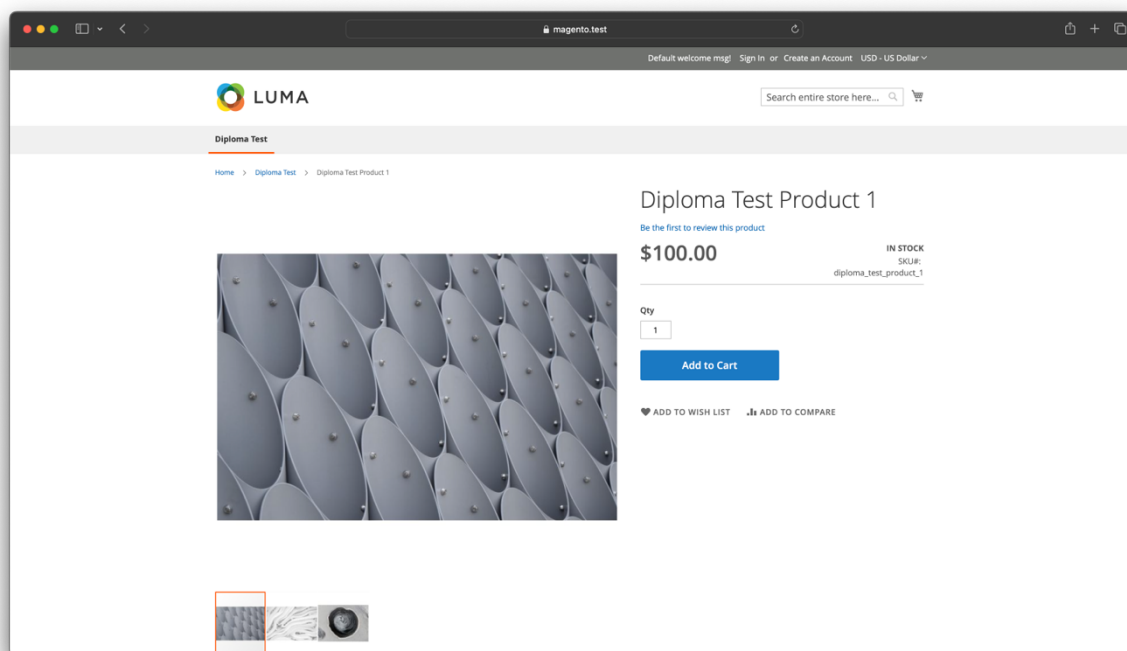


Рис 1.11. Крок 8: Перевіряємо результат збереження продукту на PDP

Як видно з наведеного прикладу, цей метод імпортування медіа-контенту є досить простим і дуже візуальним, що дозволяє людині, якій не відома ця платформа, дуже швидко розібратися і виконати поставлене завдання. Даний метод імпортування ідеально підходить для випадків, коли необхідно додати або оновити зображення для декількох продуктів, але якщо продуктів буде багато, це

займе велику кількість часу, і саме на цей випадок, у Magento 2 є інший метод імпортування медіа-файлів, а саме завантаження зображень через механізм імпортування продуктів.

Метод через механізм імпортування продуктів, не є візуальним і зрозумілим для пересічного користувача, людина яка виконує це завдання має мати доступ не тільки до адміністраторської частини сайту, а і до хостингу самого сайту через SSH або FTP/SFTP для завантаження зображень на сервер, для подальшої роботи механізму [2,3]. До такої задачі не допускають людей з малим досвідом роботи і завжди про таку роботу повідомляють команду розробників для усунення будь яких проблем, які можуть виникнути у разі потрапляння неправильної або некоректної інформації в сам механізм імпортування. Виглядає це наступним чином.

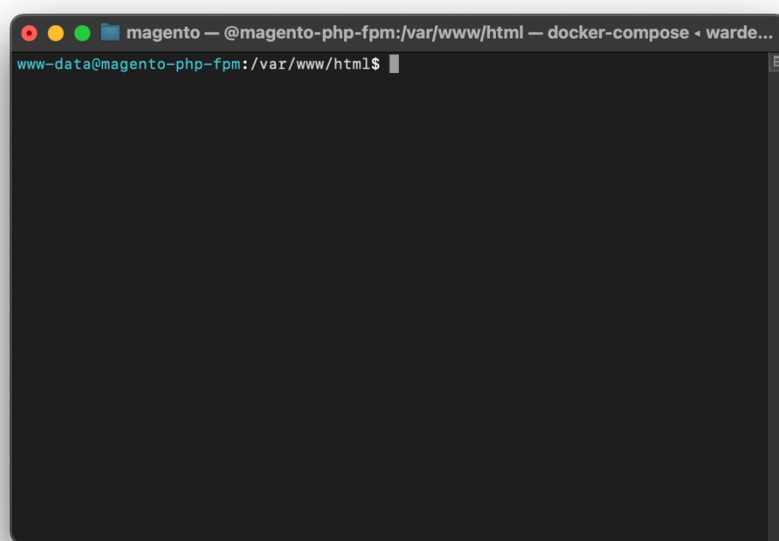


Рис 1.12. Крок 1: Підключення по SSH до серверу магазину

```
magento — @magento-php-fpm:/var/www/html — docker-compose • warde...
www-data@magento-php-fpm:/var/www/html$ cd var/import/images
bash: cd: var/import/images: No such file or directory
www-data@magento-php-fpm:/var/www/html$
```

Рис 1.13. Крок 2: Переходимо до директорії імпорту і отримуємо помилку, так як її ще не існує

```
magento — @magento-php-fpm:/var/www/html/var/import/images/diploma...
www-data@magento-php-fpm:/var/www/html$ mkdir -p var/import/images/diploma_test
www-data@magento-php-fpm:/var/www/html$ cd var/import/images/diploma_test
www-data@magento-php-fpm:/var/www/html/var/import/images/diploma_test$
```

Рис 1.14. Крок 3: Створюємо директорію імпорту і папку для наших зображень

```

magento — @magento-php-fpm:/var/www/html/var/import/images/diploma...
www-data@magento-php-fpm:/var/www/html/var/import/images/diploma_test$ ls
diploma_test_product_2__1__image_small_image.jpg  diploma_test_product_2__3.jpg
diploma_test_product_2__2__thumbnail.jpg
www-data@magento-php-fpm:/var/www/html/var/import/images/diploma_test$

```

Рис 1.15. Крок 4: Переміщуємо зображення в новостворену папку

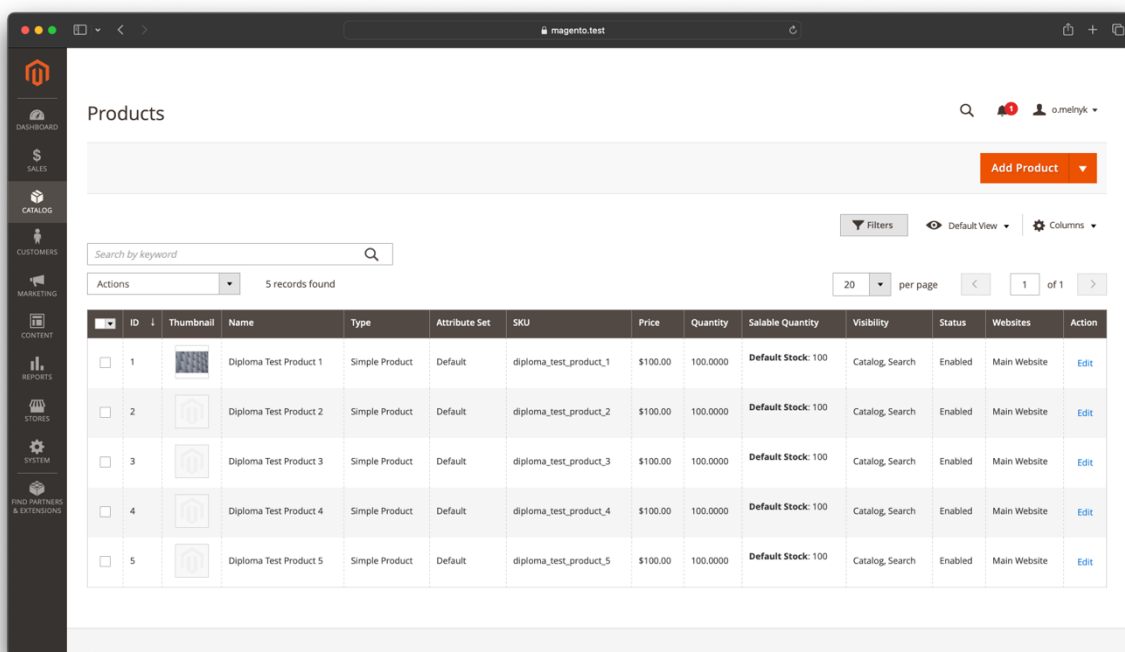


Рис 1.16. Крок 5: Відкриваємо сторінку управління всіма продуктами в адміністративній панелі для отримання SKU продукту

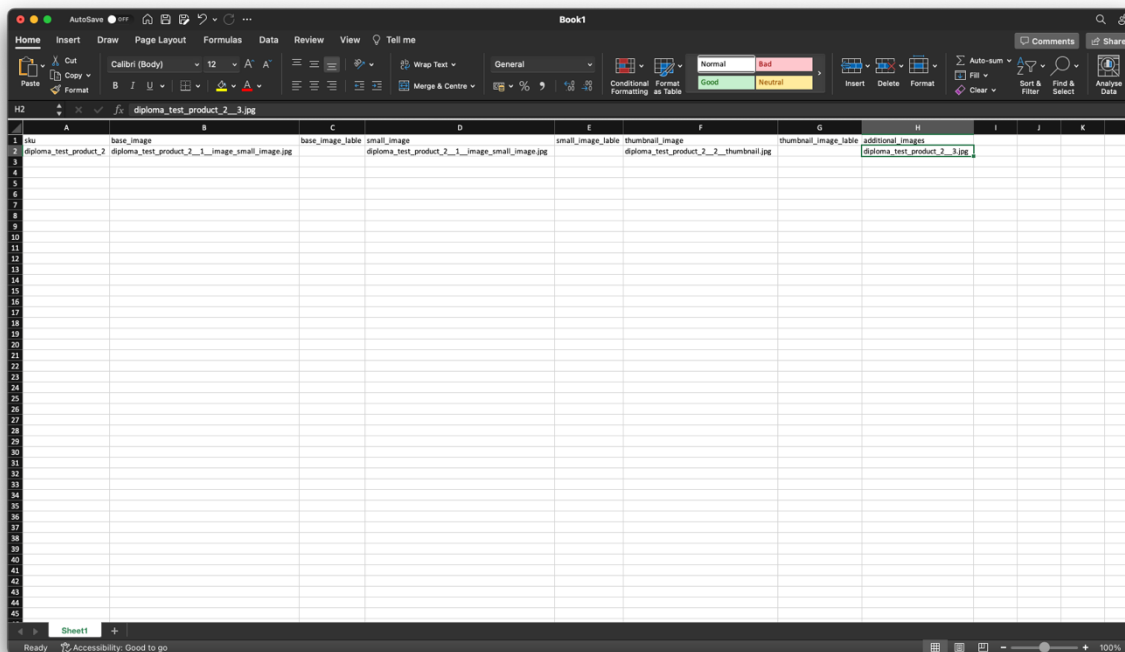


Рис 1.17. Крок 6: Створюємо файл для імпорту в зручній для цього програмі

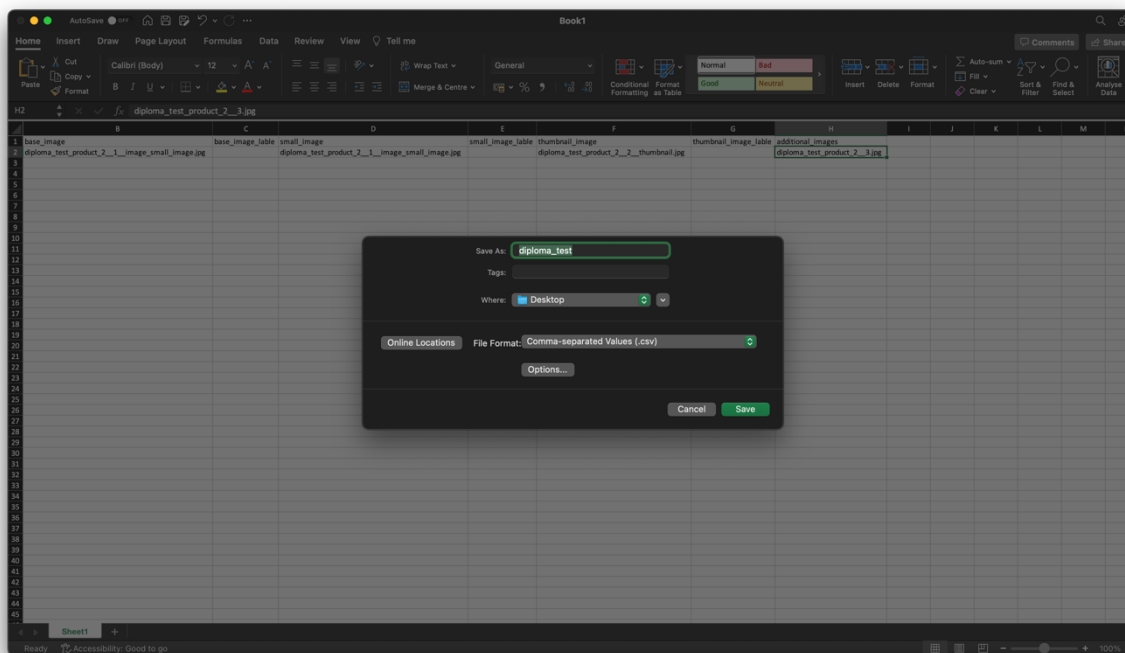


Рис 1.18. Крок 7: Зберігаємо файл в CSV форматі

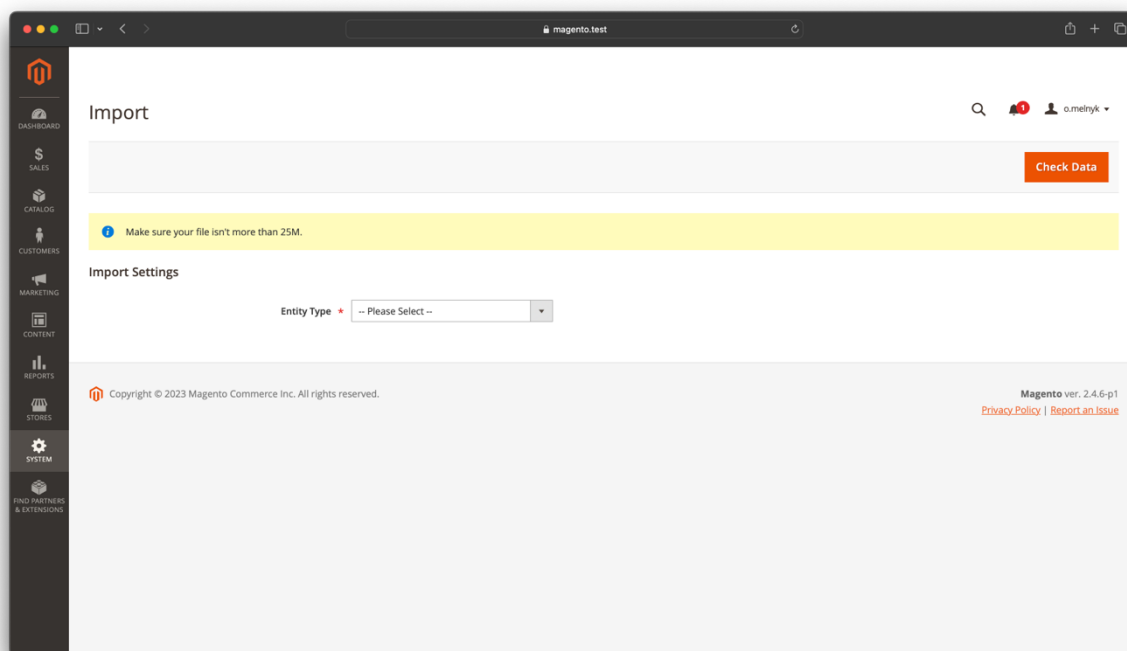


Рис 1.19. Крок 8: Відкриваємо сторінку імпорту в адміністративній панелі

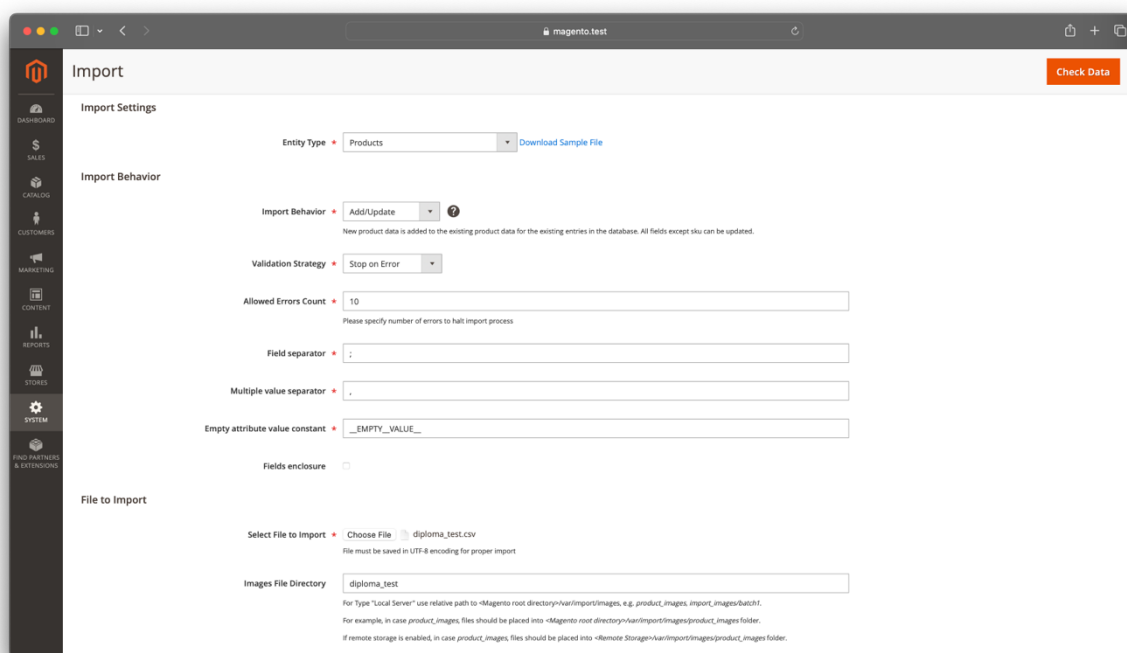


Рис 1.20. Крок 9: Налаштовуємо імпорт під новостворений файл і прописуємо папку, яку ми створили в директорії імпорту зображень

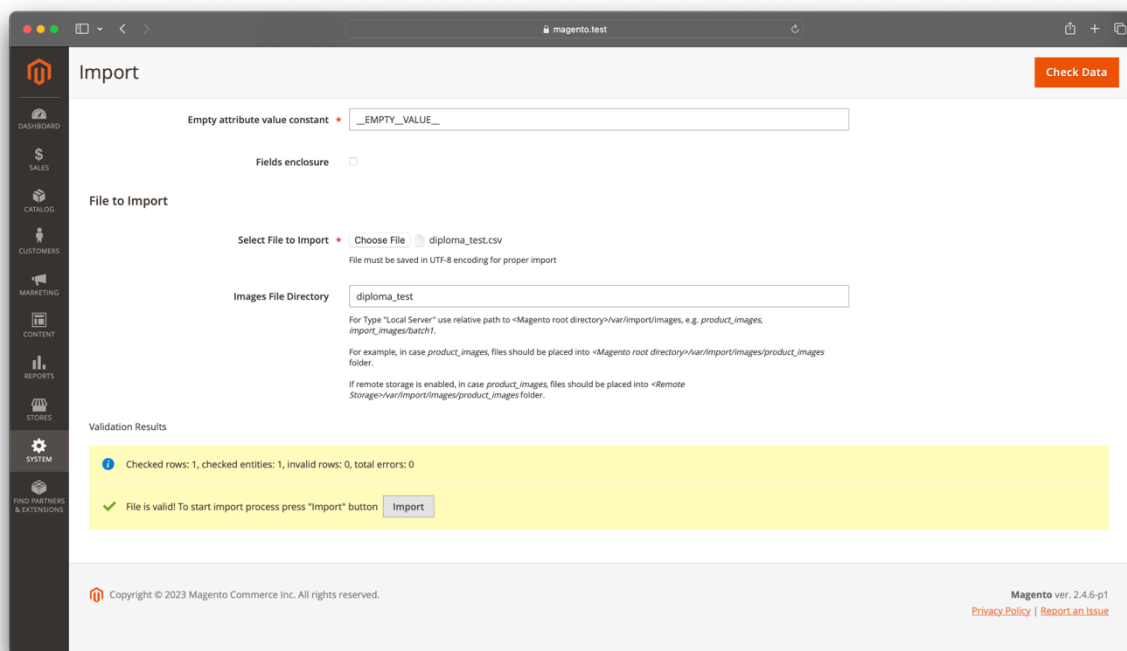


Рис 1.21. Крок 10: Натискаємо кнопку «Check Data» (перевірка даних) і отримуємо звіт перевірки; в нашому випадку все добре

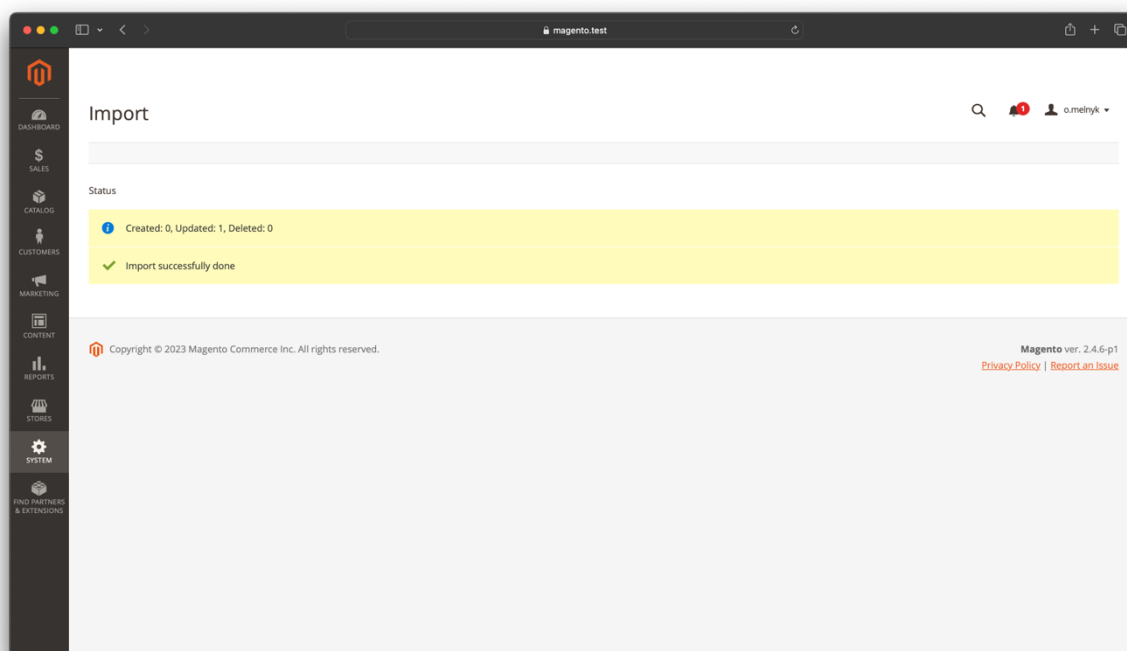


Рис 1.22. Крок 11: Натискаємо кнопку «Import» (імпорт) для імпортування зображень і бачимо повідомлення що імпорт було завершено успішно, 1 продукт був оновлений

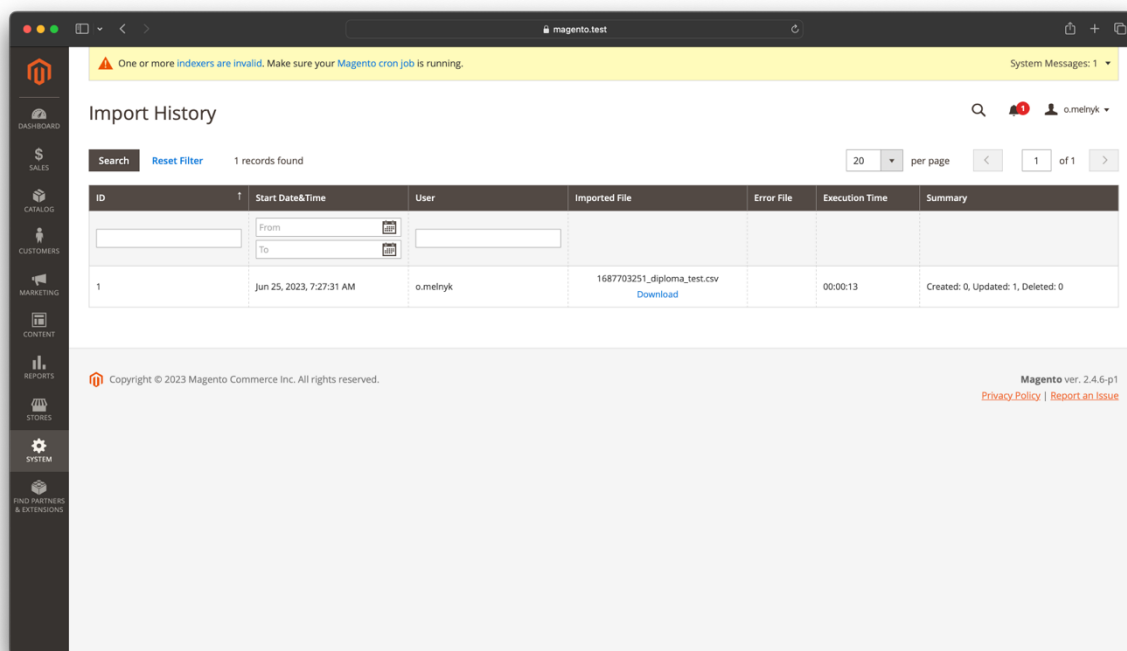


Рис 1.23. Крок 12: Перевіряємо результат імпортування на сторінці «Import History» (історії імпорту)

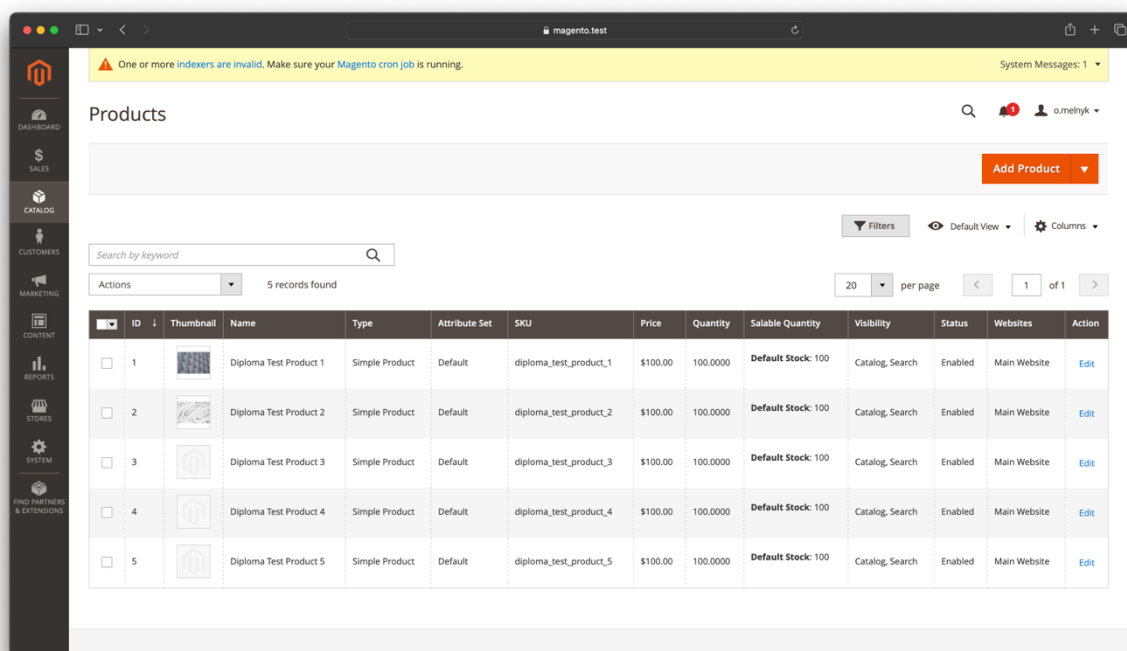


Рис 1.24. Крок 13: Перевіряємо результат імпортування на сторінці управління всіма продуктами

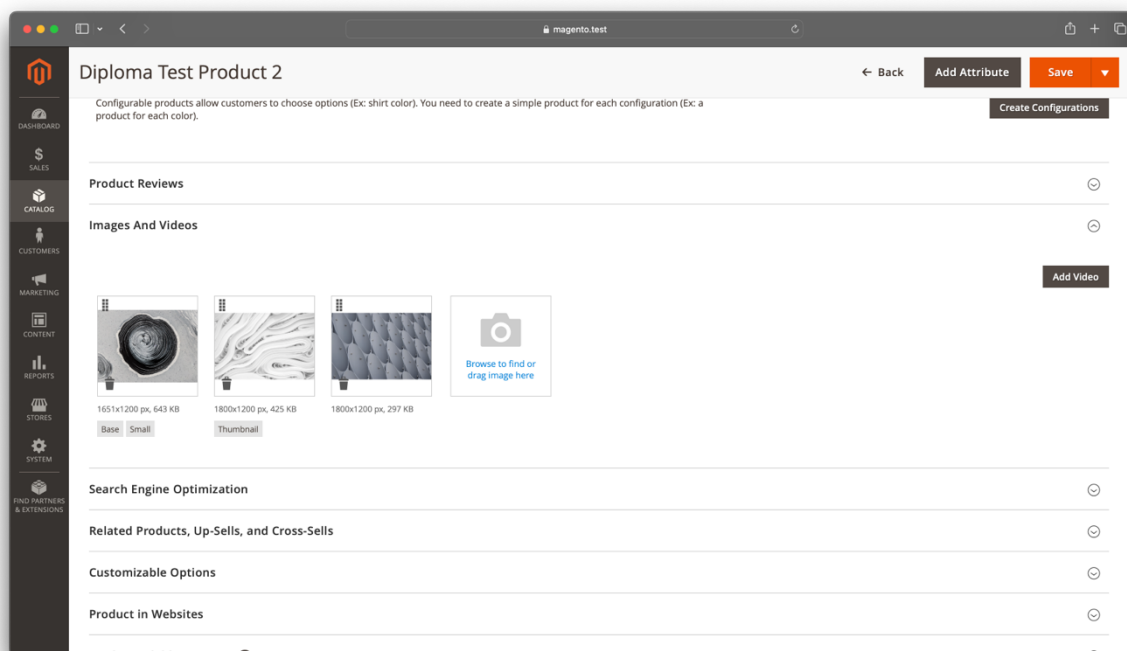


Рис 1.25. Крок 14: Перевіряємо результат імпортування на сторінці управління продуктом, приділяємо особливу увагу ролям

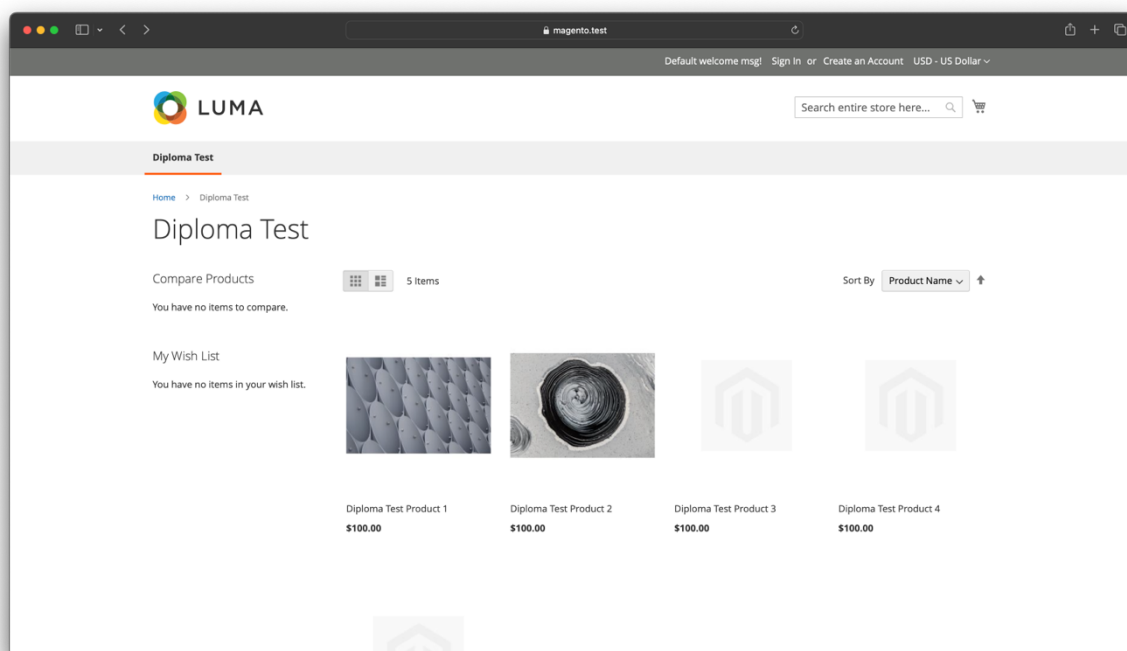


Рис 1.26. Крок 15: Перевіряємо результат імпортування на PLP

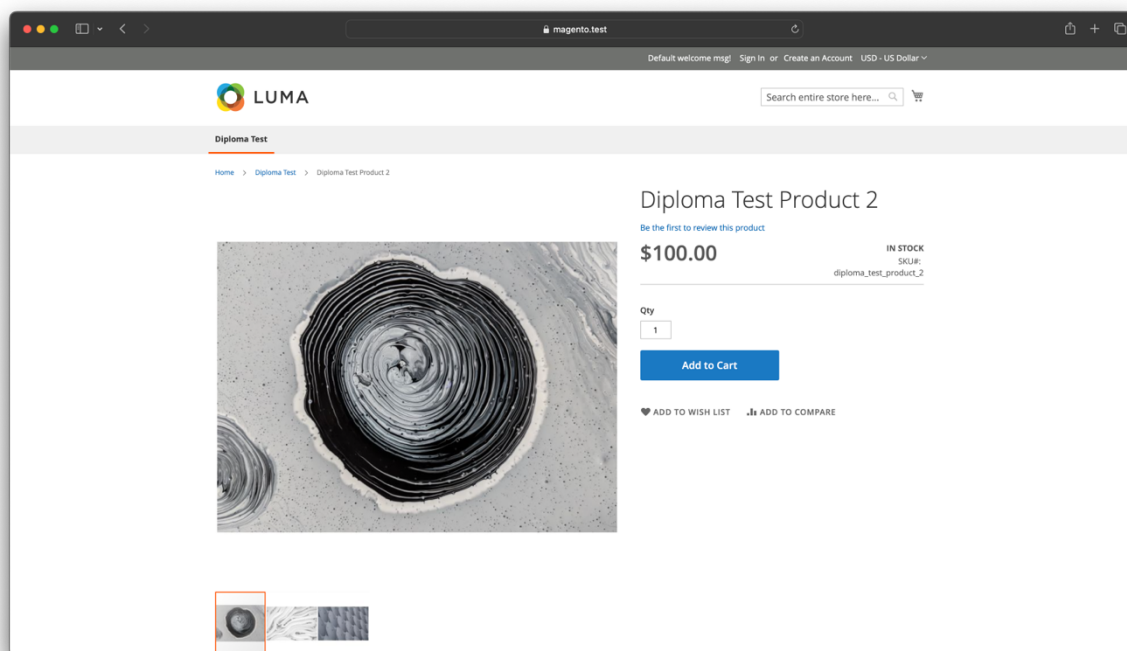


Рис 1.27. Крок 16: Перевіряємо результат імпортування на PDP

Як і було описано до цього, з цього прикладу стає зрозуміло що цей метод імпортування зображень є максимально технічним і важким для пересічної людини, в ньому значно більше кроків і він потребує більше уваги і прискіпливості, так як все відбувається не за допомогою графічного інтерфейсу, а за допомогою механізму імпортування.

В таблиці 1.1 наведено порівняння двох методів імпортування

Таблиця 1.1

Порівняльна таблиця

Назва	Простота використання	Можливість вказання позиції	Можливість вказання ролі	Можливість імпортувати для декількох продуктів відразу
Завантаження через сторінку управління продуктом	+	+	+	–
Імпорт через механізм імпортування продуктів	–	–	+	+

Як ми бачимо з порівняльної таблиці, жоден з цих методів не є ідеальним, і, насправді, вони потребують один одного для правильного і точного імпортування зображень, як того потребує клієнт, тому що, наприклад, не завжди базовим зображенням є перше зображення і тому подібне.

1.3. Формування технічного завдання

Саме в результаті такого міркування, до нас в компанію звернувся наш замовник і запросив створення зручного методу імпортування зображень, яке мало би всі, зазначені в порівняльній таблиці 1.1, характеристики. Після обміркування можливих варіантів рішення, йому було запропоновано створення стороннього модулю, який матиме просте використання, як для технічної людини, так і для звичайної пересічної людини, зможе запропонувати всі функціональні можливості і матиме можливість розширення в майбутньому.

Після фази обговорення технічних моментів з замовником, було узгоджено наступне технічне завдання:

- Створення стороннього модулю, який додає ще один метод імпортування зображень продуктів;
- Новий метод імпортування має бути простим у використанні, мати можливість вказання позиції зображення і ролі зображення, та мати можливість імпортувати зображення в декілька продуктів за одне виконання;
- Сторонній модуль має бути написаний в урахуванням можливого розширення функціоналу в майбутньому, та бути до нього адаптованим;
- Сторонній модуль має повністю відповідати код стандартам платформи Magento 2.

Висновки до першого розділу

В першому розділі було розглянуто та проаналізовано важливість якісного медіа-контенту в інтернет-магазинах, а також було розглянуто і порівняно методи імпортування вже вбудовані в платформу електронної комерції Magento 2, а саме завантаження зображень через сторінку управління продуктом та імпорт через механізм імпортування продуктів, а також визначено сильні і слабкі сторони кожного з них. Після цього було сформовано технічне завдання для стороннього модулю, яке відповідає меті і завданню кваліфікаційної роботи.

					ПЗ.КР.Б – 121 – 23 – ПЗ	Лист
Зм.	Лист	№ документа	Підпис	Дата		29

РОЗДІЛ 2. ВИЗНАЧЕННЯ ТЕХНОЛОГІЙ ТА ОБ'ЄКТНОЇ СТРУКТУРИ СТОРОННЬОГО МОДУЛЮ

2.1. Визначення варіантів використання та об'єктної структури стороннього модулю

Використовувати функціонал модулю буде лише два типи користувачів:

- замовник, або людина зі сторони замовника, від яких не потребуються технічні навички;
- технічна людина зі сторони команди.

В свою чергу варіантів використання буде три:

- запуск імпортування через панель адміністрування;
- запуск імпортування через команду в терміналі;
- відкладений запуск імпортування через Cronjob.

Модуль не має обмежень по ролям, але так як замовнику не рекомендується нічого робити на хостингу проекту, його варіанти використання менші на один пункт ніж у людини зі сторони команди, на рисунку знизу візуалізовано загальну діаграму варіантів використання

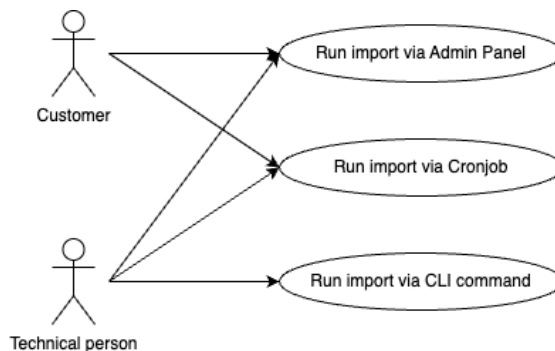


Рис 2.1. Діаграма варіантів використання

Для визначення об'єктної структури необхідно зрозуміти одну просту річ, платформа електронної комерції Magento 2 складається з модулів, всі вони мають однакову структуру і один код стандарт. Всі сторонні модулі, які можна інсталиювати, розповсюджуються в два основних способи:

- Composer пакетами;
- Архівами, файли з яких переміщуються в app/code папку проекту;

Базова структура модулю складається з наступних файлів:

- etc/module.xml;
- registration.php.

Саме після створення цих файлів і виконання команди `php bin/magento setup:upgrade` в конфігураційний файл, який знаходиться в проекті в папці `app/etc/config.php` буде додано запис про новий модуль і про те що він був увімкнений.

Офіційно, після цього, можна сказати що модуль було створено, але в ньому немає ніякою сторонньої логіки, він нічого не змінює і не додає, саме тому для правильного написання свого функціоналу потрібно розуміти, приймати і використовувати затверджену архітектуру модулю наведену в таблиці 2.1 [4].

Таблиця 2.1

Архітектура стороннього модулю Magento 2

Api	Зберігає будь-які PHP інтерфейси, які використовуються як API, по правилам і код стандартам, тут зберігаються інтерфейси моделей, або як їх називаються в професійному світі, сервісні контракти, на основі яких будуються класи і взаємодія між ними, по визначеним в інтерфейсі методам.
Block	Містить в собі PHP класи, які є частиною MVC патерну і використовуються в процесі відображення інформації на сторінку
Cron	Містить в собі PHP класи, які використовуються для виконання корисного функціоналу періодично за допомогою Cronjob процесу
CustomerData	Містить в собі PHP класи, які використовуються для надання сенситивної інформації про користувача так його даних в спеціальну секцію сайту яка не зберігається ніде окрім сесії назва якої Customer Data

Controller	Містить в собі PHP класи, які є частиною MVC патерну і використовуються в процесі роутінгу
Console	Містить в собі PHP класи, які використовуються для створення консольних команд
etc	Містить в собі конфігураційні файли модулю. Всередині цієї папки дозволяється створення папок для уточнення кордонів, в межах яких конфігураційні файли будуть виконувати свою функцію такі як: adminhtml, frontend, webapi_rest, webapi_soap
Helper	Містить в собі допоміжні PHP класи, які виконують корисний функціонал який не є частиною бізнес-логіки
i18n	Містить в собі файли локалізації
Model	Містить в собі PHP класи, які є частиною MVC патерну і використовуються для надання інформації, а також є місцем розташування бізнес-логіки
ViewModel	Містить в собі PHP класи, які є частиною MVVM патерну і використовуються для надання інформації відразу в найнижчий рівень гілки View, а саме в темплейти
Plugin	Містить в собі PHP класи, які є об'єктом DI і використовуються для модифікації або доповнення вже написаного функціоналу за допомогою трьох основних варіантів використання: beforeMethod, afterMethod або aroundMethod
Observer	Містить в собі PHP класи, які виконують свій функціонал тільки після виклику події, на яку вони підписані, менеджером подій
Preference	Містить в собі PHP класи, які є об'єктом DI і використовуються для модифікації або доповнення вже написаного функціоналу за допомогою повного

	переписування вже існуючого функціоналу. Цей метод розширення або модифікації коду використовують у випадках коли плагін написати немає можливості, наприклад коли функція не є public
UI	Містить в собі PHP класи, які використовуються для надання і формування інформації в UI компоненти, які в свою чергу є лівовою частиною адміністраторської частини сайту(динамічні таблиці і форми)
Setup	Містить в собі PHP класи, які використовуються для визначення структури бази даних та для додавання або редагування інформації в базу даних
view	Містить в собі view файли, включаючи статичні view файли, темплейти дизайну, темплейти електронних листів і layout файли

Саме на основі цієї архітектури написані абсолютно всі модулі, які існують для цієї платформи, але треба розуміти, що завдання поставлені перед сторонніми модулями бувають різними, тому це саме основа, а не єдина можлива архітектура. Magento 2 дозволяє розміщувати файли в майже будь якому варіанті, але загально прийнятий код стандарт один, і у випадку публікації свого модулю на офіційному маркетплейсі з модулями, адміністрація ресурсу не допустить критичного відходження, від загально прийнятого стандарту.

В нашому випадку ми створюємо додатковий метод імпортування зображень продуктів, тому приблизна структура нашого модулю буде наступною:

- Block
- Console
- Controller
- Cron
- etc

- adminhtml
 - routes.xml
 - system.xml
- config.xml
- crontab.xml
- di.xml
- module.xml
- Model
- view
- composer.json
- registration.php

Всі можливі варіанти використання мають свої функціональні класи, а для виклику імпортування звертаються в одну модель, в якій буде знаходитись бізнес-логіка.

2.2. Розмірковування і проектування реалізації

Так як основна мета, це створення нового методу імпортування зображень для платформи Magento 2, головною задачею є розібратися з існуючою архітектурою даних і зрозуміти як ці дані змінювати під час виконання коду.

Почнемо з самого початку, фізично імпортуванням зображень продуктів є два кроки:

- переніс медіа-файлів з папки імпорту в папку pub/media/catalog/product;
- створення або оновлення записів в базі даних.

Найкраще місця для директорії, в якій будуть знаходитись файли перед імпортуванням, це є папка var тому що:

- в ній не знаходиться жодного важливого файлу, який впливає на роботу платформи;
- її використовує сама платформа під час імпортування продуктів і медіа-контенту.

В результаті цього міркування, було вирішено, що відправною точкою для нового медіа контенту буде папка var, а якщо точніше то var/wedigify/import/image/input. Такий шлях до директорії дозволяє в майбутньому розширювати кількість методів імпортування, не тільки для зображень продуктів, а і для інших вже існуючих на платформі, або новостворених сутностей.

Основний критерій нового методу імпортування була простота, тому було вирішено, що для роботи не потрібно буде створювати і заповнювати ніяких файлів з інформацією для імпортування. Це дозволить спростити процес імпорту для клієнта і зробити його трьох кроковим:

- підготувати зображення;
- перенести в правильну директорію на сервері платформи;
- запустити процес індексу вручну, або взагалі просто дочекатись наступного автоматичного виклику.

В результаті цього виникла наступна проблема, як визначити яке зображення до якого продукту відновиться, яка його позиція і які ролі. Відповіддю на всі ці питання стала спеціальна маска назви зображення, в якій, вся потрібна інформація буде вказана через подвійне нижнє підкреслення. Маска виглядає наступним чином:

`<product_sku>__<image_position>__<image_roles>.*`

де product_sku – це значення унікально атрибуту для кожного продукту, image_postion – цифра, яка означає позицію зображення в галереї, а images_roles – перераховані через кому ролі, які необхідно призначити для цього зображення. Наприклад, якщо product_sku продукту це diploma_test_product_1 і маючи на меті завантажити його на першу позицію з ролями image і small_image ми отримаємо наступне ім'я файлу:

`diploma_test_product_1__1__image,small_image.jpg`

Розібравшись з першою частиною, залишилась сама складна, а саме, яким чином і куди записувати інформацію про нове зображення в базу даних. Для початку треба зазначити, що медіа-контент продукту, на рівні бази даних, це окрема і відокремлена від сутності продукту сутність, пов'язана с продуктом тільки

за допомогою окремої таблиці з двома зовнішніми ключами, а саме ID продукту та ID медіа-контенту. З цього випливає, що при додаванні нового зображення, необхідно додавати запис саме вищезгадану сутність. Таблиця з сутністю медіа-контенту має назву `catalog_product_entity_media_gallery` і має наступні колонки:

- `value_id`;
- `attribute_id`;
- `value`;
- `media_type`;
- `disabled`.

Для зв'язування медіа-контенту і продукту використовується наступна таблиця з назвою `catalog_product_entity_media_gallery_value_to_entity` в неї також потрібно вносити дані при імпортуванні, вона має наступні колонки:

- `value_id`;
- `entity_id`.

Позиція, текстове позначення і додаткова інформація, окрім ролі, знаходиться в третій таблиці з назвою `catalog_product_entity_media_gallery_value` і в неї теж потрібно вносити дані під час імпортування, вона має такі колонки:

- `value_id`;
- `store_id`;
- `entity_id`;
- `label`;
- `position`;
- `disabled`;
- `record_id`.

Фінальною частиною є ролі зображення, які необхідно прописувати під час імпортування. Взагалі, ролі зображення по відношенню до продукту це є його атрибути. Так як структура атрибутів продукту в Magento 2 є дуже складною, так як дозволяє створювати нові без програмування, а тільки за допомогою панелі адміністратора, назва і код атрибутів ролей зображення це записи в таблиці з назвою `eav_attribute`, в якій зберігаються дані про абсолютно всі атрибути для всіх

					ІІЗ.КР.Б – 121 – 23 – ІІЗ	Лист
						36
Зм.	Лист	№ документа	Підпис	Дата		

типів сутностей, які представлені в Magento 2 [5]. Таблиця в якій потрібно вносити зміни під час імпортування називається `catalog_product_entity_varchar` і має наступні колонки:

- `value_id`;
- `attribute_id`;
- `store_id`;
- `entity_id`;
- `value`.

Висновки до другого розділу

В другому розділі було розглянуто та проаналізовано варіанти використання та визначено об'єктну структуру стороннього модулю на основі вже існуючих стандартів.

РОЗДІЛ 3. РОЗРОБКА СТОРОННЬОГО МОДУЛЮ

3.1. Підготовка до створення проекту на базі платформи Magento 2

До початку розробки стороннього модуля для платформи Magento 2 потрібно налаштувати систему під локальну розробку і встановити саму платформу. В якості операційної системи буде виступати macOS Ventura 13.4, особливістю розробки на цій операційній системі є те, що вона встановлена тільки на комп'ютери компанії Apple Inc., які в свою чергу мають одну особливість, це їхній процесор побудований на архітектурі ARM, що вносить свої корективи в підготовку до розробки. Перш за все потрібно встановити Docker, платформу відповідальну за віртуалізацію та запуск контейнерів потрібних для коректної роботи платформи роботи [6]. Для локальної роботи потрібно Docker, а саме його графічна версія – Docker Desktop, для цього потрібно зайти на сайт і завантажити версія для Apple Silicone

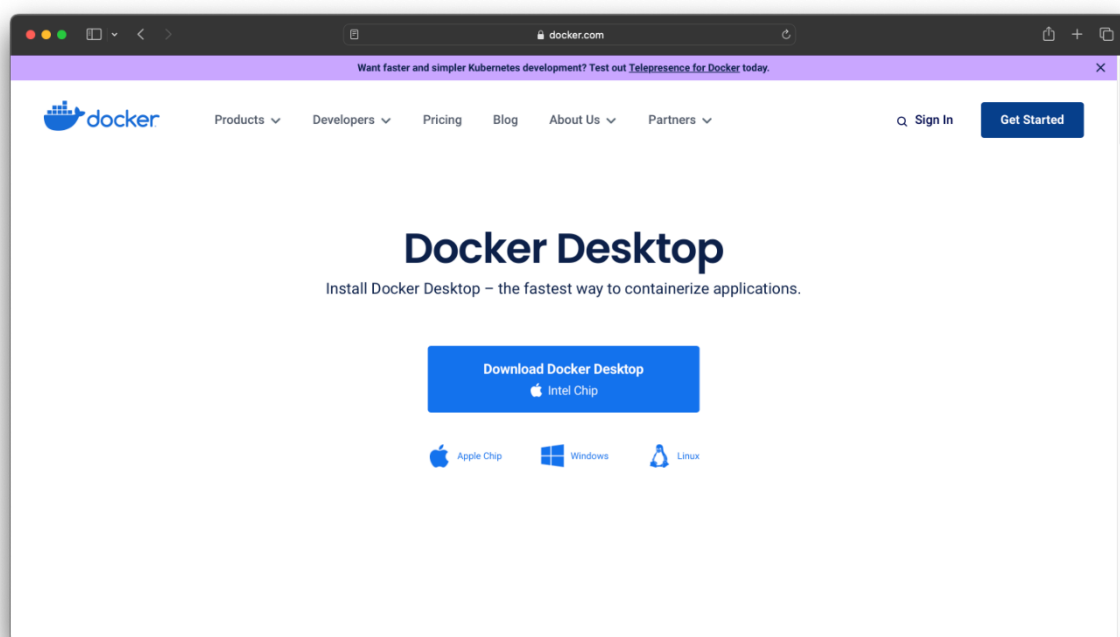


Рис 3.1. Офіційний сайт Docker, сторінка завантаження Docker Desktop
Після чого встановити, як звичайний застосунок і запустити його.

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Лист
Зм.	Лист	№ документа	Підпис	Дата		38

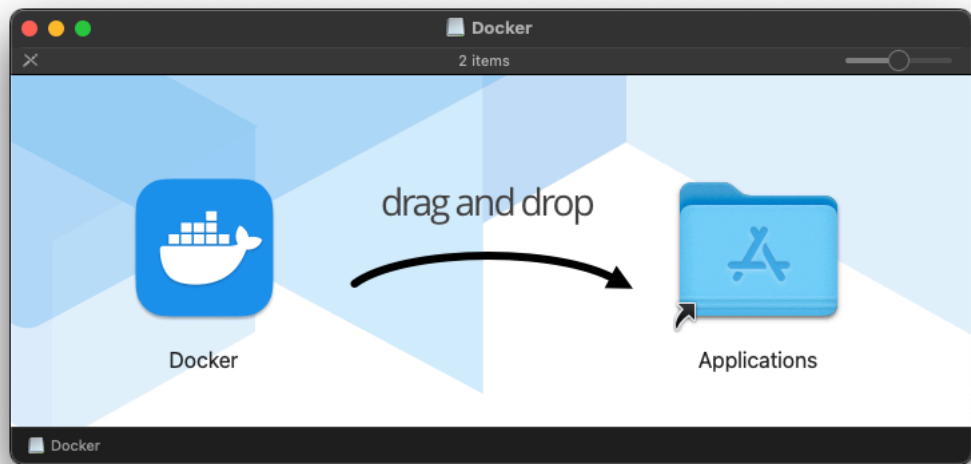


Рис 3.2. Встановлення Docker Desktop

Другим, не менш важливим пунктом, буде встановлення середовища, в якому буде відбуватись процес підняття та підтримання платформи в робочому стані. В даному випадку це буде Warden, утиліта яка працює разом з Docker для автоматичного і зручного створення проектів та їх віртуалізації [8]. Встановлення Warden буде відбуватись в два кроки, перший це встановлення homebrew, утиліти яка дозволяє встановити на macOS інші утиліти, які не пройшли жорсткі правила модерації в App Store(фірмовий магазин утиліт для продукції Apple Inc.) та були розміщені в каталозі homebrew [7]. Встановлення homebrew відбувається наступним чином, потрібно відкрити термінал, та виконати наступну команду:

```
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

Після чого, можна приступати до встановлення Warden. Процес цей дуже схожий до встановлення homebrew, тому що потрібно виконати лише одну команду:

```
brew install wardenenv/warden/warden
```

Для запуску Warden необхідно виконати наступну команду:

```
warden svc up
```

Для перевірки того, що утиліта працює, необхідно відкрити встановлену раніше програму Docker Desktop і побачити запущені контейнери утиліти Warden

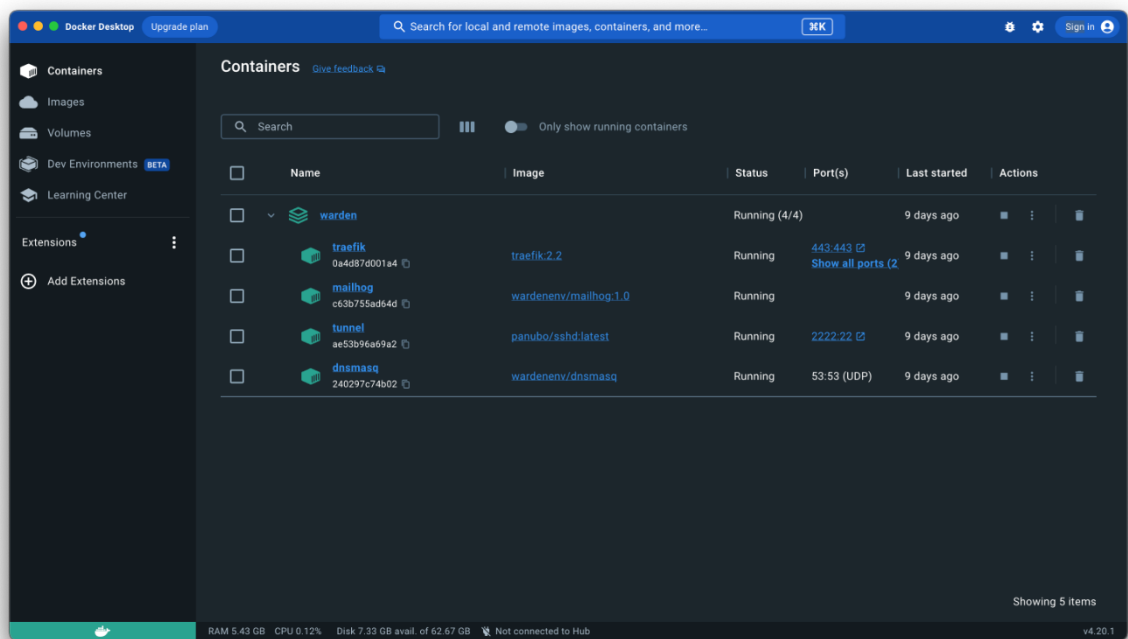


Рис 3.3. Контейнери утиліти Warden в Docker Desktop

Також, нам необхідно встановити Composer, утиліту, яка дозволить нам завантажити Magento 2, а взагалі, це менеджер залежностей, який широко використовується для встановлення сторонніх модулів як для платформи Magento 2 так і взагалі всіх популярних платформ на базі PHP [9]. Для встановлення цієї утиліти необхідно виконати наступну команду:

```
brew install composer
```

Далі, для створення проекту, необхідно створити обліковий запис на сайті Adobe (<https://www.adobe.com/>), це необхідно для того, щоб отримати доступ до маркетплейсу Magento 2 (<https://commercemarketplace.adobe.com/>), а це, в свою чергу, необхідно для отримання ключів доступу, які використовуються для завантаження платформи. Після реєстрації на сайті Adobe, заходимо до маркетплейсу Magento 2, відкриваємо My Profile -> Access Keys і натискаємо Create A New Access Key. Після виконання цих дій, ми отримаємо в своє користування пару ключів public і private, вони необхідні для завантаження платформи єдиним правильним методом, через Composer.

Name	Access Keys	Status	Actions
om	Public Key: 5a986f2d1fa69be6916 Private Key: 01285000d5bfa79ea4	Enabled	Copy Copy Disable Delete

Рис 3.4. Отримані ключі для завантаження Magento 2

3.2. Створення проекту на базі платформи Magento 2

В розділі 3.1 було продемонстрована підготовка до створення проекту на базі платформи Magento 2. Настав час для створення самого проекту, для цього ми створюємо папку з проектами і завантажуюмо Magento 2 через Composer, для цього потрібно виконати наступну команду в директорії з проектами:

```
composer create-project --repository-url=https://repo.magento.com/magento/project-community-edition <install-directory-name>
```

де замість <install-directory-name> буде назва нашого проекту, нехай це буде magento. Під час виконання команди, буде запит на ключі, які ми отримали на маркетплейсі Magento 2, їх потрібно використати. Після успішного завантаження платформи, ми починаємо ініціалізацію проекту для Warden, для цього перейшовши в новостворену директорію з проектом, потрібно виконати наступне:

```
warden env-init exampleproject magento2
```

де замість exampleproject, ми вписуємо назву нашого проекту, це буде magento. В результаті цього в директорії створиться конфігураційний файл утиліти Warden під назвою .env. Для коректної роботи нам потрібно згенерувати SSL сертифікат, робиться це за допомогою наступною командою:

```
warden sign-certificate exampleproject.test
```

де знову ми замінюємо exampleproject на magento.

Після завершення процесу генерування та додавання нового SSL сертифікату, маємо повне право запустити проект, для подальшого налаштування, зробити це можна наступною командою:

```
warden env up
```

для перевірки що все запустилось і працює, як це має бути, заходимо і перевіряємо наявність контейнерів проекту в Docker Desktop

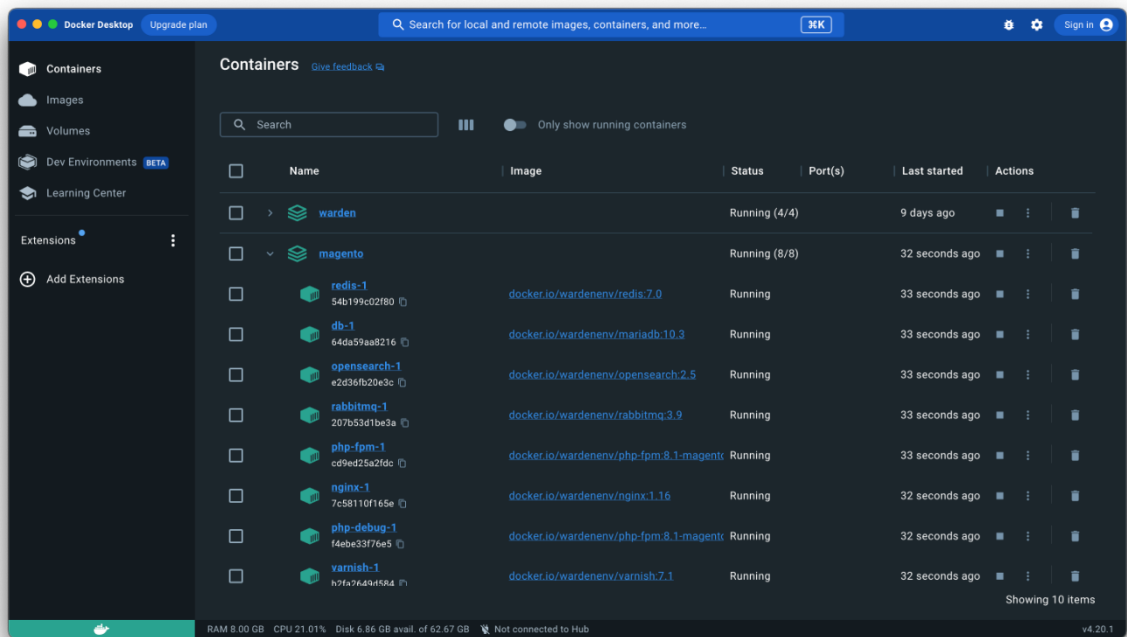


Рис 3.5. Контейнери проекту magento

Якщо, ми зараз відкриємо сторінку проекту <https://magento.test> ми побачимо сторінку, на якій вказано, що платформа працює, але вона не налаштована, для налаштування платформи ми повинні зайти в контейнер з проектом, для цього використаємо наступну команду:

```
warden shell
```

після чого необхідно для коректної роботи утиліти Composer, яка знаходиться в контейнері, нам необхідно зберегти на глобальному рівні ключі отримані з маркетплейсу Magento 2, для цього виконаємо наступне:

```
composer global config http-basic.repo.magento.com <username>
<password>
```

де <username> це public key, а <password> це private key.

Далі іде сам процес налаштування платформи для цього ми використаємо набір команди, які представлені на офіційному сайті утиліти Warden в секції Installing Magento 2, а саме:

```
## Install Application
bin/magento setup:install \
  --backend-frontname=backend \
  --amqp-host=rabbitmq \
  --amqp-port=5672 \
  --amqp-user=guest \
```

```

--amqp-password=guest \
--db-host=db \
--db-name=magento \
--db-user=magento \
--db-password=magento \
--search-engine=opensearch \
--opensearch-host=opensearch \
--opensearch-port=9200 \
--opensearch-index-prefix=magento2 \
--opensearch-enable-auth=0 \
--opensearch-timeout=15 \
--http-cache-hosts=varnish:80 \
--session-save=redis \
--session-save-redis-host=redis \
--session-save-redis-port=6379 \
--session-save-redis-db=2 \
--session-save-redis-max-concurrency=20 \
--cache-backend=redis \
--cache-backend-redis-server=redis \
--cache-backend-redis-db=0 \
--cache-backend-redis-port=6379 \
--page-cache=redis \
--page-cache-redis-server=redis \
--page-cache-redis-db=1 \
--page-cache-redis-port=6379

```

Configure Application

```

bin/magento config:set --lock-env web/unsecure/base_url \
    "https://${TRAEFIK_SUBDOMAIN}.${TRAEFIK_DOMAIN}/"

```

```

bin/magento config:set --lock-env web/secure/base_url \
    "https://${TRAEFIK_SUBDOMAIN}.${TRAEFIK_DOMAIN}/"

```

```

bin/magento config:set --lock-env web/secure/offloader_header X-
Forwarded-Proto

```

```

bin/magento config:set --lock-env web/secure/use_in_frontend 1
bin/magento config:set --lock-env web/secure/use_in_adminhtml 1
bin/magento config:set --lock-env web/seo/use_rewrites 1

```

```

bin/magento config:set --lock-env
system/full_page_cache/caching_application 2
bin/magento config:set --lock-env system/full_page_cache/ttl 604800

```

```

bin/magento config:set --lock-env catalog/search/enable_eav_indexer
1

```

```

bin/magento config:set --lock-env dev/static/sign 0

```

```

bin/magento deploy:mode:set -s developer
bin/magento cache:disable block_html full_page

```

					ІІЗ.КР.Б – 121 – 23 – ІІЗ	Лист
Зм.	Лист	№ документа	Підпис	Дата		43

```
bin/magento indexer:reindex
bin/magento cache:flush
```

Після чого створимо користувача для панелі адміністратора за допомогою наступної команди:

```
bin/magento admin:user:create \
  --admin-password="${ADMIN_PASS}" \
  --admin-user="${ADMIN_USER}" \
  --admin-firstname="Local" \
  --admin-lastname="Admin" \
  --admin-email="${ADMIN_USER}@example.com"
printf "u: %s\np: %s\n" "${ADMIN_USER}" "${ADMIN_PASS}"
```

Після виконання цих всіх команд, заходимо і перевіряємо результати налаштування платформи. Переходимо на сторінку <https://magento.test> для перевірки загальної частини платформи і на <https://magento.test/admin> для перевірки панелі адміністратора

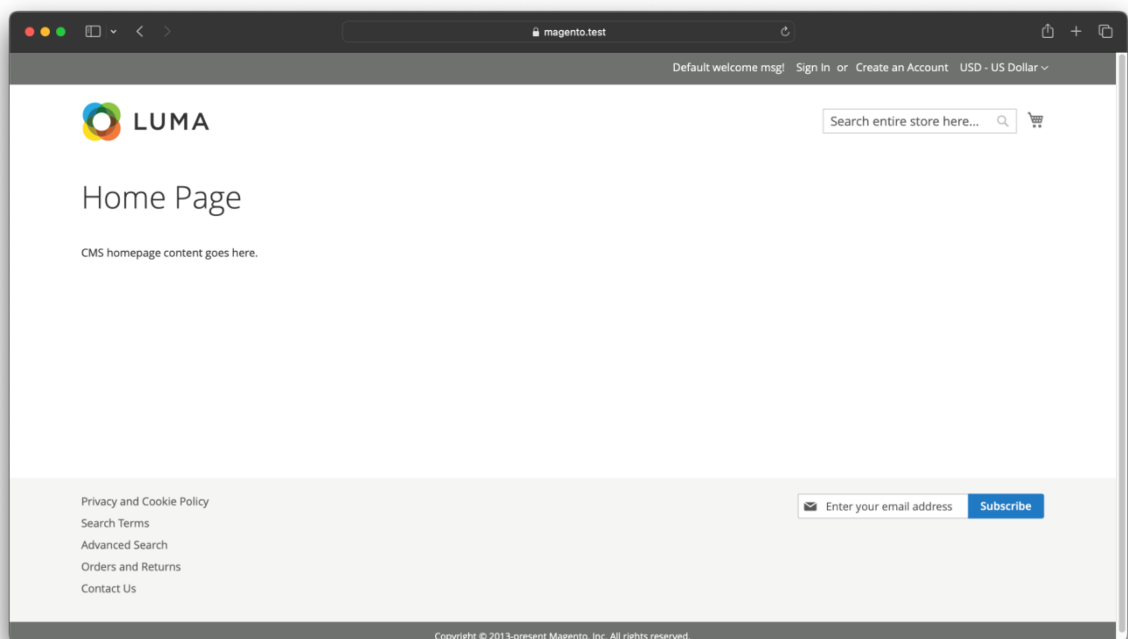


Рис 3.6. Домашня сторінка проекту magento

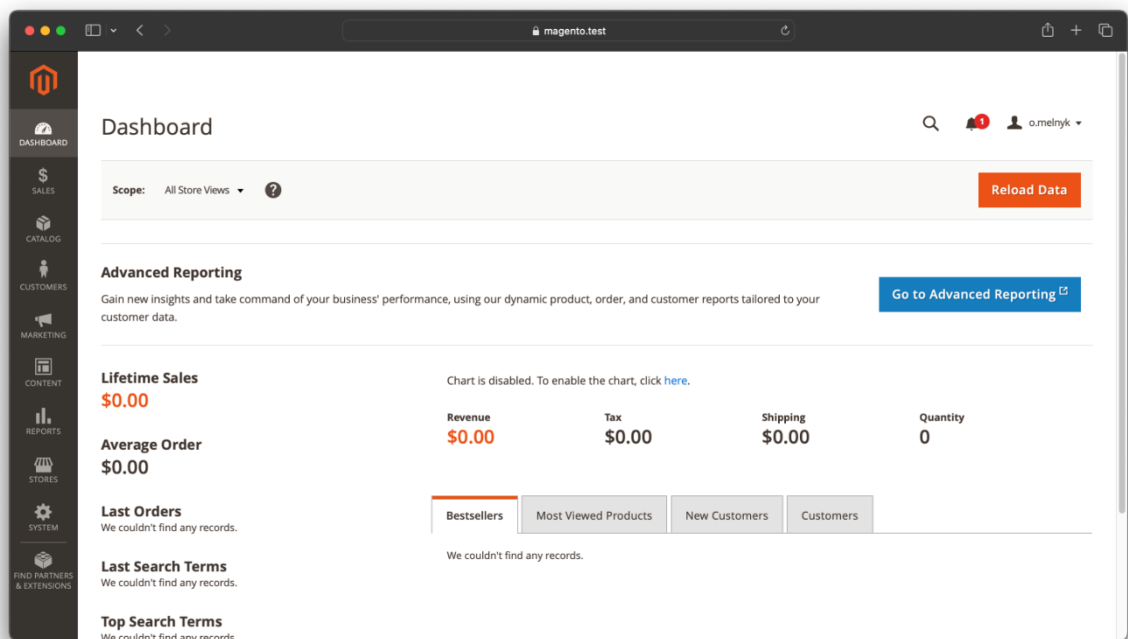


Рис 3.7. Панель адміністратора проекту magento

Отже, коли локальна інсталяція Magento готова до роботи, перейдемо до створення стороннього модулю.

3.3. Написання стороннього модулю

Створення модулю бере свій початок з створення папки вендора(того, хто створив модуль) та назви модулю в даному випадку це буде Wedigify/ImageImport, а також створення двох основних файлів, а саме:

- etc/module.xml;
- registration.php.

Після їх створення ми запускаємо команду активації модулю і отримуємо результат у вигляді нової лінії в файлі app/etc/config.php з назвою модулю, що свідчить про те, що модуль було успішно створено.

Наступним кроком буде реалізація всіх методів виклику функціоналу. Для нового методу імпортування, судячи з технічного завдання їх буде три:

- з панелі адміністратора;
- за допомогою CLI команди;
- відкладений запуск за допомогою Cronjob.

Почнемо з першого методу, а саме з панелі адміністратора. Для цього нам необхідно створити конфігураційний файл, який додає нові налаштування до панелі адміністратора у відділ Stores -> Configuration. Виклик буде реалізований за допомогою кнопки, яка буде відправляти AJAX запит на сервер і запускати функціонал модулю. З цього випливає що, наступними для створення файлами є:

- etc/adminhtml/system.xml – конфігураційний файл;
- etc/adminhtml/routes.xml – для створення точки, куди будуть відправлятися запити;
- Controller/Adminhtml/Import/Index.php – для обробки запиту;
- Block/Adminhtml/System/Config/Button/Import.php – для відображення кнопки на панелі адміністратора;
- view/adminhtml/templates/system/config/button/import.phtml – темплейт кнопки;
- Model/Import.php – основна модель, в якій знаходиться бізнес-логіка функціоналу;
- Model/ResourceModel/ProductMediaGallery.php – ресурсна модуль з бізнес-логікою для роботи з базою даних напряму, без використання репозиторіїв, в якій знаходяться функції для оновлення, видалення і створення записів в таблиці, що необхідно під час роботи коду;
- Lock/Backend.php – клас, який створює тимчасовий файл блокування, перевіряє його існування і не дозволяє запустити більше одного процесу паралельно, задля запобігання проблем.

Наступним для створення буде метод CLI команди. Він потребує створення окремого PHP класу для обробки команди, та конфігурації, яка лежатиме в файлі з назвою di.xml, отже необхідно створити наступні файли:

- etc/di.xml – загальний конфігураційний файл, в якому знаходиться блок про нову консольну команду;
- Console/Command/Import.php – клас відповідальний за створення та обробку команди, а також виклик функціоналу.

Останнім методом є відкладений запуск через Cronjob, для нього нам потрібно додати конфігурацію в панель адміністратора про частоту виконання, а також створити необхідні файли, перший конфігураційний, а другий, це той, який буде викликано під час виконання. Змінюємо вже створений файл etc/adminhtml/system.xml і створюємо наступні:

- etc/crontab.xml – конфігураційний файл;
- Cron/Import.xml – файл, який буде викликано.

Для фінішної структури модулю, не вистачає декількох файлів, які не впливають на роботу, але мають бути створеними, а саме:

- composer.json – файл, в якому вказується тип ліцензії програмного коду, необхідна версія PHP та інші важливі для утиліти Composer речі;
- README.md – файл, в якому людською мовою описується що це за модуль, для чого він і як ним користуватись.

3.4. Тестування стороннього модулю

З архітектурної точки зору перевіряти всі методи виклику нового методу імпортування зображень немає сенсу, тому що всі вони відсилаються до однієї моделі, в якій знаходиться бізнес-логіка. Тому ми перевіримо лише два основних методи.

Першим буде імпортування через кнопку в панелі адміністратора. Для початку необхідно підготувати медіа-файли і дати їм правильні імена з усією потрібною інформацією, для цього треба дізнатись значення SKU продукту, для цього переходимо в панель адміністратора на сторінку управління всіма продуктами і знаходимо це значення.

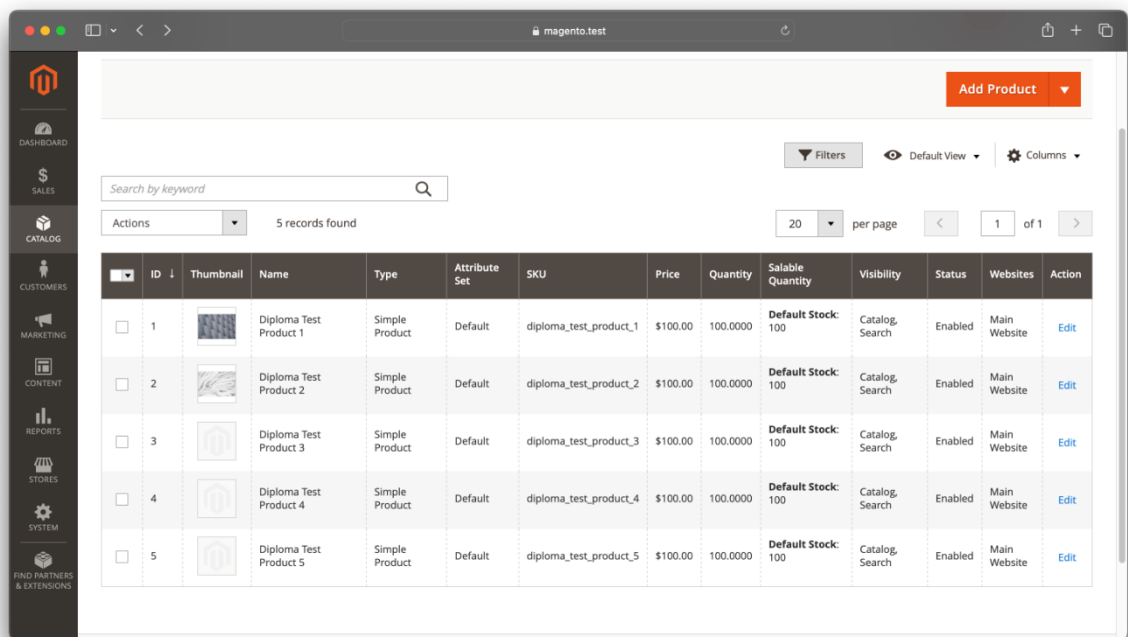


Рис 3.8. Сторінка управління усіма продуктами

Завантажувати зображення ми будемо для продукту з SKU `diploma_test_product_3`.

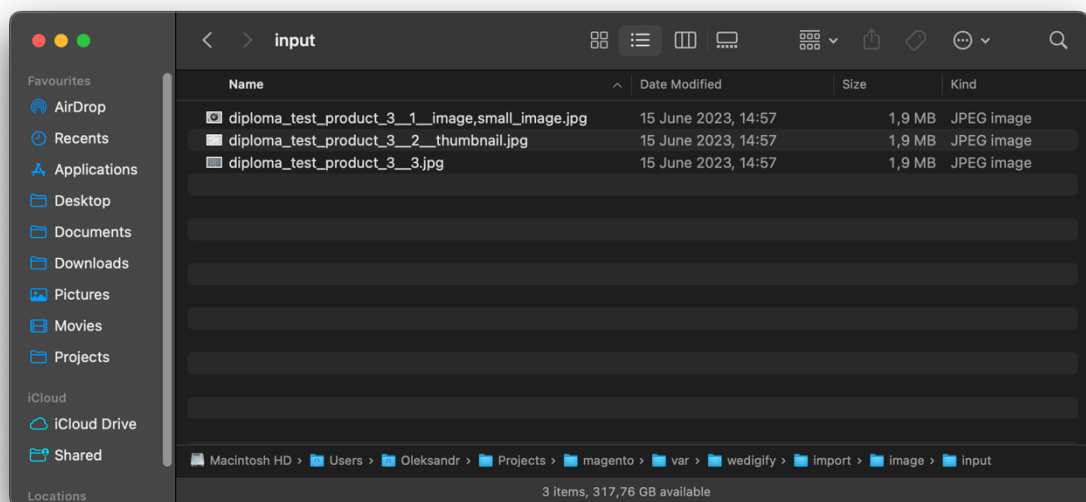


Рис 3.9. Підготовлені та перенесені зображення для імпортування

Далі, ми заходимо в панель адміністратора, переходимо в Stores -> Configuration, знаходимо налаштування модулю.

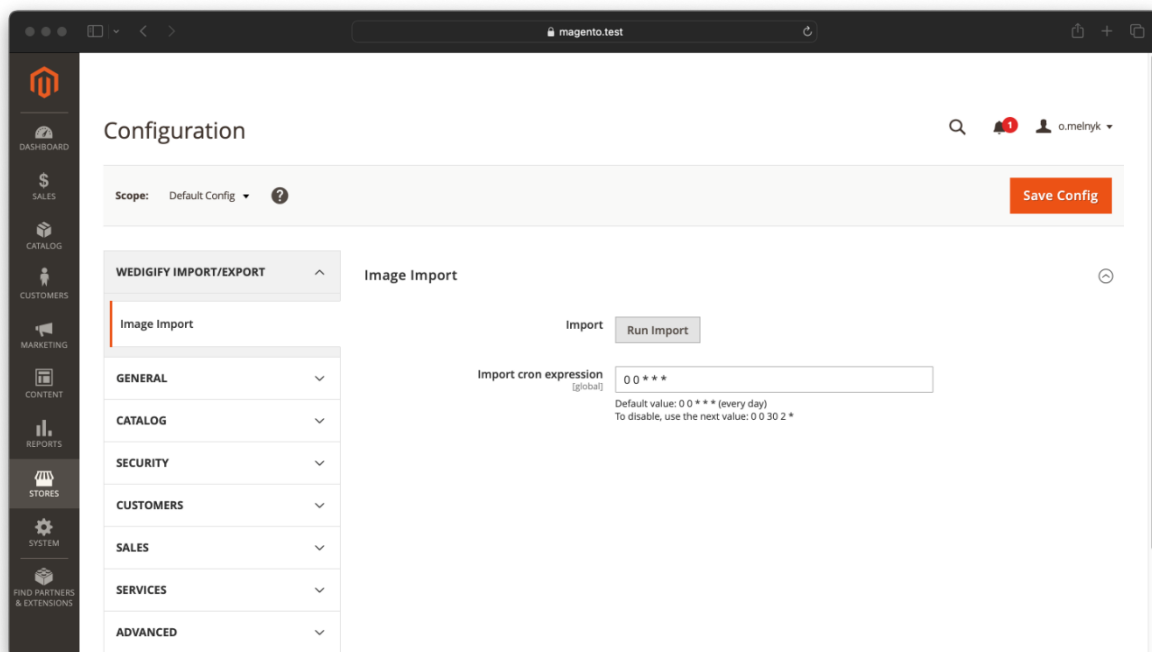


Рис 3.10. Налаштування модулю і кнопка запуску імпортування
Натискаємо кнопку Run Import і отримуємо результат виконання

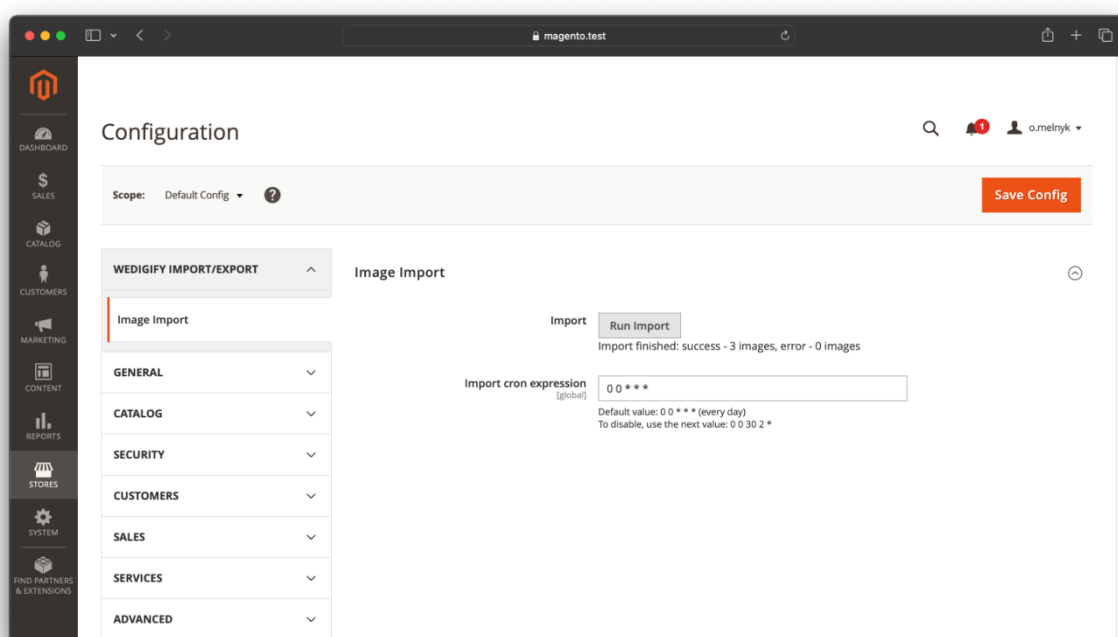


Рис 3.11. Результат виконання імпортування
Перевіряємо в панелі адміністратора на сторінці управління продуктом, та на PDP і PLP.

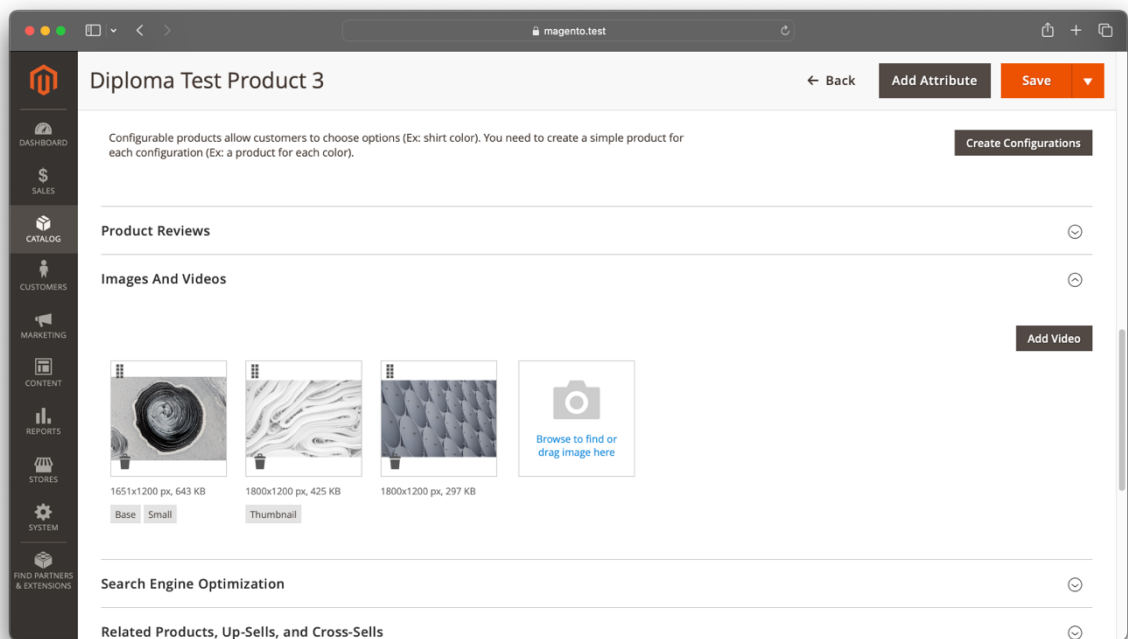


Рис 3.12. Результат виконання імпортування на сторінці управління продуктом

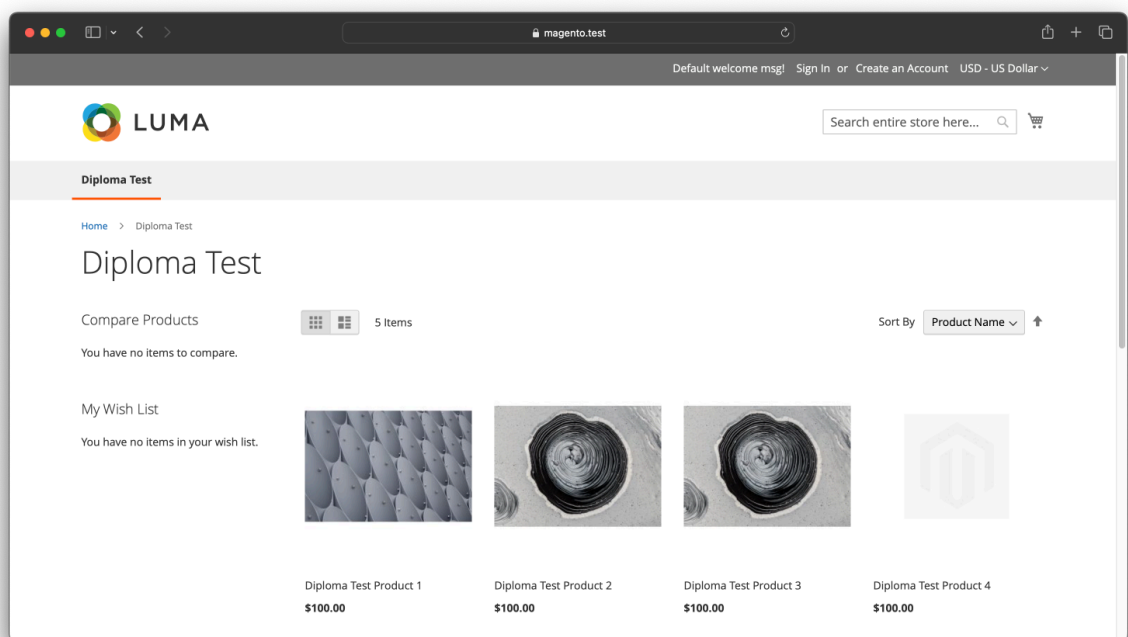


Рис 3.13. Результат імпортування на PLP

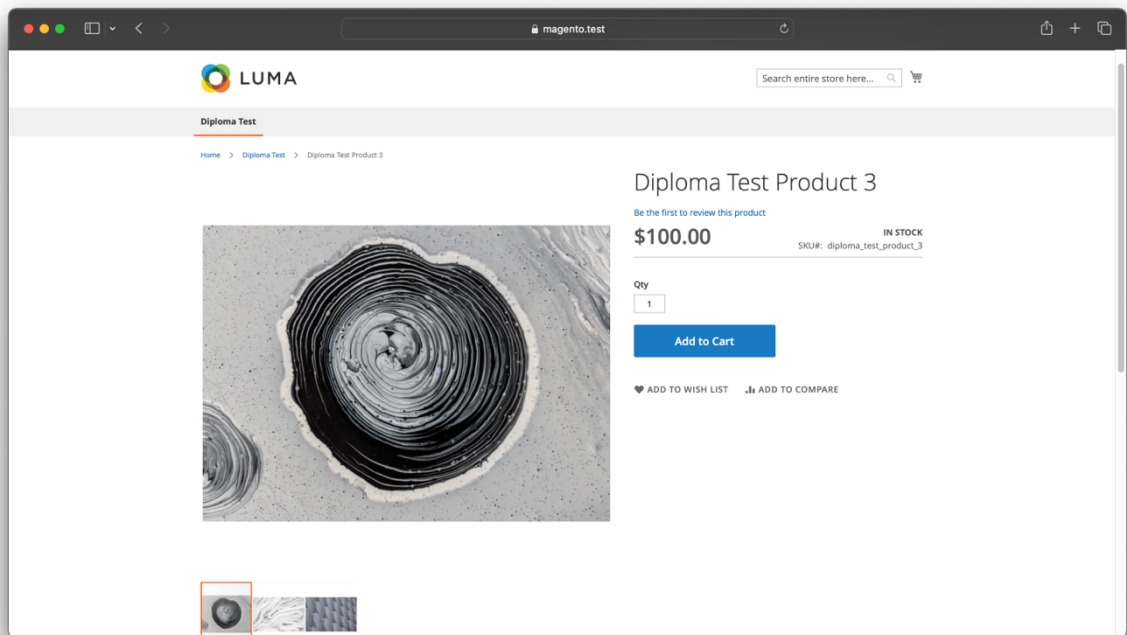


Рис 3.14. Результат імпортування на PDP

Для тестування імпортування через CLI команду, ми так само готуємо зображення і переміщуємо їх в папку імпорту. На цей раз будемо імпортувати для також для продукту зі значенням SKU `diploma_test_product_3`, але змінимо порядок картинок і їх ролі.

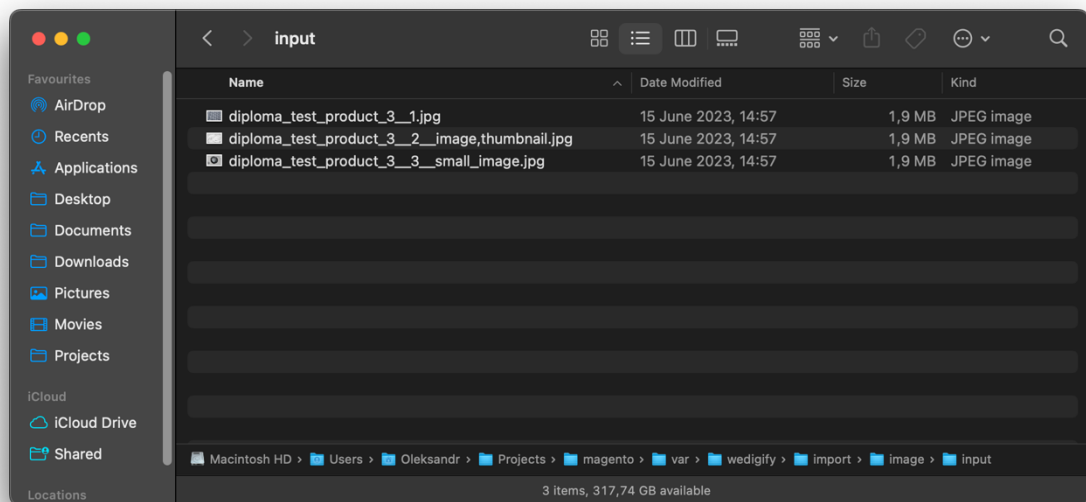


Рис 3.15. Підготовлені та перенесені зображення для імпортування

Далі ми відкриваємо термінал, заходимо в контейнер проекту і виконуємо наступну команду:

```
php bin/magento wedigify:image-import:run
```

Отримуємо результат виконання і перевіряємо.

```
magento — @magento-php-fpm:/var/www/html — docker-compose < warde...
Oleksandr@Oleksandrs-MacBook-Air magento % warden shell
www-data@magento-php-fpm:/var/www/html$ bin/magento wedigify:image-import:run
Import finished: success - 3 images, error - 0 images
www-data@magento-php-fpm:/var/www/html$
```

Рис 3.16. Результат виконання CLI команди

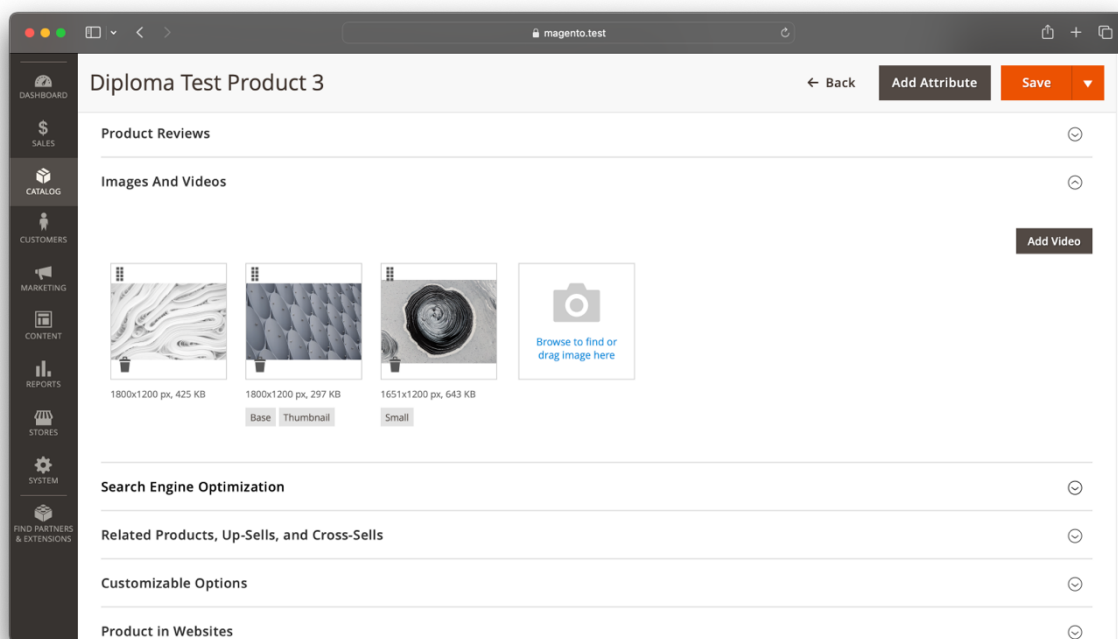


Рис 3.17. Результат виконання імпортування на сторінці управління продуктом

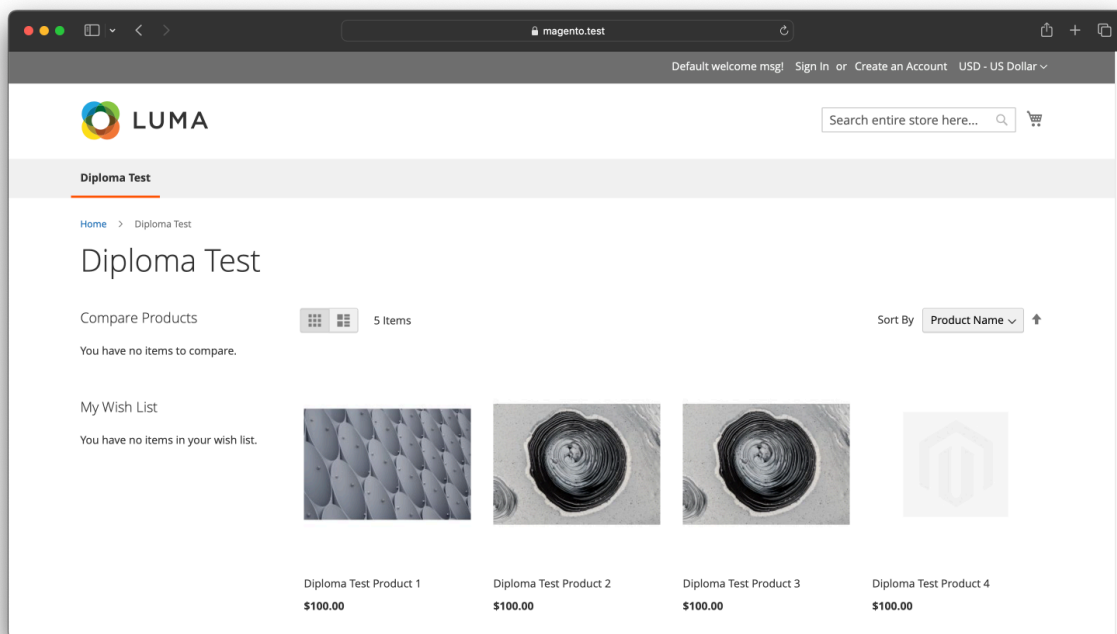


Рис 3.18. Результат виконання імпортування на PLP

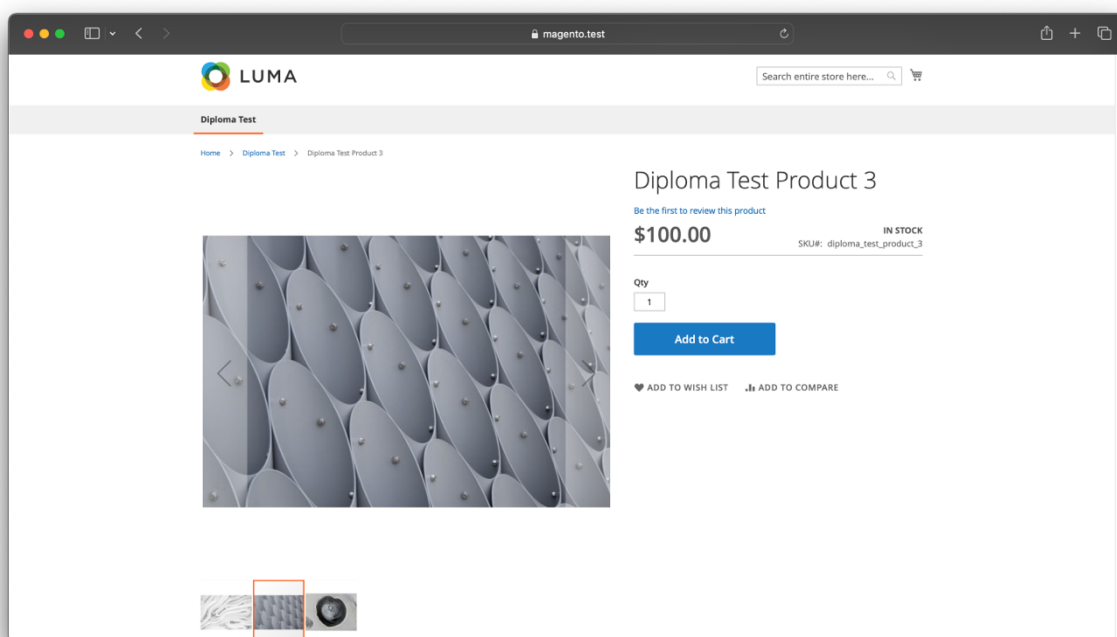


Рис 3.19. Результат виконання імпортування на POP

Отже, модуль працює і виконує поставлені перед ним задачі.

Висновки до третього розділу

В третьому розділі було підготовлено локальну машину для створення проекту, а також було створено локальну установку платформи Magento 2. Після чого було створено сам сторонній модуль, який виконує функцію додаткового методу імпортування медіа-файлів в продукти, і він був протестований двома різними методами в двох різних сценаріях.

					ПЗ.КР.Б – 121 – 23 – ПЗ	Лист
						54
Зм.	Лист	№ документа	Підпис	Дата		

ВИСНОВКИ

Розглянуто та проаналізовано важливість якісного медіа-контенту в інтернет-магазинах.

Розглянуто і порівняно методи імпортування вже вбудовані в платформу електронної комерції Magento 2, а саме завантаження зображень через сторінку управління продуктом та імпорт через механізм імпортування продуктів, а також визначено сильні і слабкі сторони кожного з них.

Сформовано технічне завдання для стороннього модулю, яке відповідає меті і завданню кваліфікаційної роботи.

Розглянуто та проаналізовано варіанти використання та визначено об'єктну структуру стороннього модулю на основі вже існуючих стандартів.

Підготовлено локальну машину для створення проекту.

Створено локальну установку платформи Magento 2.

Створено сам сторонній модуль, який виконує функцію додаткового методу імпортування медіа-файлів в продукти.

Сторонній модуль був протестований двома різними методами в двох різних сценаріях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Adobe Experience League: Manage product images and videos [Електронний ресурс] — Режим доступу до ресурсу:
<https://experienceleague.adobe.com/docs/commerce-admin/catalog/products/digital-assets/product-image.html?lang=en>
2. Adobe Experience League: Product image import [Електронний ресурс] — Режим доступу до ресурсу:
<https://experienceleague.adobe.com/docs/commerce-admin/systems/data-transfer/import/data-import-product-images.html?lang=en>
3. Adobe Experience League: Import [Електронний ресурс] — Режим доступу до ресурсу: <https://experienceleague.adobe.com/docs/commerce-admin/systems/data-transfer/import/data-import.html?lang=en>
4. Adobe Developer: Create your component file structure [Електронний ресурс] — Режим доступу до ресурсу:
<https://developer.adobe.com/commerce/php/development/build/component-file-structure/>
5. Adobe Developer: EAV and extension attributes [Електронний ресурс] — Режим доступу до ресурсу:
<https://developer.adobe.com/commerce/php/development/components/attributes/>
6. Docker [Електронний ресурс] — Режим доступу до ресурсу:
<https://www.docker.com/>
7. Homebrew [Електронний ресурс] — Режим доступу до ресурсу:
<https://brew.sh/>
8. Warden [Електронний ресурс] — Режим доступу до ресурсу:
<https://docs.warden.dev/index.html>
9. Composer [Електронний ресурс] — Режим доступу до ресурсу:
<https://getcomposer.org/>

ДОДАТКИ

					ПЗ.КР.Б – 121 – 23 – ПЗ	Лист
Зм.	Лист	№ документу	Підпис	Дата		57

Лістинг модулю:

Wedigify/ImageImport/Block/Adminhtml/System/Config/Button/Import.php

```

<?php declare(strict_types=1);

namespace Wedigify\ImageImport\Block\Adminhtml\System\Config\Button;

use Magento\Backend\Block\Template\Context;
use Magento\Config\Block\System\Config\Form\Field;
use Magento\Framework\Data\Form\Element\AbstractElement;
use Magento\Framework\Exception\LocalizedException;

class Import extends Field
{
    /**
     * @var string
     */
    protected $_template =
        'Wedigify_ImageImport::system/config/button/import.phtml';

    /**
     * @param Context $context
     * @param array $data
     */
    public function __construct(
        Context $context,
        array $data = []
    ) {
        parent::__construct($context, $data);
    }

    /**
     * Remove scope label
     *
     * @param AbstractElement $element
     *
     * @return string
     */
    public function render(AbstractElement $element)
    {
        $element->unsScope()->unsCanUseWebsiteValue()->unsCanUseDefaultValue();

        return parent::render($element);
    }

    /**
     * @param AbstractElement $element
     *
     * @return string
     */
    protected function _getElementHtml(AbstractElement $element)
    {
        return $this->_toHtml();
    }

    /**
     * @return string
     */
    public function getAjaxUrl()
    {
        return $this->getUrl('wedigify_imageimport/import');
    }
}

```

					ІІЗ.КР.Б – 121 – 23 – ІІЗ	Лист
						58
Зм.	Лист	№ документа	Підпис	Дата		

```

/**
 * @return string
 * @throws LocalizedException
 */
public function getButtonHtml()
{
    $button = $this->getLayout()->createBlock(
        'Magento\Backend\Block\Widget\Button'
    )->setData(
        [
            'id' => 'import_button',
            'label' => __('Run Import'),
        ]
    );

    return $button->toHtml();
}

```

Wedigify/ImageImport/Console/Command/Import.php

```

<?php declare(strict_types=1);

namespace Wedigify\ImageImport\Console\Command;

use Magento\Framework\App\Area;
use Magento\Framework\App\State;
use Magento\Framework\Console\Cli;
use Symfony\Component\Console\Command\Command;
use Symfony\Component\Console\Input\InputInterface;
use Symfony\Component\Console\Output\OutputInterface;
use Wedigify\ImageImport\Model\Import as Importer;

class Import extends Command
{
    /**
     * @var State
     */
    protected State $appState;

    /**
     * @var Importer
     */
    protected Importer $importer;

    /**
     * @param State $appState
     * @param Importer $importer
     * @param string|null $name
     */
    public function __construct(
        State $appState,
        Importer $importer,
        string $name = null
    ) {
        $this->appState = $appState;
        $this->importer = $importer;
        parent::__construct($name);
    }
}

```

```

/**
 * @inheritDoc
 */
protected function configure()
{
    $this->setName('wedigify:image-import:run')
        ->setDescription('Wedigify Image Import');

    parent::configure();
}

/**
 * @inheritDoc
 */
protected function execute(InputInterface $input, OutputInterface $output)
{
    try {
        $this->appState->setAreaCode(Area::AREA_GLOBAL);
        $result = $this->importer->run();
        $output->writeln($result);
    } catch (\Exception $e) {
        $output->writeln("<error>{$e->getMessage()}</error>");

        return Cli::RETURN_FAILURE;
    }

    return Cli::RETURN_SUCCESS;
}
}

```

Wedigify/ImageImport/Controller/Adminhtml/Import/Index.php

```

<?php declare(strict_types=1);

namespace Wedigify\ImageImport\Controller\Adminhtml\Import;

use Magento\Backend\App\Action;
use Magento\Backend\App\Action\Context;
use Magento\Framework\Controller\Result\JsonFactory;
use Wedigify\ImageImport\Model\Import as Importer;

class Index extends Action
{
    /**
     * @var JsonFactory
     */
    protected JsonFactory $resultJsonFactory;

    /**
     * @var Importer
     */
    protected Importer $importer;

    /**
     * @param JsonFactory $resultJsonFactory
     * @param Importer $importer
     * @param Context $context
     */
    public function __construct(
        JsonFactory $resultJsonFactory,
        Importer $importer,
        Context $context
    )

```

```

    ) {
        $this->resultJsonFactory = $resultJsonFactory;
        $this->importer = $importer;
        parent::__construct($context);
    }

    /**
     * @inheritDoc
     */
    public function execute()
    {
        $result = $this->resultJsonFactory->create();
        return $result->setData(['message' => $this->importer->run()]);
    }
}

```

Wedigify/ImageImport/Cron/Import.php

```

<?php declare(strict_types=1);

namespace Wedigify\ImageImport\Cron;

use Wedigify\ImageImport\Model\Import as Importer;

class Import
{
    /**
     * @var Importer
     */
    protected Importer $importer;

    /**
     * @param Importer $importer
     */
    public function __construct(Importer $importer)
    {
        $this->importer = $importer;
    }

    /**
     * @return void
     */
    public function execute()
    {
        $this->importer->run();
    }
}

```

Wedigify/ImageImport/etc/adminhtml/routes.xml

```

<?xml version="1.0"?>

<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="urn:magento:framework:App/etc/routes.xsd">
    <router id="admin">
        <route id="wedigify_imageimport" frontName="wedigify_imageimport">
            <module name="Wedigify_ImageImport"/>
        </route>
    </router>
</config>

```

					ПЗ.КР.Б – 121 – 23 – ПЗ	Лист
						61
Зм.	Лист	№ документа	Підпис	Дата		

Wedigify/ImageImport/etc/adminhtml/system.xml

```
<?xml version="1.0"?>

<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:noNamespaceSchemaLocation="urn:magento:module:Magento_Config:etc/system_file.x
sd">
    <system>
        <tab id="wedigify_imp_exp" translate="label" sortOrder="1">
            <label>Wedigify Import/Export</label>
        </tab>
        <section id="wedigify_image_import" translate="label" type="text"
sortOrder="10" showInDefault="1" showInWebsite="1" showInStore="1">
            <class>separator-top</class>
            <label>Image Import</label>
            <tab>wedigify_imp_exp</tab>
            <resource>Wedigify_ImageImport::acl_path</resource>
            <group id="general" translate="label" type="text" sortOrder="10"
showInDefault="1" showInWebsite="0" showInStore="0">
                <label>Image Import</label>
                <field id="import_button" translate="label comment" type="button"
sortOrder="10" showInDefault="1" showInWebsite="0" showInStore="0">
                    <label>Import</label>

<frontend_model>Wedigify\ImageImport\Block\Adminhtml\System\Config\Button\Import</
frontend_model>
                </field>
                <field id="import_cron_expression" translate="label" type="text"
sortOrder="30" showInDefault="1" showInWebsite="0" showInStore="0">
                    <label>Import cron expression</label>
                    <comment><![CDATA[Default value: 0 0 * * * (every day)</br>To
disable, use the next value: 0 0 30 2 *]]></comment>
                    <validate>validate-no-empty</validate>
                </field>
            </group>
        </section>
    </system>
</config>
```

Wedigify/ImageImport/etc/config.xml

```
<?xml version="1.0"?>

<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:noNamespaceSchemaLocation="urn:magento:module:Magento_Store:etc/config.xsd">
    <default>
        <wedigify_image_import>
            <general>
                <import_cron_expression>0 0 * * *</import_cron_expression>
            </general>
        </wedigify_image_import>
    </default>
</config>
```

Wedigify/ImageImport/etc/crontab.xml

```
<?xml version="1.0" ?>

<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:noNamespaceSchemaLocation="urn:magento:module:Magento_Cron:etc/crontab.xsd">
    <group id="default">
```

					ІІЗ.КР.Б – 121 – 23 – ІІЗ	Лист
						62
Зм.	Лист	№ документа	Підпис	Дата		

```

        <job name="wedigify_image_import"
instance="Wedigify\ImageImport\Cron\Import" method="execute">

<config_path>wedigify_image_import/general/import_cron_expression</config_path>
    </job>
</group>
</config>

```

Wedigify/ImageImport/etc/di.xml

```

<?xml version="1.0"?>

<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="urn:magento:framework:ObjectManager/etc/config.xsd"
>
    <type name="Magento\Framework\Console\CommandList">
        <arguments>
            <argument name="commands" xsi:type="array">
                <item name="wedigify_image_import"
xsi:type="object">Wedigify\ImageImport\Console\Command\Import</item>
            </argument>
        </arguments>
    </type>

    <!-- Creating own logger -->
    <virtualType name="Wedigify\ImageImport\Model\VirtualLoggerHandler"
type="Magento\Framework\Logger\Handler\Base">
        <arguments>
            <argument name="fileName"
xsi:type="const">Wedigify\ImageImport\Model\Import::LOG_FILE_PATH</argument>
        </arguments>
    </virtualType>
    <virtualType name="Wedigify\ImageImport\Model\VirtualLogger"
type="Magento\Framework\Logger\Monolog" shared="false">
        <arguments>
            <argument name="name"
xsi:type="string">wedigify_image_import</argument>
            <argument name="handlers" xsi:type="array">
                <item name="system"
xsi:type="object">Wedigify\ImageImport\Model\VirtualLoggerHandler</item>
            </argument>
        </arguments>
    </virtualType>

    <type name="Wedigify\ImageImport\Model\Import">
        <arguments>
            <argument name="logger"
xsi:type="object">Wedigify\ImageImport\Model\VirtualLogger</argument>
        </arguments>
    </type>
</config>

```

Wedigify/ImageImport/etc/module.xml

```

<?xml version="1.0"?>

<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="urn:magento:framework:Module/etc/module.xsd">
    <module name="Wedigify_ImageImport"/>
</config>

```

					ІІЗ.КР.Б – 121 – 23 – ІІЗ	Лист
						63
Зм.	Лист	№ документу	Підпис	Дата		

Wedigify/ImageImport/Lock/Backend.php

```
<?php declare(strict_types=1);

namespace Wedigify\ImageImport\Lock;

use Magento\Framework\Lock\LockManagerInterface;
use Magento\Framework\Lock\LockManagerInterfaceFactory;

class Backend implements LockManagerInterface
{
    private const DEFAULT_LOCK_TIMEOUT = 30;

    /**
     * @var LockManagerInterface
     */
    protected LockManagerInterface $lockManager;

    /**
     * @param LockManagerInterfaceFactory $lockManagerFactory
     */
    public function __construct(LockManagerInterfaceFactory $lockManagerFactory)
    {
        $this->lockManager = $lockManagerFactory->create();
    }

    /**
     * @inheritdoc
     */
    public function lock(string $name, int $timeout = -1): bool
    {
        return $this->lockManager->lock($name, $timeout == -1 ?
self::DEFAULT_LOCK_TIMEOUT : $timeout);
    }

    /**
     * @inheritdoc
     */
    public function unlock(string $name): bool
    {
        return $this->lockManager->unlock($name);
    }

    /**
     * @inheritdoc
     */
    public function isLocked(string $name): bool
    {
        return $this->lockManager->isLocked($name);
    }
}
```

Wedigify/ImageImport/view/adminhtml/templates/system/config/button/import.phtml

```
<?php /** @var $block
\Wedigify\ImageImport\Block\Adminhtml\System\Config\Button\Import */?>

<script>
    require([
        'jquery',
        'prototype'
    ], function ($) {

        $('#import_button').click(function () {
```

					ПЗ.КР.Б – 121 – 23 – ПЗ	Лист
						64
Зм.	Лист	№ документа	Підпис	Дата		

```

var params = {};
new Ajax.Request('<?php echo $block->getAjaxUrl() ?>', {
    parameters: params,
    loaderArea: false,
    asynchronous: true,
    onCreate: function () {
        $('#import_button_response_message').text('');
    },
    onSuccess: function (transport) {
        var resultText = '';
        if (transport.status > 200) {
            resultText = transport.statusText;
        } else {
            var response = JSON.parse(transport.responseText);
            resultText = response.message
        }
        $('#import_button_response_message').text(resultText);
    }
});
});
</script>
<?php echo $block->getButtonHtml(); ?>
<p>
    <span id="import_button_response_message"></span>
</p>

```

Wedigify/ImageImport/composer.json

```

{
    "name": "wedigify/image-import",
    "description": "Wedigify Image Import",
    "require": {
        "php": "^8.1"
    },
    "type": "magento2-module",
    "version": "1.0.0",
    "license": "proprietary",
    "autoload": {
        "files": [
            "registration.php"
        ],
        "psr-4": {
            "Wedigify\\ImageImport\\": ""
        }
    }
}

```

Wedigify/ImageImport/README.md

Wedigify Image Import

General information

Additional method of import product images.

There are three methods, how to start image import:

- Button "Run Import" is placed in Admin Panel Configuration;
- CLi command `bin/magento wedigify:image-import:run`
- `"wedigify_image_import"` cronjob

To use, you need to prepare the image before importing, this should be done based on the mask:

```

|     |      |             |        |      |                           |      |
|-----|------|-------------|--------|------|---------------------------|------|
|     |      |             |        |      | ІІЗ.КР.Б – 121 – 23 – ІІЗ | Лист |
|     |      |             |        |      |                           |      |
| Зм. | Лист | № документа | Підпис | Дата |                           | 65   |

```
<product_sku>__<image_position>__<image_roles>.*
\\`
```

Allowed images roles:

- image
- small\_image
- thumbnail

You can assign a few image roles to image, by providing them in name divided by a comma: image,small\_image

You can provide no image roles, file maks will be next:

```
<product_sku>__<image_position>.*
\\`
```

Allowed file extensions:

- jpg
- jpeg
- png
- gif

## Configuration

Path to configuration: Admin Panel -> Stores -> Configuration -> Wedigify  
Import/Export -> Image Import

## How to use

- Prepare images and place them to var/wedigify/import/image/input
- Start image import from Admin Panel or CLI, or just wait for the next cronjob

## Wedigify/ImageImport/registration.php

```
<?php declare(strict_types=1);
```

```
\Magento\Framework\Component\ComponentRegistrar::register(
\Magento\Framework\Component\ComponentRegistrar::MODULE,
 'Wedigify_ImageImport',
 __DIR__
);
```

|     |      |             |        |      |                         |      |
|-----|------|-------------|--------|------|-------------------------|------|
|     |      |             |        |      | ПЗ.КР.Б – 121 – 23 – ПЗ | Лист |
|     |      |             |        |      |                         | 66   |
| Зм. | Лист | № документа | Підпис | Дата |                         |      |