Sultan Moulay Slimane University
Polydisciplinary Faculty of Khouribga
Mathematics and Computer Science Department

Major: Computer Engineering and Artificial Intelligence
Module:NoSQL Databases

MINI-PROJECT REPORT :

Multi-channel product catalog

(MongoDB, Python, Streamlit, Pandas)

Author:
[**HMADNA AYMANE**]
Profil GitHub:
`https://github.com/AymaneHmadna`

Academic Year: 2025-2026

Contents

1 Project Objective :

This project aims to design a **Multi-channel product catalog** system capable of centralizing, cleaning and distributing heterogeneous product data. In a modern ecommerce context, a brand must sell on several channels (Website, Marketplace, Social media).

The main objectives are :

— **Centralization :** Create a unique repository with a document oriented database (mongoDB).

— **Data Quality :** Implement a validation workflow (Draft vs Published) to prevent the diffusion of incomplete data.

— **Flexibility :** Manage products with variable attributes (Clothing vs Electronics) thanks to JSON polymorphism.

— **Multi-Channel Distribution :** Export data in the form of adapted structured feeds (nested JSON for the Web, "flattened" CSV for Marketplaces).

2 Technology Stack Description :

Each component has been selected to meet the flexibility and performance constraints of the specifications.

2.1 MongoDB (Database) :

Document oriented NoSQL database.

— **Role :** Flexible storage of product sheets.

— **Why MongoDB ?** Unlike rigid SQL, mongoDB allows storing very different products (e.g., T-shirt with "size/color" and Phone with "storage/warranty") in the same collection without multiplying empty tables.

2.2 Python & Pandas (Processing) :

— **Role :** Data manipulation and transformation for export.

— **Why Pandas ?** The `json_normalize` function is crucial to "flatten" nested JSON structures in order to generate CSV files compatible with excel.

2.3 Streamlit (User Interface) :

— **Role :** Management interface for the product manager.

— **Why Streamlit ?** Allows rapid development of an interactive CRUD (Create, Read, Update, Delete) interface with immediate data visualization.

3 Infrastructure and Modeling :

The architecture relies on three main Python scripts :

— **`data.py` :** Simulates data ingestion. This script generates 50 realistic products with random attributes and inserts them directly into MongoDB. This allows testing the load and indexes immediately.

— **`index.py` :** Administration script that creates the required indexing structures :

- *Unique Index (SKU) :* Guarantees that no logistical duplicate is created.
- *Text Index (Name, Brand) :* Activates the performant search engine.

— **`app.py` :** The heart of the system. It manages :

- The connection to MongoDB via `pymongo`.
- The business logic (Quality rules `check_quality`).
- The user interface (Catalog, Add New, Export, Logs).

4 Execution Process and Proofs :

4.1 Data Ingestion :

Execution of the `data.py` script to populate the database. The script connects to mongoDB, cleans the existing collection and generates a batch of 50 products (Electronics and Clothing) with varied attributes.

```
python data.py
```

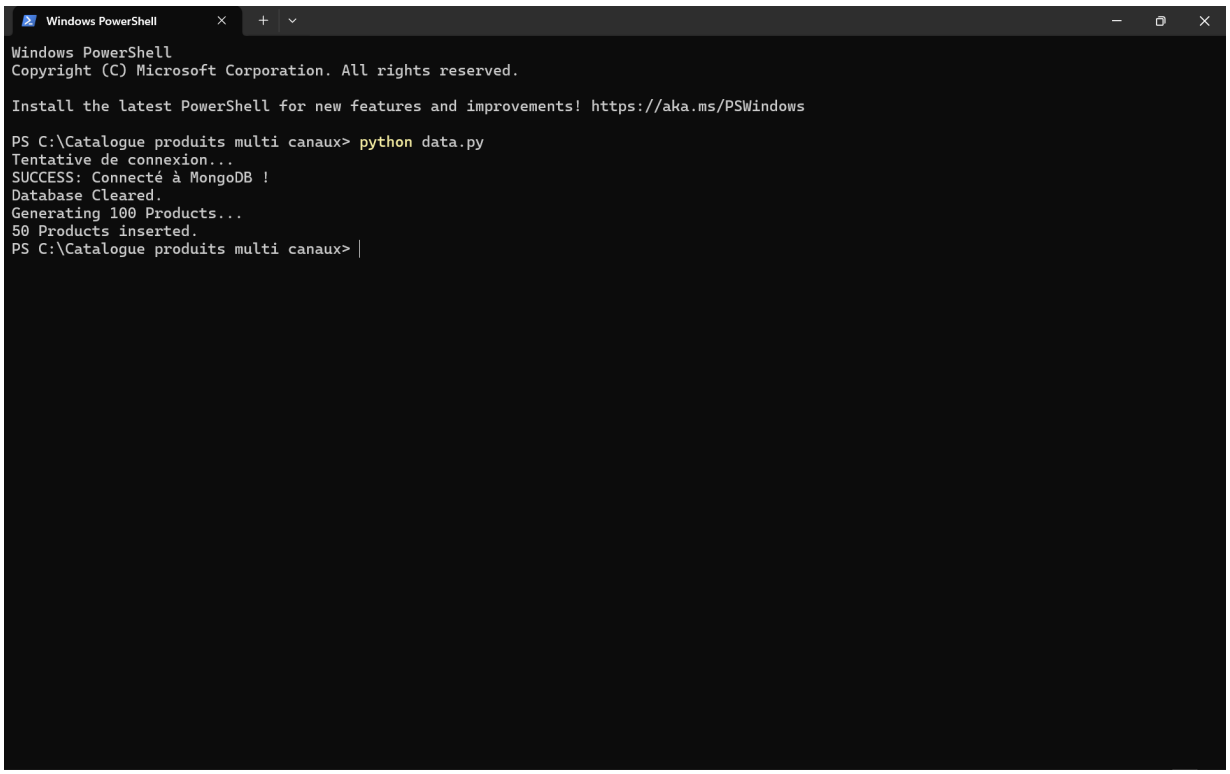


Figure 1: Success of data insertion via the python script.

4.2 Verification in MongoDB Compass :

We verify that the documents are well stored.

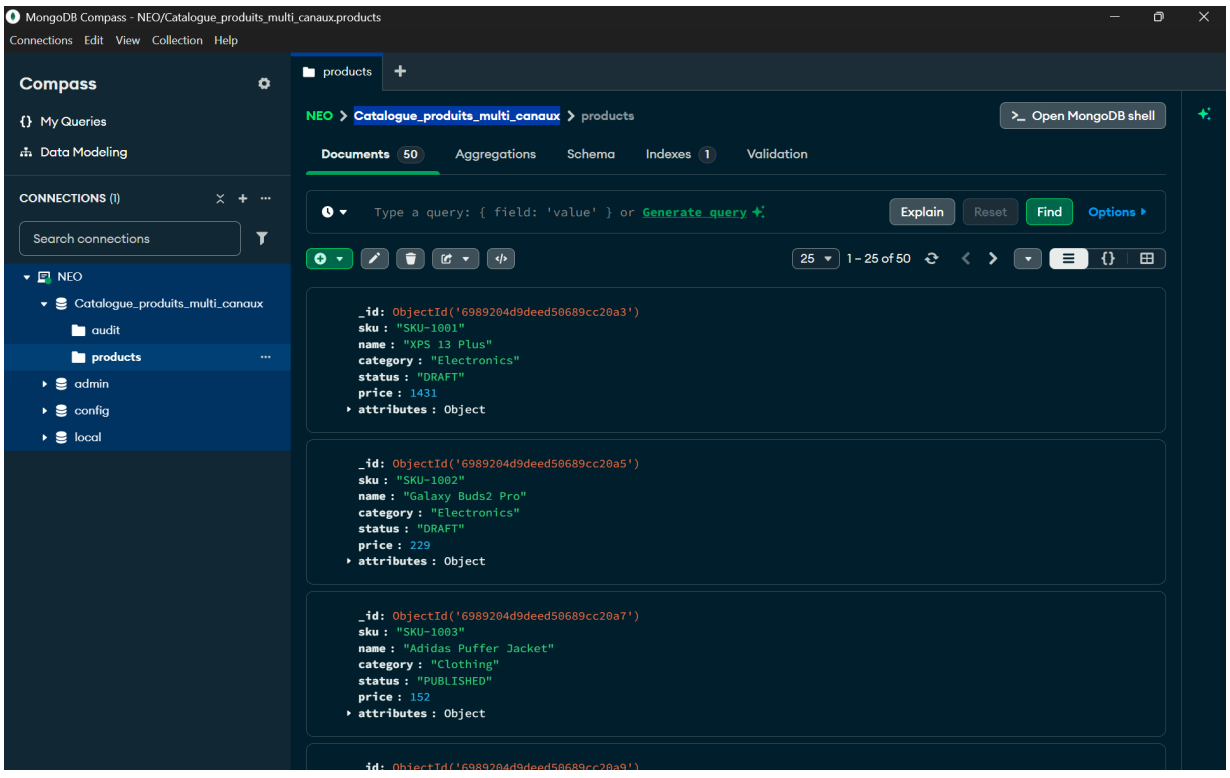


Figure 2: 50 documents inserted in the products collection.

4.3 Index creation :

Launch of the `index.py` script.

```
python index.py
```

— A **UNIQUE** index on the `sku` field to prevent duplicates.

— A **TEXT** index on `name` and `brand` for the search bar.

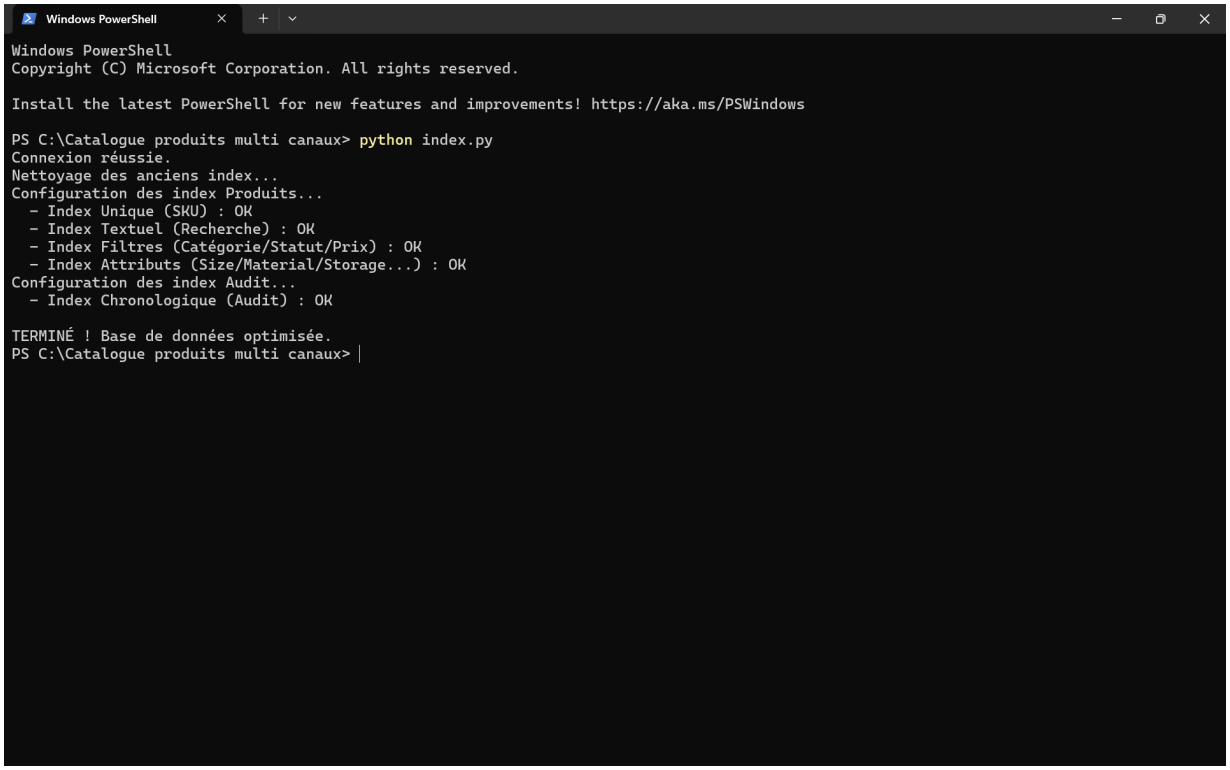


Figure 3: Terminal showing the successful creation of the different indexes.

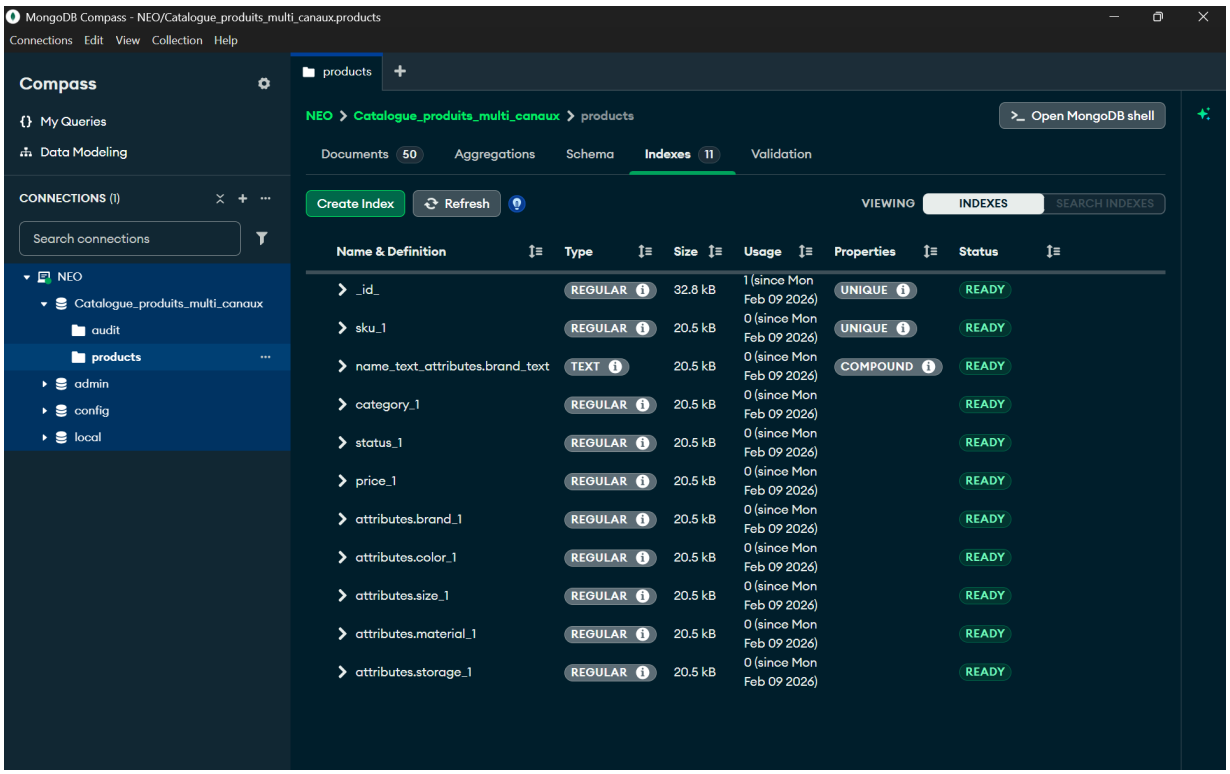


Figure 4: Visual confirmation of the indexes in compass.

4.4 Application Launch :

Starting the Streamlit server. The application is accessible locally on port 8501.

```
python -m streamlit run app.py
```

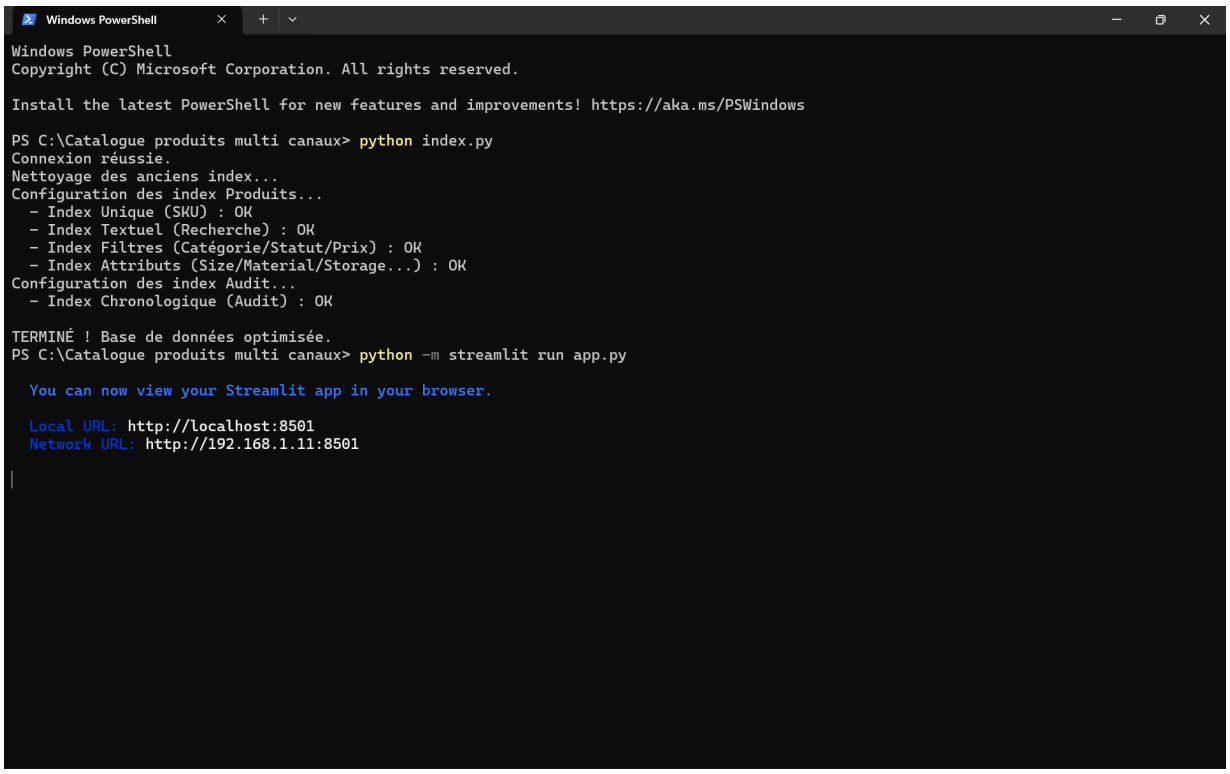


Figure 5: Starting the streamlit server.

4.5 Catalog Interface :

The home page displays the product list. We can clearly see the products in [**LIVE**] and those in [**DRAFT**]. The search bar uses the previously created text index.

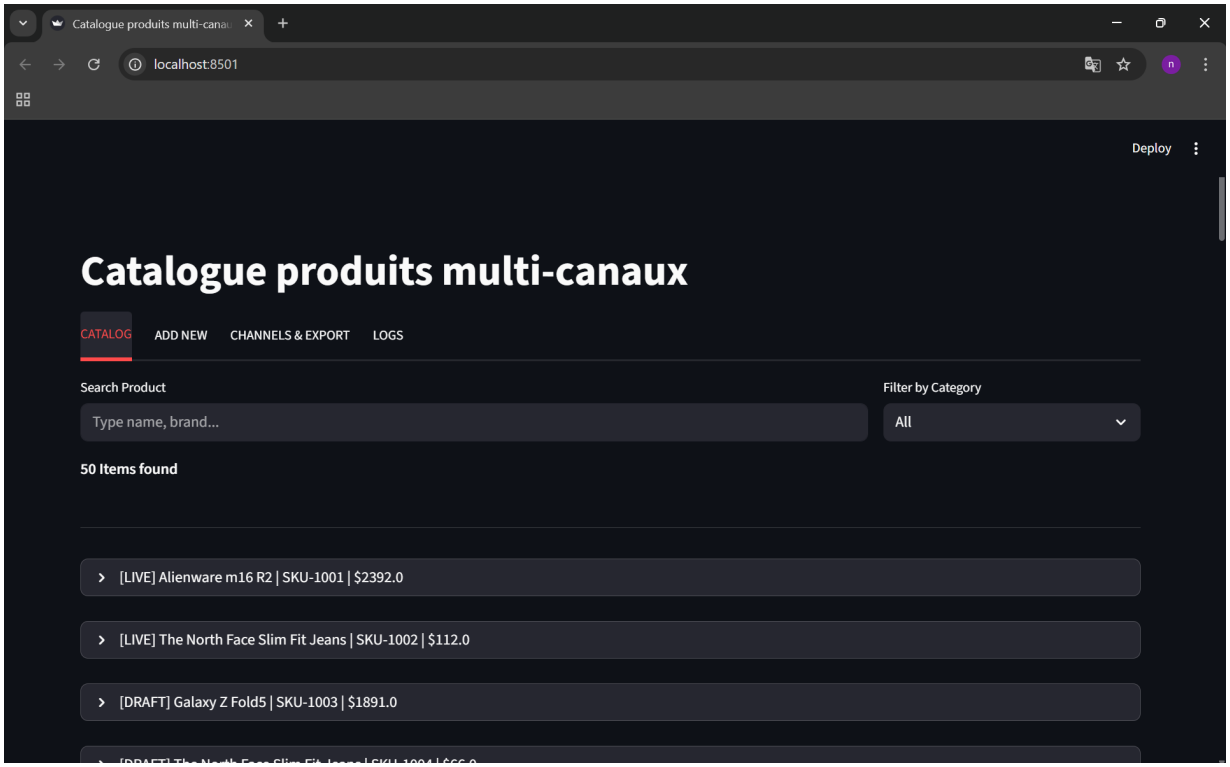


Figure 6: Global view of the multi-channel product catalog.

4.6 Quality workflow :

Demonstration of the quality rule. **Error case :** Below, an incomplete product. The "Storage" field is missing for an electronic product. The system blocks the publication and displays a red alert.

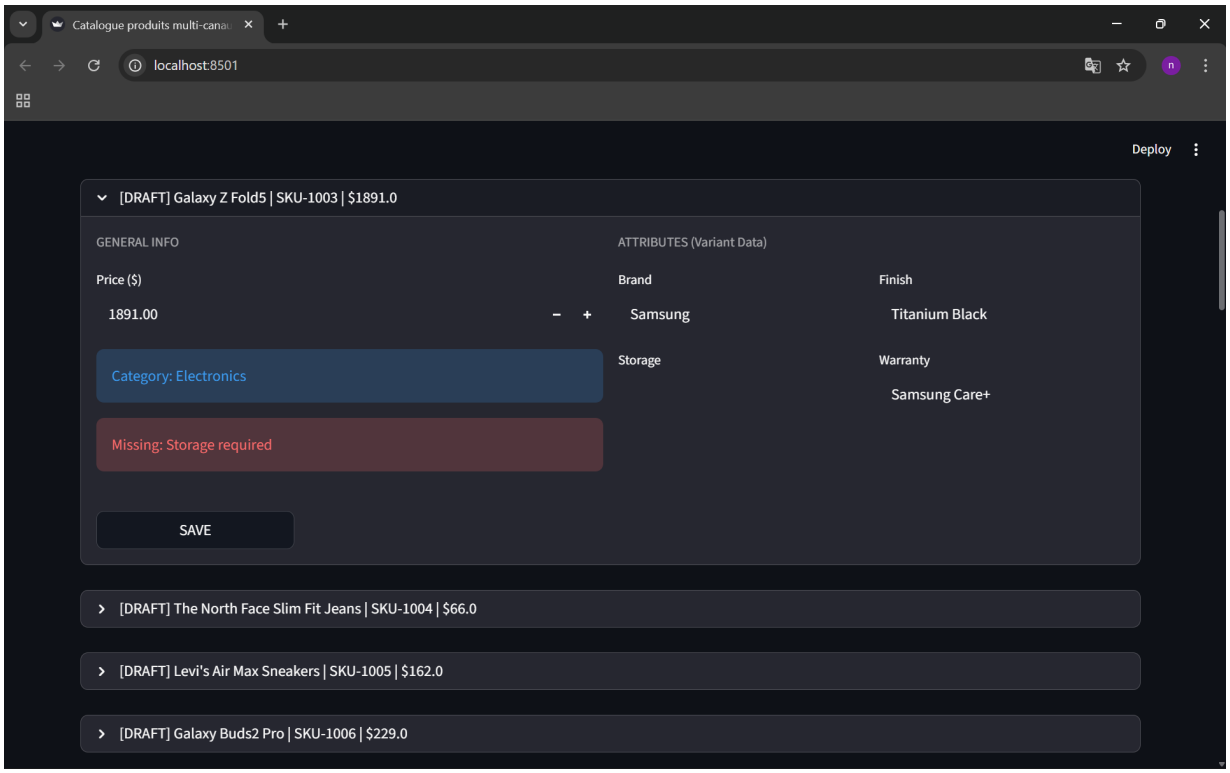


Figure 7: Failed validation : The publish button is missing, the error is explicit.

Valid case : Once the data is corrected, the red banner disappears and the **"PUBLISH TO CHANNELS"** button becomes available.

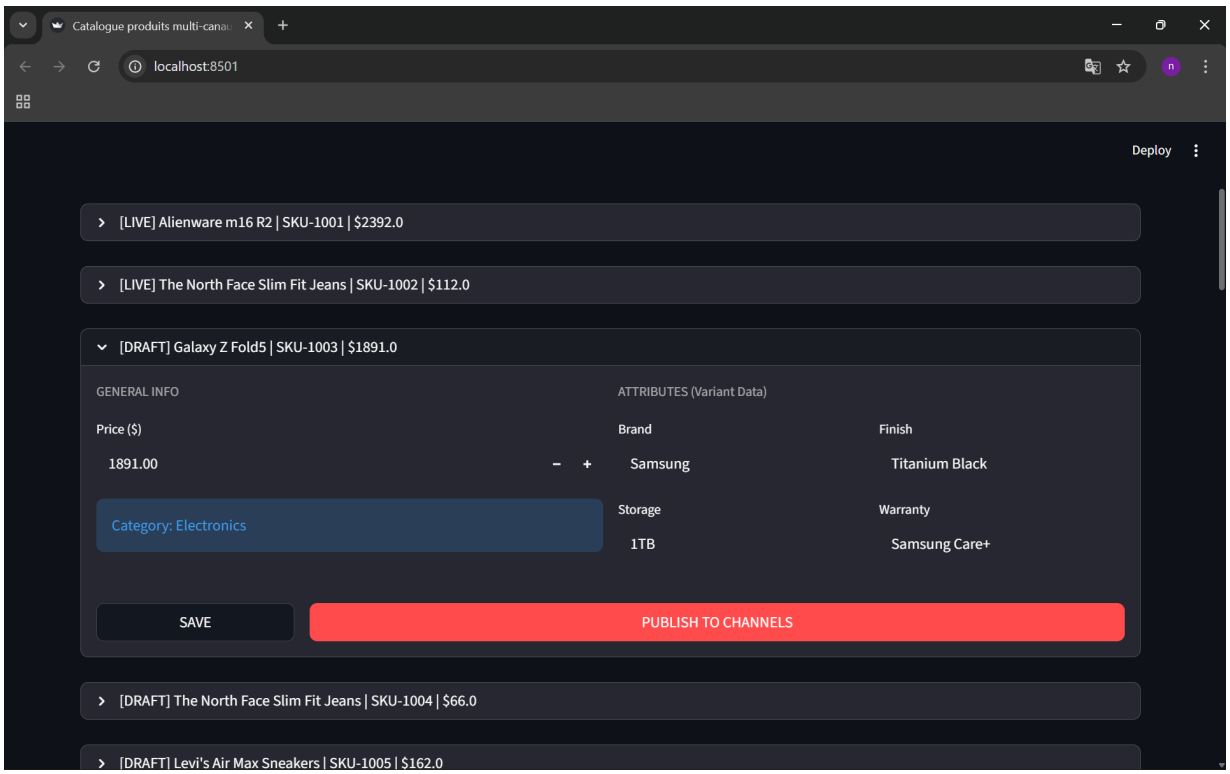


Figure 8: Successful validation : The product is ready to be broadcasted.

4.7 Manual addition of a product :

The "ADD NEW" tab allows business teams to create a product manually. The system generates a draft that will then need to be validated.

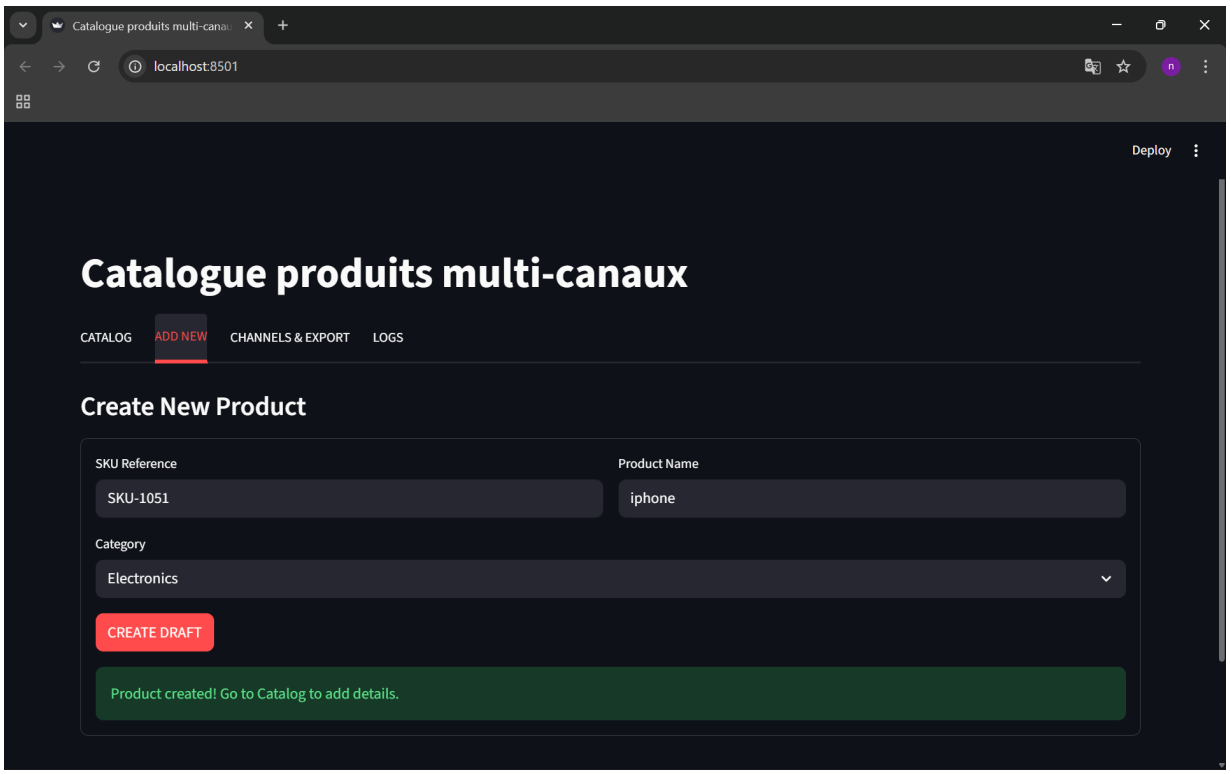


Figure 9: Creation form and success message.

4.8 Manual entry validation :

If the user tries to validate this draft without filling in the mandatory fields (Price, Storage, Warranty), the system blocks the action. We see here the previously created iphone, blocked with multiple validation errors.

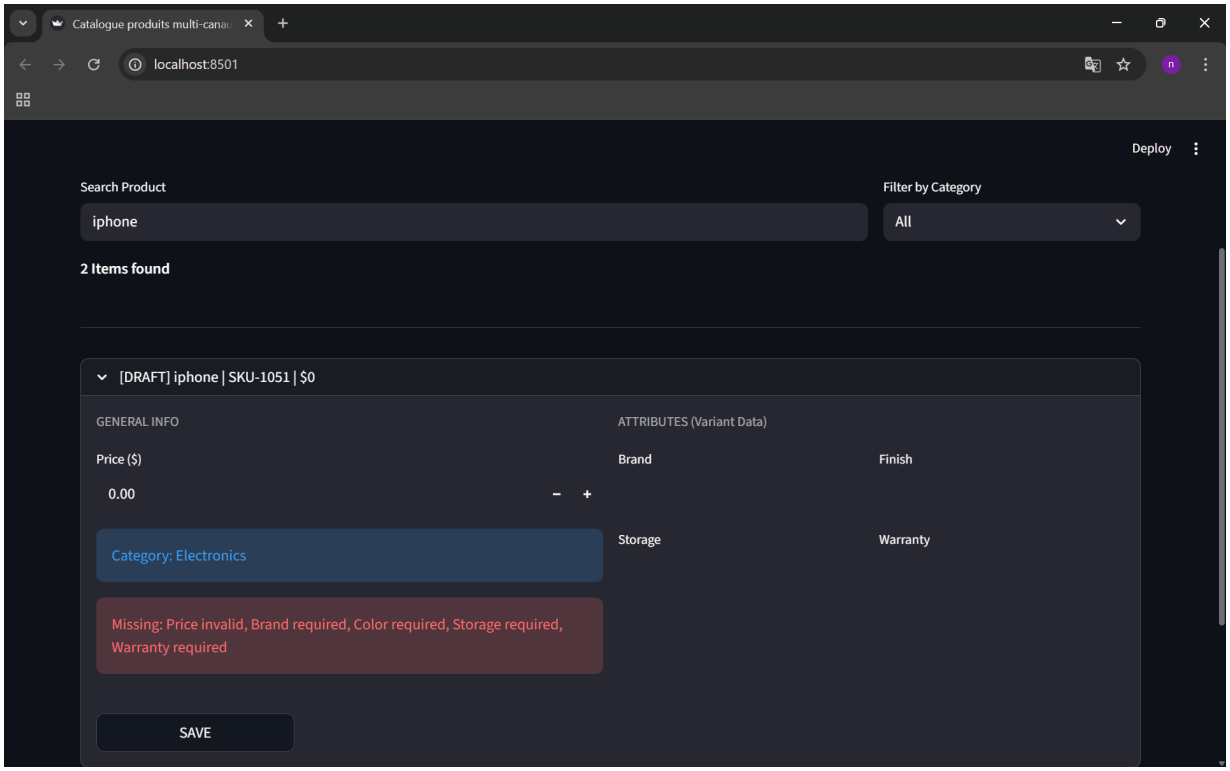


Figure 10: List of missing fields blocking publication.

4.9 Traceability :

All actions (Creation, Update, Publication) are logged. The LOGS tab allows seeing what is done and when.

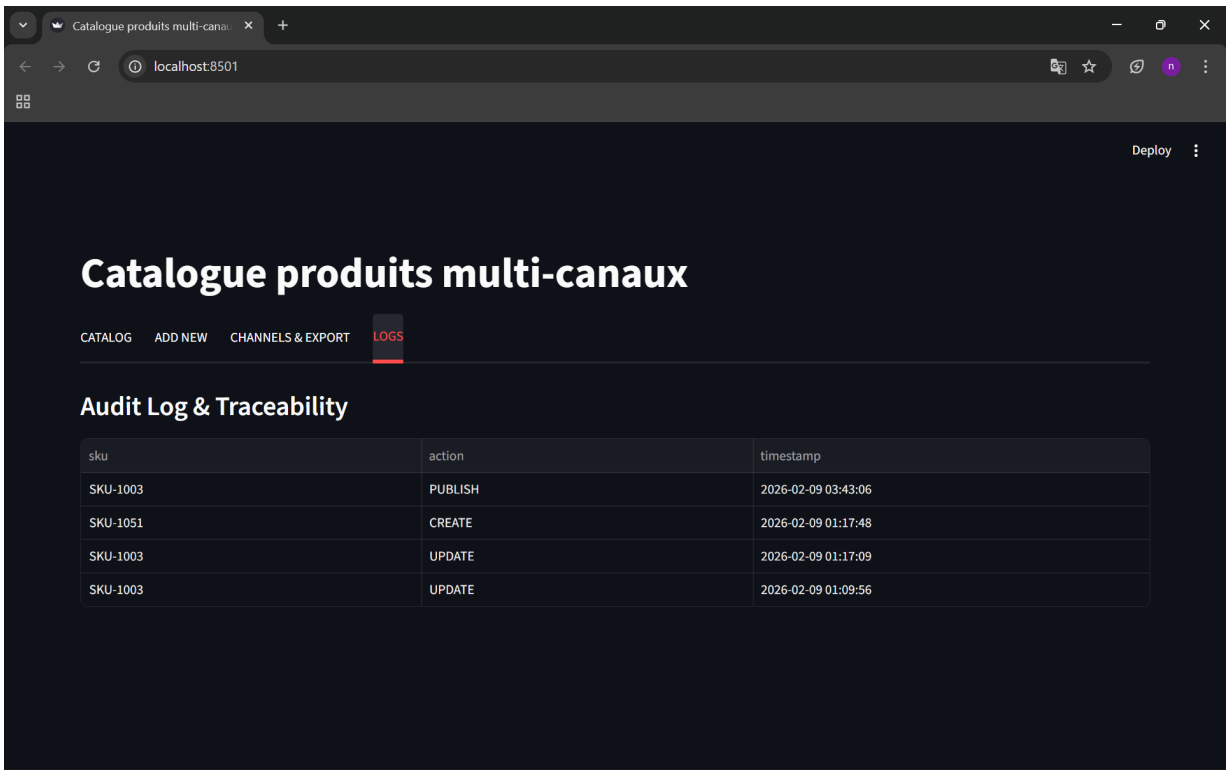


Figure 11: Action history (CREATE, UPDATE, PUBLISH) with timestamp.

4.10 Multi-channel export :

The "CHANNELS & EXPORT" module allows generating data feeds. The user chooses the target channel (Website, Marketplace or Social media).

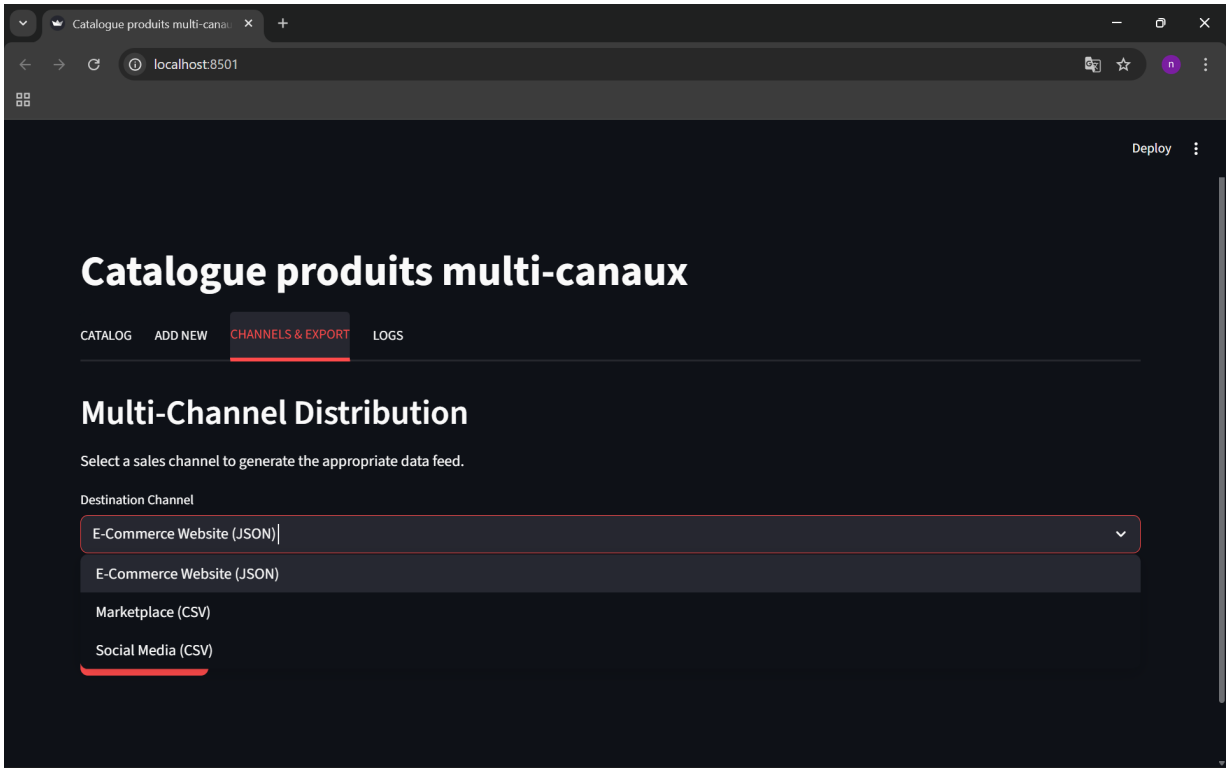


Figure 12: Sales channel selection menu.

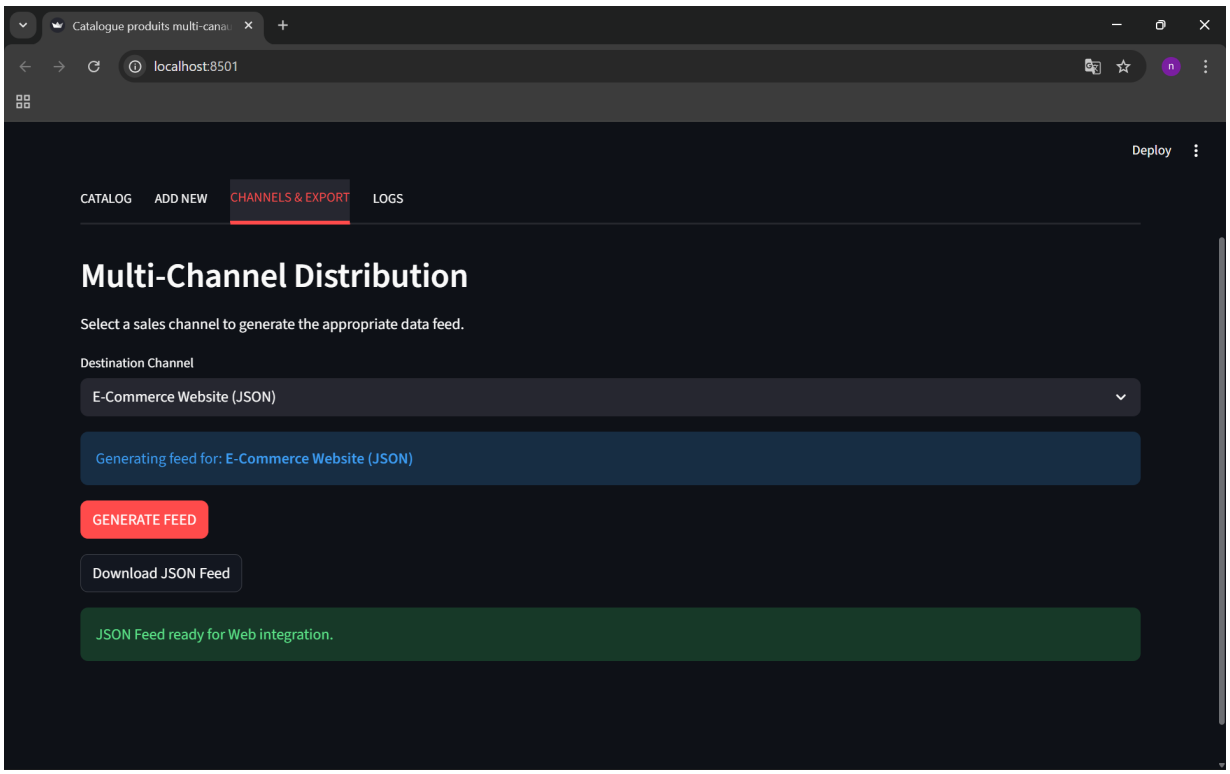


Figure 13: Successful generation of a JSON feed for the ecommerce site.

4.11 Verification of generated files :

This step validates the main objective of the project : the ability to structure data differently depending on the recipient.

1. Web Feed (JSON) : The file keeps the nested structure. Technical attributes are encapsulated in the `attributes` object.

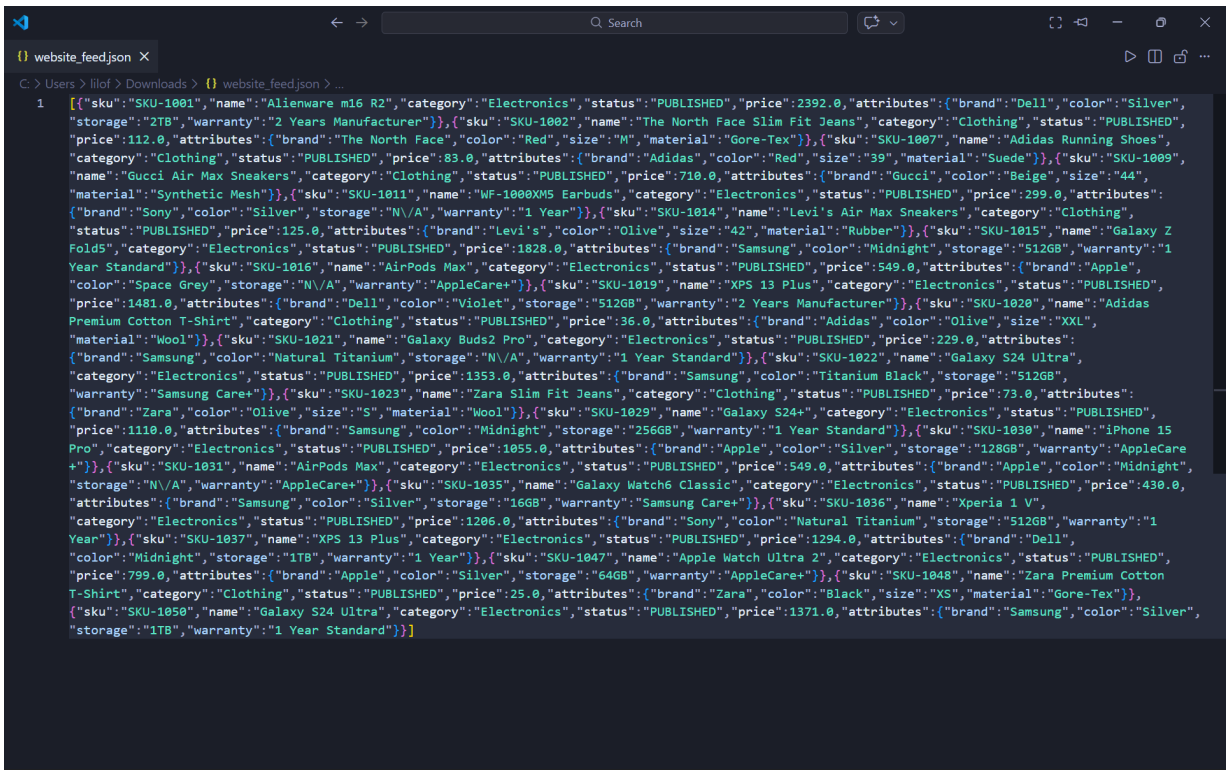
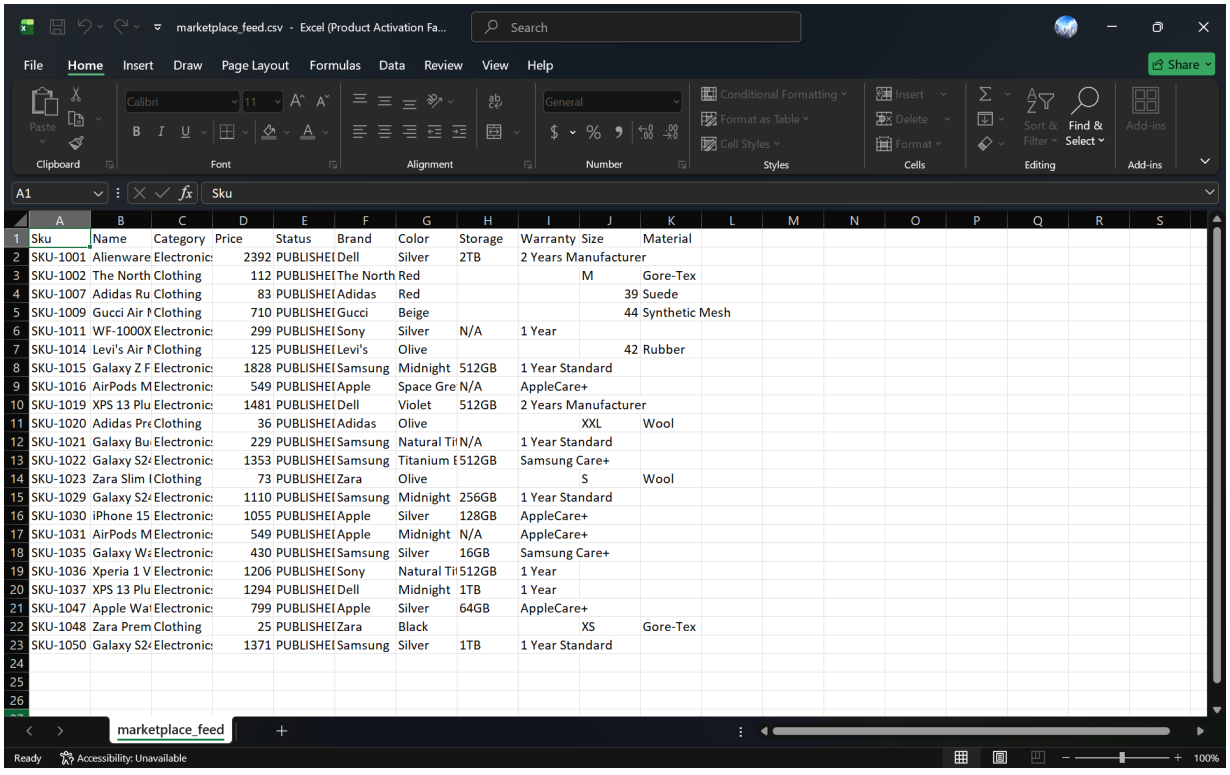


Figure 14: Generated JSON file : Hierarchical structure preserved.

2. Marketplace Feed (Flattened CSV) : For marketplaces, the JSON has been "flattened". Fields like `Storage`, `Warranty` or `Material` have become distinct columns.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	Sku	Name	Category	Price	Status	Brand	Color	Storage	Warranty	Size	Material								
2	SKU-1001	Alienware	Electronics	2392	PUBLISHED	Dell	Silver	2TB	2 Years	Manufacturer									
3	SKU-1002	The North	Clothing	112	PUBLISHED	The North	Red			M	Gore-Tex								
4	SKU-1007	Adidas Ru	Clothing	83	PUBLISHED	Adidas	Red			39	Suede								
5	SKU-1009	Gucci Air M	Clothing	710	PUBLISHED	Gucci	Beige			44	Synthetic Mesh								
6	SKU-1011	WF-1000X	Electronics	299	PUBLISHED	Sony	Silver	N/A	1 Year										
7	SKU-1014	Levi's Air M	Clothing	125	PUBLISHED	Levi's	Olive			42	Rubber								
8	SKU-1015	Galaxy Z F	Electronics	1828	PUBLISHED	Samsung	Midnight	512GB	1 Year	Standard									
9	SKU-1016	AirPods M	Electronics	549	PUBLISHED	Apple	Space Gre	N/A	AppleCare+										
10	SKU-1019	XPS 13 Plu	Electronics	1481	PUBLISHED	Dell	Violet	512GB	2 Years	Manufacturer									
11	SKU-1020	Adidas Pre	Clothing	36	PUBLISHED	Adidas	Olive			XXL	Wool								
12	SKU-1021	Galaxy Bu	Electronics	229	PUBLISHED	Samsung	Natural Tit	N/A	1 Year	Standard									
13	SKU-1022	Galaxy S2	Electronics	1353	PUBLISHED	Samsung	Titanium E	512GB	Samsung	Care+									
14	SKU-1023	Zara Slim I	Clothing	73	PUBLISHED	Zara	Olive			S	Wool								
15	SKU-1029	Galaxy S2	Electronics	1110	PUBLISHED	Samsung	Midnight	256GB	1 Year	Standard									
16	SKU-1030	iPhone 15	Electronics	1055	PUBLISHED	Apple	Silver	128GB	AppleCare+										
17	SKU-1031	AirPods M	Electronics	549	PUBLISHED	Apple	Midnight	N/A	AppleCare+										
18	SKU-1035	Galaxy W	Electronics	430	PUBLISHED	Samsung	Silver	16GB	Samsung	Care+									
19	SKU-1036	Xperia 1 V	Electronics	1206	PUBLISHED	Sony	Natural Tit	512GB	1 Year										
20	SKU-1037	XPS 13 Plu	Electronics	1294	PUBLISHED	Dell	Midnight	1TB	1 Year										
21	SKU-1047	Apple Wa	Electronics	799	PUBLISHED	Apple	Silver	64GB	AppleCare+										
22	SKU-1048	Zara Prem	Clothing	25	PUBLISHED	Zara	Black			XS	Gore-Tex								
23	SKU-1050	Galaxy S2	Electronics	1371	PUBLISHED	Samsung	Silver	1TB	1 Year	Standard									
24																			
25																			
26																			

Figure 15: Detailed technical columns.

3. Social Networks Feed (Simple CSV) : For facebook/instagram, the file is simplified to

the maximum (SKU, Name, Price, Category) to lighten advertising processing.

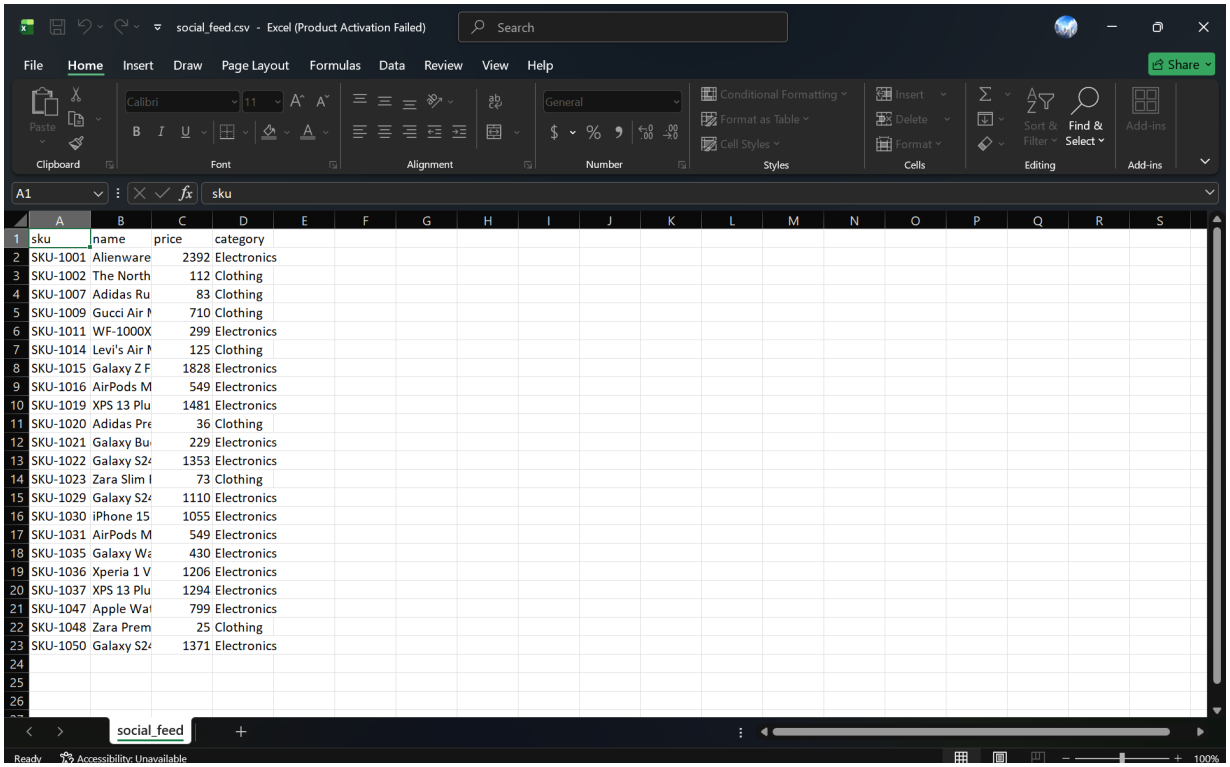


Figure 16: Simplified format.

5 Conclusion and perspectives :

This project allowed the realization of a functional multi-channel product catalog answering the problem of heterogeneous data management. The use of mongoDB proved relevant to manage product polymorphism, while python facilitated the creation of custom export feeds.

Perspectives for improvement :

— **Artificial Intelligence :** Integration of an NLP model to automatically categorize products during import (e.g., detecting that "iphone" goes into "Electronics").

— **REST API :** Develop an API so that the ecommerce site can query the catalog in real time, rather than using static JSON exports.

— **Media management :** Add support for the upload and storage of product images (via Amazon S3 or GridFS).