Zrozumienie i Implementacja Zarządzania Produktami w Shoper REST API: Atrybuty, Opcje i Stany Magazynowe

Wprowadzenie: Analiza Problemu i Kontekst Techniczny

Integracja z platformami e-commerce poprzez REST API wymaga precyzyjnego zrozumienia modelu danych i logiki biznesowej danego systemu. Zapytanie dewelopera budującego aplikację do publikowania produktów w Shoper ujawnia fundamentalne wyzwania związane z niejasnościami w oficjalnej dokumentacji API. Problemy te nie wynikają z braku wiedzy technicznej, lecz z istotnych niespójności i luk w definicjach kluczowych zasobów.

Analiza materiałów źródłowych wskazuje, że główna dezorientacja wynika z faktu, iż oficjalna dokumentacja w zasobie Resources podaje *identyczną* definicję dla "Options" (Opcji) i "Attributes" (Atrybutów): "An attribute is a thing describing the product. For example, color or material.".[1] Ten błąd w dokumentacji maskuje fundamentalną różnicę architektoniczną, która jest kluczowa dla prawidłowego zarządzania produktami.

Niniejszy raport techniczny ma na celu systematyczne rozwiązanie tych niejasności. Przeprowadzona zostanie dekonstrukcja logiki platformy Shoper, aby definitywnie rozróżnić te dwie koncepcje. Następnie, raport przedstawi szczegółową analizę przepływu pracy (workflow) oraz, co kluczowe, rozstrzygnie sprzeczności w dokumentacji dotyczące wymaganej struktury JSON, dostarczając deweloperowi kompletnego i gotowego do implementacji przewodnika.

I. Dekonstrukcja Logiki Shoper: Definitywne

Rozróżnienie "Attributes" (Atrybuty) i "Options" (Opcje)

Błędna definicja w dokumentacji [1] jest punktem wyjścia do zrozumienia, dlaczego te dwa zasoby są mylone i jak należy je poprawnie interpretować. W praktyce, "Atrybuty" i "Opcje" służą do radykalnie różnych celów i operują na innych zasobach API.

Definicja Funkcjonalna 1: "Attributes" (Atrybuty) jako Cechy Informacyjne

"Atrybuty" (Attributes) w API Shoper definiują **statyczne, informacyjne cechy produktu**. Są to dane opisowe, które klient widzi w specyfikacji technicznej, takie jak materiał wykonania, gwarancja, wymiary czy przeznaczenie.

Analiza zasobu Attributes [2] pokazuje, że jego struktura zawiera pola takie jak name (np. "Materiał"), description (opis) oraz type (typ pola: 0 - pole tekstowe, 1 - checkbox, 2 - lista rozwijana). Co najważniejsze, w definicji tego zasobu *nie ma żadnych pól ani mechanizmów* bezpośrednio powiązanych z zarządzaniem stanem magazynowym (SKU), kodem EAN czy ceną produktu. Atrybuty służą do opisywania, a nie do tworzenia wariantów handlowych.

Definicja Funkcjonalna 2: "Options" (Opcje) jako Definicje Wariantów Magazynowych

"Opcje" (Options) są fundamentem do tworzenia **wariantów produktu** – czyli różnych, sprzedawalnych wersji tego samego produktu bazowego, które różnią się np. rozmiarem czy kolorem.

Kluczowy dowód na tę różnicę znajduje się w definicji zasobu Options [3], który zawiera pole stock: boolean. Dokumentacja opisuje je jako "stock modifier". To pole sygnalizuje systemowi, że dana opcja (np. "Rozmiar") oraz jej wartości (np. "S", "M", "L") będą używane do generowania odrębnych stanów magazynowych.

Tworzenie wariantu (SKU) w Shoper to proces oparty na trzech filarach, który można nazwać

"Trójkątem Wariantów":

1. **Zasób Options (Opcje):** Definiuje *oś* wariancji (np. "Kolor", option_id: 20).[3]
2. **Zasób Option Values (Wartości Opcji):** Definiuje *punkty* na tej osi (np. "Czerwony", value_id: 200; "Niebieski", value_id: 201). Chociaż dokumentacja tego zasobu nie została szczegółowo przeanalizowana, jego istnienie jest implikowane przez zasób Product Stocks.
3. **Zasób Product Stocks (Stany Magazynowe Produktu):** Jest to kluczowy zasób, który *tworzy* fizyczny, sprzedawalny wariant (SKU). Łączy on produkt bazowy (product_id) z konkretną kombinacją opcji i wartości.[4]

Podsumowując, deweloper nie powinien traktować zasobu Options jako alternatywy dla Attributes. "Atrybuty" to cechy opisowe, natomiast "Opcje" to definicje mechanizmu wariancji, który jest implementowany przez zasób Product Stocks.

II. Zarządzanie Cechami Informacyjnymi: Głęboka Analiza Zasobu "Attributes"

Aby poprawnie zarządzać atrybutami informacyjnymi, konieczne jest zrozumienie struktury samego zasobu Attributes, który służy do *definiowania* atrybutów dostępnych w systemie (nie do przypisywania ich do produktu, co odbywa się poprzez zasób Products).

Struktura Zasobu "Attributes"

Zgodnie z dokumentacją [2], struktura obiektu Attribute jest następująca:

- attribute_id: integer: Unikalny identyfikator atrybutu.
- name: string: Nazwa atrybutu (np. "Materiał").
- description: string: Opis atrybutu.
- attribute_group_id: integer: Identyfikator grupy, do której atrybut należy. Atrybuty muszą być zorganizowane w grupy.[5]
- type: integer: Definiuje sposób prezentacji i wprowadzania danych [2]:
  - 0: Pole tekstowe (text field).
  - 1: Pole wyboru (checkbox).
  - 2: Lista rozwijana (drop down).
- active: boolean: Status aktywności.

- default: string: Wartość domyślna.
- options: string: Tablica opcji dla atrybutu typu "select" (lista rozwijana).

Rozwiązanie konfliktu nazw: Attributes.options vs. Zasób Options

Pole options wewnątrz zasobu Attribute jest kolejnym źródłem potencjalnej dezorientacji. Analiza [2] precyzuje, że to pole (zdefiniowane jako string) jest powiązane *wyłącznie* z atrybutami typu type: 2 (lista rozwijana).

Jest to po prostu predefiniowana lista wartości tekstowych (np. ``), które użytkownik będzie mógł wybrać podczas przypisywania tego atrybutu do produktu.

Należy kategorycznie stwierdzić, że pole Attributes.options **nie ma żadnego technicznego ani logicznego powiązania** z głównym zasobem Options [3] ani z zasobem Product Stocks.[4] Zasób Options używa option_id i value_id do tworzenia wariantów magazynowych, podczas gdy Attributes.options to tylko lista predefiniowanych *stringów* dla pola typu "select" w atrybucie informacyjnym.

III. Zarządzanie Wariantami Produkcyjnymi: Głęboka Analiza Zasobów "Options" i "Product Stocks"

Ta sekcja wyjaśnia poprawny proces tworzenia sprzedawalnych wariantów (SKU) produktu, co jest rzeczywistym celem korzystania z zasobu "Opcje".

Krok 1: Definiowanie Osi Wariantów (Zasób "Options")

Jak wspomniano, zasób Options [3] służy do definiowania osi wariancji. Kluczowe pola to option_id, group_id (grupa opcji), type (typ pola, np. select, radio, color) oraz stock: boolean. Ustawienie stock: true jest niezbędne, aby system wiedział, że ta opcja będzie generować oddzielne stany magazynowe.

Krok 2: Definiowanie Wartości Opcji (Zasób "Option Values")

Każda "Opcja" (np. "Kolor") musi mieć zdefiniowane "Wartości Opcji" (np. "Czerwony", "Niebieski"). Każda z tych wartości będzie posiadać własne, unikalne value_id. Te identyfikatory są niezbędne do wykonania Kroku 3.

Krok 3: Tworzenie Wariantu SKU (Zasób "Product Stocks")

Jest to kluczowy zasób dla dewelopera tworzącego produkty z wariantami. Zamiast dodawać "opcje" do produktu, deweloper musi tworzyć nowe *instancje* Product Stock dla *każdej* kombinacji opcji.

Analiza dokumentacji metody insert dla Product Stocks [4] pokazuje, że payload żądania POST przyjmuje m.in.:

- product_id: integer: Identyfikator produktu bazowego, z którym wariant jest powiązany.
- price: float (oraz price_type: integer): Cena (lub różnica w cenie) dla tego konkretnego wariantu.
- stock: float: Stan magazynowy *tylko dla tego wariantu*.[6]
- code: string: Nowy, unikalny kod SKU dla wariantu.
- ean: string: Unikalny kod EAN dla wariantu.[7]
- options: array: Kluczowy mechanizm powiązania. Jest to tablica obiektów, która definiuje wariant.

Struktura tablicy options w Product Stocks jest następująca [4]:

JSON

```
"options": [
 {
  "option_id": 10,
  "value_id": 100
 },
 {
  "option_id": 20,
  "value_id": 200
```

```
  }
]
```

Powyższy przykład tworzy wariant dla produktu product_id, który odpowiada kombinacji opcji 10 (np. "Rozmiar") o wartości 100 (np. "L") ORAZ opcji 20 (np. "Kolor") o wartości 200 (np. "Czerwony").

Tabela 1: Porównanie Funkcjonalne: Attributes vs. Options vs. Product Stocks

Poniższa tabela syntetyzuje analizę z sekcji I-III, wizualnie podsumowując różnice między trzema mylonymi koncepcjami.

Koncepcja	Główny Zasób API	Przeznaczenie	Wpływ na Magazyn/ Cenę/SKU?	Przykład Użycia	Źródła
Atrybuty (Attributes)	/attributes	Cechy informacyjne, opisowe.	**Nie**	"Materiał: Bawełna", "Gwarancja: 24 miesiące"	[2]
Opcje (Options)	/options	Definicja *osi* wariantów.	**Tak** (poprzez pole stock: boolean)	"Rozmiar", "Kolor" (jako *koncepcje*)	[3]
Stany Magazynowe (Product Stocks)	/product-stocks	Fizyczny, sprzedawalny wariant (SKU).	**Tak** (definiuje własną cenę, EAN i stan stock)	"T-shirt, Rozmiar L, Kolor Czerwony" (jako *instancja*)	[4]

IV. Definitywny Przepływ Pracy (Workflow) Tworzenia Produktu

Ta sekcja odpowiada na pytania "czy wysyłać atrybuty po utworzeniu produktu czy w trakcie" oraz "Użyć POST czy PUT".

Metodologia: POST vs. PUT

Terminologia API Shoper mapuje metody zasobów na standardowe czasowniki HTTP [1]:

- insert = POST (Tworzenie nowego obiektu)
- update = PUT (Modyfikacja/aktualizacja istniejącego obiektu)

Zalecany Przepływ Pracy: "Separation of Concerns"

Użytkownik pyta, czy może dodać atrybuty *podczas* tworzenia produktu, czyli w jednym żądaniu POST /products. Dokumentacja jest w tej kwestii niejasna [1] i, jak zostanie wykazane w Sekcji V, zawiera krytyczne sprzeczności dotyczące struktury JSON dla atrybutów w żądaniu POST /products/insert [8] w porównaniu do głównego zasobu Products.[9]

Próba wysłania wszystkich danych w jednym, złożonym żądaniu POST jest ryzykowna i podatna na błędy z powodu tych niespójności.

Najbezpieczniejszym, najbardziej atomowym i niezawodnym przepływem pracy jest **rozdzielenie operacji** (Separation of Concerns). Pozwala to na precyzyjne zarządzanie każdym aspektem produktu i ułatwia debugowanie.

Zalecany Przepływ Pracy (Krok po Kroku):

1. **(Opcjonalnie) Definiowanie globalnych atrybutów/opcji:**
    - Jeśli atrybuty (np. "Materiał") lub opcje (np. "Rozmiar") jeszcze nie istnieją w systemie Shoper, należy je jednorazowo zdefiniować za pomocą POST /attributes [2] oraz POST /options i POST /option-values.[3] Aplikacja powinna pobrać i przechować zwrócone id.
2. **Krok 1: Stworzenie Produktu Bazowego (POST).**

    - Wykonaj żądanie POST /products.[8]
    - **Payload JSON:** Prześlij *minimalny* zestaw danych wymagany do stworzenia produktu: np. name, code, category_id, producer_id, unit_id, tax_id.
    - *Zalecenie: Nie dołączaj* klucza attributes na tym etapie, aby uniknąć problemów ze sprzeczną dokumentacją.
    - **Odpowiedź:** Odbierz odpowiedź i zapisz product_id nowo utworzonego produktu.
3. **Krok 2: Przypisanie Atrybutów Informacyjnych (PUT).**
    - Wykonaj żądanie PUT /products/{id}, gdzie id to product_id z Kroku 1.
    - **Payload JSON:** W ciele żądania przekaż klucz attributes z poprawną, zagnieżdżoną strukturą JSON (szczegółowo opisaną w Sekcji V).[9]
    - *Rezultat:* Produkt zostaje zaktualizowany o cechy informacyjne (np. "Materiał: Bawełna").
4. **Krok 3: Stworzenie Wariantów Magazynowych (POST do product-stocks).**
    - Dla *każdego* wymaganego wariantu (np. "S, Czerwony", "M, Czerwony", "S, Niebieski" itd.):
    - Wykonaj *osobne* żądanie POST /product-stocks.[4]
    - **Payload JSON:** W ciele żądania przekaż product_id (z Kroku 1) oraz tablicę options [4] definiującą ten konkretny wariant (wraz z jego unikalnym code, ean, price, stock).

Odpowiedzi na pytania użytkownika:

- Atrybuty (informacyjne) wysyłaj **PO** utworzeniu produktu.
- Użyj PUT /products/{id} do przypisania atrybutów.
- Użyj POST /product-stocks do stworzenia wariantów (Opcji).

V. Rozwiązanie Sprzeczności w Dokumentacji: Poprawna Struktura JSON dla Atrybutów

To jest kluczowy problem techniczny blokujący dewelopera. Analiza dokumentacji ujawnia trzy różne, oficjalnie udokumentowane struktury JSON dla "atrybutów".

Identyfikacja Problemu: Trzy Sprzeczne Struktury

1. **Struktura A (Zagnieżdżona, z Grupą):**
    - **Źródło:** Dokumentacja głównego zasobu Products (/resources/products).[9]
    - **Opis:** "a nested associative array - keys of main array = attribute group identifiers,

values: an associative array (keys: attribute identifiers, values: attribute value)".

- **Format JSON:**

```JSON
{
 "attributes": {
  "(group_id)": {
   "(attribute_id)": "value"
  }
 }
}
```

2. **Struktura B (Płaska, bez Grupy):**
   - **Źródło:** Dokumentacja metody products: insert (/resources/products/insert).[8]
   - **Opis:** "an associative array (keys: attribute identifiers, values: attribute value)".
   - **Format JSON:**

```JSON
{
 "attributes": {
  "(attribute_id)": "value"
 }
}
```

3. **Struktura C (Tablica Obiektów w Webhooku):**
   - **Źródło:** Dokumentacja webhooka product.create.[10]
   - **Opis:** Format, w jakim API *zwraca* dane o atrybutach.
   - **Format JSON:**

```JSON
{
 "attributes": [
  {
   "id": "(group_id)",
   "attributes": [
    {
     "id": "(attribute_id)",
     "value": "value"
    }
   ]
  }
 ]
}
```

Analiza i Werdykt

Struktura B [8] jest anomalią. Pomija ona attribute_group_id, który jest fundamentalnym elementem organizacji atrybutów w Shoper (co potwierdza Struktura A i C). Użycie tej struktury jest wysoce ryzykowne, ponieważ attribute_id mogą nie być globalnie unikalne, a system najprawdopodobniej wymaga kontekstu grupy. Jest to niemal na pewno błąd w dokumentacji strony /products/insert lub niebezpieczne uproszczenie.

Struktury A i C są koncepcyjnie tożsame – obie przekazują te same trzy kluczowe informacje: ID grupy, ID atrybutu i Wartość. Różnią się jedynie formatem (mapa asocjacyjna vs tablica obiektów).

Zgodnie z zalecanym przepływem pracy (Sekcja IV), deweloper będzie używał metody PUT /products/{id} do przypisywania atrybutów. Ta metoda jest częścią głównego zasobu Products, dla którego obowiązuje dokumentacja ze **Struktury A**.[9]

Definitywne Zalecenie:
Deweloper powinien zignorować sprzeczną dokumentację metody products: insert (Struktura B).8
Poprawna, bezpieczna i zalecana struktura JSON do *przypisywania* atrybutów do produktu (w żądaniu PUT /products/{id}) to **zagnieżdżona tablica asocjacyjna (Struktura A)**, gdzie kluczem zewnętrznym jest ID grupy atrybutów, a kluczem wewnętrznym jest ID atrybutu.

VI. Przewodnik Implementacyjny: Kompletny Przykład Tworzenia Produktu

Poniższy scenariusz łączy wszystkie powyższe koncepcje w praktyczny przykład.

Scenariusz: Publikujemy produkt "Techniczny T-shirt".

- **Produkt Bazowy:** category_id: 5, producer_id: 15.
- **Atrybut (Informacyjny):** "Materiał: Poliester". Założenia ID: attribute_group_id: 1 ("Specyfikacja"), attribute_id: 101 ("Materiał").
- **Warianty (Opcje/Magazyn):** "Rozmiar" (S, M) i "Kolor" (Czarny, Biały). Założenia ID:
    - Opcje: option_id: 20 ("Rozmiar"), option_id: 30 ("Kolor").
    - Wartości: value_id: 201 ("S"), value_id: 202 ("M"), value_id: 301 ("Czarny"), value_id:

302 ("Biały").

Krok 1: POST /products (Tworzenie produktu bazowego)

Wyślij żądanie POST https://shop.url/webapi/rest/products z minimalnym payloadem.[8]

Payload JSON:

JSON

```
{
  "producer_id": 15,
  "category_id": 5,
  "unit_id": 1,
  "tax_id": 1,
  "code": "TSHIRT-TECH-BASE",
  "translations": {
    "pl_PL": {
      "name": "Techniczny T-shirt",
      "description": "Opis bazowy produktu."
    }
  }
}
```

Odpowiedź: {"status": 1, "id": 123}. Zapisz product_id: 123.

Krok 2: PUT /products/123 (Przypisywanie atrybutów informacyjnych)

Wyślij żądanie PUT https://shop.url/webapi/rest/products/123, używając **poprawnej Struktury A**.[9]

Payload JSON:

JSON

```
{
  "attributes": {
    "1": {
      "101": "Poliester"
    }
  }
}
```

Rezultat: Produkt 123 ma teraz przypisany atrybut "Materiał: Poliester" w grupie "Specyfikacja".

Krok 3: POST /product-stocks (Tworzenie wariantu S / Czarny)

Wyślij żądanie POST https://shop.url/webapi/rest/product-stocks, używając struktury options z.[4]

Payload JSON:

JSON

```
{
  "product_id": 123,
  "stock": 50,
  "active": true,
  "code": "TSHIRT-TECH-S-BLK",
  "ean": "123456789001",
  "price": 99.99,
  "price_type": 1,
  "options": [
    { "option_id": 20, "value_id": 201 },
    { "option_id": 30, "value_id": 301 }
```

```
 ]
}
```

Rezultat: Utworzono sprzedawalny wariant "S, Czarny" dla produktu 123, z własnym SKU, EAN, ceną i stanem magazynowym.

Krok 4: POST /product-stocks (Tworzenie wariantu M / Biały)

Wyślij kolejne żądanie POST https://shop.url/webapi/rest/product-stocks.[4]

Payload JSON:

JSON

```
{
 "product_id": 123,
 "stock": 75,
 "active": true,
 "code": "TSHIRT-TECH-M-WHT",
 "ean": "1234567890O2",
 "price": 105.00,
 "price_type": 1,
 "options": [
  { "option_id": 20, "value_id": 202 },
  { "option_id": 30, "value_id": 302 }
 ]
}
```

Rezultat: Utworzono drugi sprzedawalny wariant "M, Biały".

Tabela 2: Kluczowe Struktury Danych JSON (Szybki Podręcznik)

Poniższa tabela zestawia dwie kluczowe i różne struktury JSON wymagane w zalecanym

przepływie pracy.

Cel	Endpoint	Kluczowe Pole	Struktura JSON Payload	Źródła
Przypisywanie **Atrybutów** (Cech informacyjnych)	PUT /products/{id}	attributes	{ "attributes": { "(ID_Grupy)": { "(ID_Atrybutu)": "Wartość" } } }	[9]
Tworzenie **Wariantu** (SKU z Opcji)	POST /product-stocks	options	{ "product_id": (ID_Produktu), "code": "...", "stock":..., "options": }	[4]

VII. Podsumowanie Zaleceń i Ostateczna Odpowiedź na Pytanie "ID czy Nazwa"

Niniejszy raport wykazał, że problemy napotkane przez dewelopera są wynikiem znaczących niespójności w dokumentacji API Shoper. Kluczem do sukcesu jest przyjęcie poprawnego modelu koncepcyjnego i ścisłego, rozdzielonego przepływu pracy.

Ostateczna Odpowiedź: "Używać nazwy czy id"

Zapytanie dewelopera dotyczyło również, czy do powiązanych obiektów należy odwoływać się za pomocą id czy tekstowych name. Chociaż jeden z przeanalizowanych fragmentów sugerował, że dokumentacja nie precyzuje tej kwestii [1], analiza przeprowadzona w tym raporcie jednoznacznie obala ten wniosek.

API Shoper jest systemem w pełni opartym na identyfikatorach (ID).

- **Dowody:**
  - Przypisywanie atrybutów (Struktura A) wymaga group_id i attribute_id.[9]
  - Tworzenie wariantów Product Stocks wymaga product_id, option_id i value_id.[4]
  - Pobieranie zasobów (np. Attributes) odbywa się przez GET /attributes/{id}.[11]
  - Webhooki systemowe zwracają dane powiązane przez id grupy i id atrybutu.[10]

Wniosek: Aplikacja kliencka *musi* odwoływać się do wszystkich powiązanych zasobów (kategorii, producentów, atrybutów, grup, opcji, wartości opcji) wyłącznie za pomocą ich unikalnych identyfikatorów liczbowych. Oznacza to, że przed próbą publikacji produktów, aplikacja musi posiadać mechanizm pobierania (np. z endpointów list) i mapowania (lub przechowywania) tych identyfikatorów. Użycie tekstowych name w żądaniach API zakończy się niepowodzeniem.

Lista Kontrolna Implementacji

1. **Rozróżnienie Koncepcyjne:** Traktuj Attributes [2] jako *informacje* (specyfikacje). Traktuj Options [3] i Product Stocks [4] jako system do tworzenia *wariantów* (SKU).
2. **Przepływ Pracy (Workflow):** Zawsze używaj rozdzielonego przepływu pracy:
   - POST /products (tworzenie produktu bazowego, minimalny payload).
   - PUT /products/{id} (przypisywanie atrybutów informacyjnych).
   - POST /product-stocks (tworzenie każdego wariantu (SKU) osobno).
3. **Struktura Atrybutów:** Do przypisywania atrybutów w PUT /products/{id} zawsze używaj zagnieżdżonej struktury z ID grupy: {"attributes": {"(group_id)": {"(attribute_id)": "value"}}}.[9] Zignoruj sprzeczną dokumentację products: insert.[8]
4. **Struktura Wariantów:** Do tworzenia wariantów w POST /product-stocks zawsze używaj tablicy options z identyfikatorami opcji i wartości: {"options": [{"option_id":..., "value_id":...}]}.[4]
5. **Referencje:** Wszystkie powiązania w API Shoper muszą być realizowane przy użyciu identyfikatorów (id), a nie nazw (name).

Cytowane prace

1. Shoper Developers, otwierano: listopada 17, 2025, https://developers.shoper.pl/developers/api/resources
2. Attributes - Shoper Developers, otwierano: listopada 17, 2025, https://developers.shoper.pl/developers/api/resources/attributes
3. Shoper Developers, otwierano: listopada 17, 2025, https://developers.shoper.pl/developers/api/resources/options
4. Shoper Developers, otwierano: listopada 17, 2025, https://developers.shoper.pl/developers/api/resources/product-stocks/insert

5. Attributes: list - Shoper Developers, otwierano: listopada 17, 2025, https://developers.shoper.pl/developers/api/resources/attributes/list
6. Product Stock: update - Shoper Developers, otwierano: listopada 17, 2025, https://developers.shoper.pl/developers/api/resources/product-stocks/update
7. Product Stock: list - Shoper Developers, otwierano: listopada 17, 2025, https://developers.shoper.pl/developers/api/resources/product-stocks/list
8. Products: insert - Shoper Developers, otwierano: listopada 17, 2025, https://developers.shoper.pl/developers/api/resources/products/insert
9. Products - Shoper Developers, otwierano: listopada 17, 2025, https://developers.shoper.pl/developers/api/resources/products
10. product.create - Shoper Developers, otwierano: listopada 17, 2025, https://developers.shoper.pl/developers/webhooks/methods/product-create
11. Attributes: get - Shoper Developers, otwierano: listopada 17, 2025, https://developers.shoper.pl/developers/api/resources/attributes/get