Szczegółowa Analiza Danych Produktów - Rezon.eu

1. Źródła Danych

API: https://rezon-api.vercel.app/api/v1/products

- **Status**: ✅ Dostępne
- **Format**: JSON
- **Liczba produktów**: 323
- **Struktura odpowiedzi**:


```
{
  "status": "success",
  "numberOfProducts": 323,
  "data": {
    "products": [...]
  }
}
```


Strona Web: https://www.rezon.eu/search?search=brelok

- **Status**: ⚠️ Wymaga JavaScript (Next.js)
- **Renderowanie**: Client-side (SPA)
- **Dostęp do danych**: Wymaga browser console lub headless browser

2. Pola Dostępne w API (21 pól)

Pole	Typ	Przykład	Opis
`_id`	String	“66a330b38898d9514e542666”	Unikalny identyfikator MongoDB
`createdAt`	DateTime	“2024-07-26T05:14:27.473Z”	Data utworzenia
`pc_id`	String	“DK42-009B”	Kod produktu (identyfikator biznesowy)
`name`	String	“SCYZORYK DREWNO”	Nazwa produktu
`slug`	String	“scyzoryk-drewno”	URL-friendly nazwa
`category`	String	“akcesoria podróżne”	Kategoria produktu
`price`	Number	9	Cena w PLN
`description`	String	“Drewniany scyzoryk…”	Opis produktu
`imageCover`	String	“/images/akcesoria_podróżne/…”	Główny obraz
`images`	Array	[“/images/…”, “https://www.wholesaleresortaccessories.com/cdn/shop/products/601_Pocket_Knife_-_Copy.png?v=1468962760”]	Wszystkie obrazy
`technology`	Number	3	ID technologii produkcji
`stock`	Number	6000	Stan magazynowy
`stock_optimal`	Number	6000	Optymalny stan
`stock_ordered`	Number	0	Zamówione sztuki
`dimensions`	String	“0 cm x 0 cm”	Wymiary
`new`	Boolean	false	Czy nowy produkt

Pole	Typ	Przykład	Opis
`active`	Boolean	true	Czy aktywny
`form`	String	"/formularze/dorośli1.pdf"	Formularz (opcjonalne)
`compilation`	Boolean	true	Czy kompilacja (opcjonalne)
`compilation_price`	Number	1760	Cena kompilacji (opcjonalne)
`stand`	Boolean	true	Czy ma stojak (opcjonalne)

3. Mapowanie na Bazę Danych

Tabela PRODUCT

```
CREATE TABLE Product (
  id SERIAL PRIMARY KEY,
  identifier VARCHAR(50) UNIQUE,  -- pc_id z API
  index INTEGER,                   -- do ręcznego sortowania
  slug VARCHAR(100) UNIQUE,        -- slug z API
  description TEXT,                -- description z API
  price DECIMAL(10,2),            -- price z API
  imageUrl VARCHAR(500),          -- imageCover z API
  images JSONB,                   -- images z API jako JSON array
  category VARCHAR(100),          -- category z API
  productionPath VARCHAR(200),    -- do uzupełnienia ręcznie
  isActive BOOLEAN DEFAULT true, -- active z API
  new BOOLEAN DEFAULT false       -- new z API
);
```

Tabela INVENTORY

```
CREATE TABLE Inventory (
  id SERIAL PRIMARY KEY,
  productId INTEGER REFERENCES Product(id),
  quantity INTEGER,               -- stock z API
  reserved INTEGER,               -- stock_ordered z API
  optimal INTEGER,                -- stock_optimal z API
  lastUpdated TIMESTAMP DEFAULT NOW()
);
```

4. Przykłady Mapowania Danych

Przykład 1: Scyzoryk Drewno

API → Database

```
// Dane z API
const apiProduct = {
  "_id": "66a330b38898d9514e542666",
  "pc_id": "DK42-009B",
  "name": "SCYZORYK DREWNO",
  "slug": "scyzoryk-drewno",
  "category": "akcesoria podróżne",
  "price": 9,
  "description": "Drewniany scyzoryk wielofukcyjny z pięcioma narzędziami...",
  "imageCover": "/images/akcesoria_podróżne/scyzoryk_drewno1.jpg",
  "images": [
    "/images/akcesoria_podróżne/scyzoryk_drewno1.jpg",
    "/images/akcesoria_podróżne/scyzoryk_drewno2.jpg"
  ],
  "stock": 6000,
  "stock_optimal": 6000,
  "stock_ordered": 0,
  "new": false,
  "active": true
}

// Mapowanie na Product
const productData = {
  identifier: apiProduct.pc_id,           // "DK42-009B"
  slug: apiProduct.slug,                  // "scyzoryk-drewno"
  description: apiProduct.description,    // "Drewniany scyzoryk..."
  price: apiProduct.price,                // 9
  imageUrl: apiProduct.imageCover,        // "/images/akcesoria_podróżne/..."
  images: JSON.stringify(apiProduct.images), // JSON array
  category: apiProduct.category,          // "akcesoria podróżne"
  isActive: apiProduct.active,           // true
  new: apiProduct.new                     // false
}

// Mapowanie na Inventory
const inventoryData = {
  quantity: apiProduct.stock,            // 6000
  reserved: apiProduct.stock_ordered,    // 0
  optimal: apiProduct.stock_optimal      // 6000
}
```

5. Analiza URL-i Obrazów

Typy URL-i w API:

1. **Lokalne ścieżki**: `/images/akcesoria_podróżne/scyzoryk_drewno1.jpg`
2. **Cloudinary CDN**: `https://i.ytimg.com/vi/FvrdQNB1eiE/maxresdefault.jpg

Struktura lokalnych ścieżek:

```
/images/[kategoria]/[nazwa_pliku]
```

Przykłady:

- `/images/akcesoria_podróżne/scyzoryk_drewno1.jpg`
- `/images/akcesoria_podróżne/PIERSIOWKA_1.jpg`
- `/images/akcesoria_podróżne/017.jpg`

Cloudinary URL-e:

```
https://i.ytimg.com/vi/HBKk-URU0po/hq720.jpg?sqp=-oaymwEhCK4FEIIDSFryq4qpAxMIARU-AAAAAGAElAADIQj0AgKJD&rs=AOn4CLD-Dxrqy14lRPs1NfqUfy6W7JUmJw
```

6. Rekomendacje Implementacji

6.1 Synchronizacja Danych

```
// Funkcja synchronizacji z API
async function syncProductsFromAPI() {
  try {
    const response = await fetch('https://rezon-api.vercel.app/api/v1/products');
    const data = await response.json();

    for (const apiProduct of data.data.products) {
      // Sprawdź czy produkt już istnieje
      const existingProduct = await prisma.product.findUnique({
        where: { identifier: apiProduct.pc_id }
      });

      if (existingProduct) {
        // Aktualizuj istniejący
        await updateProduct(existingProduct.id, apiProduct);
      } else {
        // Utwórz nowy
        await createProduct(apiProduct);
      }
    }
  } catch (error) {
    console.error('Błąd synchronizacji:', error);
  }
}
```

6.2 Obsługa Obrazów

```
// Funkcja do konwersji ścieżek obrazów
function processImageUrl(imagePath) {
  if (imagePath.startsWith('http')) {
    return imagePath; // Już pełny URL (Cloudinary)
  }

  if (imagePath.startsWith('/images/')) {
    return `https://i.ytimg.com/vi/o17Fk4Dcn-w/maxresdefault.jpg // Dodaj bazowy URL
  }

  return imagePath;
}

// Przetwarzanie tablicy obrazów
function processImages(images) {
  return images.map(processImageUrl);
}
```

6.3 Mapowanie Kategorii

```
// Mapowanie kategorii z API na nasze kategorie
const categoryMapping = {
  'akcesoria podróżne': 'travel-accessories',
  'breloki': 'keychains',
  'gadżety': 'gadgets'
  // ... inne mapowania
};

function mapCategory(apiCategory) {
  return categoryMapping[apiCategory] || apiCategory.toLowerCase().replace(/\s+/g, '-')
;
}
```

6.4 Walidacja Danych

```
// Schema walidacji dla produktu z API
const apiProductSchema = {
  _id: 'string',
  pc_id: 'string',
  name: 'string',
  price: 'number',
  active: 'boolean',
  // ... inne pola
};

function validateApiProduct(product) {
  // Implementacja walidacji
  return Object.keys(apiProductSchema).every(key =>
    typeof product[key] === apiProductSchema[key]
  );
}
```

7. Przykład Kompletnej Implementacji

7.1 Model Prisma (aktualizacja)

```
model Product {
  id              Int       @id @default(autoincrement())
  identifier      String    @unique // pc_id z API
  index           Int?
  slug            String    @unique
  description     String?
  price           Decimal   @db.Decimal(10, 2)
  imageUrl        String?   // imageCover z API
  images          Json?     // images z API jako JSON
  category        String?
  productionPath String?
  isActive        Boolean   @default(true)
  new             Boolean   @default(false)

  // Dodatkowe pola z API
  technology      Int?
  dimensions      String?
  form            String?
  compilation     Boolean?
  compilationPrice Decimal? @db.Decimal(10, 2)
  stand           Boolean?

  // Relacje
  inventory       Inventory[]
  orderItems      OrderItem[]

  createdAt       DateTime  @default(now())
  updatedAt       DateTime  @updatedAt

  @@map("products")
}

model Inventory {
  id          Int      @id @default(autoincrement())
  productId   Int
  quantity    Int      @default(0)
  reserved    Int      @default(0) // stock_ordered z API
  optimal     Int      @default(0) // stock_optimal z API
  lastUpdated DateTime @default(now())

  product     Product  @relation(fields: [productId], references: [id])

  @@map("inventory")
}
```

7.2 API Route dla Synchronizacji

```
// pages/api/sync/products.js
export default async function handler(req, res) {
  if (req.method !== 'POST') {
    return res.status(405).json({ message: 'Method not allowed' });
  }

  try {
    const response = await fetch('https://rezon-api.vercel.app/api/v1/products');
    const apiData = await response.json();

    let created = 0;
    let updated = 0;

    for (const apiProduct of apiData.data.products) {
      const productData = {
        identifier: apiProduct.pc_id,
        slug: apiProduct.slug,
        description: apiProduct.description,
        price: apiProduct.price,
        imageUrl: processImageUrl(apiProduct.imageCover),
        images: apiProduct.images ? processImages(apiProduct.images) : [],
        category: apiProduct.category,
        isActive: apiProduct.active,
        new: apiProduct.new || false,
        technology: apiProduct.technology,
        dimensions: apiProduct.dimensions,
        form: apiProduct.form,
        compilation: apiProduct.compilation,
        compilationPrice: apiProduct.compilation_price,
        stand: apiProduct.stand
      };

      const inventoryData = {
        quantity: apiProduct.stock || 0,
        reserved: apiProduct.stock_ordered || 0,
        optimal: apiProduct.stock_optimal || 0
      };

      const existingProduct = await prisma.product.findUnique({
        where: { identifier: apiProduct.pc_id }
      });

      if (existingProduct) {
        await prisma.product.update({
          where: { id: existingProduct.id },
          data: productData
        });

        await prisma.inventory.upsert({
          where: { productId: existingProduct.id },
          update: inventoryData,
          create: { ...inventoryData, productId: existingProduct.id }
        });

        updated++;
      } else {
        const newProduct = await prisma.product.create({
          data: productData
        });

        await prisma.inventory.create({
          data: { ...inventoryData, productId: newProduct.id }
```

```
        });

        created++;
      }
    }

    res.status(200).json({
      success: true,
      message: `Synchronizacja zakończona. Utworzono: ${created}, zaktualizowano: ${up-
dated}`
    });

  } catch (error) {
    console.error('Błąd synchronizacji:', error);
    res.status(500).json({ success: false, error: error.message });
  }
}
```

8. Podsumowanie

Zalety API:

- ✅ Kompletne dane produktów (323 produkty)
- ✅ Szczegółowe informacje o stanie magazynowym
- ✅ Strukturalne dane JSON
- ✅ Obrazy w dwóch formatach (lokalne + Cloudinary)
- ✅ Metadane produktów (technologia, wymiary, formularze)

Wyzwania:

- ⚠️ Strona web wymaga JavaScript do analizy
- ⚠️ Różne formaty URL-i obrazów
- ⚠️ Brak bezpośredniego mapowania na niektóre pola DB

Rekomendacje:

1. **Użyj API jako głównego źródła danych** - zawiera wszystkie potrzebne informacje
2. **Zaimplementuj regularną synchronizację** - codziennie lub co kilka godzin
3. **Obsłuż oba typy URL-i obrazów** - lokalne ścieżki i Cloudinary
4. **Mapuj pc_id na identifier** - jako unikalny klucz biznesowy
5. **Przechowuj images jako JSON** - zachowaj elastyczność
6. **Monitoruj zmiany w API** - dodaj logi i powiadomienia o błędach