**Gdańsk, styczeń 2025**

# PromoComparer

Jakub Kapica

Julia Kusy

Wiktoria Wiśniewska

## 1. Opis aplikacji

**PromoComparerAPI** to aplikacja oparta na technologii RESTful APU, której celem jest zarządzanie danymi o promocjach z różnych sklepów. Aplikacja umożliwia analizowanie informacji o gazetkach promocyjnych, promocjach, sklepach oraz kategoriach produktów. System wykorzystuje mechanizmy **web scrapingu**, **OCR**, **sztucznej inteligencji OpenAI** oraz **bazę danych** do przetwarzania i analizy danych.

## 2. Główne funkcjonalności

- **Pobieranie gazetek promocyjnych** – automatyczne scrapowanie danych o gazetkach promocyjnych w formacie PDF.
- **Konwersja PDF na obrazy** – przekształcanie plików PDF na obrazy, aby ułatwić ich analizę.
- **Analiza obrazów z OpenAI** – wykorzystanie **sztucznej inteligencji OpenAI** do ekstrakcji danych o promocjach z przetworzonych obrazów.
- **Zarządzanie promocjami** – przechowywanie i obsługa danych o promocjach dostępnych w różnych sklepach.
- **Zarządzanie sklepami i kategoriami produktów** – strukturalizacja danych o sieciach handlowych i produktach objętych promocjami.
- **Automatyczna aktualizacja danych** – cykliczne pobieranie i przetwarzanie danych w celu utrzymania aktualności informacji o promocjach.

## 3. Architektura systemu

Aplikacja opiera się na **architekturze warstwowej**, w której każda warstwa pełni określoną funkcję:

- **Warstwa API (Controllers)** – odpowiada za obsługę żądań użytkowników oraz zwracanie danych w formacie JSON.
- **Warstwa logiki biznesowej (Services)** – implementuje główne operacje aplikacji, np. pobieranie gazetek, analiza obrazów, obsługa promocji.
- **Warstwa dostępu do danych (Repository/Database)** – realizuje interakcję z bazą danych SQL, przechowując wszystkie informacje o promocjach, gazetkach i sklepach.

## 4. Technologie i narzędzia

- **ASP.NET Core 7** – framework do budowy API.
- **Entity Framework Core** – ORM do zarządzania bazą danych.

- **SQL Server** – relacyjna baza danych.
- **OpenAI API** – analiza obrazów.
- **HtmlAgilityPack** – web scraping.
- **ImageMagick** – konwersja PDF na obrazy.
- **Coravel** – automatyzacja zadań.

## 6. Przepływ danych w systemie

1. **ScrapingController** pobiera dane o gazetkach promocyjnych w formacie PDF ze stron internetowych sklepów.
2. **PdfHandlerService** konwertuje pliki PDF na obrazy.
3. **OpenAIService** przesyła obrazy do OpenAI w celu ekstrakcji informacji o promocjach.
4. Otrzymane dane o promocjach są przetwarzane i zapisywane w bazie danych przez **PromotionService**.
5. API udostępnia dane użytkownikom za pomocą kontrolerów **PromotionController**, **CategoryController**, **StoreController**.

## 7. Automatyzacja

- Aplikacja wykorzystuje **Coravel Scheduler** do automatycznego cyklicznego pobierania i aktualizowania danych o promocjach.
- **Scheduler** uruchamia **ScrapingController**, **PdfHandlerService** i **OpenAIService**, aby zapewnić aktualność bazy danych.

## 8. Baza danych

### 8.1. Tabele

*Tabela 8.1. Spis tabel*

| Nazwa tabeli | Opis |
|---|---|
| Categories | Informacje o typach produktów |
| Leaflets | Informacje o gazetkach promocyjnych |
| Promotions | Dostępne promocje |
| Stores | Dane o sklepach, w których obowiązują promocje |

### 8.1.1. Categories

Tabela *Categories* służy do przechowywania informacji o kategoriach, do których przypisane są produkty objęte promocjami.

- **Id -** Każda kategoria posiada unikalny identyfikator typu UNIQUEIDENTIFIER, który pozwala jednoznacznie odróżnić kategorie w systemie.
- **Name -** Pole tekstowe przechowujące nazwę kategorii, np. "Elektronika", "Artykuły spożywcze", "Produkty dla dzieci".

### 8.1.2. Leaflets

Tabela *Leaflets* służy do przechowywania informacji o gazetkach promocyjnych publikowanych przez sklepy. Gazetki stanowią zbiory promocji dostępnych w określonych ramach czasowych.

- **Id -** Każda gazetka posiada unikalny identyfikator typu UNIQUEIDENTIFIER, który pozwala jednoznacznie ją identyfikować.
- **StartDate -** Pole przechowujące datę, od której gazetka zaczyna obowiązywać.
- **EndDate -** Pole przechowujące datę, do której gazetka jest ważna.
- **PdfLink -** Pole tekstowe przechowujące link do pliku PDF gazetki, umożliwiając użytkownikom jej pobranie lub przeglądanie.
- **StoreId -** Klucz obcy wskazujący na tabelę Stores, identyfikujący sklep, który opublikował gazetkę.

### 8.1.3. Promotions

Tabela *Promotions* przechowuje szczegółowe informacje o promocjach dostępnych w gazetkach. Każdy wiersz reprezentuje pojedynczą promocję na określony produkt.

- **Id -** Każda promocja posiada unikalny identyfikator (UNIQUEIDENTIFIER), który umożliwia jej jednoznaczną identyfikację.
- **ProductName -** Pole tekstowe przechowujące nazwę produktu objętego promocją, np. "Smartfon XYZ", "Sok jabłkowy".
- **UnitType -** Pole określające jednostkę sprzedaży produktu, np. "szt", "kg".
- **OriginalPrice -** Pole numeryczne przechowujące cenę produktu przed promocją.

- **PriceAfterPromotion -** Pole numeryczne przechowujące cenę produktu po uwzględnieniu promocji.
- **PromotionType -** Pole tekstowe określające rodzaj promocji, np. "zniżka procentowa", "gratis", "2+1".
- **StartDate -** Data rozpoczęcia obowiązywania promocji.
- **EndDate -** Data zakończenia obowiązywania promocji.
- **UntilOutOfStock -** Flaga (BIT) wskazująca, czy promocja trwa do wyczerpania zapasów.
- **RequiredApp -** Flaga (BIT) wskazująca, czy promocja wymaga użycia aplikacji mobilnej.
- **LeafletId -** Klucz obcy wskazujący na tabelę Leaflets, identyfikujący gazetkę, w której zawarta jest promocja.
- **CategoryId -** Klucz obcy wskazujący na tabelę Categories, określający kategorię produktu.

#### 8.1.4. Stores

Tabela Stores zawiera informacje o sieciach handlowych lub sklepach, które udostępniają promocje. Każdy sklep jest powiązany z gazetkami promocyjnymi, które mogą obejmować określone promocje.

- **Id -** Unikalny identyfikator (UNIQUEIDENTIFIER), który jednoznacznie identyfikuje każdy sklep w systemie.
- **Name -** Pole tekstowe przechowujące nazwę sklepu, np. "Lidl", "Biedronka".
- **Stem -** Pole tekstowe przechowujące uproszczoną nazwę sklepu, używaną np. w adresach URL lub w wyszukiwarkach, np. "lidl", "biedronka".

### 8.2. Procedury

*Tabela 2.2. Spis procedur*

| Nazwa procedury | Opis |
|---|---|
| **GetPromotionsByStore** | Pobiera promocje dostępne w danym sklepie |
| **GetTop10LargestPromotions** | Wykazuje 10 największych promocji pod względem rabatu |
| **GetPromotionsByCategory** | Pobiera promocje dla określonej kategorii |

### 8.2.1. GetPromotionsByStore

Procedura pobiera listę promocji dostępnych w danym sklepie na podstawie jego identyfikatora.

**Wykorzystywane tabele:**

- Promotions
- Categories
- Leaflets
- Stores

**Działanie:**

1. Pobiera listę gazet promocyjnych (Leaflets) powiązanych z danym sklepem (Stores) poprzez StoreId.
2. Pobiera wszystkie promocje (Promotions) powiązane z tymi gazetkami.
3. Oblicza:
    a. Kwotę rabatu jako różnicę między OriginalPrice a PriceAfterPromotion.
    b. Procentowy rabat jako (OriginalPrice - PriceAfterPromotion) / OriginalPrice * 100.
4. Formatuje daty rozpoczęcia i zakończenia promocji.
5. Zwraca listę promocji z następującymi polami:
    a. Id promocji
    b. Nazwa produktu
    c. Cena przed i po promocji
    d. Kwota rabatu
    e. Procentowy rabat
    f. Kategoria produktu
    g. Nazwa sklepu

### 8.2.2. GetTop10LargestPromotions

Procedura wyznacza 10 największych promocji na podstawie wartości rabatu (w złotówkach oraz procentowo).

**Wykorzystywane tabele:**

- Promotions
- Categories
- Leaflets

- Stores

**Działanie:**

1. Pobiera wszystkie aktywne promocje z tabeli Promotions.
2. Oblicza:
    a. Kwotę rabatu jako różnicę między OriginalPrice a PriceAfterPromotion.
    b. Procentowy rabat.
3. Sortuje promocje malejąco według wartości rabatu w złotówkach, a w przypadku remisu według wartości procentowego rabatu.
4. Pobiera 10 najlepszych promocji.
5. Zwraca listę 10 promocji z następującymi polami:
    a. Id promocji
    b. Nazwa produktu
    c. Cena przed i po promocji
    d. Kwota rabatu
    e. Procentowy rabat
    f. Kategoria produktu
    g. Nazwa sklepu
    h. Id sklepu

### 8.2.3. GetPromotionsByCategory

Procedura pobiera listę promocji dla określonej kategorii na podstawie CategoryId.

**Wykorzystywane tabele:**

- Promotions
- Categories
- Leaflets
- Stores

**Działanie:**

1. Pobiera wszystkie promocje powiązane z daną kategorią na podstawie CategoryId.
2. Filtruje promocje, aby uwzględnić tylko aktywne promocje.
3. Oblicza:
    a. Kwotę rabatu jako różnicę między OriginalPrice a PriceAfterPromotion.
    b. Procentowy rabat.
4. Formatuje daty rozpoczęcia i zakończenia promocji.
5. Zwraca listę promocji z następującymi polami:

a. Id promocji
b. Nazwa produktu
c. Cena przed i po promocji
d. Kwota rabatu
e. Procentowy rabat
f. Kategoria produktu
g. Nazwa sklepu

# 9. Architektura i Działanie Systemu API

## 9.1. Kontrolery

### 9.1.1 OpenAIController

**Rola:** Kontroler odpowiada za integrację z usługą OpenAI, umożliwiając analizę obrazów.

**Endpointy:**

- POST /openai/analize-images - uruchamia analizę obrazów przez OpenAI.
- Odpowiedzi:
    - 200 OK – Integracja OpenAI zakończona sukcesem.
    - 500 Internal Server Error – Błąd podczas integracji OpenAI.

**Działanie:**

- Wywołuje metodę ParseImagesToFunction() z _openAIService.
- Loguje sukces lub błąd operacji.
- Zwraca odpowiedni status HTTP.

### 9.1.2. ScrapingController

**Rola:** Obsługuje pobieranie i konwersję plików PDF z promocjami sklepów.

**Endpointy:**

- GET /scraping/download-pdfs - pobiera pliki PDF z promocjami dla dostępnych sklepów.

- Odpowiedzi:
    - 200 OK – Proces pobierania PDF rozpoczęty.
    - 404 Not Found – Brak sklepów do pobrania.
    - 500 Internal Server Error – Błąd podczas pobierania PDF.

- GET /scraping/convert-to-images - konwertuje pobrane pliki PDF na obrazy.
- Odpowiedzi:
    - 200 OK – Proces konwersji zakończony sukcesem.
    - 500 Internal Server Error – Błąd podczas konwersji PDF na obrazy.

**Działanie:**

- Pobiera listę sklepów z _storeService.
- Dla każdego sklepu wywołuje _pdfHandlerService.ScrappPromotionData().
- Konwertuje pliki PDF na obrazy.
- Loguje operację i zwraca odpowiedni status.

### 9.1.3. PromotionController

**Rola:** Obsługuje operacje na promocjach, w tym pobieranie aktywnych i najlepszych ofert.

**Endpointy:**

- GET /api/promotions/active – pobiera aktywne promocje.
- GET /api/promotions/top – pobiera najlepsze promocje.
- GET /api/promotions/store/{storeId} – pobiera promocje dla konkretnego sklepu.
- GET /api/promotions/category/{categoryId} – pobiera promocje dla konkretnej kategorii.

**Działanie:**

- Pobiera promocje z _promotionService.
- Loguje ewentualne błędy.
- Zwraca dane w formacie JSON.

### 9.1.4. CategoryController

**Rola:** Obsługuje operacje na kategoriach produktów.

**Endpointy:**

- GET /api/categories – pobiera wszystkie kategorie.
- POST /api/categories/from-list – tworzy kategorie na podstawie listy.

**Działanie:**

- Pobiera listę kategorii z _categoryService.
- Umożliwia masowe dodanie kategorii.
- Loguje sukcesy i błędy.

### 9.1.5. StoreController

**Rola:** Zarządza danymi o sklepach.

**Endpointy:**

- GET /api/stores – pobiera listę sklepów.
- POST /api/stores/all – tworzy sklepy na podstawie konfiguracji.

**Działanie:**

- Pobiera dane sklepów z _storeService.
- Pozwala na jednorazowe dodanie sklepów.
- Loguje zdarzenia i błędy.

### 9.1.6. LeafletController

**Rola:** Obsługuje operacje na gazetkach promocyjnych.

**Endpointy:**

- GET /api/leaflets – Pobiera wszystkie gazetki.
- POST /api/leaflets – Tworzy nową gazetkę.

**Działanie:**

- Pobiera dane gazetek z _leafletService.
- Obsługuje dodawanie nowych gazetek.
- Loguje błędy i zwraca odpowiednie statusy HTTP.

## 9.2. Serwisy

### 9.2.1. OpenAIService

**Rola:** Obsługuje analizę obrazów z wykorzystaniem OpenAI, przetwarza obrazy gazet promocyjnych na dane w formacie JSON i zapisuje je w bazie danych.

**Metody:**

- **ParseImagesToFunction()**
  - Pobiera listę folderów zawierających obrazy gazet promocyjnych.
  - Iteruje przez foldery i pobiera obrazy.
  - Przetwarza obrazy za pomocą OpenAI.
  - Tworzy obiekty promocji na podstawie zwróconych danych.
  - Zapisuje dane w bazie danych.

- **GetJsonPromotions(string imageFilePath)**
  - Pobiera listę dostępnych kategorii.
  - Wczytuje obraz z podanej ścieżki.
  - Wysyła obraz do OpenAI wraz z przygotowanym promtem.
  - Otrzymuje JSON zawierający szczegóły promocji.
  - Zwraca JSON do dalszego przetwarzania.

### 9.2.2. Scheduler

**Rola:** Automatyzuje proces pobierania, konwersji i przetwarzania gazetek promocyjnych.

**Metody:**

- **Start()**
  - Pobiera listę sklepów.
  - Pobiera pliki PDF dla każdego sklepu.
  - Konwertuje PDF na obrazy.
  - Przekazuje obrazy do OpenAI.
  - Parsuje wyniki do bazy danych.

### 9.2.3. StoreService

**Rola:** Zarządza danymi sklepów w systemie.

**Metody:**

- **GetAllStoresAsync()**
    - Pobiera wszystkie sklepy z bazy danych.
    - Zwraca listę sklepów jako DTO.

- **GetStoreByIdAsync(Guid id)**
    - Pobiera sklep o podanym ID.
    - Zwraca dane sklepu w formacie DTO.

- **GetIdFromStemAsync(string shop_stem)**
    - Wyszukuje sklep na podstawie stem.
    - Zwraca unikalny identyfikator sklepu.

- **CreateStoresFromConfAsync()**
    - Pobiera listę sklepów z konfiguracji.
    - Iteruje przez listę i dodaje nowe sklepy.
    - Zapisuje zmiany w bazie danych.

#### 9.2.4. CategoryService

**Rola:** Zarządza kategoriami produktów.

**Metody:**

- **GetAllCategoriesAsync()**
    - Pobiera wszystkie kategorie z bazy danych.
    - Zwraca listę kategorii jako DTO.

- **GetCategoryByIdAsync(Guid id)**
    - Pobiera kategorię na podstawie podanego ID.
    - Zwraca dane kategorii w formacie DTO.

- **CreateCategoryAsync(CategoryDto categoryDto)**
    - Sprawdza, czy kategoria już istnieje.
    - Tworzy nową kategorię.
    - Zapisuje dane w bazie.

- **CreateCategoryFromListAsync()**
    - Pobiera predefiniowaną listę kategorii.
    - Iteruje przez listę i dodaje kategorie do bazy.

- Zapisuje zmiany.

### 9.2.5. PromotionService

**Rola:** Obsługuje operacje na promocjach.

**Metody:**

- **GetAllPromotionsAsync()**
  - Pobiera wszystkie promocje z bazy danych.
  - Zwraca listę promocji w formacie DTO.
- **GetActivePromotionsAsync()**
  - Pobiera promocje, które są aktualnie aktywne.
  - Zwraca listę aktywnych promocji.

- **GetTopPromotionsAsync()**
  - Pobiera 10 najlepszych promocji.
  - Sortuje je według największego rabatu.
  - Zwraca listę promocji.

- **CreatePromotionsAsync(ChatCompletion completion, Guid guidLeaflet)**
  - Pobiera dane JSON z OpenAI.
  - Tworzy obiekty promocji na podstawie danych.
  - Zapisuje promocje w bazie danych.

### 9.2.6. LeafletService

**Rola:** Obsługuje operacje na gazetkach promocyjnych.

**Metody:**

- **GetAllLeafletsAsync()**
  - Pobiera listę wszystkich gazetek promocyjnych.
  - Zwraca listę w formacie DTO.

- **GetLeafletByIdAsync(Guid id)**
  - Pobiera gazetkę po ID.
  - Zwraca dane gazetki w formacie DTO.

- **CreateLeafletAsync(string dateRange, string shop_stem, string pdfLink)**
  - Tworzy nową gazetkę na podstawie daty i sklepu.
  - Dodaje gazetkę do bazy danych.

- Zwraca ID nowej gazetki.

### 9.2.7. PdfHandlerService

**Rola:** Obsługuje pobieranie i konwersję plików PDF gazet promocyjnych.

**Metody:**

- **ScrappPromotionData(string shop)**
  - Pobiera stronę internetową sklepu.
  - Wyszukuje pliki PDF z gazetkami.
  - Pobiera pliki PDF.
  - Zapisuje je w systemie.

- **ConvertAllPdfsToImagesAndDelete()**
  - Pobiera wszystkie pliki PDF.
  - Konwertuje je na obrazy.
  - Usuwa oryginalne pliki PDF.

## 10. Frontend – aplikacja webowa

Aplikacja frontendowa PromoComparer umożliwia zarządzanie promocjami, sklepami i kategoriami produktów. Jest zbudowana w technologii React.js z wykorzystaniem nowoczesnych narzędzi webowych i opiera się na interakcjach z API backendowym PromoComparerAPI.

### 10.1. Główne funkcjonalności frontendu

- **Wyświetlanie promocji:**
  - **Lista wszystkich promocji:** Użytkownik może przeglądać wszystkie promocje dostępne w systemie, w tym szczegóły, takie jak nazwa produktu, cena przed i po promocji, oraz typ promocji (np. zniżka procentowa).
  - **Największe promocje:** Strona główna zawiera link do sekcji z największymi promocjami, sortowanymi według wartości rabatu.
  - **Przykład interakcji:** Po kliknięciu na szczegóły promocji użytkownik przechodzi do dedykowanej strony produktu, gdzie znajdzie szczegółowe informacje.
- **Zarządzanie sklepami:**
  - **Lista sklepów:** Widok zawiera szczegółowe informacje o każdym sklepie, w tym nazwę i dostępne promocje.

    - **Przeglądanie promocji sklepu:** Użytkownik może kliknąć na nazwę sklepu, aby zobaczyć wszystkie promocje powiązane z danym sklepem.
    - **Przykład interakcji:** Wyszukiwanie sklepu według nazwy w pasku wyszukiwania dynamicznie filtruje wyniki.

- **Kategorie produktów:**
    - **Lista kategorii:** Użytkownik może wybrać kategorię, aby przeglądać promocje na produkty w niej zawarte.
    - **Promocje powiązane z kategorią:** Widok zawiera sortowalne i filtrowalne listy produktów.
    - **Przykład interakcji:** Po wyborze kategorii, np. "Elektronika", system ładuje tylko promocje z tej kategorii i umożliwia dalsze filtrowanie według ceny.
- **Nawigacja:**
    - **Intuicyjny system:** Pasek nawigacji jest zawsze widoczny i zapewnia szybki dostęp do kluczowych sekcji aplikacji (Promocje, Sklepy, Kategorie).
    - **Przykład interakcji:** Po najechaniu na elementy menu rozwijają się podmenu z dodatkowymi opcjami, takimi jak wyszukiwanie lub filtrowanie.

## 10.2. Architektura aplikacji

Aplikacja frontendowa opiera się na architekturze komponentowej, co umożliwia jej łatwą rozbudowę i utrzymanie. Każdy komponent jest odpowiedzialny za jednoznacznie określoną funkcjonalność, co ułatwia wyszukiwanie błędów oraz rozwój systemu.

### 10.2.1 Struktura aplikacji

- **src/ – Główny katalog aplikacji zawierający wszystkie moduły.**
    - App.js – Główny komponent zarządzający routingiem i układem aplikacji.
    - Navigation/ – Folder zawierający pasek nawigacji.
    - Home/ – Moduł strony głównej aplikacji.
    - Promotions/ – Komponenty odpowiedzialne za wyświetlanie listy promocji, w tym podstrony z największymi promocjami i promocjami według kategorii.
    - Shops/ – Widoki i logika związana z zarządzaniem sklepami.

- Categories/ – Obsługa widoku listy kategorii oraz promocji powiązanych z kategoriami.
- hooks/ – Hooki customowe, takie jak useShopPromotionsData oraz useShopsData.
- assets/ – Folder przechowujący zasoby statyczne, takie jak grafiki czy ikony.

### 10.2.2 Komponenty i ich relacje:

- **Komponenty główne:**
  - `App.js` – Zarządza routingiem i zapewnia podstawowy układ aplikacji.
  - `Navigation` – Pasek nawigacji zapewniający dostęp do kluczowych sekcji.
- **Podstrony:**
  - `Home` – Strona główna z linkiem do największych promocji.
  - `Promotions` – Lista promocji z opcją filtrowania według sklepu lub kategorii.
  - `ShopsPage` – Lista sklepów dostępnych w systemie.
  - `CategoriesPage` – Strona wyboru kategorii produktów.
- **Hooki:**
  - `useShopPromotionsData` – Obsługuje logikę pobierania promocji dla danego sklepu, zarządzając stanami ładowania i błędów.
  - `useShopsData` – Zarządza pobieraniem listy sklepów, umożliwiając dynamiczne odświeżanie danych.
- **Relacje między komponentami:**
  - Komponent `App.js` zarządza routingiem i określa, który widok powinien być renderowany w zależności od adresu URL.
  - Komponenty stron, takie jak `Home` czy `CategoriesPage`, są odpowiedzialne za wyświetlanie danych, które pobierają za pomocą dedykowanych hooków.
  - Hooki, takie jak `useShopPromotionsData`, komunikują się z backendem za pomocą API i zwracają przetworzone dane do komponentów.

## 10.3. Technologie i narzędzia

- **React.js (v18.2.0):** Framework do budowy interfejsu użytkownika, zapewniający komponentową strukturę aplikacji.

- **React Router (v6.14.1):** Zarządzanie trasami aplikacji i nawigacją między widokami.
- **CSS:** Modularna stylizacja komponentów, zapewniająca odrębność stylów dla każdego widoku.
- **Fetch API:** Komunikacja z backendem, używana do pobierania danych z API PromoComparer.
- **ESLint (v8.39.0):** Narzędzie do analizy statycznej kodu, zapewniające spójność stylu kodowania.
- **Prettier (v2.8.8):** Narzędzie do automatycznego formatowania kodu.
- **Node.js (v18.17.0) i npm:** Środowisko uruchomieniowe JavaScript i menedżer pakietów używane do zarządzania zależnościami aplikacji.

## 10.4. Przepływ danych w aplikacji

1. **Pobieranie danych:**
    a. Komponenty wykorzystują hooki, takie jak `useShopsData` i `useShopPromotionsData`, do asynchronicznego pobierania danych z backendu.
    b. W trybie testowym aplikacja korzysta z danych z folderu `dummyData`.
2. **Wyświetlanie:**
    a. Dane są przetwarzane i renderowane za pomocą komponentów odpowiednich dla każdej sekcji aplikacji.
3. **Interakcje użytkownika:**
    a. Wybór kategorii na stronie `CategoriesPage` przekierowuje użytkownika do promocji powiązanych z tą kategorią.

## 10.5. Automatyzacja

Aplikacja frontendowa integruje się z backendem PromoComparerAPI, który automatycznie aktualizuje dane o promocjach. Po stronie frontendu dane te są dynamicznie pobierane i odświeżane w momencie wejścia użytkownika na daną sekcję.

## 10.6. Stylizacja

Stylizacja aplikacji opiera się na plikach CSS, co zapewnia modularność i łatwą edycję wyglądu.

- `App.css` – Globalny styl aplikacji.
- `CategoriesPage.css` – Stylizacja widoku kategorii.

- `Home.css` – Wygląd strony głównej.

### 10.7. Przykłady interakcji z API

- **Lista promocji sklepu:**
  - Endpoint: `GET /api/shops/:shopId/discounts`
  - Wykorzystany w hooku `useShopPromotionsData`.
- **Lista sklepów:**
  - Endpoint: `GET /api/shops`
  - Wykorzystany w hooku `useShopsData`.

### 10.8. Możliwości rozbudowy

- **Zaawansowane filtrowanie promocji:** Dodanie filtrów według daty obowiązywania promocji, wartości rabatu lub dostępności.
- **Dynamiczne powiadomienia:** Umożliwienie użytkownikom ustawienia powiadomień o nowych promocjach w wybranych kategoriach lub sklepach.
- **Opcje personalizacji:** Wdrożenie mechanizmów pozwalających użytkownikom na tworzenie własnych list ulubionych promocji.
- **Historia przeglądania:** Zapisywanie ostatnio przeglądanych promocji w celu łatwiejszego powrotu.
- **Integracja z mapami:** Wyświetlanie lokalizacji sklepów z promocjami w oparciu o mapy Google.
- **Obsługa języków:** Dodanie funkcji wielojęzyczności, umożliwiającej użytkownikom wybór języka interfejsu.
- **Tryb offline:** Umożliwienie użytkownikom dostępu do ostatnio pobranych promocji w trybie offline.