**Słodka Chwila** to strona wizytówka cukierni, która zapewnia szybki dostęp do kluczowych informacji. Użytkownicy mogą przeglądać menu, aktualności, stronę główną oraz łatwo skontaktować się poprzez formularz wiadomości. Prosty i intuicyjny układ pozwala na wygodne korzystanie z serwisu.

  Strona Główna  
Strona główna zawiera powitanie oraz krótkie wprowadzenie do cukierni. Zawiera również najnowsze informacje i promocje, umożliwiając użytkownikom szybki dostęp do kluczowych treści.

 **Menu**  
Sekcja menu przedstawia pełną ofertę produktów cukierni, w tym ciasta, torty i ciasteczka. Użytkownicy mogą przeglądać produkty wraz z cenami i składnikami, co ułatwia dokonanie wyboru.

 **Aktualności**  
W tej sekcji użytkownicy mogą zapoznać się z najnowszymi wydarzeniami cukierni, takimi jak nowe produkty, promocje i specjalne oferty. Pomaga utrzymać klientów na bieżąco z nowościami.

 **Kontakt**  
Sekcja kontaktowa zawiera dane kontaktowe, takie jak numer telefonu, e-mail i adres fizyczny. Dodatkowo, zawiera mapkę i linki do mediów społecznościowych, aby ułatwić kontakt.

* **Formularz Kontaktowy**  
  Formularz kontaktowy umożliwia użytkownikom szybkie przesyłanie zapytań lub opinii do cukierni. Jest prosty i łatwy w obsłudze, z polami na imię, e-mail oraz wiadomość.

# ·Front-end: React + Tailwind CSS , React Hook Form do formularzy, React Router do routingu(AOS.js dla animacji na stronie).

**CSS Framework:**

**Tailwind CSS**:  do tworzenia responsywnych i estetycznych interfejsów. Minimalizuje pisanie własnego CSS.

**Formularze w React:**

**React Hook Form**: Lekka biblioteka do obsługi formularzy. Działa świetnie z walidacją i jest bardzo intuicyjna.

**Routing (do różnych sekcji strony):**

**React Router**: Pozwala tworzyć różne podstrony, np. /galeria, /kontakt.

**Animacje(opcjonalnie):**

**AOS.js (Animate On Scroll)**: Łatwe w implementacji animacje, wyzwalane podczas przewijania strony.

# Back-end: Node.js z Express.js do obsługi formularza, nodemailer do wysyłania e-maili.

·        **Express.js**:

- Lekki, szybki i prosty framework do obsługi zapytań.

- Świetny do tworzenia API obsługujących formularze.

·        **Dodatki do backendu:**

**nodemailer**: Do wysyłania e-maili z formularza.

**body-parser**: Do obsługi danych przesyłanych przez formularz.

**CORS**: Do umożliwienia komunikacji między front-endem i back-endem (jeśli są na różnych serwerach).

# Hosting: Netlify/Vercel dla front-endu, Render/Heroku dla backendu.

·   **Front-end (React):**

**Netlify**: Darmowa platforma z obsługą Reacta.

·        **Back-end (Node.js)**:

**Render**: Darmowy hosting dla aplikacji Node.js.

# API:

(dodatkowo nie połączono z bd)

### **Endpoint testowy (GET)**

* **Ścieżka:** /
* **Metoda:** GET
* **Opis:** Jest to endpoint testowy, który po prostu zwraca komunikat "Backend działa poprawnie!". Może być użyty do sprawdzenia, czy serwer działa i czy backend jest dostępny.
* **Przykład:**
  + **URL:** https://slodkachwila.onrender.com/
  + **Odpowiedź:** Backend działa poprawnie!

1. **Google Maps API**

* Do integracji mapy z lokalizacją cukierni.
* **GET /api/location** – Pobiera współrzędne cukierni do wyświetlenia na mapie.

**Dostawca mapy**: OpenStreetMap

**Biblioteka mapowa**: Leaflet

### API do obsługi bazy danych

#### Endpointy API dla tabeli photos:

1. **Dodanie zdjęcia**
   * **Metoda:** POST /api/photos
   * **Opis:** Dodaje nowe zdjęcie do bazy danych.
2. **Pobranie wszystkich zdjęć**
   * **Metoda:** GET /api/photos
   * **Opis:** Zwraca listę wszystkich zdjęć z bazy danych.
3. **Pobranie konkretnego zdjęcia**
   * **Metoda:** GET /api/photos/{id}
   * **Opis:** Zwraca dane jednego zdjęcia na podstawie jego id.
4. **Usunięcie zdjęcia**
   * **Metoda:** DELETE /api/photos/{id}
   * **Opis:** Usuwa zdjęcie na podstawie jego id.

#### Endpointy API dla tabeli messages:

1. **Dodanie wiadomości**
   * **Metoda:** POST /api/messages
   * **Opis:** Dodaje wiadomość przesłaną przez formularz kontaktowy do bazy danych.
2. **Pobranie wszystkich wiadomości**
   * **Metoda:** GET /api/messages
   * **Opis:** Zwraca wszystkie wiadomości z bazy danych.
3. **Pobranie konkretnej wiadomości**
   * **Metoda:** GET /api/messages/{id}
   * **Opis:** Zwraca treść jednej wiadomości na podstawie jej id.
4. **Usunięcie wiadomości**
   * **Metoda:** DELETE /api/messages/{id}
   * **Opis:** Usuwa wiadomość na podstawie jej id.

 Baza danych( Postgres ) **składa się z dwóch tabelek:**

#### 1. Tabela photos (do przechowywania zdjęć)

* **Struktura tabeli:**
  + id (INT, Primary Key, Auto Increment) – unikalny identyfikator zdjęcia.
  + url (VARCHAR) – ścieżka do zdjęcia (np. link do zasobu na serwerze).
  + description (VARCHAR) – opcjonalny opis zdjęcia (np. „Tort czekoladowy”).
  + created\_at (TIMESTAMP) – data dodania zdjęcia.

#### 2. Tabela messages (do przechowywania wiadomości od użytkowników)

* **Struktura tabeli:**
  + id (INT, Primary Key, Auto Increment) – unikalny identyfikator wiadomości.
  + nameSurname (VARCHAR) – imię i nazwisko użytkownika.
  + email (VARCHAR) – adres e-mail użytkownika.
  + message (TEXT) – treść wiadomości.
  + created\_at (TIMESTAMP) – data przesłania wiadomości.

Użycie bd z api w projekcie:

### **Endpoint do kontaktu (POST):**

* **Ścieżka:** /contact
* **Metoda:** POST
* **Opis:** Ten endpoint służy do odbierania danych z formularza kontaktowego. Przyjmuje dane w formacie JSON, które zawierają informacje o imieniu, adresie e-mail i wiadomości. Endpoint powinien obsługiwać te dane (np. zapisać je do bazy danych lub wysłać e-mail).
* **Przykład:**
  + **URL:** https://twoj-backend.onrender.com/contact
  + **Metoda:** POST
  + **Przykładowe dane do wysłania w ciele żądania:**

{

"name": "Jan Kowalski",

"email": "jan.kowalski@example.com",

"message": "Chciałbym się dowiedzieć więcej o Waszych usługach."

}

#### Galeria zdjęć:

* Strona główna wywołuje GET /api/photos, by wyświetlić listę zdjęć w galerii. Zdjęcia są pobierane z tabeli photos.