Przewodnik integracji Chatfuel z monday.com, Booksy i innymi systemami

Instrukcja techniczna

Spis tresći

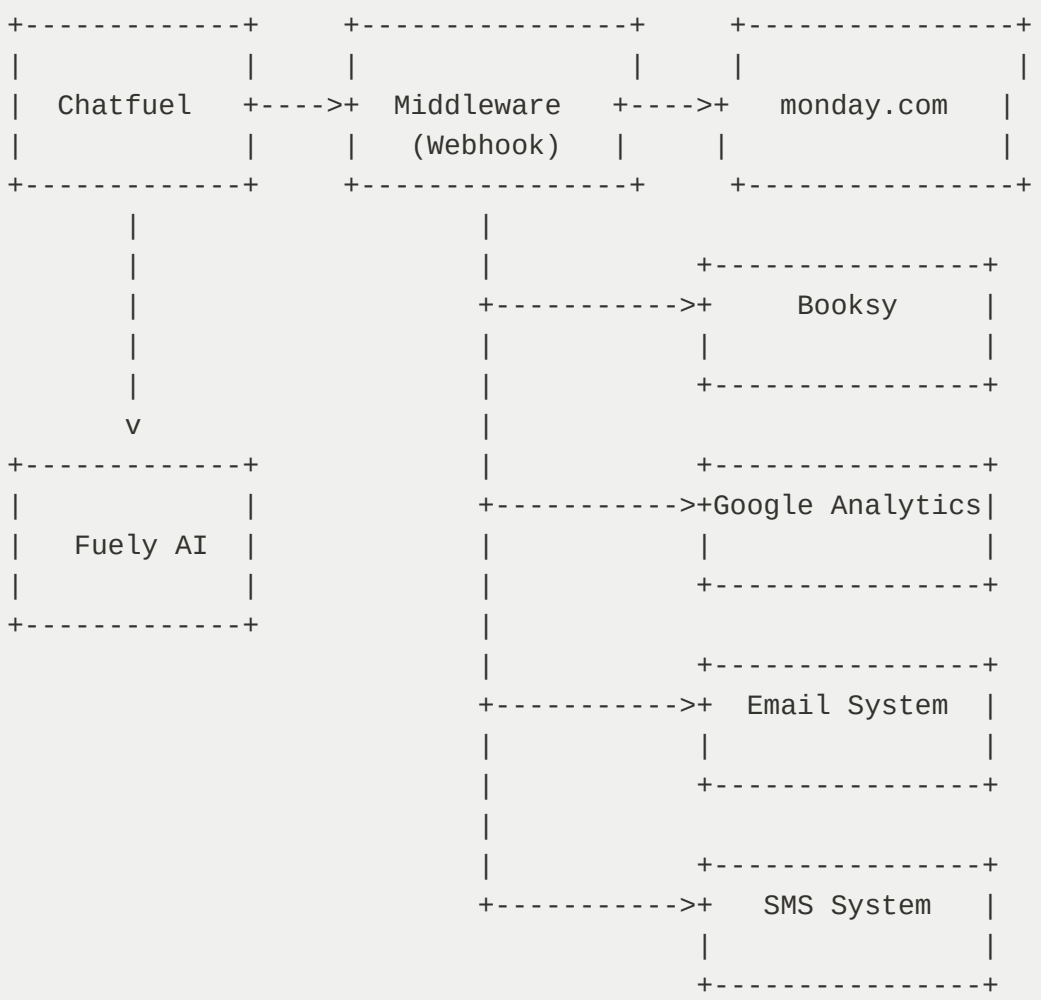
Przegląd integracji

Integracja chatbota Chatfuel z systemami zewnętrznymi NovaHouse pozwoli na:

- Automatyczne tworzenie leadów w monday.com
- Synchronizację terminów spotkań z Booksy
- Śledzenie konwersji i interakcji w Google Analytics
- Automatyczne powiadomienia email i SMS dla klientów i konsultantów

Architektura integracji

Plain Text



Podejście do integracji

Dla integracji Chatfuel z systemami zewnętrznymi, zastosujemy dwa podejścia:

1. **Integracje natywne** - wykorzystanie wbudowanych integracji Chatfuel
2. **Integracje przez webhook** - wykorzystanie middleware do łączenia systemów

Integracja z monday.com

Wymagania wstępne

- Konto monday.com z uprawnieniami administratora
- Klucz API monday.com

- Struktura tablic i kolumn w monday.com
- Konto Chatfuel Business

Konfiguracja monday.com

1. Generowanie klucza API

1. Zaloguj się do monday.com
2. Przejdź do "Admin" > "API"
3. Wygeneruj nowy klucz API z uprawnieniami:
   - `boards:read`
   - `boards:write`
   - `items:read`
   - `items:write`
4. Zapisz klucz API w bezpiecznym miejscu

2. Identyfikacja ID tablic i kolumn

1. Otwórz tablicę "Leady" w monday.com
2. Sprawdź URL, aby znaleźć ID tablicy (np. `https://company.monday.com/boards/123456789`)
3. Otwórz narzędzia deweloperskie przeglądarki (F12)
4. Przejdź do zakładki "Network"
5. Wykonaj akcję na tablicy (np. dodaj nowy element)
6. Znajdź żądanie API i zidentyfikuj ID kolumn

3. Przygotowanie struktury danych

Przygotuj mapowanie pól Chatfuel do kolumn monday.com:

Dane Chatfuel	Kolumna monday.com	ID Kolumny
Imię i nazwisko	Nazwa	text
Email	Email	email
Telefon	Telefon	phone
Temat	Temat	text1
Zŕódło	Zŕódło	status1
Data spotkania	Data spotkania	date
Godzina spotkania	Godzina	time
Pakiet	Pakiet	status2
Status	Status	status

Integracja przez Zapier (opcja 1)

1. Konfiguracja Zapier

1. Utwórz konto Zapier
2. Utwórz nowy Zap
3. Wybierz Chatfuel jako trigger
4. Wybierz "New Message" jako wydarzenie
5. Połącz konto Chatfuel
6. Skonfiguruj trigger (wybierz bota i flow)

2. Konfiguracja akcji monday.com

1. Wybierz monday.com jako akcję
2. Wybierz "Create Item" jako wydarzenie
3. Połącz konto monday.com

4. Wybierz tablicę "Leady"
5. Mapuj pola Chatfuel do kolumn monday.com
6. Testuj i aktywuj Zap

Integracja przez webhook (opcja 2)

1. Konfiguracja webhook w Chatfuel

1. Zaloguj się do panelu Chatfuel Business
2. Przejdź do sekcji "Flows"
3. Wybierz flow, który ma wyzwalać integrację
4. Dodaj akcję "Webhook"
5. Wprowadź URL webhook (np. `https://your-middleware.com/monday-webhook`)
6. Skonfiguruj dane do wysłania (format JSON)

2. Implementacja middleware

Poniżej znajduje się przykładowy kod Node.js dla middleware:

JavaScript

```
const express = require('express');
const axios = require('axios');
const bodyParser = require('body-parser');

const app = express();
app.use(bodyParser.json());

// Konfiguracja
const MONDAY_API_KEY = 'your_monday_api_key';
const MONDAY_API_URL = 'https://api.monday.com/v2';
const MONDAY_BOARD_ID = 'your_board_id';

// Endpoint dla webhook Chatfuel
app.post('/monday-webhook', async (req, res) => {
  try {
    // Dane z Chatfuel
```

```
    const { name, email, phone, topic, source, meetingDate, meetingTime,
package, status } = req.body;

    // Tworzenie elementu w monday.com
    const query = `
      mutation {
        create_item (
          board_id: ${MONDAY_BOARD_ID},
          item_name: "${name}",
          column_values: "{
            \\"email\\": {\\"email\\": \\"${email}\\", \\"text\\":
\\"${email}\\"},
            \\"phone\\": {\\"phone\\": \\"${phone}\\",
\\"countryShortName\\": \\"PL\\"},
            \\"text1\\": \\"${topic}\\",
            \\"status1\\": \\"${source}\\",
            \\"date\\": \\"${meetingDate}\\",
            \\"time\\": \\"${meetingTime}\\",
            \\"status2\\": \\"${package}\\",
            \\"status\\": \\"${status}\\"
          }"
        ) {
          id
        }
      }
    `;

    // Wysłanie żądania do monday.com
    const response = await axios.post(MONDAY_API_URL, { query }, {
      headers: {
        'Content-Type': 'application/json',
        'Authorization': MONDAY_API_KEY
      }
    });

    // Odpowiedź dla Chatfuel
    res.json({
      success: true,
      monday_item_id: response.data.data.create_item.id
    });
  } catch (error) {
    console.error('Error creating item in monday.com:', error);
    res.status(500).json({
      success: false,
      error: error.message
    });
  }
});
```

```
// Uruchomienie serwera
const PORT = process.env.PORT || 3000;
app.listen(PORT, () => {
  console.log(`Server running on port ${PORT}`);
});
```

3. Wdrożenie middleware

1. Wdróż middleware na serwerze (np. Heroku, AWS, Google Cloud)
2. Skonfiguruj zmienne środowiskowe (klucze API, ID tablic)
3. Zabezpiecz endpoint (np. przez token autoryzacyjny)
4. Testuj integrację

Testowanie integracji z monday.com

1. Przeprowadź testową konwersację w Chatfuel
2. Wypełnij formularz kontaktowy lub umów spotkanie
3. Sprawdź, czy element został utworzony w monday.com
4. Sprawdź, czy wszystkie dane zostały poprawnie zmapowane

Integracja z Booksy

Wymagania wstępne

- Konto Booksy z uprawnieniami administratora
- Klucz API Booksy
- Lista usług i pracowników w Booksy
- Konto Chatfuel Business

Konfiguracja Booksy

1. Generowanie klucza API

1. Zaloguj się do panelu administracyjnego Booksy
2. Przejdź do "Ustawienia" > "Integracje" > "API"
3. Wygeneruj nowy klucz API
4. Zapisz klucz API w bezpiecznym miejscu

2. Identyfikacja ID usług i pracowników

1. Zaloguj się do panelu administracyjnego Booksy
2. Przejdź do "Usługi"
3. Zapisz ID usług, które mają być dostępne przez chatbota
4. Przejdź do "Pracownicy"
5. Zapisz ID pracowników, którzy mają być dostępni przez chatbota

3. Przygotowanie struktury danych

Przygotuj mapowanie pól Chatfuel do Booksy:

Dane Chatfuel	Pole Booksy
Imię i nazwisko	customer_name
Email	customer_email
Telefon	customer_phone
Data	date
Godzina	time
Usługa	service_id
Pracownik	staff_id

Integracja przez webhook

1. Konfiguracja webhook w Chatfuel

1. Zaloguj się do panelu Chatfuel Business
2. Przejdź do sekcji "Flows"
3. Wybierz flow "Umów spotkanie"
4. Dodaj akcję "Webhook"
5. Wprowadź URL webhook (np. `https://your-middleware.com/booksy-webhook`)
6. Skonfiguruj dane do wysłania (format JSON)

2. Implementacja middleware

Poniżej znajduje się przykładowy kod Node.js dla middleware:

JavaScript

```
const express = require('express');
const axios = require('axios');
const bodyParser = require('body-parser');

const app = express();
app.use(bodyParser.json());

// Konfiguracja
const BOOKSY_API_KEY = 'your_booksy_api_key';
const BOOKSY_API_URL = 'https://api.booksy.com/api/v1';

// Endpoint dla webhook Chatfuel
app.post('/booksy-webhook', async (req, res) => {
  try {
    // Dane z Chatfuel
    const { name, email, phone, date, time, serviceId, staffId } = req.body;

    // Tworzenie rezerwacji w Booksy
    const response = await axios.post(`${BOOKSY_API_URL}/appointments`, {
      customer_name: name,
      customer_email: email,
      customer_phone: phone,
      date: date,
      time: time,
      service_id: serviceId,
      staff_id: staffId
```

```
    }, {
      headers: {
        'Content-Type': 'application/json',
        'Authorization': `Bearer ${BOOKSY_API_KEY}`
      }
    });

    // Odpowiedź dla Chatfuel
    res.json({
      success: true,
      booksy_appointment_id: response.data.id,
      confirmation_details: {
        date: response.data.date,
        time: response.data.time,
        service: response.data.service_name,
        staff: response.data.staff_name
      }
    });
  } catch (error) {
    console.error('Error creating appointment in Booksy:', error);
    res.status(500).json({
      success: false,
      error: error.message
    });
  }
});

// Endpoint do sprawdzania dostępności
app.post('/booksy-availability', async (req, res) => {
  try {
    // Dane z Chatfuel
    const { date, serviceId, staffId } = req.body;

    // Sprawdzanie dostępności w Booksy
    const response = await axios.get(`${BOOKSY_API_URL}/availability`, {
      params: {
        date: date,
        service_id: serviceId,
        staff_id: staffId
      },
      headers: {
        'Authorization': `Bearer ${BOOKSY_API_KEY}`
      }
    });

    // Odpowiedź dla Chatfuel
    res.json({
      success: true,
```

```
      available_slots: response.data.available_slots
    });
  } catch (error) {
    console.error('Error checking availability in Booksy:', error);
    res.status(500).json({
      success: false,
      error: error.message
    });
  }
});

// Uruchomienie serwera
const PORT = process.env.PORT || 3000;
app.listen(PORT, () => {
  console.log(`Server running on port ${PORT}`);
});
```

3. Wdrożenie middleware

1. Wdróż middleware na serwerze (np. Heroku, AWS, Google Cloud)
2. Skonfiguruj zmienne środowiskowe (klucze API)
3. Zabezpiecz endpoint (np. przez token autoryzacyjny)
4. Testuj integrację

Testowanie integracji z Booksy

1. Przeprowadź testową konwersację w Chatfuel
2. Przejdź przez proces umawiania spotkania
3. Sprawdź, czy rezerwacja została utworzona w Booksy
4. Sprawdź, czy powiadomienia zostały wysłane

Integracja z Google Analytics

Wymagania wstępne

- Konto Google Analytics

- ID śledzenia Google Analytics
- Konto Chatfuel Business

Konfiguracja Google Analytics

1. Utworzenie konta i właściwości

1. Zaloguj się do Google Analytics
2. Utwórz nową właściwość dla chatbota
3. Skonfiguruj cele konwersji:
   - Rozpoczęcie rozmowy
   - Wypełnienie formularza kontaktowego
   - Umówienie spotkania
   - Przekierowanie do konsultanta
4. Zapisz ID śledzenia (UA-XXXXXXXX-X lub G-XXXXXXXXXX)

Integracja natywna

1. Konfiguracja w Chatfuel

1. Zaloguj się do panelu Chatfuel Business
2. Przejdź do sekcji "Settings" > "Integrations"
3. Znajdź integrację "Google Analytics"
4. Wprowadź ID śledzenia
5. Włącz śledzenie zdarzeń

2. Konfiguracja zdarzeń

1. Przejdź do sekcji "Flows"

2. Dla każdego kluczowego kroku, dodaj akcję "Track Event":
    - Kategoria: "Chatbot"
    - Akcja: np. "StartConversation", "BookMeeting", "ContactForm"
    - Etykieta: np. "Website", "Instagram", "WhatsApp"
    - Wartość: opcjonalna wartość liczbowa

Integracja przez webhook (alternatywna)

1. Konfiguracja webhook w Chatfuel

1. Zaloguj się do panelu Chatfuel Business
2. Przejdź do sekcji "Flows"
3. Dla każdego kluczowego kroku, dodaj akcję "Webhook"
4. Wprowadź URL webhook (np. `https://your-middleware.com/ga-webhook`)
5. Skonfiguruj dane do wysłania (format JSON)

2. Implementacja middleware

Poniżej znajduje się przykładowy kod Node.js dla middleware:

JavaScript

```
const express = require('express');
const axios = require('axios');
const bodyParser = require('body-parser');
const { v4: uuidv4 } = require('uuid');

const app = express();
app.use(bodyParser.json());

// Konfiguracja
const GA_TRACKING_ID = 'your_ga_tracking_id';
const GA_API_URL = 'https://www.google-analytics.com/collect';

// Endpoint dla webhook Chatfuel
app.post('/ga-webhook', async (req, res) => {
```

```
  try {
    // Dane z Chatfuel
    const { category, action, label, value, userId, source } = req.body;

    // Przygotowanie danych dla Google Analytics
    const data = {
      v: 1,
      tid: GA_TRACKING_ID,
      cid: userId || uuidv4(),
      t: 'event',
      ec: category || 'Chatbot',
      ea: action,
      el: label,
      ev: value,
      ds: source || 'chatbot'
    };

    // Wysłanie danych do Google Analytics
    await axios.post(GA_API_URL, new URLSearchParams(data).toString(), {
      headers: {
        'Content-Type': 'application/x-www-form-urlencoded'
      }
    });

    // Odpowiedź dla Chatfuel
    res.json({
      success: true
    });
  } catch (error) {
    console.error('Error sending data to Google Analytics:', error);
    res.status(500).json({
      success: false,
      error: error.message
    });
  }
});

// Uruchomienie serwera
const PORT = process.env.PORT || 3000;
app.listen(PORT, () => {
  console.log(`Server running on port ${PORT}`);
});
```

3. Wdrożenie middleware

1. Wdróż middleware na serwerze (np. Heroku, AWS, Google Cloud)

2. Skonfiguruj zmienne środowiskowe (ID śledzenia)
3. Zabezpiecz endpoint (np. przez token autoryzacyjny)
4. Testuj integrację

Testowanie integracji z Google Analytics

1. Przeprowadź testowe konwersacje w Chatfuel
2. Wykonaj różne akcje (rozpoczęcie rozmowy, umówienie spotkania, itp.)
3. Sprawdź, czy zdarzenia są rejestrowane w Google Analytics
4. Sprawdź, czy cele konwersji są poprawnie zliczane

Integracja z systemem powiadomień email

Wymagania wstępne

- Konto usługi email (np. SendGrid, Mailchimp, Amazon SES)
- Klucz API usługi email
- Szablony email
- Konto Chatfuel Business

Konfiguracja usługi email

1. Utworzenie konta i generowanie klucza API

1. Zarejestruj się w wybranej usłudze email (np. SendGrid)
2. Wygeneruj klucz API
3. Zapisz klucz API w bezpiecznym miejscu

2. Przygotowanie szablonów email

Przygotuj szablony email dla różnych scenariuszy:

- Potwierdzenie rezerwacji spotkania
- Przypomnienie o spotkaniu
- Potwierdzenie kontaktu z konsultantem
- Podziękowanie za rozmowę

Integracja przez webhook

1. Konfiguracja webhook w Chatfuel

1. Zaloguj się do panelu Chatfuel Business
2. Przejdź do sekcji "Flows"
3. Dla każdego punktu, który ma wyzwalać email, dodaj akcję "Webhook"
4. Wprowadź URL webhook (np. `https://your-middleware.com/email-webhook`)
5. Skonfiguruj dane do wysłania (format JSON)

2. Implementacja middleware

Poniżej znajduje się przykładowy kod Node.js dla middleware z SendGrid:

JavaScript

```
const express = require('express');
const bodyParser = require('body-parser');
const sgMail = require('@sendgrid/mail');

const app = express();
app.use(bodyParser.json());

// Konfiguracja
const SENDGRID_API_KEY = 'your_sendgrid_api_key';
sgMail.setApiKey(SENDGRID_API_KEY);

// Endpoint dla webhook Chatfuel
app.post('/email-webhook', async (req, res) => {
  try {
    // Dane z Chatfuel
    const {
```

```
      to,
      templateId,
      name,
      meetingDate,
      meetingTime,
      service,
      consultant,
      dynamicData
    } = req.body;

    // Przygotowanie danych dla SendGrid
    const msg = {
      to: to,
      from: 'kontakt@novahouse.pl',
      templateId: templateId,
      dynamic_template_data: {
        name: name,
        meeting_date: meetingDate,
        meeting_time: meetingTime,
        service: service,
        consultant: consultant,
        ...dynamicData
      }
    };

    // Wysłanie email
    await sgMail.send(msg);

    // Odpowiedź dla Chatfuel
    res.json({
      success: true
    });
  } catch (error) {
    console.error('Error sending email:', error);
    res.status(500).json({
      success: false,
      error: error.message
    });
  }
});

// Uruchomienie serwera
const PORT = process.env.PORT || 3000;
app.listen(PORT, () => {
  console.log(`Server running on port ${PORT}`);
});
```

3. Wdrożenie middleware

1. Wdróż middleware na serwerze (np. Heroku, AWS, Google Cloud)
2. Skonfiguruj zmienne środowiskowe (klucze API)
3. Zabezpiecz endpoint (np. przez token autoryzacyjny)
4. Testuj integrację

Testowanie integracji z systemem email

1. Przeprowadź testowe konwersacje w Chatfuel
2. Wykonaj akcje, które powinny wyzwalać emaile
3. Sprawdź, czy emaile są wysyłane
4. Sprawdź, czy dane dynamiczne są poprawnie wstawiane do szablonów

Integracja z systemem powiadomień SMS

Wymagania wstępne

- Konto usługi SMS (np. Twilio, MessageBird)
- Klucz API usługi SMS
- Szablony wiadomości SMS
- Konto Chatfuel Business

Konfiguracja usługi SMS

1. Utworzenie konta i generowanie klucza API

1. Zarejestruj się w wybranej usłudze SMS (np. Twilio)
2. Wygeneruj klucz API i SID
3. Zapisz klucz API i SID w bezpiecznym miejscu

2. Przygotowanie szablonów SMS

Przygotuj szablony SMS dla różnych scenariuszy:

- Potwierdzenie rezerwacji spotkania
- Przypomnienie o spotkaniu
- Potwierdzenie kontaktu z konsultantem

Integracja przez webhook

1. Konfiguracja webhook w Chatfuel

1. Zaloguj się do panelu Chatfuel Business
2. Przejdź do sekcji "Flows"
3. Dla każdego punktu, który ma wyzwalać SMS, dodaj akcję "Webhook"
4. Wprowadź URL webhook (np. `https://your-middleware.com/sms-webhook`)
5. Skonfiguruj dane do wysłania (format JSON)

2. Implementacja middleware

Poniżej znajduje się przykładowy kod Node.js dla middleware z Twilio:

JavaScript

```
const express = require('express');
const bodyParser = require('body-parser');
const twilio = require('twilio');

const app = express();
app.use(bodyParser.json());

// Konfiguracja
const TWILIO_ACCOUNT_SID = 'your_twilio_account_sid';
const TWILIO_AUTH_TOKEN = 'your_twilio_auth_token';
const TWILIO_PHONE_NUMBER = 'your_twilio_phone_number';

const client = twilio(TWILIO_ACCOUNT_SID, TWILIO_AUTH_TOKEN);
```

```
// Endpoint dla webhook Chatfuel
app.post('/sms-webhook', async (req, res) => {
  try {
    // Dane z Chatfuel
    const { to, templateId, name, meetingDate, meetingTime, service,
consultant } = req.body;

    // Wybór szablonu
    let messageBody;
    switch (templateId) {
      case 'booking_confirmation':
        messageBody = `Witaj ${name}! Potwierdzamy Twoje spotkanie w
NovaHouse dnia ${meetingDate} o godz. ${meetingTime}. Temat: ${service}.
Konsultant: ${consultant}. Pytania? Zadzwoń: 123456789`;
        break;
      case 'booking_reminder':
        messageBody = `Przypomnienie: jutro (${meetingDate}) o godz.
${meetingTime} masz spotkanie w NovaHouse. Temat: ${service}. Konsultant:
${consultant}. Do zobaczenia!`;
        break;
      case 'consultant_contact':
        messageBody = `Witaj ${name}! Twoje zapytanie zostało przekazane do
konsultanta (${consultant}). Skontaktuje się z Tobą w ciągu 24h. Dziękujemy
za zainteresowanie NovaHouse!`;
        break;
      default:
        messageBody = `Witaj ${name}! Dziękujemy za kontakt z NovaHouse. W
razie pytań, zadzwoń: 123456789`;
    }

    // Wysłanie SMS
    await client.messages.create({
      body: messageBody,
      from: TWILIO_PHONE_NUMBER,
      to: to
    });

    // Odpowiedź dla Chatfuel
    res.json({
      success: true
    });
  } catch (error) {
    console.error('Error sending SMS:', error);
    res.status(500).json({
      success: false,
      error: error.message
    });
  }
```

```
});

// Uruchomienie serwera
const PORT = process.env.PORT || 3000;
app.listen(PORT, () => {
  console.log(`Server running on port ${PORT}`);
});
```

3. Wdrożenie middleware

1. Wdróż middleware na serwerze (np. Heroku, AWS, Google Cloud)
2. Skonfiguruj zmienne środowiskowe (klucze API, SID, numer telefonu)
3. Zabezpiecz endpoint (np. przez token autoryzacyjny)
4. Testuj integrację

Testowanie integracji z systemem SMS

1. Przeprowadź testowe konwersacje w Chatfuel
2. Wykonaj akcje, które powinny wyzwalać SMS-y
3. Sprawdź, czy SMS-y są wysyłane
4. Sprawdź, czy dane dynamiczne są poprawnie wstawiane do szablonów

Testowanie integracji

Testy jednostkowe

1. **Test integracji z monday.com:**
   - Tworzenie nowego leada
   - Aktualizacja istniejącego leada
   - Pobieranie danych z monday.com
2. **Test integracji z Booksy:**

- Sprawdzanie dostępnośći terminów
- Tworzenie nowej rezerwacji
- Zmiana/odwołanie rezerwacji

3. **Test integracji z Google Analytics:**
   - Śledzenie rozpoczęcia rozmowy
   - Śledzenie umówienia spotkania
   - Śledzenie wypełnienia formularza kontaktowego
4. **Test integracji z systemem email:**
   - Wysyłanie potwierdzenia rezerwacji
   - Wysyłanie przypomnienia o spotkaniu
   - Wysyłanie potwierdzenia kontaktu z konsultantem
5. **Test integracji z systemem SMS:**
   - Wysyłanie potwierdzenia rezerwacji
   - Wysyłanie przypomnienia o spotkaniu
   - Wysyłanie potwierdzenia kontaktu z konsultantem

Testy end-to-end

1. **Scenariusz umawiania spotkania:**
   - Rozpoczęcie rozmowy
   - Wybór pakietu
   - Wybór terminu
   - Podanie danych kontaktowych
   - Potwierdzenie spotkania

- Sprawdzenie, czy:
    - Spotkanie zostało utworzone w Booksy
    - Lead został utworzony w monday.com
    - Zdarzenie zostało zarejestrowane w Google Analytics
    - Email z potwierdzeniem został wysłany
    - SMS z potwierdzeniem został wysłany

2. **Scenariusz kontaktu z konsultantem:**
    - Rozpoczęcie rozmowy
    - Zadanie pytania bez odpowiedzi w bazie wiedzy
    - Przekierowanie do konsultanta
    - Podanie danych kontaktowych
    - Sprawdzenie, czy:
        - Lead został utworzony w monday.com
        - Zdarzenie zostało zarejestrowane w Google Analytics
        - Email z potwierdzeniem został wysłany
        - SMS z potwierdzeniem został wysłany

Testy obciążeniowe

1. Symulacja wielu równoczesnych rozmów
2. Monitorowanie wydajnośći middleware
3. Monitorowanie limitów API (monday.com, Booksy, Twilio, SendGrid)
4. Identyfikacja wąskich gardeł

Najlepsze praktyki

Bezpieczeństwo

- Używaj HTTPS dla wszystkich endpointów
- Implementuj uwierzytelnianie dla webhooków (np. tokeny, podpisy)
- Przechowuj klucze API w zmiennych środowiskowych
- Regularnie rotuj klucze API
- Implementuj rate limiting dla endpointów

Niezawodność

- Implementuj mechanizmy ponownych prób dla nieudanych żądań
- Loguj wszystkie żądania i odpowiedzi
- Monitoruj wydajność i dostępność middleware
- Implementuj mechanizmy powiadamiania o błędach
- Testuj scenariusze awaryjne

Wydajność

- Optymalizuj zapytania do API
- Implementuj buforowanie dla często używanych danych
- Używaj asynchronicznych operacji
- Monitoruj czas odpowiedzi
- Skaluj middleware w zależności od obciążenia

Utrzymanie

- Dokumentuj wszystkie endpointy i integracje

- Używaj systemu kontroli wersji
- Implementuj testy automatyczne
- Monitoruj wykorzystanie API i koszty
- Regularnie aktualizuj zależności

Przygotował: Michał Marini

Data: 8 lipca 2025