



# Twilio Chat App — Kompleksowa dokumentacja

ver3.1.0 •  
2025-12-13

Opis produktu, architektura, scenariusze biznesowe & wskazówki wdrożeniowe

## Przegląd rozwiązania

Twilio Chat App to dedykowane rozwiązanie dla firm, które potrzebują bezpiecznego, kontrolowanego kanału komunikacji SMS/MMS z możliwością automatyzacji i integracji z modelami AI. Produkt łączy prostotę obsługi z polityką audytowalności i pełnym zapisem danych.

### Panel operatora

Intuicyjny interfejs dla zespołów obsługi

### Auto reply & AI

Tryby automatycznych odpowiedzi z logiką biznesową

### REST API

Integracja z systemami CRM i narzędziami zewnętrznymi

### SQLite

Łatwa persystencja i prosty backup plikowy

ver3.1.0

Flask + Twilio

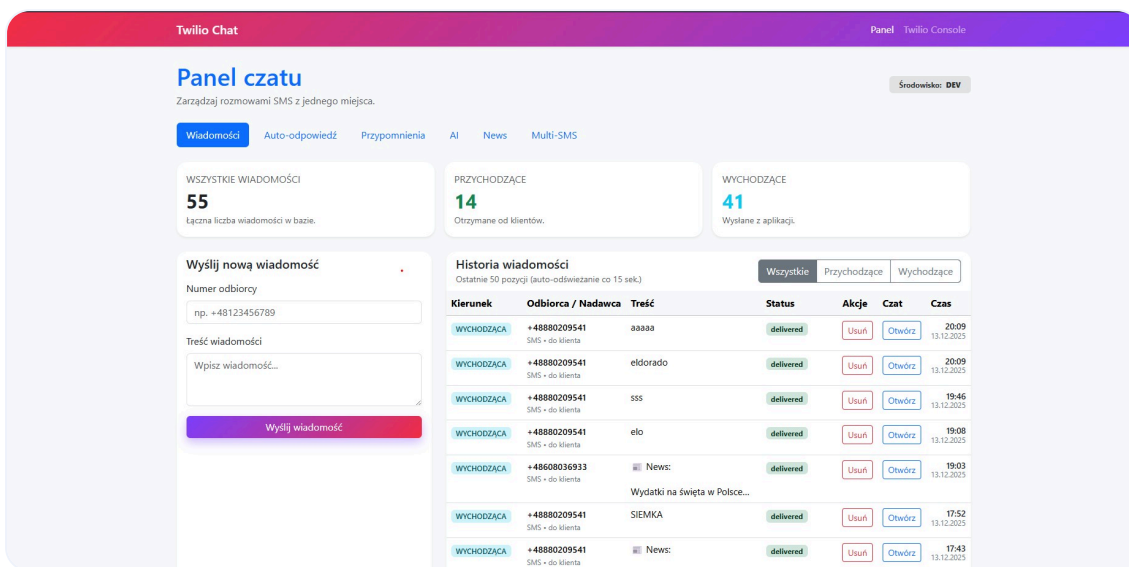
Workery w tle

FAISS / RAG

Multi SMS

## Kluczowe funkcje

- Webhooks Twilio: przyjmowanie inbound i status callbacks
- Auto reply (klasyczny) + AI auto reply: tryby współdzielą jeden magazyn danych
- Przypomnienia: scheduler cykliczny do powiadomień
- Multi SMS: jednorazowe kampanie batchowe z historią odbiorców
- News / FAISS / RAG: scraping, indeksowanie i broadcast streszczeń



Widok wiadomości w panelu — jeden ekran do nadzoru, wysyłki i wglądu w historię

### Co widzisz

- Historię wiadomości (kierunek inbound/outbound, numery, skrót treści).
- Statusy dostarczenia i znaczniki czasu.
- Szybką nawigację do widoku czatu dla konkretnego numeru.

### Co możesz zrobić

- Przejść do rozmowy i kontynuować chat z klientem.
- Weryfikować skuteczność wysyłek po statusach.
- Operacyjnie diagnozować błędy (np. numer, treść, status callback).

### Co dzieje się w tle

- Panel pobiera dane przez `GET /api/messages`.
- Statusy są aktualizowane callbackiem `POST /twilio/status`.
- Wszystko jest audytowane w SQLite w tabeli `messages`.

## Architektura techniczna

Modularna aplikacja Flask z dedykowanymi modułami integracji i workerami. Poniżej związane mapowanie komponentów.

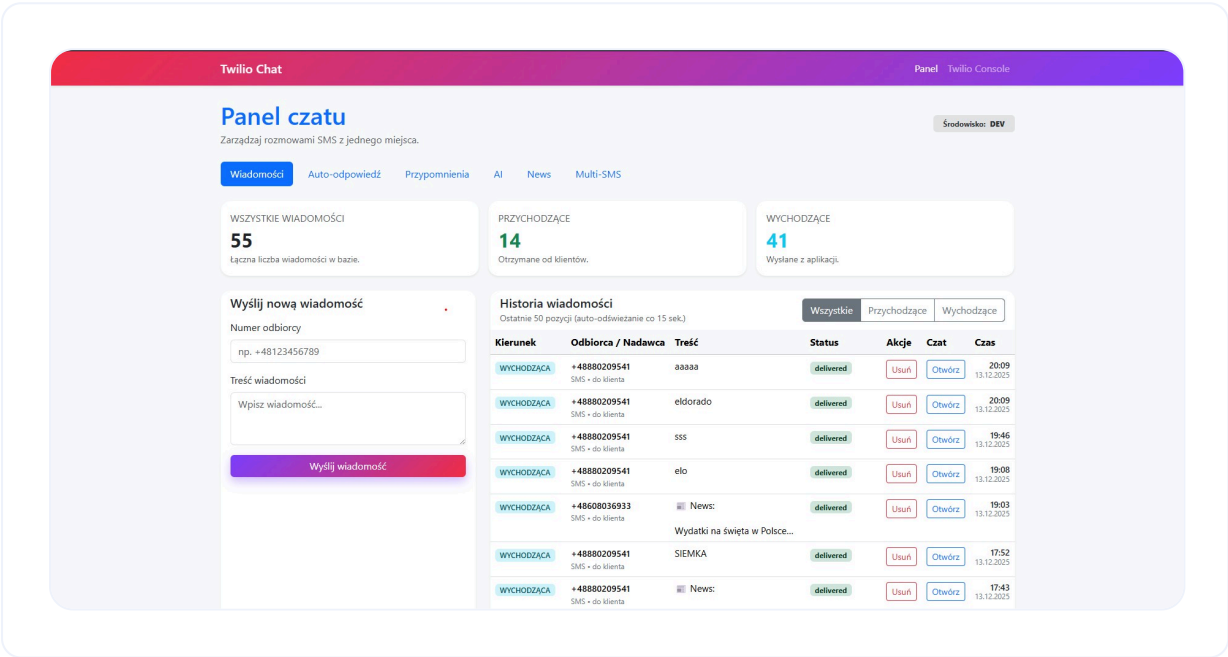
Komponent	Rola
<code>`webhooks_bp`</code>	HTTP webhooki Twilio i API
<code>`ai_service`</code>	Generowanie odpowiedzi AI i budowa kontekstu
<code>`multi_sms.py`</code>	Worker batchowy dla kampanii Multi SMS i rezerwacja zadań
<code>`twilio_client`</code>	Warstwa wysyłki wiadomości i zarządzania wątkami
<code>`database.py`</code>	SQLite, migracje, helpery biznesowe

# Panel WWW — UX i scenariusze użycia

Zaprojektowany pod kątem wydajności operatora: czytelne karty, inteligentne filtry i klikalne wiersze historii upraszczają pracę.

## Historia wiadomości

Jednowierszowy podgląd treści, meta kanału i rola (od/do klienta), statusy oraz czas rozbity na godzinę i datę.



Przykładowy wiersz z meta kanału oraz przyciskiem szybkiej nawigacji do czatu

## Co widzisz

- Wiersze z wiadomościami i metadanymi (od/do, treść, status, data/godzina).
- Rozróżnienie inbound vs outbound (ułatwia kontrolę reakcji).

## Możliwe interakcje

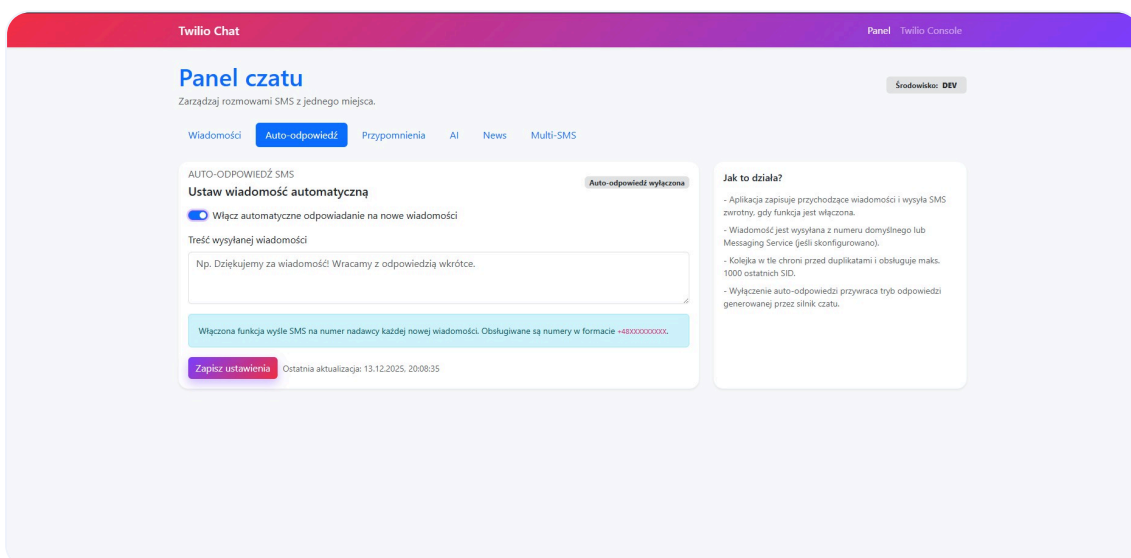
- Wejście do konwersacji klienta i dalsza obsługa w trybie czatu.
- Kontrola statusu dostarczenia i szybkie wykrywanie błędów.
- Praca „na historii” — obsługa wielu klientów bez utraty kontekstu.

## Co się zapisuje

- Każda wiadomość jest przechowywana w SQLite wraz z SID i statusem.
- System normalizuje numery, aby spinać wątki i raportowanie.

## Auto odpowiedź

Włączenie auto odpowiedzi to szybki sposób na obsłużenie powtarzalnych zapytań. Operator ustala treść i może w dowolnym momencie włączyć/wyłączyć funkcję bez restartu systemu.



Przełącznik ON/OFF i pole treści — gotowe do natychmiastowego użycia.

## Co widzisz

- Stan aktywacji (ON/OFF) oraz aktualną treść automatycznej odpowiedzi.
- Ustawienia działające natychmiast po zapisaniu.

## Możliwe interakcje

- Włączenie/wyłączenie auto reply dla całego systemu.
- Zmiana treści odpowiedzi (np. informacja o godzinach pracy, FAQ).

AI auto reply może przejąć sterowanie, gdy jest aktywne.

## Co dzieje się w tle

- Inbound webhook uruchamia logikę enqueue (bez blokowania requestu).
- System odpowiada tylko na wiadomości nowsze niż moment aktywacji (bez „retro-odpowiedzi”).
- Wysyłka realizowana przez Twilio z pełnym logiem w `messages`.

## Przypomnienia

Przypomnienia umożliwiają cykliczną wysyłkę do wybranych numerów (np. powiadomienia, statusy, nudges). Worker w tle egzekwuje harmonogram i zapisuje wyniki.

The screenshot shows the 'Twilio Chat' interface with a sidebar menu containing 'Wiadomości', 'Auto-odpowiedź', 'Przypomnienia', 'AI', 'News', and 'Multi-SMS'. The 'Przypomnienia' section is active, displaying a form for creating a new reminder and a table for active and scheduled reminders.

**Panel czatu**  
Zarządzaj rozmowami SMS z jednego miejsca.

Środowisko: DEV

Wiadomości Auto-odpowiedź **Przypomnienia** AI News Multi-SMS

**NOWE PRZYPOMNIENIE SMS**  
Wyslij cyklicznie

Numer odbiorcy  
+48123456789

Treść wiadomości  
Napisz treść przypomnienia

Interwał (minuty)  
60

Zapisz przypomnienie

**PRZYPOMNIENIA**  
Aktywne i zaplanowane

Numer	Treść	Interwał	Status	Akcje
Brak danych.				

Planowanie i podgląd aktywnych przypomnień w panelu.

**Co widzisz**

- Listę zadań cyklicznych (numer, treść, interwał, status włączenia).
- Informacje potrzebne do nadzoru (czy działa, od kiedy, do kogo).

**Możliwe interakcje**

- Dodanie nowego przypomnienia (treść + interwał).
- Wstrzymanie/wznowienie oraz usunięcie harmonogramu.

**Co dzieje się w tle**

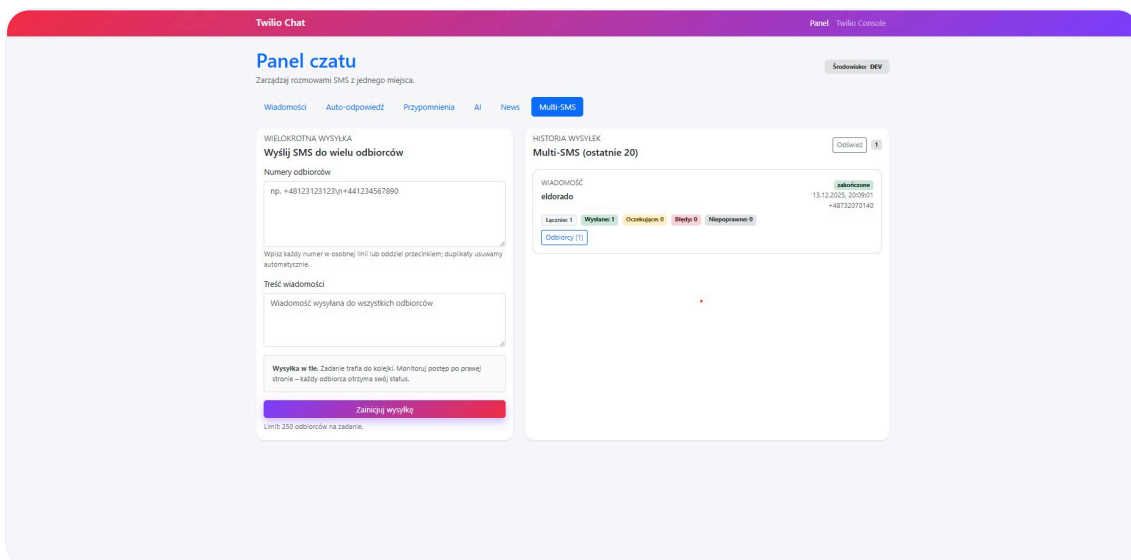
- Scheduler sprawdza due tasks i wysyła SMS przez Twilio.
- Każda wysyłka jest zapisywana w historii, a błędy są logowane.
- Operacje wykonywane są przez API `/api/reminders`.

## Multi SMS — wysyłka jednej wiadomości do wielu odbiorców

Multi SMS umożliwia operatorowi wklejenie listy numerów (dowolny format), utworzenie zadania batchowego i śledzenie postępu wysyłki realizowanej w tle.

### Jak to działa (w praktyce)

1. Operator wkleja numery (przecinki / średniki / nowe linie) i treść wiadomości.
2. Backend normalizuje numery i zapisuje batch + rekordy odbiorców w SQLite.
3. Worker w tle rezerwuje najbliższy batch **pending** i przetwarza odbiorców.
4. Każda próba wysyłki aktualizuje status odbiorcy oraz liczniki batcha (sukcesy/błędy).



Batch w panelu: status, liczniki sukcesów/błędów oraz rozwijana lista odbiorców.



### Co widzisz

- Formularz utworzenia batcha (treść + lista numerów).
- Historię batchy z licznikami: wysłane / błędy / łącznie oraz status batcha.
- Możliwość rozwinięcia listy odbiorców i podejrzenia statusów per numer.

### Możliwe interakcje

- Wklejenie numerów w różnych formatach (przecinki/linie) — system je rozbije, oczyści i znormalizuje.
- Utworzenie batcha bez czekania na wysyłkę (wysyłka idzie w tle).
- Kontrola postępu: odświeżanie historii batchy i rozwijanie odbiorców dla wglądu w statusy.
- Diagnostyka błędów: podejrzenie *error* per numer (np. niepoprawny format, błąd Twilio, brak routingu).

### Co dzieje się w tle

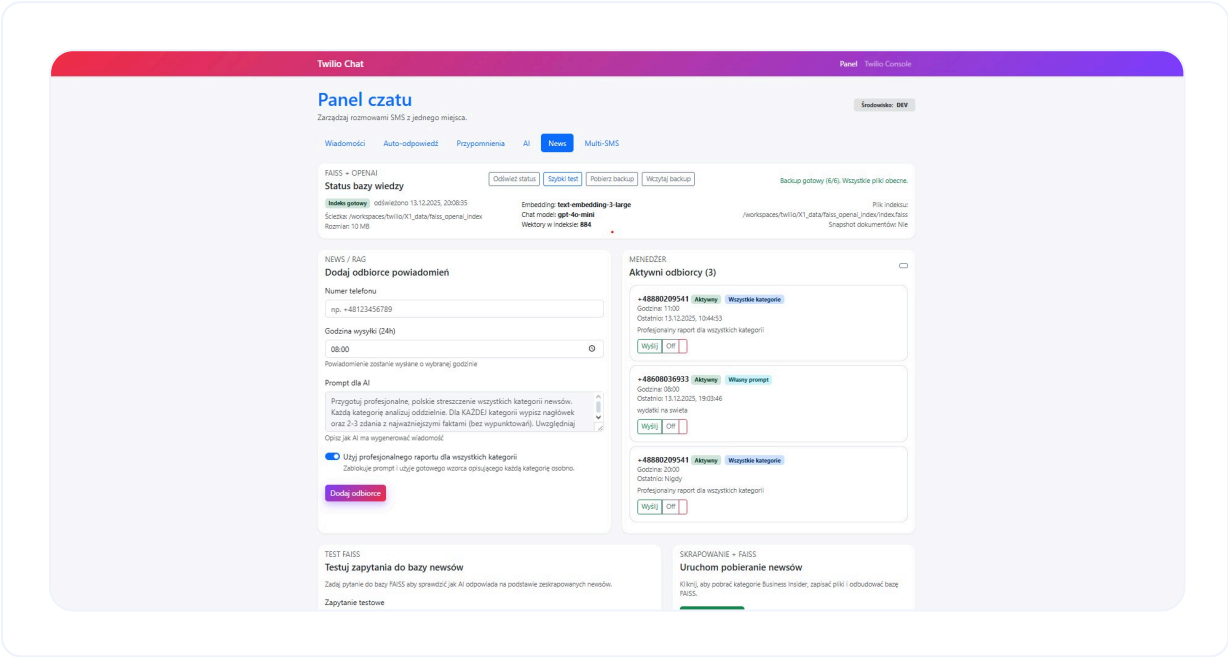
- Backend tworzy batch i odbiorców w SQLite (kolejka DB oparta o statusy).
- Worker rezerwuje batch do przetworzenia, wysyła SMS per odbiorca i aktualizuje statusy w tabelach `multi_sms_*`.
- Każda próba wysyłki jest zapisywana w historii (`messages`), co daje pełny audyt.
- UI pobiera batchy przez `GET /api/multi-sms/batches` (opcjonalnie wraz z listą odbiorców).

### Kontrola ryzyka

- **Limit odbiorców na batch:** 250 (ochrona kosztów i przypadkowej wysyłki masowej).
- **Odporność na restart:** stan utrzymywany w bazie, worker po restarcie kontynuuje kolejkę.
- **Audit:** każda wysyłka trafia do historii wiadomości oraz ma wynik per odbiorca.

# News / FAISS (RAG) — scraping, indeksowanie i dystrybucja

Moduł News automatyzuje pobieranie treści (scraping), budowę indeksu FAISS oraz generowanie streszczeń (RAG). Panel pozwala zarządzać odbiorcami i wykonywać testy bez wysyłki.



Widok zarządzania modułem News: konfiguracja, test i uruchomienie scrapingu.

## Co widzisz

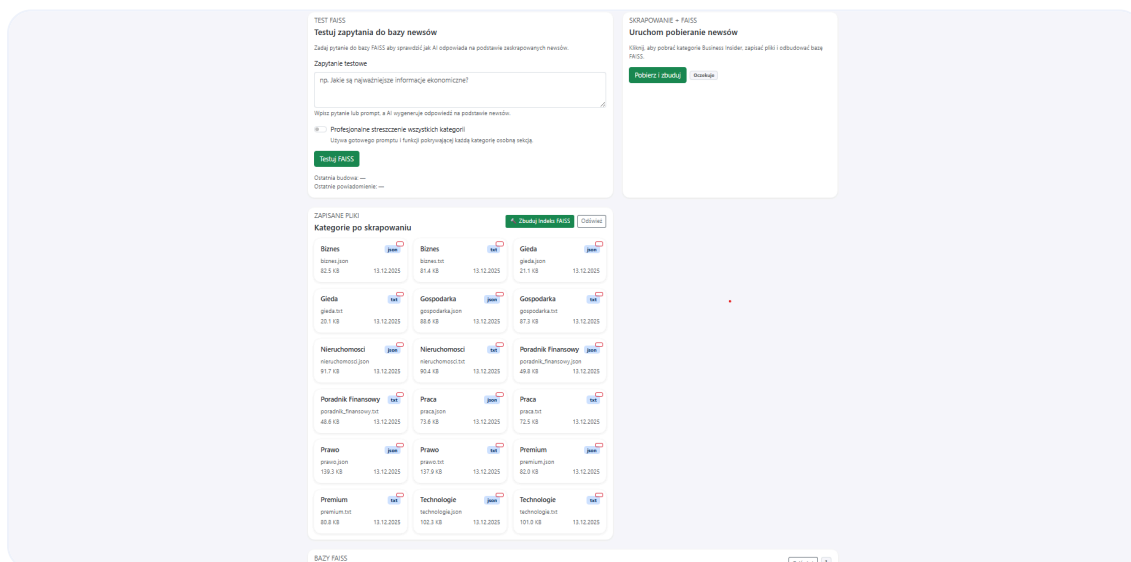
- Akcje administracyjne: test połączenia, uruchomienie scrapingu, podgląd plików źródłowych.
- Informacje o ostatnim buildzie i stanie modułu (czy dane są aktualne).

## Możliwe interakcje

- Test generacji bez wysyłki (bezpieczne sprawdzenie jakości streszczeń i konfiguracji promptu).
- Uruchomienie scrapingu „na żądanie” (ręczny refresh danych) oraz szybka weryfikacja, czy pliki zostały zapisane.

## Co dzieje się w tle

- Scraper pobiera treści i zapisuje pliki w katalogu danych (dla każdej kategorii osobno).
- Po pobraniu danych budowany/aktualizowany jest indeks FAISS, aby zapytania RAG miały świeży kontekst.
- API: `POST /api/news/scrape`, `GET /api/news/files`, `GET /api/news/files/<filename>`.



Definicja odbiorców: numer, tryb promptu oraz godzina wysyłki.

## Co widzisz

- Listę odbiorców powiadomień newsowych (numer + godzina + tryb).
- Flagi **enabled**, informację o ostatniej wysyłce oraz tryb generacji.

## Możliwe interakcje

- Dodanie odbiorcy (numer w E.164) i wybór trybu: wszystkie kategorie (domyślnie) vs prompt niestandardowy.
- Włączenie/wyłączenie odbiorcy oraz test wiadomości bez wysyłki (preview).
- Force-send (wysyłka ręczna) dla odbiorcy w celach operacyjnych i weryfikacji routingu Twilio.

## Co dzieje się w tle

- Konfiguracja odbiorców jest zapisywana w pliku **news\_config.json**.
- Scheduler/worker wysyła o określonej godzinie wygenerowane streszczenia (jeśli odbiorca jest włączony).
- Wysyłka jest realizowana przez Twilio i powinna być widoczna w historii wiadomości.
- API: **GET/POST/DELETE /api/news/recipients** + akcje per odbiorca: **toggle**, **test**, **send**.

The screenshot displays a web interface for managing news recipients and database status. The top section, titled 'ZAPISANE PLIKI' (Saved Files), shows a grid of 18 categories, each with a file icon, name, size, and date. The categories are: Biznes, Gleda, Gospodarka, Nieruchomosci, Poradnik Finansowy, Praca, Prawo, and Premium. The bottom section, titled 'BAZY FAISS', shows a table with the status of the FAISS database. The table has columns for Name, Created, Size, Status, and Actions. The current entry is 'faiss\_openal\_index', created on 13.12.2025 at 18:38, with a size of 10862637 vekt, and a status of 'aktywny' (active). There are buttons for 'Wybierz aktywną bazę / podgląd' (Select active database / view) and 'Ustaw aktywną' (Set active).

Nazwa	Utworzono	Rozmiar	Status	Akcje
faiss_openal_index	13.12.2025 18:38	10862637 vekt.	aktywny	Ustaw aktywną Usun

Status indeksu i operacje: eksport/import backupu, test FAISS i budowa indeksu.

### Co widzisz

- Status indeksu (czy załadowany, ile wektorów, jakie modele embeddings/chat).
- Gotowość backupu (wymagane pliki) oraz operacje import/export.

### Możliwe interakcje

- Eksport backupu (ZIP) i import w celu odtworzenia środowiska (np. po wdrożeniu lub awarii).
- Budowa indeksu z istniejących plików oraz test zapytań FAISS (diagnostyka jakości i „świeżości” danych).

### Co dzieje się w tle

- FAISS przechowuje wektory embeddings + mapowanie do dokumentów.
- Zapytania RAG pobierają kontekst z indeksu i opcjonalnie używają LLM do sformatowania odpowiedzi.
- API: GET /api/news/faiss/status , GET /api/news/faiss/export , POST /api/news/faiss/import , POST /api/news/test-faiss .

### API modułu News (skrót)

- **Konfiguracja/odbiorcy:** GET /api/news/config , GET/POST/DELETE /api/news/recipients , POST /api/news/recipients/<id>/(toggle|test|send)
- **Scraping:** POST /api/news/scrape , pliki: GET/DELETE /api/news/files
- **FAISS:** GET /api/news/faiss/status , GET /api/news/faiss/export , POST /api/news/faiss/import , POST /api/news/indices/build

Import backupu ma limit bezpieczeństwa 250 MB.

## Dane (SQLite) — trwałość, backup i migracje

Aplikacja przechowuje stan w SQLite. Ścieżka bazy jest konfigurowana przez `DB_PATH` (domyślnie `data/app.db`).

### Co jest zapisywane

- **messages**: historia inbound/outbound, SID Twilio, status, błędy i timestamps.
- **multi\_sms\_\***: batch + odbiorcy (statusy, liczniki, błędy per numer).
- **reminders**: harmonogramy cyklicznych wysyłek.
- **news\_config.json**: konfiguracja odbiorców News (plik w katalogu danych).

### Backup (praktycznie)

- Najprostszy backup to kopia pliku DB (przy zatrzymanej aplikacji lub w oknie niskiej aktywności).
- FAISS posiada dedykowany eksport/import ZIP z manifestem (API modułu News).

## API — integracja i automatyzacja

Panel WWW korzysta z REST API. Poniżej najważniejsze punkty integracyjne (bez pełnego OpenAPI).

- **Wiadomości i konwersacje**: `GET /api/messages`, `GET /api/conversations`, `GET /api/conversations/<participant>`
- **Wysyłka**: `POST /api/send-message` (SMS/MMS + Content SID)
- **Multi SMS**: `POST /api/multi-sms/batches`, `GET /api/multi-sms/batches`, `GET /api/multi-sms/batches/<id>`, `GET /api/multi-sms/batches/<id>/recipients`
- **Status callback**: `POST /twilio/status` (aktualizacja statusów wiadomości)

## Glosariusz i najważniejsze pojęcia

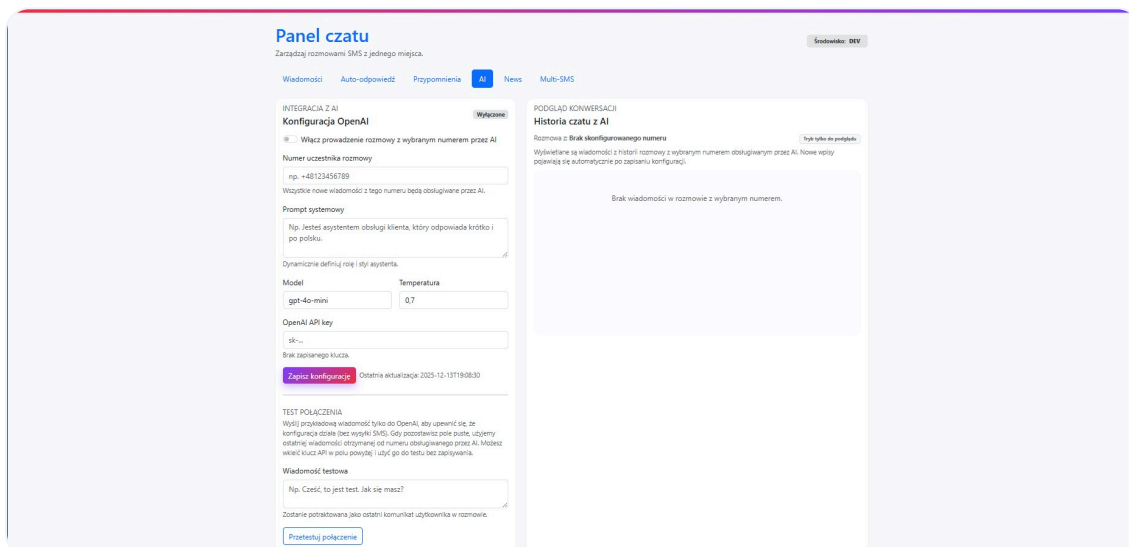
Ten rozdział porządkuje terminologię spotykaną w panelach i API, aby PDF był samowystarczalny dla zespołów biznesowych i technicznych.

- **Batch (Multi SMS)** — zadanie wysyłki jednej wiadomości do wielu odbiorców; posiada status ( `pending` , `processing` , `completed` , `failed` ) i liczniki skuteczności.
- **Recipient (Multi SMS)** — pojedynczy numer przypisany do batcha; system śledzi status ( `pending` / `sent` / `failed` ) oraz ewentualny błąd Twilio.
- **Messaging Service SID** — identyfikator grupujący numery w Twilio; jeśli skonfigurowany, Multi SMS i UI używają go jako nadawcy zamiast pojedynczego numeru.
- **Webhook** — endpoint HTTP nasłuchujący żądań Twilio (np. inbound wiadomości, status dostarczenia). Każdy webhook jest podpisany nagłówkiem `X-Twilio-Signature`.
- **FAISS** — biblioteka wektorowa używana do szybkiego wyszukiwania podobnych dokumentów, podstawowy komponent modułu News/RAG.
- **Scraper** — proces pobierający artykuły i zapisujący je w katalogu danych; jego wynik zasila FAISS i moduł News.
- **Reminder** — harmonogram cyklicznej wysyłki SMS. Każde przypomnienie opisuje numer docelowy, treść i interwał.
- **Auto reply** — mechanizm automatycznych odpowiedzi (klasyczny lub AI). W trybie AI każde zapytanie jest przekazywane do modelu OpenAI z kontekstem rozmowy.
- **Normalized number** — numer telefonu po oczyszczeniu (E.164). Pozwala scalać rekordy w historii i zapobiegać duplikatom.
- **News recipient** — wpis w `news_config.json` ; określa numer, godzinę wysyłki i tryb promptu (wszystkie kategorie vs. prompt niestandardowy).

## AI & Auto reply — zasady działania

AI działa jako globalny tryb auto reply; klasyczny auto reply jest wyłączany, gdy AI jest aktywne. Dodatkowo każdy tryb operuje tylko na wiadomościach nowszych niż punkt aktywacji (bez retro odpowiedzi).

1. AI buduje historię rozmowy z tabeli `messages` i wysyła zapytanie do OpenAI.
2. Odpowiedzi są wysyłane tak, aby zachować wątek konwersacji (preferowana metoda `send\_reply\_to\_inbound`).
3. Operator może przetestować model w zakładce AI bez wysyłania SMS (endpoint `POST /api/ai/test`).



Interfejs konfiguracji AI oraz podgląd rozmowy w formie bąbelków



### Co widzisz

- Konfigurację AI (włączenie, parametry, prompt/system) oraz stan aktywacji.
- Podgląd rozmowy w formie bąbelków, dzięki czemu operator widzi kontekst, na którym bazuje model.

### Możliwe interakcje

- Włączenie AI auto reply i ustawienie promptu systemowego pod styl firmy i politykę odpowiedzi.
- Test generacji odpowiedzi bez wysyłki (bezpieczny preview) oraz dopasowanie parametrów.

### Co dzieje się w tle

- AI buduje kontekst z historii wiadomości i wywołuje OpenAI wg ustawień (model/temperatura/limit historii).
- Jeśli tryb auto reply jest aktywny, inbound webhook enqueue'uje zadanie odpowiedzi (bez blokowania requestu).
- Wysyłka odpowiedzi przez Twilio jest rejestrowana w `messages`; statusy mogą się aktualizować callbackiem.
- System unika odpowiedzi na starsze wiadomości sprzed aktywacji trybu (brak retro odpowiedzi).

## Bezpieczeństwo — kluczowe zasady

System jest zbudowany tak, aby minimalizować ryzyko nieautoryzowanych wywołań webhooków, wycieków kluczy oraz niekontrolowanej wysyłki.

### Webhooki Twilio: walidacja podpisu

- Callbacki Twilio są weryfikowane nagłówkiem `X-Twilio-Signature` (RequestValidator).
- Walidację można wyłączyć tylko dla testów lokalnych przez `TWILIO_VALIDATE_SIGNATURE=false` (produkcyjnie: zostawić włączoną).

### Klucze i konfiguracja

- Dane dostępowe Twilio są pobierane ze zmiennych środowiskowych i nie powinny trafiać do repozytorium.
- DB jest plikiem (SQLite) — w produkcji zadбай o uprawnienia do katalogu `data/` oraz backup.

### Kontrola kosztów

- Multi SMS ma limit odbiorców per batch.
- News wysyła wiadomości tylko do włączonych odbiorców; testy mogą działać bez wysyłki.

## Operacje — uruchomienie i utrzymanie

Aplikacja uruchamia workery w tle (m.in. Multi SMS, przypomnienia, News). Najważniejsze operacyjnie jest przewidywalne środowisko, logi i backup danych.

### Środowisko

- **DB:** `DB_PATH` (domyślnie `data/app.db` )
- **Twilio:** `TWILIO_ACCOUNT_SID` , `TWILIO_AUTH_TOKEN` , oraz `TWILIO_DEFAULT_FROM` lub `TWILIO_MESSAGING_SERVICE_SID`
- **AI/FAISS:** konfiguracja OpenAI (jeśli używana) + pliki indeksu w katalogu danych

### Diagnostyka

- Jeśli Multi SMS nie wysyła: sprawdź logi workera oraz czy tabele `multi_sms_batches` i `multi_sms_recipients` istnieją w DB.
- Jeśli News/FAISS nie działa: sprawdź status `GET /api/news/faiss/status` i czy scraping zbudował indeks.

### Backup

- SQLite: kopia pliku DB + rotacja (np. dzienna).
- FAISS: eksport ZIP przez endpoint i/lub zrzut katalogu indeksu.

## Deployment & generowanie PDF

Plik HTML jest przygotowany do konwersji do PDF. Poniżej przykładowe polecenia do wygenerowania wysokiej jakości PDF lokalnie.

### Jak wygenerować PDF

Rekomendowane: użyj headless Chrome lub wkhtmltopdf. Poniżej znajdują się dwie metody.

#### Headless Chrome

```
google-chrome --headless --disable-gpu --no-sandbox --print-to-pdf=twilio-chat-app.pdf  
file://$(pwd)/deploy/releases/full_documentation.html
```

#### wkhtmltopdf

```
wkhtmltopdf deploy/releases/full_documentation.html twilio-chat-app.pdf
```

Obie metody pozwalają na generację zaawansowanego, paginowanego PDF z poprawnymi stylami CSS. Jeżeli potrzebujesz, przygotuję skrypt automatyzujący proces — udostępniłem przykładowe skrypty w katalogu `scripts/`.