DOKUMENTACJA TECHNICZNA
DLA PROJEKTU:
„AI Asystent kalendarza Google”

Zrobiona przez:
Volodymyr Haideichuk, 282267
Sławomir Salamon, 280486

Wrocław, 2025

Spis treści

Wykorzystane technologie:

- pyTelegramBotAPI:
  Opis: biblioteka Pythona będąca wrapperem dla Telegram Bot API. Zapewnia prosty interfejs do tworzenia i zarządzania botami Telegrama, obsługując m.in. wysyłanie wiadomości, reagowanie na komendy, obsługę przycisków inline oraz webhooków. Wspiera zarówno tryb polling, jak i webhook. Idealna do szybkiego prototypowania botów.

- Google Gemini AI:
  Opis: zaawansowany model językowy opracowany przez Google, służący do przetwarzania i generowania tekstu. W projekcie wykorzystywany jako komponent AI do przetwarzania języka naturalnego (NLP).

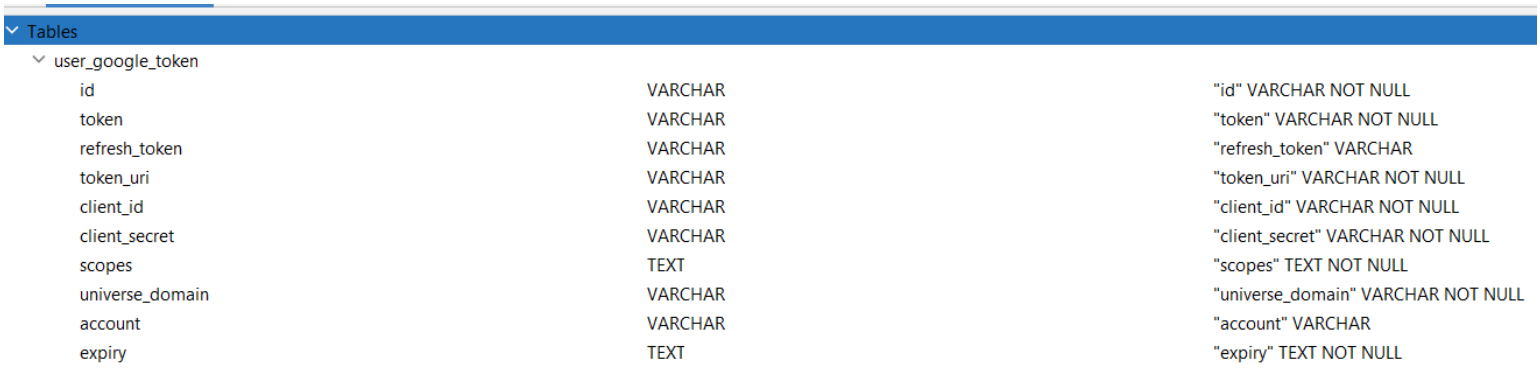
- Google Auth:
  Opis: mechanizm uwierzytelniania użytkowników za pomocą konta Google, oparty na protokole OAuth 2.0. Umożliwia bezpieczne logowanie do aplikacji z użyciem danych Google, eliminując potrzebę tworzenia osobnych kont. W projekcie wykorzystany do autoryzacji użytkowników i zarządzania dostępem kalendarza.

- Google Calendar API:
  Opis: interfejs programistyczny umożliwiający integrację aplikacji z Kalendarzem Google. Umożliwia tworzenie, odczytywanie, aktualizowanie i usuwanie wydarzeń w kalendarzu użytkownika. W projekcie wykorzystywany do zarządzania wydarzeniami i wykonywania wyszukiwani według podanych kryteriów.

Baza danych:

W projekcie wykorzystywana jest jedna tabela: *user_google_token.* Struktura ta jest w pełni wystarczająca do działania aplikacji, ponieważ umożliwia powiązanie użytkownika Telegrama (na podstawie jego unikalnego user_id) z tokenem dostępu do Google Calendar API. Tabela przechowuje również wszystkie niezbędne dane uwierzytelniające, co pozwala na bezpieczne i ciągłe zarządzanie wydarzeniami w kalendarzu Google z poziomu bota Telegrama. Dane są przechowywane w sposób zapewniający wysoki poziom bezpieczeństwa.

Poniżej została przedstawiona struktura bazy danych:



Opis głównych funkcjonalności:

Telegram:

- send_welcome
  - Argumenty:
    - message – obiekt który zawiera dane wiadomości wysłane przez użytkownika, np: id użytkownika, id chatu gdzie została wysłana wiadomość i td.
  - Co robi:
    - Sprawdza czy użytkownik jest zalogowany. Wysyła użytkownikowi powitalną wiadomość i prosi się zalogować jeśli użytkownik nie jest jeszcze zalogowany
- send_colors
  - Argumenty:
    - message – obiekt który zawiera dane wiadomości wysłane przez użytkownika, np: id użytkownika, id chatu gdzie została wysłana wiadomość i td.
  - Co robi:
    - Wysyła użytkownikowi listę dostępnych kolorów dla wydarzenia
- logout_user
  - Argumenty:
    - message – obiekt który zawiera dane wiadomości wysłane przez użytkownika, np: id użytkownika, id chatu gdzie została wysłana wiadomość i td.
  - Co robi:
    - Umożliwia użytkownikowi wylogowanie się, żeby bot nie miał dostępu do kalendarza
- send_command_message
  - Argumenty:
    - message – obiekt który zawiera dane wiadomości wysłane przez użytkownika, np: id użytkownika, id chatu gdzie została wysłana wiadomość i td.
  - Co robi:
    - Sprawdza czy użytkownik jest zalogowany i w przypadku gdy nie jest – przypomina o zalogowaniu. Parsuje tekst który wprowadził użytkownik i wysyła wiadomość-potwierdzenie jeśli wiadomość jest sensowna, w przeciwnym przypadku prosi żeby użytkownik napisał klarowniej

- send_callback
    - Argumenty:
        - call – obiekt który jest tworzony w przypadku gdy użytkownik klika w przycisk InlineKeyboardButton i zawiera w sobie np. id użytkownika, id chatu i td
    - Co robi:
        - Jeśli użytkownik potwierdził wykonanie określonej czynności przez bota to bot wysyła wiadomość że wykonuje tą czynność. W przypadku niepotwierdzenia bot prosi o wpisanie dokładniejszego polecenia
- send_remove_answer
    - Argumenty:
        - message – obiekt który zawiera dane wiadomości wysłane przez użytkownika, np: id użytkownika, id chatu gdzie została wysłana wiadomość i td.
    - Co robi:
        - Jeśli użytkownik wybrał opcję „Anuluj” – bot wysyła wiadomość o tym, że żadne wydarzenie nie zostało usunięte. W przeciwnym przypadku usuwa wybrane wydarzenie oraz bot powiadamia o tym użytkownika
- send_update_answer
    - Argumenty:
        - message – obiekt który zawiera dane wiadomości wysłane przez użytkownika, np: id użytkownika, id chatu gdzie została wysłana wiadomość i td.
    - Co robi:
        - Jeśli użytkownik wybrał opcję „Anuluj” – bot wysyła wiadomość o tym, że żadne wydarzenie nie zostało zaktualizowane. W przeciwnym przypadku aktualizuje wybrane wydarzenie oraz bot powiadamia o tym użytkownika
- send_notifications
    - Argumenty:
        - bot – obiekt naszego bota.
    - Co robi:
        - Działa w osobnym wątku. Co 10 sekund sprawdza czy są nadchodzące wydarzenia dla których trzeba wysłać przypomnienie. Jeśli tak, to bot asynchronicznie wysyła przypomnienie o nadchodzących wadarzeniach.

- send_notifications_for_day
  - Argumenty:
    - bot – obiekt naszego bota.
  - Co robi:
    - Działa w osobnym wątku. Codziennie o 6 rano bot asynchronicznie wysyła wydarzenia jakie użytkownik ma na obecny dzień

Utils:

- parse_json_to_bot_answer
  - Argumenty:
    - data – sparsowany json obiekt odpowiedzi użytkownika
  - Zwraca:
    - string
  - Co robi:
    - Przekształca sparsowany json obiekt na czytelny format dla użytkownika
- manage_events
  - Argumenty:
    - parsed_ai_json – sparsowany json obiekt wiadomości użytkownika
    - user_id – id użytkownika w Telegram
  - Zwraca:
    - string
    - list[Event]
  - Co robi:
    - Zarządza wydarzeniami. Jeśli użytkownik chce dodać wydarzenie to od razu go dodaje i zwraca string, że wydarzenie zostało dodane. W przeciwnym przypadku wyszukuje potrzebne wydarzenia i zwraca listę wydarzeń
- construct_events
  - Argumenty:
    - events – lista wydarzeń użytkownika w kalendarzu
    - title – tytuł wiadomości
  - Zwraca:
    - string
  - Co robi:

        - Przekształca listę wydarzeń z Google na klarowną wiadomość dla użytkownika
- update_event
    - Argumenty:
        - event – wydarzenie które użytkownik chce zaktualizować
        - user_id – id użytkownika w Telegram
        - parsed_ai_json - sparsowany json obiekt wiadomości użytkownika
    - Co robi:
        - Wyciąga dane które użytkownik chce zaktualizować, za pomocą funkcji z Google Calendar API aktualizuje wydarzenie

Gemini AI

- detect_event_type
    - Argumenty:
        - user_input – tekst użytkownika
    - Zwraca:
        - type: EventType
    - Co robi:
        - Wyznacza jakiego typu jest operacja którą użytkownik chce zrobić. W przypadku nieznanej operacji jest zwracana wartość Unknown
- ai_parse_text
    - Argumenty:
        - user_input – tekst użytkownika
    - Zwraca:
        - result – sparsowany tekst użytkownika w formacie json według podanej schemy
    - Co robi:
        - Parsuje tekst użytkownika na format json według podanej schemy

- Klasa EventType – enum, które określa jaki typ operacji użytkownik chce wykonać: Add, Edit, Remove, Show, Unknown
- Klasa AddEvent – schema według której AI parsuje tekst użytkownika jeśli celem jest dodanie wydarzenia
- Klasa ShowAndRemoveEvent – schema według której AI parsuje tekst użytkownika jeśli celem jest pokazanie lub usuwanie wydarzenia
- Klasa EditEvent – schema według której AI parsuje tekst użytkownika jeśli celem jest edycja wydarzenia

GoogleCalendarAPI.py

- getCredentials
  - Argumenty:
    - user_token_dict – przyjmuje obiekt słownika (dict) posiadający dane z danych .json od google calendar api
  - Zwraca:
    - Obiekt google. oauth2.credentials
  - Wyjątki:
    - Google.auth.exceptions.RefreshError
    - Google.oauth2.credentials.exceptions.DeafultCredentialsError
  - Co robi:
    - Konwertuje słownik (user_token_dict) na obiekt Credentials jeśli to możliwe. Jeśli można odświeżą token
- getEvents
  - Argumenty:
    - user_token_dict:dict,
    - calendar_id: str = 'primary',
    - time_min: str = datetime(1970, 1, 1, tzinfo=timezone.utc).isoformat(),
    - time_max: str = datetime(2100, 1, 1, tzinfo=timezone.utc).isoformat(),
    - max_results: int = 200,
    - order_by: str = 'startTime',
    - show_deleted: bool = False,
    - single_events: bool = True,
    - show_hidden_invitations: bool = False,
    - updated_min: Optional[datetime] = None,
    - always_include_email: bool = True,
    - query=''
  - Zwraca:
    - List[Event]
    - None
  - Co robi:
    - Wyszukuje eventy na podstawie argumentów i zwraca ich liste
- getEvents_by_id
  - Argumenty:
    - user_token_dict:dict
    - event_id
    - calendar_id: str = 'primary'

- Zwraca:
  - Event
  - None
- Co robi:
  - Wszukuje eventy na podstawie id i go zwraca lub zwraca None

- addEvent
  - Argumenty:
    - user_token_dict:dict
    - event: Event,
    - calendar_id: str = 'primary',
  - Zwraca:
    - Event
    - None
  - Co robi:
    - Dodaje event i zwraca dodany event lub zwraca None
- updateEvent
  - Argumenty:
    - user_token_dict:dict,
    - event: Event,
    - calendar_id: str = 'primary'
  - Wyjątki:
    - ValueError
  - Zwraca:
    - Event
    - None
  - Co robi:
    - Aktualizuje event i zwraca dodany event lub zwraca None
- updateEvent
  - Argumenty:
    - user_token_dict:dict,
    - event: Event,
    - calendar_id: str = 'primary'
  - Wyjątki:
    - ValueError
  - Zwraca:
    - Event

        - None
    - Co robi:
        - usuwa event i zwraca dodany True lub zwraca False
- Klasa Reminder
    - Funkcje:
        - setUserToken:
            - argumenty:
                - user_token_dict
            - co robi:
                - przypisuje _user_token_dict:=user_token_dict
        - update:
            - argumenty:
                - time_max
            - co robi:
                - aktualizuje i dodaje przyszłe przypomnienia(od czasu teraźniejszego do time_max) w postaci tuple(identyfikator eventu,data przypomnienia), i sortuje
        - get:
            - argumenty:
                - now=None
            - zwraca
                - list[Event]
            - co robi:
                - zwraca eventy których data <= czasu teraźniejszego

Event.py

- Klasa Event
    - Funkcje:
        - __init__
            - Argumenty
                - Data:dict=None
            - Co robi
                - Tworzy obiekt Event na podstawie słownika

- Klasa EventBuilder
    - Funkcje:

- __init__
  - Argumenty
    - Event=Event()
  - Co robi
    - Tworzy obiekt EventBuilder
- as_calendar_event
  - Co robi:
    - Ustawia event jako calendar#event
- with_summary
  - argumenty:
    - summary:str
  - Co robi:
    - Ustawia tytuł eventu
- with_description
  - argumenty:
    - description:str
  - Co robi:
    - Ustawia opis eventu
- with_location
  - argumenty:
    - location:str
  - Co robi:
    - Ustawia lokalizacje eventu
- with_start_date
  - argumenty:
    - dateTime:str (poprawny format: *2026-07-01T15:31:00+00:00 ISO)*
  - Co robi:
    - Ustawia czas rozpoczęcia eventu
- with_end_date
  - argumenty:
    - dateTime:str (poprawny format: *2026-07-01T15:31:00+00:00 ISO)*
  - Co robi:
    - Ustawia czas zakończenia eventu
- with_attendees
  - argumenty:

        - attendees_emails:list | str (jeśli str email@domena.pl,email2@domena.pl)
    - Co robi:
        - Dodaje uczestników do eventu
- with_color_id
    - argumenty:
        - color_id:str poprawne wartości (0,1,2,3,4,5,6,7,8,9)
    - Co robi:
        - Dodaje kolor do eventu
- with_reminders
    - argumenty:
        - useDefault:bool,
        - overrides:list[dict[str,str | int]] format: [{'method':'popup' | 'email', 'minutes':int>=0}]
    - Co robi:
        - Dodaje przypomnienia
- add_reminder
    - argumenty:
        - method {‘popup’ | ‘emai’}
        - minutes {int >=0}
    - Co robi:
        - Dodaje przypomnienie
- with_status
    - argumenty:
        - status {‘confirmed’,tentative’,’canceled’}
    - Co robi:
        - Ustawia status eventowi
- with_creator
    - argumenty:
        - email:str
        - displayName:str = None
    - Co robi:
        - Dodaje twórce do eventu
-
- with_anyoneCanAddSelf
    - argumenty:
        - b:bool
    - Co robi:

            - Ustawia atrybut anyoneCanAddSelf eventowi
        - with_guestsCanInviteOthers
            - argumenty:
                - b:bool
            - Co robi:
                - Ustawia atrybut guestsCanInviteOthers eventowi
        - with_guestsCanModify
            - argumenty:
                - b:bool
            - Co robi:
                - Ustawia atrybut guestsCanModify eventowi
        - with_guestsCanSeeOtherGuests
            - argumenty:
                - b:bool
            - Co robi:
                - Ustawia atrybut guestsCanSeeOtherGuests eventowi
        - Build
            - Co robi:
                - Tworzy ostateczny event

CredenctialsFunctions.py

- create_authorization_url
    - Argumenty:
        - user_id
    - Zwraca:
        - str
    - Co robi:
        - Tworzy URL dla użytkownika do logowania
- create_authorization_url
    - Argumenty:
        - user_id
    - Zwraca:
        - str
    - Co robi:
        - Tworzy URL dla użytkownika do logowania
- check_user_credentials
    - Argumenty:
        - user_id
    - Zwraca:

        - bool
    - Co robi:
        - Sprawdza czy user posiada credentials w bazie danych
- get_user_credentials
    - Argumenty:
        - user_id
    - Zwraca:
        - dict
    - Co robi:
        - Zwraca credentiale user zapisane w bazie
- delete_user_credentials
    - Argumenty:
        - user_id
    - Zwraca:
        - dict
    - Co robi:
        - Usuwa credentiale user zapisane w bazie