DOKUMENTACJA TECHNICZNA

DLA PROJEKTU:

„AI Asystent kalendarza Google”

Zrobiona przez:

Volodymyr Haideichuk, 282267

Sławomir Salamon, 280486

Wrocław, 2025

Spis treści

[Wykorzystane technologie: 3](#_Toc200800704)

[Baza danych: 5](#_Toc200800705)

[Opis głównych funkcjonalności: 6](#_Toc200800706)

[Telegram: 6](#_Toc200800707)

[Utils: 8](#_Toc200800708)

[Gemini AI 9](#_Toc200800709)

[GoogleCalendarAPI.py 10](#_Toc200800710)

[Event.py 12](#_Toc200800711)

[CredenctialsFunctions.py 15](#_Toc200800712)

# 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.

Obraz zawierający tekst, zrzut ekranu, numer, oprogramowanie

Zawartość wygenerowana przez AI może być niepoprawna.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