Dokumentacja aplikacji Unordered

Adam Dybcio
Łukasz Czapski
Igor Ciżewski
Denis Jabłoński
Mateusz Sztankiewicz

20 maja 2025

Spis treści

1 Wprowadzenie

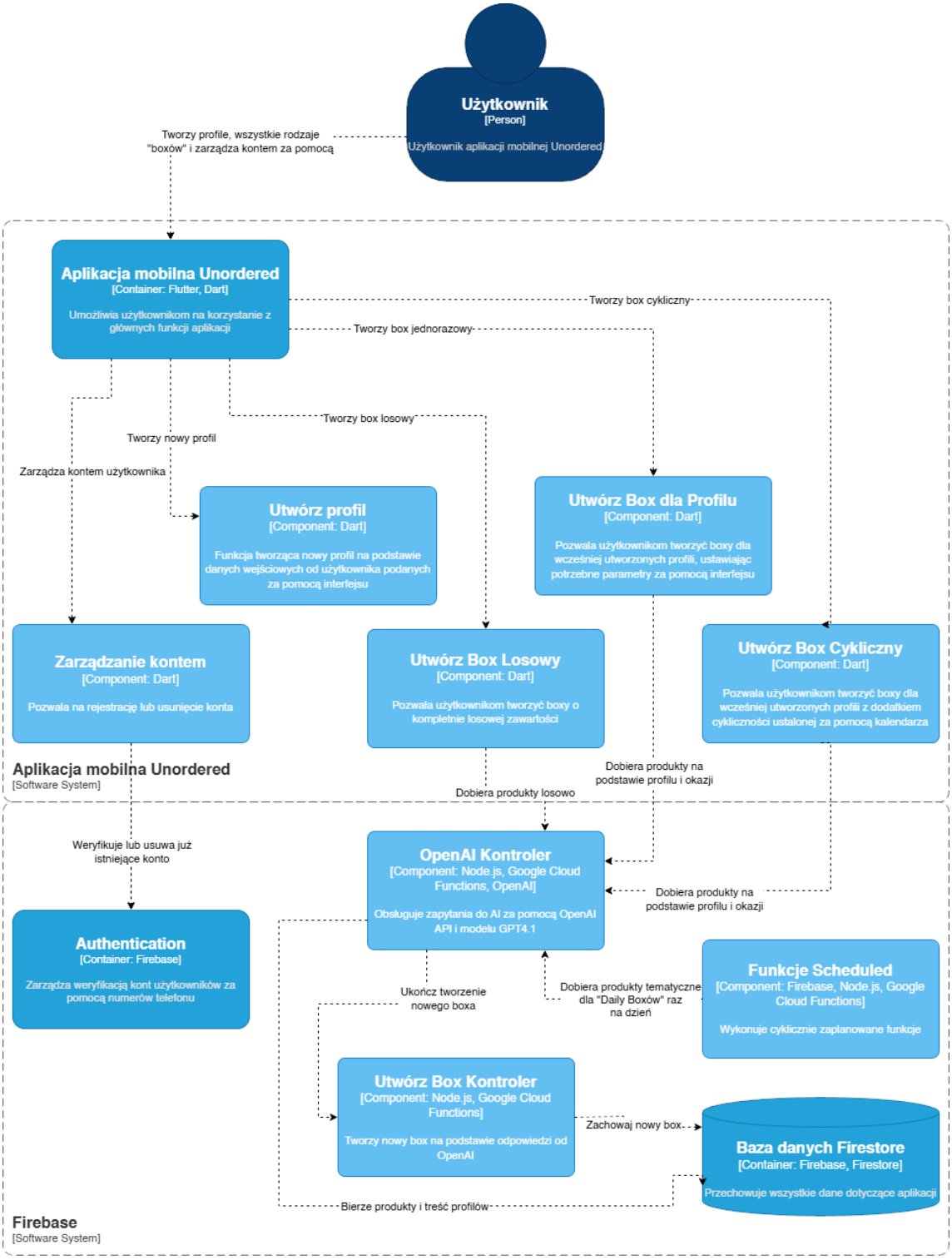
1.1 Cel dokumentu

Dokumentacja ma na celu przedstawienie architektury oraz funkcjonalności aplikacji Unordered. Zawiera również informacje o integracji z OpenAI oraz wdrożeniu aplikacji.

1.2 Opis aplikacji

2 Architektura

2.1 Diagram wysokopoziomowy



Rysunek 1: Architektura systemu

2.2 Frontend (Flutter)

- Wersja Flutter: [X.Y.Z]
- Główne pakiety: `firebase_core`, `cloud_functions`, `http`, etc.
- Struktura katalogów:

2.3 Backend (Firebase)

- **Baza danych:** Firestore - przechowuje dane użytkowników, zapisane profile, utworzone boxy, listy produktów i jakiekolwiek inne potrzebne dane do działania aplikacji.
- **Funkcjonalość:** Cloud Functions (Node.js) - obsługuje główne działania backendu aplikacji, takie jak generowanie boxów, komunikacja i przetwarzanie danych z OpenAI czy pobieranie aktualnej listy produktów.
- **Uwierzytelnianie:** Firebase Authentication - pozwala użytkownikom na rejestrację i logowanie się do aplikacji za pomocą numeru telefonu, używając weryfikacji SMS.
- **Automatyzacja:** Cloud Functions Scheduler - służy do cyklicznego uruchamiania funkcji, obsługuje m.in. generowanie "Daily Boxów" - codziennych propozycji prezentów o losowych kategoriach.

3 Funkcjonalności

3.1 Generowanie boxów

3.1.1 Typy generowania

- **Dla profilu** - Generacja boxów na podstawie profilu, który został wcześniej utworzony. Użytkownik podaje budżet oraz okazję, na którą box ma być wygenerowany.
- **Cykliczny** - Generacja boxów cyklicznych, które będą generowane co określony czas (np. co miesiąc, co rok). Wymagane parametry jak w przypadku boxów "Dla profilu".
- **Losowy** - Generacja losowych propozycji prezentów, bez podawania żadnych parametrów
- **Manualny** - aktualnie brak implementacji

3.1.2 Profil użytkownika

Do wygenerowania boxów typu "Dla profilu" lub "Cykliczny" wymagane jest utworzenie profilu, dla którego dobierane będą propozycje. Dane profilu to:

- **Nazwa profilu** (np. "Mama", "Tata", "Kolega")
- **Opis osoby** (Preferencje, zainteresowania, itp.)
- **Zainteresowania** (Wybór kategorii z gotowej listy)

3.2 Struktura boxa

Struktura boxów jest taka sama dla wszystkich typów generowania. Boxy są przechowywane w bazie danych Firestore i zawierają następujące dane:

- **Budżet** (Kwota, którą użytkownik chce przeznaczyć na prezenty)
- **Data wygenerowania** (Data, kiedy box został wygenerowany)
- **Czy jest cykliczny** (Wartość boolean, czy box jest cykliczny)
- **Frekwencja cykliczności** (Użyte tylko dla boxów cyklicznych, np. co miesiąc, co rok)
- **Okazja** (Okazja, na którą box został wygenerowany, np. urodziny, imieniny)
- **Planowana data** (Data, na którą box został zaplanowany)
- **Cena** (Suma cen wszystkich produktów w boxie)
- **Lista produktów** (Produkty, które zostały wygenerowane w boxie)

4 Integracja z OpenAI

4.1 Opis API

Wybranym do generacji modelem jest **gpt-o4-mini**. Wybór modelu jest uzasadniony jego działaniem w trybie "reasoning" który pozwala na generacje biorącą pod uwagę wszystkie parametry podane przez użytkownika. Jest on również tańszym wyborem, w porównaniu do modeli **gpt-4.1** oraz **gpt-o3**.

4.2 Prompt engineering

Przy inżynierii promptów uwaga była skupiona na wykorzystaniu najmniejszej ilości tokenów przy zachowaniu jak najlepszej jakości odpowiedzi. Strukturą wszystkich promptów jest: **ZASADA→OPIS ZASADY**

- **Generowanie boxów dla profilu** - generacja boxów na podstawie profilu użytkownika.

  ANALIZA→Uwzględnij zainteresowania, okazję (...)
  RÓŻNORODNOŚĆ→Unikaj powtórzeń, (...)
  BUDŻET→Nigdy nie przekraczaj budżetu!(...)
  ODPOWIEDŹ→Zwróć TYLKO JSON array (...)

- **Generowanie "Daily Boxów"** - generacja boxów o różnych tematach, raz dziennie.

  DOBÓR→Uwzględnij temat, unikaj innych kategorii. (...)
  RÓŻNORODNOŚĆ→Unikaj powtórzeń, (...)
  BUDŻET→Nigdy nie przekraczaj budżetu! (...)
  ODPOWIEDŹ→Zwróć TYLKO JSON array (...)

- **Generowanie losowych boxów** - generacja boxów bez podawania żadnych parametrów.

  LOSOWOŚĆ→Wybierz losowe przedmioty (...)
  RÓŻNORODNOŚĆ→Dobieraj różne kategorie.
  BUDŻET→Utrzymaj realistyczny budżet.
  ODPOWIEDŹ→Zwróć TYLKO JSON array (...)

4.3 Przetwarzanie odpowiedzi

Formatem zwróconej odpowiedzi jest JSON array, który zawiera listę samych ID produktów, aby zminimalizować ilość tokenów. Następnie ID są sprawdzane oraz przetwarzane przez backend.

5 Wdrożenie

5.1 Konfiguracja Firebase

Aby wdrożyć funkcje chmurowe (Cloud Functions) za pomocą Firebase, należy użyć narzędzia `firebase` dostępnego przez npm, oraz zainicjować projekt i wdrożyć funkcję do projektu Firebase. Po wdrożeniu należy upewnić się, że adresy (linki) do nowych funkcji są zgodne zarówno w kodzie backendu (Cloud Functions), jak i w wywołaniach po stronie frontendu Flutter. Spójność nazw i endpointów jest kluczowa dla poprawnej komunikacji między aplikacją a backendem.

6 Podsumowanie i dalszy rozwój

6.1 Znane ograniczenia

- Ograniczenia, limity związane np z OpenAI, firebase, etc.

6.2 Roadmap

Planowane funkcje (np. integracja z sklepami).

Literatura

[1] Flutter documentation, `https://flutter.dev/docs`

[2] Firebase documentation, `https://firebase.google.com/docs`

[3] OpenAI API reference, `https://platform.openai.com/docs/api-reference`