Zaawansowane metody programowania

Bimbeer

1 Opis funkcjonalny systemu

Bimbeer to innowacyjny system składający się z aplikacji webowej, desktopowej i mobilnej. Jest to klon popularnej aplikacji randkowej Tinder, ale z unikalnym podejściem skierowanym do miłośników piwa. Dzięki Bimbeer użytkownicy mogą poznać innych miłośników kosztowania browarów i umówić się z nimi na spotkanie w realnym świecie.

1.1 Funkcjonalność aplikacji obejmuje:

- Uwierzytelnianie użytkownika przy pomocy maila;
- Uwierzytelnianie przy pomocy social media;
- Customizacja profilu;
- Chat real-time;
- Statusy wiadomości;
- Proponowanie użytkowników na podstawie preferencji;
- Proponowanie użytkowników na podstawie lokalizacji;
- Swipe'owanie profili;
- Panel wyboru lubianych browarów;
- Tworzenie par;
- Rozłączanie par.

2 Streszczenie opisu technologicznego

2.1 WEB

- VUE.js - otwartoźródłowy framework do aplikacji webowych typu front-end, oparty na języku JavaScript;

- Firebase - platforma do tworzenia i zarządzania aplikacjami mobilnymi i webowymi, oferująca m.in. bazę danych w czasie rzeczywistym oraz autentykację użytkowników.
- Vercel - platforma chmurowa jako firma usługowa. Firma utrzymuje framework do tworzenia stron internetowych Next.js. Architektura Vercel jest zbudowana wokół Jamstack, a wdrożenia są obsługiwane przez repozytoria Git.
- ChakraUI for VUE.js - biblioteka komponentów interfejsu użytkownika dla Reacta, umożliwiająca szybkie tworzenie atrakcyjnych i spójnych interfejsów.
- GitHub - miejsce do kontroli wersji kodu. Pozwala na łatwą synchronizację między członkami grupy oraz na prostą kontrolę zmian w kodzie.
- ESLint - narzędzie do analizy statycznej kodu JavaScript w celu wykrycia problemów i niezgodności ze standardami kodowania.
- Prettier - narzędzie do formatowania kodu, zapewniające spójny styl kodowania w całym projekcie.

2.2 Desktop

- Electron - framework umożliwiający tworzenie aplikacji desktopowych przy użyciu technologii webowych.
- React - biblioteka JavaScript do tworzenia interfejsów użytkownika.
- React Router - biblioteka do zarządzania nawigacją w aplikacjach Reacta.
- React Fast Refresh - funkcja umożliwiająca szybkie odświeżanie komponentów Reacta podczas rozwoju aplikacji.
- Webpack - narzędzie do budowania i pakowania aplikacji JavaScript.
- Firebase - platforma do tworzenia i zarządzania aplikacjami mobilnymi i webowymi, oferująca m.in. bazę danych w czasie rzeczywistym oraz autentykację użytkowników.
- ChakraUI - biblioteka komponentów interfejsu użytkownika dla Reacta, umożliwiająca szybkie tworzenie atrakcyjnych i spójnych interfejsów.
- GitHub - miejsce do kontroli wersji kodu. Pozwala na łatwą synchronizację między członkami grupy oraz na prostą kontrolę zmian w kodzie.
- ESLint - narzędzie do analizy statycznej kodu JavaScript w celu wykrycia problemów i niezgodności ze standardami kodowania.
- Prettier - narzędzie do formatowania kodu, zapewniające spójny styl kodowania w całym projekcie.

2.3 Mobile

Aplikacja mobilna została napisana przy wykorzystaniu języka Dart wraz z cross-platformowym framework'iem Flutter. Implementacja obejmuje jedynie część wymaganą do uruchomienia aplikacji na platformie android.

Lista najważniejszych technologii i bibliotek, na bazie których powstała aplikacja:

- Dart + Flutter - język programowania i framework do tworzenia aplikacji na różne platformy. Flutter oferuje szybkość i wydajność, a także hot reload, ułatwiający iteracyjną pracę deweloperów.
- Bloc - wzorzec projektowy umożliwiający separację logiki biznesowej od warstwy prezentacyjnej w aplikacjach. Logika biznesowa jest zaimplementowana jako klasa BLoC, która obsługuje zdarzenia i generuje stany aplikacji.
- Firebase - platforma backendowa oferująca wiele usług, takich jak autentykacja użytkowników, baza danych Firestore i przechowywanie plików w chmurze. Firebase zapewnia również narzędzia do analityki i monitorowania aplikacji.
- Rxdart - biblioteka do programowania reaktywnego w języku Dart, umożliwiająca tworzenie i transformację strumieni danych w aplikacjach.
- Formz - biblioteka do tworzenia i walidacji formularzy w aplikacjach Flutter, oferująca wiele gotowych walidatorów i ułatwiająca obsługę różnych typów pól.
- SwipeCards - biblioteka umożliwiająca przesuwanie kart w aplikacjach z wieloma konfigurowalnymi opcjami.
- Github - platforma hostingowa dla projektów programistycznych oferująca narzędzia do kontroli wersji, zarządzania projektem i łatwej współpracy z innymi deweloperami.

3 Streszczenie wykorzystanych wzorców projektowych

3.1 WEB

- Conditional Rendering - wzorzec pozwalający na wyświetlanie różnych komponentów lub elementów w zależności od spełnienia określonych warunków.
- Komponenty prezentacyjne i kontenerowe (Presentational and container components) - wzorzec polegający na oddzieleniu logiki zarządzania stanem i danymi od prezentacji.
- Komponenty wyższego rzędu (Higher-order components, HOC) - wzorzec służący do ponownego wykorzystania logiki komponentów.
- Provider pattern - wzorzec umożliwiający udostępnianie wartości w całym drzewie komponentów bez konieczności przekazywania ich przez każdy poziom.

3.2 Desktop

- Komponenty wyższego rzędu (Higher-order components, HOC) - wzorzec służący do ponownego wykorzystania logiki komponentów.
- Provider pattern - wzorzec umożliwiający udostępnianie wartości w całym drzewie komponentów bez konieczności przekazywania ich przez każdy poziom.
- Hooks Pattern - wzorzec umożliwiający wykorzystanie stanu i innych funkcjonalności Reacta w komponentach funkcyjnych.
- Conditional Rendering - wzorzec pozwalający na wyświetlanie różnych komponentów lub elementów w zależności od spełnienia określonych warunków.
- Komponenty prezentacyjne i kontenerowe (Presentational and container components) - wzorzec polegający na oddzieleniu logiki zarządzania stanem i danymi od prezentacji.
- Singleton - wzorzec pozwalający zapewnienie istnienia wyłącznie jednej instancji danej klasy.

3.3 Mobile

- Feature first approach - sposób strukturyzacji projektu, polegający na umieszczaniu elementów oprogramowania dotyczących określonej funkcjonalności w przeznaczonych dla nich folderach
- Bloc (Business Logic Component) to wzorzec projektowy, który jest oparty na podejściu "Separation of Concerns" (SoC), czyli separacji zadań. W praktyce oznacza to, że Bloc oddziela logikę biznesową od prezentacji (warstwy widoku) oraz od zarządzania stanem (warstwy modelu). Bloc wykorzystuje podejście unidirectional data flow, w którym zdarzenia (Events) przekazywane są do Bloca, który przetwarza je i zwraca nowy stan (State). Widok (View) reaguje na zmiany stanu i aktualizuje interfejs użytkownika. Dzięki temu podejściu, Bloc zapewnia separację zadań, co ułatwia utrzymanie i testowanie aplikacji.
- Repository Pattern - to wzorzec projektowy, który ma na celu oddzielenie logiki biznesowej od sposobu dostępu do danych. Repository udostępnia jednolite API dla warstwy biznesowej, niezależnie od źródła danych (np. bazy danych, plików, serwisów sieciowych).
- Value Object - to wzorzec projektowy, który ma na celu reprezentowanie prostych obiektów, których wartość jest najważniejsza, a nie ich identyfikator. Obiekty te są immutable (niemodyfikowalne), co oznacza, że ich stan nie może być zmieniony po utworzeniu.
- Dependency Injection - polega na przekazywaniu obiektów, na których inny obiekt polega, jako argumentów do konstruktora lub metod tego obiektu, zamiast tworzenia ich wewnątrz obiektu. Dzięki temu, zależności są odseparowane od obiektu, co ułatwia testowanie, rozwijanie i zarządzanie nimi. Wzorzec ten pozwala na elastyczne podmienianie zależności w zależności od potrzeb, co zwiększa modularność i łatwość utrzymywania kodu.

- Walidator - wzorzec używany do sprawdzania poprawności danych wejściowych w różnych kontekstach aplikacji. Wzorzec ten zazwyczaj składa się z klas walidujących, które weryfikują dane wejściowe i zwracają informacje o ewentualnych błędach.
- Obserwator - to wzorzec projektowy, który umożliwia obiektom reagowanie na zmiany stanu innego obiektu (zwany podmiotem) poprzez subskrybowanie jego zdarzeń. Podmiot powiadamia swoich obserwatorów o zmianach, a obserwatorzy podejmują odpowiednie działania w zależności od otrzymanych informacji.

4 Instrukcja lokalnego i zdalnego uruchomienia

4.1 WEB

4.1.1 Uruchomienie systemu

- Lokalnie
  - Zainstaluj Node.js
  - Zainstaluj klienta Git
  - Uruchom konsole i wpisz:

    ```
    git clone https://github.com/bimbeer/WEB.git
    ```

  - Następnie przejdź do folderu z projektem, otwórz konsolę i wpisz:

    ```
    npm install
    ```

  - Poczekaj na koniec procesu i wpisz w konsoli:

    ```
    npm start
    ```

  Po tych krokach powinna uruchomić się przeglądarka internetowa z adresem **localhost:3000** i po chwili wyświetlić stronę startową projektu.

- Zdalnie
  - Wystarczy przejść na stronę

    ```
    https://bimbeer.vercel.app/
    ```

    a aplikacja otworzy się w oknie przeglądarki

4.2 Desktop

4.2.1 Uruchomienie testów

- Lokalnie - Lokalne uruchomienie testów wymaga pobrania repozytorium, instalacji zależności i manualne uruchomienie testów.

  - Pobranie i instalacja Node.js ze strony internetowej: https://nodejs.org/
  - Sklonowanie repozytorium: git clone –branch main https://github.com/bimbeer/desktop
  - Przejście do folderu projektu: cd your-project-name
  - Instalacja zależności: npm install
  - Uruchomienie testów: npm test

- Zdalnie - uruchomienie testów wymaga jedynie otwarcia Github Actions, w których są automatycznie przeprowadzane testy CLI.
  - Otwarcie repozytorium: https://github.com/bimbeer/desktop
  - Przejście do zakładki Actions
  - Otwarcie workflow'u testowego Node.js CI

4.2.2 Uruchomienie systemu

- Manualne uruchomienie
  - Sklonowanie repozytorium: git clone –branch main https://github.com/bimbeer/desktop
  - Przejście do folderu projektu: cd your-project-name
  - Instalacja zależności: npm install
  - Uruchomienie projektu: npm start
- Instalacja przy użyciu pliku wykonywalnego
  - Pobranie pliku archiwum: https://github.com/bimbeer/desktop/releases/tag/v1.0.0
  - Rozpakowanie archiwum
  - Przejście do folderu build (release/build)
  - Uruchomienie instalatora "Bimbeer Setup 1.0.0.exe"
  - Uruchomienie aplikacji Bimbeer

5 Wnioski projektowe

Proces tworzenia systemu okazał się dużym wyzwaniem pod względem wymagań funkcjonalnych. Projektowanie systemu obejmującego trzy platformy: webową, desktopową i mobilną wymagało od naszej drużyny starannego planowania i koordynacji działań. Musieliśmy wziąć pod uwagę różnice między platformami i zapewnić spójność doświadczenia użytkownika na wszystkich z nich. Jednym z kluczowych wyzwań było zapoznanie się z nieznanymi nam wcześniej platformami i technologiami. Musieliśmy szybko nauczyć się nowych narzędzi i metod pracy, aby sprostać wymaganiom projektu. Dodatkowo musieliśmy pilnować by przestrzegać zasady czystego

kodu oraz stosować wzorce projektowe i dobre praktyki w celu zapewnienia skalowalności i łatwości utrzymania kodu. Podczas tworzenia projektu mieliśmy okazję wzmocnić kompetencje miękkie, takie jak: umiejętność pracy w zespole, komunikatywność, samodyscyplina, kreatywność, a także cierpliwość i wytrwałość, które były niezwykle kluczowe w najtrudniejszych momentach tworzenia systemu. Komunikacja w zespole była kluczowa dla sukcesu projektu. Głównym narzędziem komunikacji był Discord, który umożliwił nam skuteczne dzielenie się informacjami i koordynowanie działań. Dzięki temu udało nam się zrealizować projekt na czas i zgodnie z oczekiwaniami. Podsumowując, proces tworzenia systemu dla trzech platform był pełen wyzwań, ale dzięki staranemu planowaniu, zastosowaniu dobrych praktyk i skutecznej komunikacji w zespole udało nam się doprowadzić projekt do końca.