Wydział Zamiejscowy w Chorzowie

Systemy urządzeń mobilnych

Projekt

Radosław Tchórzewski

Chorzów 2025

1 Wstęp

Celem projektu było zaprojektowanie aplikacji PWA umożliwiającej śledzenie wydatków i przychodów. W celach przechowywania danych użytkownika należało użyć localstorage. Do persystencji danych o wydatkach w wymaganiach sprecyzowane zostało MongoDB. W celu informowania użytkowników o ważnych wydarzeniach finansowych zaimplementowano powiadomienia push.

2 Wykorzystane technologie

Projekt został zrealizowany korzystając z frameworka NextJS pozwalające na stworzenie backendu oraz frontendu aplikacji w pojedynczym projekcie. Do obsługi funkcjonalności PWA wykorzystano bibliotekę serwist. W celu komunikacji z bazą danych MongoDB zastosowano bibliotekę mongoose.

3 Funkcjonalności aplikacji

W ramach projektu stworzono aplikacje pozwalającą na zarządzanie przychodami oraz wydatkami użytkowników wraz z prezentowaniem wykresów o wydatkach aktualnego miesiąca. Dane przechowywane są w MongoDB oraz localstorage, dzięki czemu aplikacja umożliwia przeglądanie historii wydatków z aktualnego miesiąca w trybie offline. Użytkownik ma możliwość ustawienia miesięcznego budżetu na wydatki. W przypadku zbliżenia bądź przekroczenia ustawionej kwoty wysyłane jest powiadomienie push z informacją jeśli użytkownik włączył tę funkcjonalność. Zaimplementowane zostało 5 ekranów:

- Ekran powitalny zawierający krótki opis aplikacji prezentowany niezależnie od tego czy użytkownik zalogował się do aplikacji
- Ekran logowania pozwalający na zalogowanie w trybie offline
- Ekran rejestracji pozwalający na stworzenie użytkownika
- Ekran statystyk prezentujący zestawienie wydatków aktualnego miesiąca oraz stan budżetu
- Ekran wydatków, który pozwala na przeglądanie, dodawanie oraz usuwanie przychodów i wydatków

Framework NextJS został wybrany w celu zdobycia dodatkowego doświadczenia w nieznanej technologii. Implementacja trybu offline wymusiła odejście od najnowszej konwencji tworzenia komunikacji frontend-backend w projektach korzystających z NextJS. Zamiast akcji serwerowych należało wykorzystać starsze podejście używające komunikacji REST. W celu stylizacji wykorzystano bibliotekę Tailwind 4 oraz HeroUI. Wersja Tailwind wymusiła korzystanie z wersji beta biblioteki HeroUI.

W całej aplikacji zastosowano również komunikaty toast pozwalające na uspójnienie ostrzeżeń oraz komunikatów błędów.

3.1 Nagłówek oraz stopka

Nagłówek aplikacja pozwala na nawigacje pomiędzy ekranami. Ekrany statystyk oraz wydatków dostępne są jedynie po zalogowaniu. Dla wielkości ekranu telefonów nagłówek prezentuje wysuwające się menu natomiast dla wielkości tabletowej oraz komputerów wszystkie informacje zawarte są w nierozwijalnym nagłówku. Stopka zawiera informacje o autorze aplikacji. Subskrypcja powiadomień push dostępna jest w nagłówku za pomocą przełącznika prezentującego aktualny stan.

3.2 Ekran powitalny

Ekran powitalny zawiera krótki opis aplikacji. Nie prezentuje on żadnych danych użytkownika. Zaimplementowana została funkcja sprawdzająca, czy w sesji znajdują się dane użytkownika. Brak danych powoduje przekierowanie na ten ekran.

3.3 Ekran logowania

Ekran logowania pozwala na zalogowanie do aplikacji. Po wpisaniu poprawnych danych backend zwraca id użytkownika oraz wstępne dane zapisywane w localstorage. Użytkownik po zalogowaniu może przeglądać swoje dane w trybie offline. Nie zaimplementowano walidacji pól. Próba zalogowania się bez połączenia z internetem powoduje pojawienie się komunikatu o braku dostępu do sieci.

3.4 Ekran logowania

Ekran rejestracji pozwala na rejestrowanie użytkowników. Podobnie do ekranu logowania nie zaimplementowano walidacji pól. Po utworzeniu użytkownika następuje przekierowanie do ekranu logowania. Próba utworzenia użytkownika w trybie offline powoduje pojawienie się odpowiedniego komunikatu.

3.5 Ekran statystyk

Ekran statystyk pozwala na monitorowanie i zmianę miesięcznego budżetu, odświeżenie danych w localstorage oraz analizę miesięcznych wydatków prezentowanych na wykresie kołowym. W celu przedstawienia zebranych danych wykorzystano bibliotekę ChartJS, z która zaznajomiono się w trakcie ćwiczenia 3 oraz 4. Ekran ten jest niedostępny dla niezalogowanych użytkowników.

3.6 Ekran wydatków i przychodów

Ekran wydatków i przychodów pozwala na wybieranie zakresu wyświetlanych danych. Zalogowany użytkownik może przeglądać, usuwać oraz dodawać transakcje przypisane do jego konta. Przychody oznaczone są zielonym kolorem. W trybie offline dostępność danych zmniejsza się do aktualnego miesiąca, a dla usuwania i dodawania transakcji prezentowany jest toast o braku dostępności funkcjonalności bez dostępu do sieci. W celu uwzględnienia nowych wydatków na ekranie statystyk użytkownik musi kliknąć przycisk odświeżenia danych. Jeśli przy dodawaniu wydatku użytkownik przekroczy bądź zbliży się do ustawionego budżetu, wysyłany jest komunikat push.

4 Funkcjonalności PWA

W celu zaimplementowania działania pwa dostępne były 4 rozwiązania:

- workbox
- biblioteka next-pwa
- biblioteka ducanh2912/next-pwa będąca forkiem next-pwa
- serwist

Po przeprowadzonej analizie większość źródeł rekomendowała korzystanie z serwist co było powodem wybrania tej biblioteki.

4.1 Service worker

Logika service workera została utworzona w pliku sw.ts. Serwist umożliwiło sprawne dodanie wymaganych funkcjonalności. W celu dodania powiadomień push stworzono endpoint do subskrybowania wraz z zapisem danych Subskrypcji w MongoDB oraz endpoint pozwalający na wysyłanie powiadomień push. Celem aplikacji nie było zaimplementowanie autoryzacji i autentykacji, dlatego endpoint wysyłający powiadomienia sprawdza, czy w wysyłanych danych zawarty jest również klucz ustawiany w zmiennych środowiskowych deploymentu. Dla trybu offline dodano również przechwytywane akcji fetch, w której jeśli zapytanie dotyczy pobrania danych listy a użytkownik jest w trybie offline zwracane są dane z localstorage. Wybrano również domyślną strategię cachowania udostępnianą przez serwist.

4.2 Manifest

Plik manifestu oparto o ćwiczenia 3 i 4. Dodano ikony pasujące do tematyki projektu, natomiast plik umieszczono w folderze *static*.

4.3 Tryb offline

Tryb offline pozwala zalogowanemu użytkownikowi na sprawdzenie transakcji dodanych w aktualnym miesiącu. Dostępne jest również przeglądanie pozostałego budżetu oraz wykresu kołowego wydatków danego miesiąca. W celu edycji danych użytkownik musi przejść do trybu online.

5 Wdrożenie aplikacji

W celu wdrożenia aplikacji na hostingu wykorzystano serwis Vercel rekomendowany przez twórców NextJS. Baza danych została uruchomiona korzystając z Atlas Database. Zarówno Vercel jak i Atlas Database posiadały wystarczająco zadowalające darmowe plany. Probles wdrożenia aplikacja przebiegł sprawnie. Jedynym problemem, który napotkano był brak zdefiniowanego adresu ip bazy danych w serwisie Vercel. Skutkiem tego był brak możliwości zalogowania użytkownika oraz korzystania z funkcjonalności aplikacji.

6 Napotkane problemy

6.1 Zapętlenie budowania aplikacji po wdrożeniu PWA

Skonfigurowaniu biblioteki serwist uniemożliwiło korzystanie z aplikacji. Generowane pliki service workera odświeżały budowanie aplikacji przez webpack. Dla uruchomienia aplikacji w trybie dev wymagane było dodanie dodatkowej konfiguracji wyłączającej pwa. Testowanie funkcjonalności offline było czasochłonne, ponieważ wymagało zbudowania aplikacji i uruchomienia jest w trybie release.

6.2 Brak wsparcia Tailwind CSS 4 w bibliotece HeroUI

W celu skorzystania z HeroUI należało dodatkowo skonfigurować zależności projektu oraz dostępne pliki. Style biblioteki nie były automatycznie wykrywane przez projekt. Rozwiązaniem okazało się manualne dodanie zależności w pliku globals.css.

6.3 Stylizacja projektu po dodaniu spinnera

Po uwzględnieniu ekranu ładowania musiała nastąpić reorganizacja kodu projektu. Style globalne autoryzacji oraz zwykłych ekranów przeniesiono poziom niżej, dzięki czemu ekran ładowania został uspójniony dla każdego widoku.

7 Podsumowanie

W ramach projektu dodatkowym celem było zapoznanie się z nowymi technologiami. Wykorzystanie NextJS do aplikacji obsługującej tryb offline nie było optymalnym rozwiązaniem. Udało się jednak stworzyć aplikacje spełniającą wymogi projektu. W ramach pracy pogłębiono wiedzę o IndexedDB, bazach nierelacyjnych oraz hostowaniu aplikacji. Poznano również nieużywane wcześniej technologie takie jak React, Tailwind oraz NextJS. Różniły się one w pewnym stopniu od narzędzi dotychczas stosowanych przez autora takich jak Angular oraz Bootstrap.

7.1 Potencjalny rozwój projektu

Pierwszy potencjalny krok rozwoju aplikacji to integracja biblioteki umożliwiającej autoryzacji oraz dodanie walidacji pól. Dodatkowe usprawnienia uwzględniają poszerzenie dostępnych ekranów statystyk, analizę trybu offline o zakres danych, które warto prezentować użytkownikowi oraz dodanie panelu administratorskiego pozwalającego na tworzenie powiadomień push z przeglądarki.