

Dokumentacja projektu

„Shopping list”

Przedmiot: Programowanie urządzeń mobilnych

Autor: Dawid Kiełbasa

github: https://github.com/breeg554/shoppinglist

O projekcie

Projekt został stworzony jako praca zaliczeniowa na 5 semestrze studiów inżynierskich na kierunku informatyka. Głównym celem zadania było stworzenie aplikacji mobilnej która miała cechować się podstawowymi funkcjami CRUD wykorzystując technologię dostępne na rynku.

Tematem projektu jest "Mobilna lista zakupów" która w założeniu ma ułatwiać robienie zakupów oraz odciążać pamięć dzięki prostemu organizowaniu tak częstej czynności, jaką są zakupy.

Technologie

Baza danych została zaprojektowana w oparciu o platformę programistyczną Firebase. Wykorzystany został moduł o nazwie "Realtime Database" który pozwala aktualizować zmiany w czasie rzeczywistym. Interfejs graficzny w formie aplikacji mobilnej powstał przy użyciu programu Visual Studio Code i biblioteki React Native.

Baza danych

Baza danych, jako baza NoSQL, nie posiada tabel ani relacji. Dane przechowywane są w formacie JSON. Jako ze jest to baza z dostępem rzeczywistym, mamy pewność, że użytkownik widzi tylko aktualne dane. Warto dodać, że w przypadku gdy użytkownik nie ma dostępu do Internetu, dane zapisywane są lokalnie, a po przywróceniu połączenia automatycznie synchronizowane z bazą danych.



Rysunek 1. Struktura bazy danych

Baza danych składa się z kolekcji i właściwości. Głowna kolekcją jest kolekcja "users". Każdy użytkownik posiada kolekcję "lists" w której znajdują się takie właściwości jak:

-archived - która określa czy lista zakupów jest zarchiwizowana

-color - określajacy kolor poszczególnych elementów listy

-name - określający nazwe listy

-todos - tablica przechowująca produkty do kupienia. Każdy produkt ma właściwość "completed" określającą czy został już zakupiony oraz właściwość "name".

W celu uproszczenia korzystania z aplikacji uwierzytelnianie w bazie danych działa w sposób anonimowy. Użytkownik podczas pierwszego uruchomienia automatycznie tworzy konto tymczasowe które pozwala mu na swobodny dostęp do danych z danego urządzenia.

Struktura aplikacji

|  |  |
| --- | --- |
| Struktura projektu jest typowa dla aplikacji napisanych w React Native. Oprócz plików konfiguracyjnych takich jak package.json, app.json czy folderu node\_modules gdzie znajdują się moduły potrzebne do uruchomienia aplikacji mamy również folder „app” gdzie są umieszczone wszystkie komponenty odpowiedzialne za wygląd i działanie aplikacji.  Plik „Colors.js” zawiera zdefiniowane kolory używane do stylowania aplikacji, a plik „isCorrectExpression.js” funkcje sprawdzające poprawność wprowadzanych danych.  Plik „fire.js” odpowiedzialny jest za łączenie się z baza danych. Zawiera również wszystkie metody służące do podstawowych operacji CRUD.  Screen z kodu w sekcji „Przykład kodu”. | Rysunek 2. Struktura projektu |

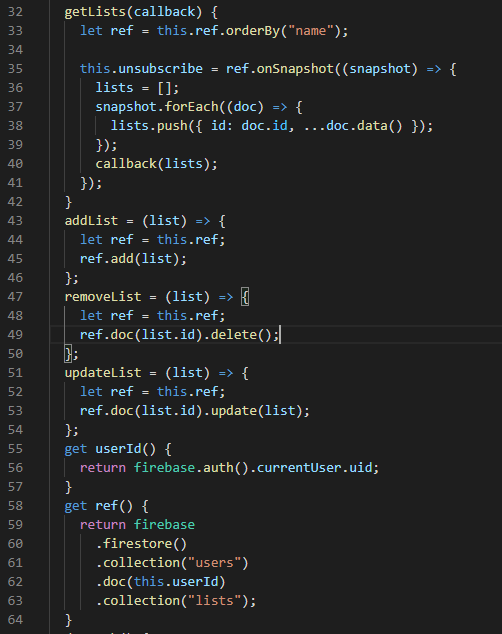
Przykład kodu



Rysunek 3. Plik konfiguracyjny połączenie do bazy danych

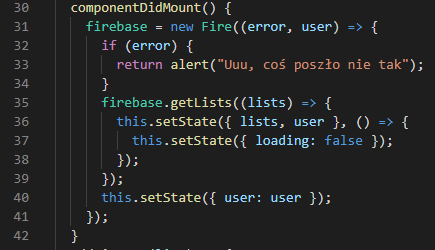
Powyższy fragment kodu przedstawia klasę „Fire” która odpowiedzialna jest za łączenie z bazą danych. Na samej górze pliku mamy zdefiniowaną stałą „config” która zawiera niezbędne informacje potrzebne do połączenia się z API Firebase takie jak np. „apiKey” .

W konstruktorze klasy wywoływana jest metoda „init()” która w wypadku gdy połączenie z baza nie jest zainicjalizowane, inicjalizuje je, a następnie używając obserwatora „onAuthStateChanged()” na obiekcie „auth” sprawdza czy obiekt „auth” nie jest w stanie pośrednim.



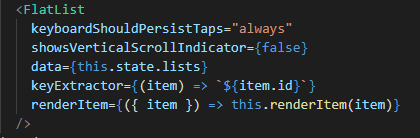
Rysunek 4. Metody klasy Fire

Powyżej pokazane są metody klasy Fire odpowiedzialne za podstawowe funkcjonalności CRUD.



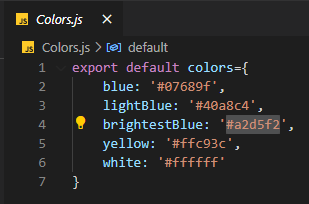
Rysunek 5. Utworzenie instancji klasy Fire

W głównym komponencie „LandingPage” w metodzie ‘componentDidMount” tworzona jest instancja klasy Fire , pobierane są wszystkie listy zalogowanego użytkownika i zapisywane do state’u. W wypadku niepowodzenia zwracany jest odpowiedni alert.



Rysunek 6. Wyświetlanie list trzymanych w state’cie

Zaciągnięte „listy”, które przechowywane są w state’cie naszej aplikacji, w metodzie render komponentu „LandingPage” przekazywane są do komponentu „FlatList”, który jest komponentem bezpośrednio importowanym z „react-native” , i renderowane na ekranie. Metoda „render” aktualizuje nam widok aplikacji za każdym razem gdy zmieni się „state”, dlatego gdy dodamy albo usuniemy listę, widok zostanie zaktualizowany.

Rysunek 7. Pliku colors Rysunek 8. Używanie zdefiniowanych kolorów

W pliku „Colors” zdefiniowane są wszystkie kolory użyte w aplikacji. Taki sposób definiowania zmiennych/stałych ułatwia i skraca nam prace gdyż w wypadku gdy zechcemy zmienić odcień np. koloru żółtego, nie musimy robić tego w całej aplikacji. Wystarczy że zmienimy go w tym miejscu.

Prezentacja aplikacji

|  |  |
| --- | --- |
| Rysunek 9.Widok startowy | Rysunek 10.Modal do tworzenia listy |

Po uruchomieni aplikacji naszym oczom ukaże się widok startowy aplikacji (rys. 9)

Na samym środku widnieje duży przycisk z napisem „Utwórz pierwsza listę!”. Zarówno po kliknięciu na ten przycisk jak i na żółty z plusem w prawym dolnym rogu ukaże nam się modal w którym możemy dodać naszą pierwszą listę. (rys. 10). W polu tekstowym wprowadzamy nazwę dla naszej listy (pole nie może być puste rys.11) oraz wybieramy jeden z dostępnych kolorów.

|  |  |
| --- | --- |
| Rysunek 11.Walidacja pola tekstowego i wybór koloru | Rysunek 12.Utworzona lista zakupów |

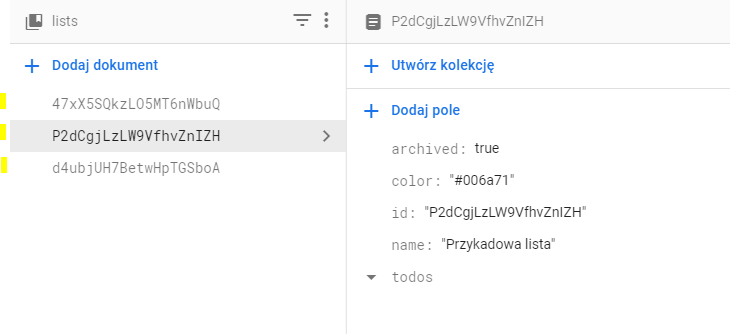
Po kliknięciu przycisku „dodaj”, modal zostaje zamknięty, a nasza lista dodaje się do widoku startowego (rys. 12).

Po stworzeniu listy mamy dwie możliwości: możemy ja zarchiwizować w przypadku gdy nie jest nam potrzebna lub klikając w nią przejść do widoku listy.

Zacznijmy od czegoś szybszego więc od archiwizacji. Po kliknięciu w przycisk „archiwizuj” nasza lista znika z widoku startowego i dodaje się do list zarchiwizowanych. Żeby je przejrzeć klikamy w ikonę „kosza” w prawym górnym rogu. Po kliknięciu wysuwa nam się modal z listami zarchiwizowanymi (rys. 13).

|  |  |
| --- | --- |
| Rysunek 13.Spis list zarchiwizowanych | Rysunek 14. Usuwanie list |

Nasza lista dodała się do list zarchiwizowanych. Możemy ją przywrócić klikając w przycisk „Przywróć” lub permanentnie usunąć klikając w „Usuń”. Usuńmy pozostałe listy zostawiając tylko 3 pierwsze.



Rysunek 15. Baza danych po usunięciu list

Jak widzimy na rys. 14 i 15 nasze listy usunęły się zarówno po stronie klienta jak i bazy danych.

Przywróćmy teraz listę którą stworzyliśmy, wróćmy na ekran startowy i kliknijmy w nią.

|  |  |
| --- | --- |
| Rysunek 16. Widok pojedynczej listy | Rysunek 17. Dodanie produktu do listy |

Na rys.16 pokazany jest widok pojedynczej listy. Możemy tutaj zarządzać produktami do kupienia. Uzupełniając pole „Nazwa produktu” i klikając ikonę „plusa” dodajemy produkt do naszej listy (rys. 17).

Jak widać po dodaniu produktu licznik w prawym dolnym rogu zmienił się. Mamy teraz jeden produkt do kupienia i jak na razie 0 kupionych.

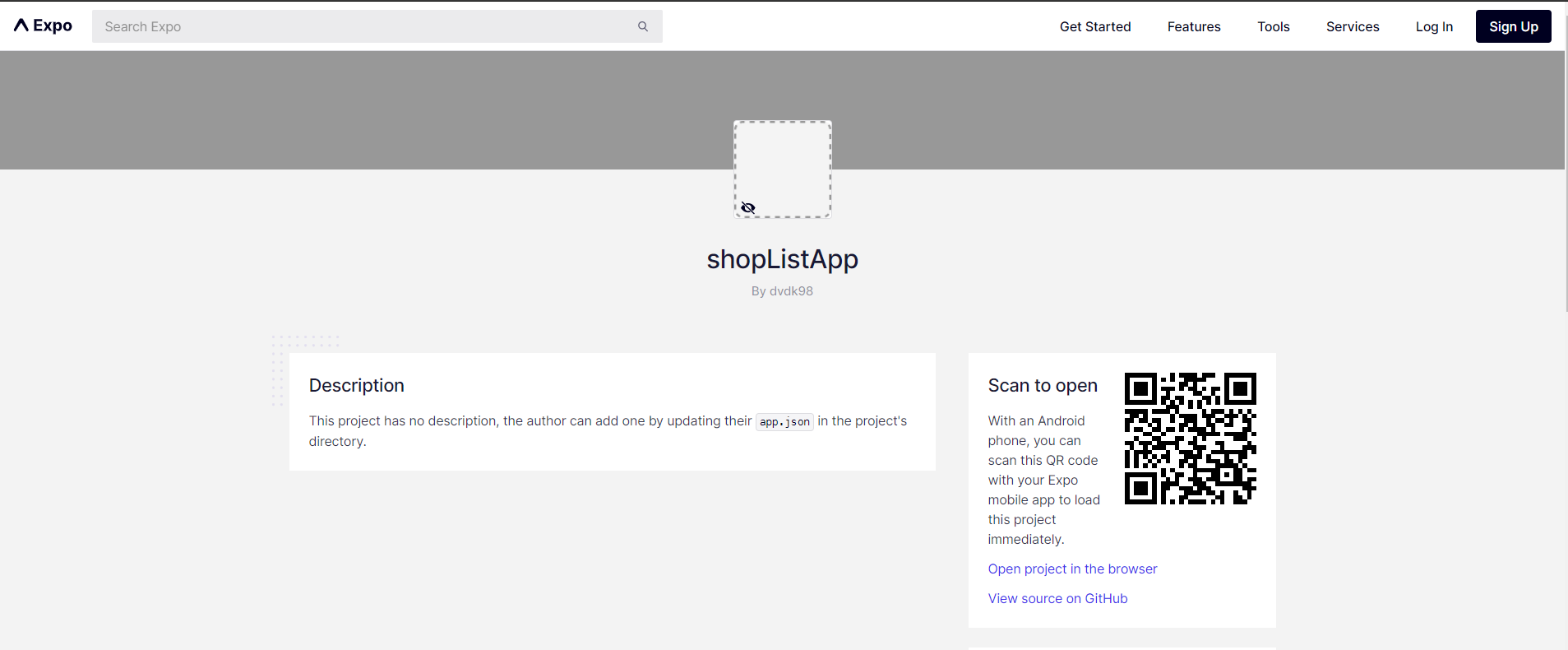
|  |  |
| --- | --- |
| Rysunek 18. Dodanie produktu do listy | Rysunek 19. Licznych produktów |

Po dodaniu kilku produktów (rys.18) mamy możliwość oznaczenia ich jako produkt kupiony ( włożony do koszyka etc.). Jak widać na screen’ie ikona przy nazwie jak i kolor nazwy produktu zmienia się w zależności od stanu.

Jak widać na rys. 18 i 19, licznik naszych produktów działa w sposób automatyczny i aktualizuje się w zależności od stanu naszego koszyka.

Po kliknięciu w przycisk „Delete” (rys. 18) przy jakimkolwiek produkcie, zostanie od usunięty na stałe z listy.

***Jeśli chcemy wytestować działanie aplikacji u siebie na smartfonie***  wystarczy mieć pobraną aplikację „Expo” którą znajdziemy w sklepie Play, wejść w ten [LINK](https://expo.io/@dvdk98/projects/shopListApp) i zeskanować skanerem z naszej aplikacji (rys.21) kod QR (rys. 20).



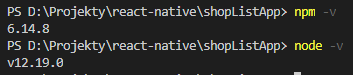
Rysunek 20. Kod QR aplikacji

|  |  |
| --- | --- |
| Rysunek 21. Aplikacja expo na telefonie | Rysunek 22. Działająca aplikacja na naszym telefonie |

Uruchamianie projektu

Jeżeli chcemy pobrać aplikacje do siebie i uruchomić ją lokalnie musimy spełnić kilka wymagań.

Po pierwsze, mieć zainstalowany u siebie menadżer pakietów „npm” oraz „Node.js”. Możemy to sprawdzić wpisując w dowolnym terminalu komendy „ npm -v „ i „node -v”. Jeżeli są one już na naszym komputerze powinniśmy widzieć podobny widok (rys. 23)



Rysunek 23. Wersja npm i noda

Jeśli nie posiadamy ich jeszcze zainstalowanych wystarczy ze wejdziemy na główną stronę [node.js](https://nodejs.org/en/download/), pobierzemy wersje odpowiedna dla naszego systemu i zainstalujemy go. Wraz z node’em, menadżer pakietów „npm” zainstaluje się automatycznie.

Po drugie, musimy zainstalować „expo CLI”. Wpisujemy w terminalu komendę

„npm install expo-cli –global”. Po krótkiej chwili „expo” zostanie zainstalowane.

Po trzecie mieć zainstalowany jakiś emulator urządzeń wirtualnych np. Android Studio

Po konfiguracji naszego środowiska możemy pobrać repozytorium z projektem.

W terminalu wpisujemy komendę „git clone <https://github.com/breeg554/shoppinglist.git>”.

Po krótkiej chwili repozytorium zostanie pobrane na nasz komputer. Wchodzimy do folderu z projektem i wpisujemy komendę „ npm install „ która pobierze wszystkie niezbędne pakiety potrzebne do uruchomienia projektu.

Po zainstalowaniu niezbędnych modułów możemy odpalić nasza aplikację. W konsoli wpisujemy komendę „ npm start „ i po paru sekundach nasza aplikacja wystartuje.



Rysunek 24. Odpalenie aplikacji

Jeśli wszystko poszło dobrze , powinniśmy widzieć taki widok. Teraz Jeśli chcemy odpalić aplikację na maszynie wirtualnej naszego komputera wystarczy w konsoli wcisnąć przycisk „a” uprzednio uruchamiając maszynę wirtualną.

Aplikację możemy również uruchomić skanując kod QR naszym telefonem używając aplikacji Expo.

Od teraz możemy dowolnie modyfikować projekt. Po każdym zapisie aplikacja zostanie przeładowana.