Programowanie mobilne

Hybrydowe i cross-platformowe aplikacje mobilne

dr inż. Andrzej Wilczyński - www.awilczynski.me

Aplikacje hybrydowe (web view)

Aplikacje łączące charakterystyczne cechy aplikacji natywnych i aplikacji webowych. Budowane z wykorzystaniem technologii webowych lecz dzięki zastosowaniu specyficznej dla danego oprogramowania otoczce, wyglądem przypominają aplikacje natywne. Przykładowe narzędzia do transferu kodu pisanego pod „przeglądarki” na kod rozumiany przez urządzenie mobilne to Apache Cordova lub Titanium.

Zalety aplikacji hybrydowych

- Funkcjonalność aplikacji pochodzi głównie z serwera aplikacji sieci Web
- Aplikacja działa na wielu platformach
- Koszty wytworzenia są relatywnie niższe niż w przypadku aplikacji natywnych, ponieważ największa część kodu programu jest niezależna od platformy
- Niskie koszty utrzymania
- Dostęp do natywnego API urządzenia

Wady aplikacji hybrydowych

- Zwykle działają wolniej niż aplikacje natywne
- Jedna baza kodu – aplikacja działa tak samo na wszystkich urządzenia, co może powodować problemy z używaniem niektóry funkcji dostępnych tylko na jednej z platform
- Potrzebny web deweloper do backendu, natywny deweloper do reszty kodu

Apache Cordova

Platforma służąca do budowania aplikacji mobilnych przy użyciu HTML, CSS i JavaScript. Składa się z interfejsów do programowania aplikacji pod dane urządzenie (API), pozwalających deweloperom na tworzenie dedykowanych urządzeniom aplikacji bez użycia przez nich kodu natywnego. Obsługuje najpopularniejsze systemy mobilne (najlepsze wsparcie jest dla Androida oraz iOS). Pod platformę tworzone są liczne frameworki takie jak np. Ionic.

Apache Cordova – najważniejsze zalety

- Nieduży koszt wejścia
- Instalacja aplikacji w identyczny sposób jak aplikacji natywnej
- Open-source
- Stale rosnącą popularność technologii HTML5, CSS czy JS i standaryzacja przeglądarek

Apache Cordova – wady i ograniczenia

- Brak natywnego wsparcia dla wszystkich wersji danego SDK
- Na starszych telefon mogą może sprawiać problemy wydajnościowe
- Momentami „spaghetti code”
- Problemy z niektórymi tagami HTML5 czy Google Maps

Ionic

Framework do tworzenia aplikacji, która ma służyć zarówno jako aplikacja internetowa jak i aplikacja Android czy iOS. Używane są tutaj te same narzędzie, których używam do budowy klasycznej aplikacji jednostronicowej (SPA). Komponenty do tworzenia interfejsu użytkownika dostarczane przez Ionic są zaprojektowane przede wszystkim z myślą o urządzeniach mobilnych.

Ionic – najważniejsze zalety

- Pakiet SDK typu open source, zapewniający również kilka szablonów początkowych
- Debugowanie w Ionic jest proste
- Stylizacja interfejsu użytkownika
- Wygląd i styl dostosowany do platformy

Aplikacje cross-platformowe (kompilowane do natywnych)

Aplikacje cross-platformowe wykorzystują natywne elementy dostarczane przez producentów np. Androida czy iOS'a. Wszystkie platformy mają tę samą bazę kodu, ale posiadają różnice w UI. Elementy natywne są wyświetlane za pomocą HTML/CSS oraz wykorzystaniu na przykład Angulara jako jądra aplikacji.
Najpopularniejsze frameworki do tworzenia aplikacji cross-platformowych to React Native lub Flutter.

Zalety aplikacji cross-platformowych

- Korzystne czasowo i kosztowo
- Łatwiej trafić do większej ilości potencjalnych odbiorców
- Dodawanie nowych funkcji nie wymaga dużo pracy
- Wydajność UI jest porównywalna do rozwiązań natywnych

Wady aplikacji cross-platformowych

- Tworzenie może być bardziej skomplikowane i wymaga większych kompetencji od deweloperów
- Zależność od frameworka jeśli chodzi o system operacyjny czy funkcjonalności UI
- Niektóre części kodu muszą być pisane osobno

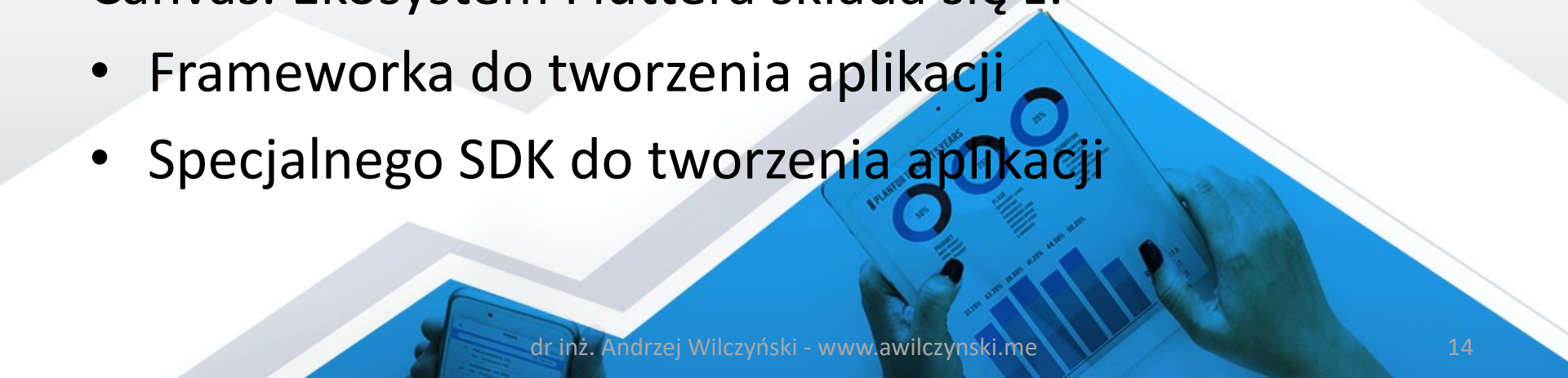
Flutter (1)

Framework zbudowany w 2017 roku przez organizację Google umożliwiający tworzenie wydajnych i szybkich aplikacji działających na kilku platformach, przede wszystkim na systemach Android i iOS. Wykorzystuje język Dart do tworzenia aplikacji, które wyglądają i zachowują się niemal identycznie na wszystkich platformach z wydajnością zbliżona do aplikacji natywnych pisanych bezpośrednio pod daną platformę.

Flutter (2)

Framework ten podobnie jak React Native używa innego podejścia niż frameworki do tworzenia aplikacji hybrydowych, ponieważ nie wykorzystuje WebView. Zamiast mapowania elementów na natywne odpowiedniki sam bierze odpowiedzialność za tworzenie całego UI, wykorzystując przy tym natywny Canvas. Ekosystem Fluttera składa się z:

- Frameworka do tworzenia aplikacji
- Specjalnego SDK do tworzenia aplikacji



Flutter - zalety

- Czas tworzenia
- Łatwość obsługi
- Niskie koszty
- Możliwość szybkiej weryfikacji zmian wprowadzonych w kodzie
- Niski próg wejścia
- Szeroki wybór widgetów

React Native

Framework umożliwiający tworzenie aplikacji mobilnych na różne platformy systemowe, najczęściej na Androida oraz iOS'a. Tworząc aplikację w React Native deweloper przygotowuje kod, który następnie jest tłumaczony na natywny kod obsługiwany przed odpowiednią platformę. Kod natywny jest z kolei kompilowany do właściwej aplikacji. Aplikacji napisanych w React Native praktycznie nie da się odróżnić od aplikacji natywnych.

React Native – początki istnienia technologii

React Native powstał w związku z rozpoczęciem w 2021 poszukiwania przez założyciela Facebooka Marka Zuckerberga rozwiązania, które usprawni i skróci czas budowy aplikacji. Mark Zuckerberg zauważył, wtedy, że coraz więcej użytkowników zaczęło korzystać z aplikacji mobilnej Facebooka. Od tego momentu organizacja zaczęła wykorzystywać technologie natywne w swoich rozwiązaniach.

React Native – najważniejsze zalety

- Szybszy proces dewelopmentu
- Współdzielony kod
- Lepszy niż aplikacje hybrydowe

React Native – najważniejsze wady i ograniczenia

- Potrzeba użycia natywnego kodu
- Brak obsługi natywnych API i SDK
- Mniejsza wydajność
- Dłuższe debugowanie

React Native – kiedy stosować?

- Należy rozważyć przy starcie nowego projektu
- Przy prostych aplikacjach
- Przy budowaniu MVP
- Przy niskim budżecie i ograniczonej ilości deweloperów

React Native a aplikacja hybrydowa

Z technicznego punktu widzenia aplikacja napisana w React Native nie jest ”aplikacją hybrydową”, która z definicji jest hybrydą aplikacji webowej oraz aplikacji natywnej. Za pomocą React Native otrzymujemy aplikację natywną, natomiast technologia ta bazuje na koncepcjach wytworzonych wcześniej w technologiach hybrydowych i dlatego jest jeszcze przez niektórych czasami przypisywana do tej właśnie grupy.

React Native i Flutter - różnice

- Język programowania
- Interfejs użytkownika
- Dokumentacja

React Native a React

React Native i React to dwa zupełnie inne pojęcia. React zwany również inaczej ReactJS jest biblioteką JavaScript, służącą do tworzenia interfejsów użytkowników. Tak samo jak React Native został zbudowany przez twórców Facebooka. ReactJS przede wszystkim pozwala na tworzenie izolowanych komponentów i deklaratywnych widoków. React Native z kolei nie jest biblioteką a frameworkiem umożliwiającym tworzenie wieloplatformowych aplikacji mobilnych.

React Native – budowa projektu

Projekt można zbudować na dwa sposoby:

- Poprzez narzędzie Expo
- Poprzez budowę projektu, który będzie zawierał osobne pliki dla Androida'a jak i iOS'a.

React Native – pierwszy projekt

1. Instalacja Node.js i npm.
2. Instalacja *React Native Command Line Interface.*

```
npm install -g react-native-cli
```

3. Stworzenie projektu za pomocą polecenia:

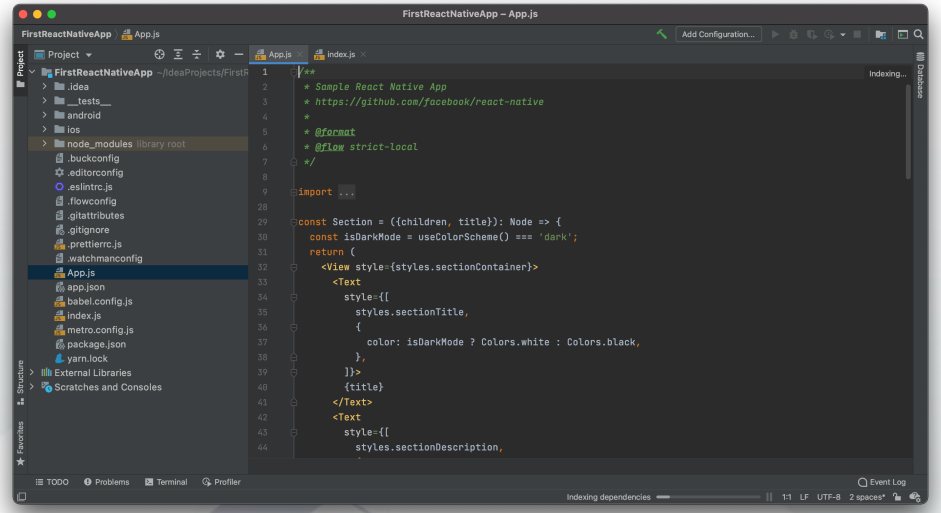
```
react-native init FirstReactNativeApp
```

Dokładna instrukacja znajduje się n stronie:

https://reactnative.dev/docs/environment-setup

React Native – struktura projektu

- `android` – kod specyficzny dla platfromy Android
- `ios` – kod i zależności specyficzne dla platformy iOS
- `node_modules` – folder z zależnościami
- `.flowconfig` – konfiguracja Flow
- `.watchmanconfig` – konfiguracja dla Watchmana
- `index.js` – punkt wejścia aplikacji
- `App.js` – domyślny import główny używany w pliku index.js
- `package.json` – konfiguracja npm



React Native – uruchomienie aplikacji

1. Uruchomienie serwera React Native za pomocą polecenia.

```
react-native start
```

2. W osobnej konsoli budujemy i uruchamiamy aplikację:

```
react-native run-android
react-native run-ios
```

Dziękuję! ☺

Bibliografia

1. Nader Dabit, *OS 12. React Native w akcji. Twórz aplikacje na iOS i Android w JavaScripcie. Wydawnictwo Naukowe PWN SA*, 2020
2. *Flutter – dlaczego warto zainteresować się nowym narzędziem od Google?*, 4.06.2021, https://it-solve.pl/google-flutter-co-to-jest-dlaczego-warto-go-uzywac/
3. *5 kroków do napisania pierwszej aplikacji we Flutterze*, 4.06.2021, https://devstyle.pl/2019/10/14/5-krokow-do-napisania-pierwszej-aplikacji-we-flutterze/
4. *Tworzenie aplikacji w React Native*, 4.06.2021, https://impicode.pl/react-native/
5. *Apache Cordova*, 4.06.2021, https://datamobile.wordpress.com/2015/06/06/apache-cordova/
6. *Case Study: Który framework do aplikacji mobilnych jest najlepszy – Ionic, Flutter czy React Native?*, 4.06.2021, https://geek.justjoin.it/case-study-ktory-framework-do-aplikacji-mobilnych-jest-najlepszy-ionic-flutter-czy-react-native
7. *Aplikacje cross-platformowe vs natywne vs hybrydowe vs PWA*, 4.06.2021, https://itcraftapps.com/pl/blog/aplikacje-cross-platformowe-vs-natywne-vs-hybrydowe-vs-pwa/
8. *Aplikacja mobilna - czy da się dobrze i tanio?*, 4.06.2021, https://jcd.pl/aplikacja-mobilna-czy-da-sie-dobrze-i-tanio
9. *React Native pierwsza aplikacja -> StormSnapshot*, 4.06.2021, https://stormit.pl/react-native-pierwsza-aplikacja/