1 Pierwsze kroki z React Native

Tematyka rozdziału:

- wprowadzenie do React Native,
- mocne strony React Native,
- tworzenie komponentów,
- tworzenie pierwszego projektu.

Tworzenie natywnych aplikacji mobilnych może być złożone. Przy skomplikowanych środowiskach, rozbudowanych frameworkach i długich czasach kompilacji, z którymi stykają się programiści, opracowanie wysokiej jakości natywnej aplikacji mobilnej nie jest łatwym zadaniem. Nic dziwnego, że na rynku pojawiają się pewne rozwiązania, które próbują rozwiązać problemy związane z tworzeniem natywnych aplikacji mobilnych.

U podstaw tej złożoności leżą trudności związane z programowaniem wieloplatformowym. Poszczególne platformy są zasadniczo odmienne i nie współdzielą znacznej części swoich środowisk programistycznych, interfejsów API lub kodu. Z tego powodu musimy mieć osobne zespoły pracujące na każdej platformie, co jest zarówno drogie, jak i nieefektywne.

Ale obecnie mamy do czynienia z ekscytującym momentem w rozwoju aplikacji mobilnych. Jesteśmy świadkami nowego paradygmatu w zakresie tworzenia aplikacji mobilnych, a React Native zajmuje czołowe miejsce w tej zmianie sposobu ich

tworzenia i inżynierii. Jesteśmy teraz w stanie tworzyć natywne aplikacje międzyplatformowe, a także aplikacje webowe w jednym języku i w jednym zespole. Wraz ze wzrostem liczby urządzeń mobilnych i związanym z tym wzrostem zapotrzebowania na talenty, co napędza wzrost pensji programistów, React Native oferuje możliwość dostarczania wysokiej jakości aplikacji na wszystkie platformy w ułamku wcześniejszych czasów i kosztów, przy jednoczesnym zapewnieniu wysokiej jakości produktów i wyśmienitych wrażeń programistycznych.

1.1. Wprowadzenie do React i React Native

React Native to platforma do tworzenia natywnych aplikacji mobilnych przy użyciu JavaScriptu i biblioteki React; kod React Native jest kompilowany do rzeczywistych, natywnych komponentów. React to open source'owa biblioteka JavaScriptu udostępniona i używana przez Facebook. Pierwotnie była wykorzystywana do tworzenia interfejsów użytkownika dla aplikacji webowych. Od tego czasu ewoluowała i teraz może być wykorzystywana do tworzenia aplikacji serwerowych i mobilnych (przy użyciu React Native).

Z React Native jest związanych wiele rzeczy. Oprócz tego, że został stworzony i udostępniony przez Facebook, to ma także za sobą liczną społeczność zmotywowanych ludzi. React Native obsługuje Facebook Groups, liczące miliony użytkowników, a także Facebook Ads Manager. Airbnb, Bloomberg, Tesla, Instagram, Ticketmaster, SoundCloud, Uber, Walmart, Amazon i Microsoft to tylko część firm, które inwestują lub wykorzystują React Native w produkcji.

Dzięki React Native programiści mogą za pomocą JavaScriptu budować natywne widoki i uzyskiwać dostęp do komponentów specyficznych dla danej platformy. To odróżnia React Native od innych hybrydowych frameworków, takich jak Cordova i Ionic, które opakowują webowe widoki zbudowane przy użyciu HTML i CSS w natywną aplikację. Natomiast React Native korzysta z JavaScriptu i kompiluje go w prawdziwą natywną aplikację, która może korzystać z interfejsów API i komponentów specyficznych dla danej platformy. Alternatywy takie jak Xamarin stosują to samo podejście, natomiast aplikacje Xamarin są budowane przy użyciu C#, a nie JavaScriptu. Wielu programistów zna JavaScript, co ułatwia im przejście z tworzenia aplikacji webowych na aplikacje mobilne.

Wybór React Native jako frameworka dla aplikacji mobilnych ma wiele zalet. Ponieważ aplikacja bezpośrednio renderuje natywne komponenty oraz korzysta z natywnych API, szybkość i wydajność są znacznie lepsze niż w przypadku frameworków hybrydowych, takich jak Cordova czy Ionic. Dzięki React Native możemy napisać całą aplikację przy użyciu jednego języka programowania: JavaScript. Możemy ponownie wykorzystywać kod, zmniejszając w ten sposób czas potrzebny na dostarczenie aplikacji dla wielu platform. A znajdowanie i zatrudnianie wysokiej jakości programistów JavaScript jest znacznie łatwiejsze i tańsze niż zatrudnianie programistów Java, Objective C lub Swift, co prowadzi łącznie do mniej kosztownego procesu.

UWAGA Aplikacje React Native są tworzone przy użyciu JavaScriptu i JSX. JSX omówimy dokładnie w tej książce, natomiast na razie należy traktować to jako rozszerzenie składni języka JavaScript, które wygląda jak HTML lub XML.

Głębiej w React wejdziemy w rozdziale 2. Na razie w ramach wprowadzenia omówimy kilka podstawowych koncepcji.

1.1.1. Prosta klasa React

Komponenty są elementami składowymi aplikacji React lub React Native. Punktem wejścia aplikacji jest komponent, który wymaga innych komponentów i z nich się składa. Te komponenty mogą również wymagać kolejnych komponentów i tak dalej.

Istnieją dwa główne typy komponentów React Native: *stanowe* (*stateful*) i bezstanowe (*stateless*). Oto przykład stanowego komponentu używającego klasy ES6:

```
class HelloWorld extends React.Component {
  constructor() {
    super()
    this.state = { name: 'Chris' }
  }

  render () {
    return (
      <SomeComponent />
    )
  }
}
```

A teraz przykład komponentu bezstanowego:

```
const HelloWorld = () => (
  <SomeComponent />
```

Główną różnicą jest to, że komponenty bezstanowe nie mogą być podłączone do żadnych metod związanych z cyklem życia i nie mają własnego stanu, zatem wszelkie przetwarzane dane muszą być pobierane jako właściwości (*props*). Metody związane z cyklem życia dokładniej omówimy w rozdziale 2, ale na razie rzućmy okiem na nie oraz na poniższą klasę.

Listing 1.1 Tworzenie prostej klasy React Native

```
import React from 'react'
import { View, Text, StyleSheet } from 'react-native'

class HelloWorld extends React.Component {
  constructor () {
    super()
    this.state = {
      name: 'React Native in Action'
    }
  }
  componentDidMount () {
    console.log('mounted..')
  }
  render () {
    return (
      <View style={styles.container}>
```

Konstruktor tworzy obiekt stanowy z właściwością name

Końcowa metoda cyklu życia

Wywołanie render()

```
          <Text>{this.state.name}</Text>
        </View>
      )
    }

const styles = StyleSheet.create({
  container: {
    marginTop: 100,
    flex: 1
  }
})
```

UWAGA Podczas omawiania powyższych metod należy pamiętać o koncepcji *montowania* (*mounting*). Kiedy tworzony jest komponent, powstaje instancja cyklu życia komponentu React, która wywołuje metody zastosowane na listingu 1.1.

Na początku pliku potrzebujemy deklaracji importu `React` z `'react'` oraz `View`, `Text` i `StyleSheet` z `'react-native'`. `View` jest najbardziej podstawowym elementem składowym do tworzenia komponentów React Native oraz w ogóle interfejsu użytkownika, dlatego można go traktować jak znacznik `div` w HTML. `Text` pozwala tworzyć elementy tekstowe i jest porównywalny ze znacznikiem `span` w HTML. `StyleSheet` pozwala tworzyć obiekty stylów do użycia w aplikacji. Te dwa pakiety (`react` i `react-native`) są dostępne jako moduły npm.

Gdy komponent jest ładowany po raz pierwszy, ustawiamy w konstruktorze obiekt `state` z właściwością `name`. Aby dane w aplikacji React Native były dynamiczne, należy je ustawić jako stanowe lub przekazać jako właściwości. W tym przypadku ustawiliśmy stan w konstruktorze, więc w razie potrzeby możemy go zmienić przez wywołanie

```
this.setState({
  name: 'Some Other Name'
})
```

które ponownie wyrenderuje komponent. Ustawienie zmiennej jako stanowej pozwala zaktualizować wartość w innym miejscu komponentu.

Następnie jest wywoływany `render`: sprawdza właściwości i stan, a następnie musi zwrócić pojedynczy element React Native, `null` lub `false`. Jeśli będziemy mieli wiele elementów potomnych, to będą one musiały być opakowane w element nadrzędny. Tutaj komponenty, style oraz dane zostały połączone, aby utworzyć to, co zostanie wyrenderowane w interfejsie użytkownika.

Ostatnią metodą w cyklu życia jest `componentDidMount`. Jeśli byłoby potrzebne jakiekolwiek wywołanie API lub żądanie AJAX w celu zresetowania stanu, jest to zwykle najlepsze miejsce, aby to zrobić. Na koniec na urządzeniu jest renderowany interfejs użytkownika i dzięki temu można zobaczyć wynik.

1.1.2. Cykl życia React

Po utworzeniu klasy komponentu React Native tworzone są instancje metod, do których można się podłączyć. Te metody są nazywane *metodami cyklu życia* i omówimy je szczegółowo w rozdziale 2. Metody z listingu 1.1 to `constructor`, `componentDidMount`