Github: https://bit.ly/36vvTaJ

Ćwiczenie 1 – konfiguracja

- Wejdź na https://snack.expo.io
- Zarejestruj się
- Ściągnij aplikację „Expo” na telefon (http://onelink.to/u65hxa)
- W przeglądarce naciśnij run i zeskanuj telefonem kod QR, który się pojawi
- Spróbuj coś zmienić w kodzie i zobacz efekty

Ćwiczenie 2 – używanie komponentów

- W pliku App.js zaimportuj **Button** i **Alert** z pakietu **react-native**
- Dodaj komponent <**Button** /> w metodzie **render**( ) wewnątrz <**View**>
- Dodaj mu właściwość **onPress={() ⇒ {}}**
- Wewnątrz **onPress** wywołaj **Alert**.
- https://facebook.github.io/react-native/docs/alert.html
- https://facebook.github.io/react-native/docs/button.html

Ćwiczenie 3 – szkielet aplikacji

- Commit: **Exercise 3**
- Stwórz katalog **services** a w nim pliki **package.js** i **paczkomaty.js**
- Plik **paczkomaty.js** zostaw pusty
- W **package.js**:
  - stwórz klasę **Packages** z asynchroniczną metodą **fetch**, która zwróci:

```
await Promise.all(staticPackages);
```

  - Nad klasą dodaj stałą, np.:

```
const staticPackages = [
  '68031363694170001599870 7',
  '64170084694031801959513 8',
  '63202071794170011846767 4',
];
```

- W pliku **App.js**:
  - Zaimportuj klasę **Packages** z pliku **services/package.js**
  - W klasie **App**:
    - Dodaj stan: **{ data: [], loading: true }**
    - Stwórz metodę asynchroniczną **fetchData()** i wypełnij jej ciało.
    - W metodzie **componentDidLoad** wywołaj **fetchData**.
    - W metodzie **render** dodaj 2 komponenty **Text** które wyświetlą stan komponentu, np.:

```
<Text>{(this.state.data || []).length}</Text>
<Text>{(this.state.loading) >? ‘Loading…’ : ‘Loaded.’}</Text>
```

Ćwiczenie 4 – FlatList

- Commit: **Exercise 4**
- W pliku **App.js** w metodzie Render zaimplementuj **FlatList** oraz **ActivityIndicator** gdy dane się jeszcze wczytują.
- Możesz posiłkować się kodem na GitHubie, dokumentacją React Native i React Native Paper.
- ***this.state.data*** *jest tablicą stringów, więc w komórce wyświetl tylko numer śledzenia przesyłki.*

Ćwiczenie 5 – Używanie React Navigation

- Commit: **Exercise 5**
- Przerób aplikacje tak, aby korzystała z React Navigation.
  - Stwórz katalog **components**, a w nim pliki **ListView.js** i **DetailsView.js**
  - Przenieś kod z **App.js** do **ListView.js**
  - W Pliku **DetailsView.js** stwórz komponent **DetailsView** wyświetlający tekst, np. “to do”
  - W pliku **App.js** wpisz deklaracje ścieżek w aplikacji (pomoc na GitHubie)
  - Dodaj parametr **onPress** do elementów listy, przekaż obiekt paczki do widoku podrzędnego.
- Zobacz czy aplikacja się uruchamia i czy da się przejść do widoku detali po kliknięciu na wiersz

Ćwiczenie 6 – Flexbox

Zobacz jak działa flexbox korzystając z **yoga playground** (https://yogalayout.com/playground/)

Ćwiczenie 7 – Pobieranie danych

- Commit: **Exercise 7**
- Stwórz plik **services/paczkomaty.js**
- Dodaj stałą **`const TRACKING_URL = 'https://api-shipx-pl.easypack24.net/v1/tracking/';`**
- Dodaj klasę Paczkomaty z metoda asynchroniczną **fetch(trackingNumber)**

- Zwróć rezultat wywołania metody fetch, pamiętając, aby zwracany obiekt posiadał właściwość key z unikalną wartością (np. numer przesyłki)
- W pliku services/package.js:
  - zaimportuj klasę **Paczkomaty**
  - W metodzie **fetch** pobierz informacje dla każdego elementu tablicy i zwróć je.
- Zaktualizuj widok listy, aby obsługiwał nowy format zwracanych danych (**List.Item** i proporcje **title** i **description**)
- Możesz dodać obsługę błędów (try, catch).

Ćwiczenie 8 – Widok szczegółów

- Commit: **Exercise 8**
- W pliku **ListView.js** dodaj przekazywanie parametrów do widoku podrzędnego (details view):

```
    this.props.navigation.navigate('Details',
{number: item.key, item: item})
```

- W pliku **DetailsView.js**:
  - Dodaj propercję `state = { item: {} }`
  - Dodaj statyczną propercje **navigationOptions** z metodą, która ustawi tytuł korzystając z przekazanego w nawigacji parametru.
  - W metodzie **componentWillMount** za pomocą **setState** ustaw przekazany w nawigacji obiekt paczki.
  - W DetailsView wyświetl obiekt (`JSON.stringify()`) z danymi (pomoc na GitHubie)

Ćwiczenie 9 – Widoku szczegółów c.d.

- Commit: **Exercise 9**
- Dodaj widok karty szczegółów przesyłki
  - Możesz wykorzystać biblioteki React Paper (komponent **Card**)
  - Podpowiedź na GitHubie
  - Wygląd każdego elementu można dowolnie zmienić korzystając z właściwości **style** komponentu, poeksperymentuj. https://facebook.github.io/react-native/docs/stylesheet
  - Za pomocą komponentu **Linking** z pakietu **‚expo'** dodaj przycisk otwierający link do śledzenia w przeglądarce.

Ćwiczenie 9a – Historia paczki w widoku szczegółów

- Commit: **Exercise 9a**
- Dodaj historię statusów przesyłki:

- Zobacz jak to jest zrobione na GitHubie
- Wygeneruj listę komponentów na bazie właściwości **item.tracking_details.**
- Wyświetl tą listę w widoku szczegółów.
- Zamień widok nadrzędny komponentu na **ScrollView**.
- Ustaw mu odpowiednie style we właściwości **contentContainerStyle**. {paddingBottom:30}.
- *Uwaga, na githubie jest błąd właściwość **containerInnerStyle** powinna nazywać się **contentContainerStyle!***
- Ustaw style dodanym komponentom wedle uznania.

Ćwiczenie 10 – Dodawanie nowej przesyłki

- Commit: **Exercise 10**
- UI: W pliku **ListView.js**:
  - Podpowiedź na GitHubie
  - Dodaj przycisk w nagłówku i w metodzie **componentDidMount** dodaj:
    ```
    this.props.navigation.setParams({ addPackage:
    this.addPackage.bind(this) });
    ```
  - Dodaj metody **addPackage, finishAddingPackage** i **cancelAddingPackage**, które odpowiednio ustawia stan komponentu.
  - Dodaj komponent **Dialog** posiłkując się kodem na GitHub.
- Logika:
  - Dodaj właściwości **`addingPackage: false, number: ''`** do stanu komponentu
  - Podpowiedź na GitHubie
  - W pliku **package.js** dodaj metodę **addPackage**, która doda nową przesyłkę do wewnętrznej listy przesyłek w komponencie.
  - Uzupełnij metodę **finishAddingPackage** w pliku **ListView.js** o wywołanie addPackage i odświeżenie danych. Warto dodać przynajmniej prostą walidację numeru. Analogicznie postąp z **cancelAddingPackage**.
  - Dla chętnych: wykorzystanie AsyncStorage do przechowywania paczek w pamięci telefonu.

Cwiczenie 11 – Pull to refresh

- Commit: **Exercise 11**
- W pliku **ListView.js**:
  - Dodaj właściwość **`refreshing: false`** do stanu komponentu
  - Dodaj metodę **refreshData** na podstawie metody **fetchData,** pobierającą dane i ustawiającą odpowiednio flagę **refreshing.**
  - Do komponentu **FlatList** dodaj właściwości **onRefresh** i **refreshing**
  - Podpowiedź na GitHubie