Uniwersytet Rzeszowski
Kolegium Nauk Przyrodniczych

Kalkulator walut - Aplikacja mobilna oparta o React Native

Programowanie urządzeń mobilnych

Prowadzący: **dr inż. Piotr Lasek**

Autor: Marcin Wielgos

Rok akademicki 2020/2021, semestr zimowy

Opis projektu

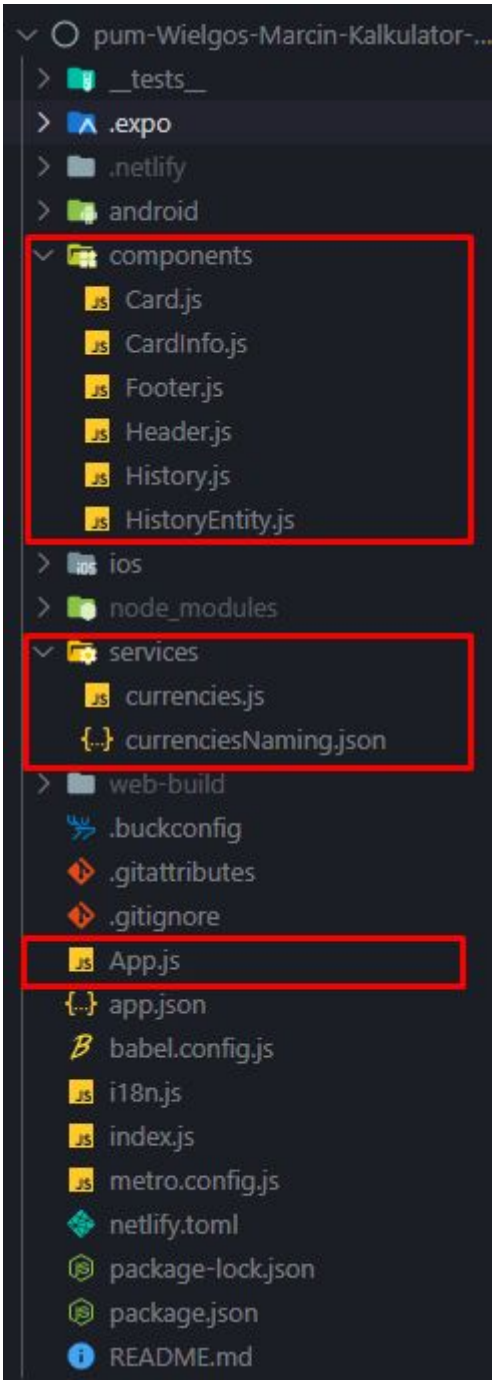
Celem projektu jest zaprojektowanie aplikacji mobilnej umożliwiającej konwersję walut w oparciu o aktualnie obowiązujące kursy. Aplikacja została wykonana w wersji anglojęzycznej.

Aplikacja hostowana pod adresem: https://currency-conv.wielgos.me/ przez serwis Netlify (continuous deployment).
Repozytorium projektu: https://github.com/mwvabes/pum-Wielgos-Marcin-Kalkulator-walut

Funkcjonalności

1. Łączenie z zewnętrznym API
   Aplikacja łączy się z zewnętrznym API przy użyciu biblioteki Axios https://github.com/axios/axios. Wykorzystano API dostępne w serwisie https://exchangeratesapi.io/, które zapewnia aktualne kursy walut i dodatkowo https://purecatamphetamine.github.io/country-flag-icons/ zapewniające flagi państw. Lokalnie przechowywane są informacje o walutach, między którymi można wykonywać konwersję. Po wybraniu odpowiedniej waluty przez użytkownika wysyłane jest zapytanie do API o kursach dla tej waluty.
2. Obliczanie kursu zależnie od wybranej waluty i uzupełnionej ilości na podstawie zapytania API.
3. Odporność na błędy takie jak wprowadzenie wartości innej niż numeryczna lub wybranie takiej samej waluty źródłowej i destynacji.

Struktura plików



Kolorem czerwonym oznaczono najważniejsze elementy.

- App.js - Komponent wywoływany przez index.js. Zarządza stanem aplikacji i wywołuje wszystkie jej komponenty z folderu /components.
- /components/Header.js - Komponent nagłówka aplikacji
- /components/Card.js - Komponent z kartą główną do wprowadzania wartości do kalkulacji

Widoki

1. Karta główna po wejściu do aplikacji (wartości domyślne: waluty - 0, waluta bazowa - PLN, waluta destynacji - EUR)

Po wpisaniu wartości następuje zapytanie do API i obliczenie waluty. Zapytanie do API odbywa się przy pomocy poniższej funkcji (./services/currencies.js)

```
const getByCurrencyValue = (value) => {
  const request =
axios.get(`${baseUrl}?base=${value}`).catch(error =>
console.log(error))
  return request.then(response => response.data)
}
```

Następnie informacje te aktualizowane są do stanu komponentu przez funkcję (./App.js)

```
  const refreshCalculation = () => {

currenciesService.getByCurrencyValue(sourceSelect).then((
response) => {

setDestinationCurrencyValue(response.rates[destinationSel
ect].toFixed(2))

setDestinationTextInput((response.rates[destinationSelect
] * sourceTextInput).toFixed(2))
    })
  }
```

W dolnej części pojawia się informacja o przeliczniku za daną jednostkę i flaga państwa korzystająca z danej waluty.

Do pobierania wartości dla waluty użyto komponentu TextInput wbudowanego w ReactNative (./components/Card.js)

```
        <TextInput
           style={styles.textInput}
           onChangeText={(text) => { handleTextInput(text,
"source") }}
           value={sourceTextInput}
           keyboardType={"decimal-pad"}
         />
```

Komponent korzysta z funkcji handleTextInput zależnej od pola. Dzięki temu jest w stanie określić, czy przypisać wartość stanu dla waluty źródłowej bądź destynacji. (./App.js)

```
  const handleTextInput = (text, type) => {
    if (type === "source") {
      setSourceTextInput(text)
    } else {
      setDestinationTextInput(text)
    }
  }
```

Wybór walut odbywa się przez komponent Picker pakietu @react-native-picker/picker (./components/Card.js)

2. Ikony karty głównej
    a. Ikona dodawania do historii

Po kliknięciu następuje zapis wartości do bazy danych (SQLite), dzięki czemu można zapisać kalkulowane wartości i powrócić do nich w przyszłości. Dodawanie odbywa się przez funkcję (./App.js)

```
  const handleAdd = (sourceValue, sourceCurrency,
destinationCurrency) => {
    db.transaction(tx => {
      tx.executeSql(
```

```
        `INSERT INTO HISTORY (sourceValue,
sourceCurrency, destinationCurrency) VALUES (?, ?,
?)`,
        [sourceValue, sourceCurrency,
destinationCurrency],
        (success, result) => {
          setHistoryKey(historyKey + 1)
        },
        (error, result) => {

        })
    }, error => {

    }, (success, result) => {

    })
  }
```

b. Ikona przełączania “swap”

Pozwala na zamianę aktualnych walut miejscami. Waluta źródłowa staje się walutą destynacji, a waluta destynacji staje się walutą źródłową.
Korzysta z funkcji swapValues (./App.js)

```
  const swapValues = () => {
    let source = sourceSelect
    setSourceSelect(destinationSelect)
    setDestinationSelect(source)
  }
```

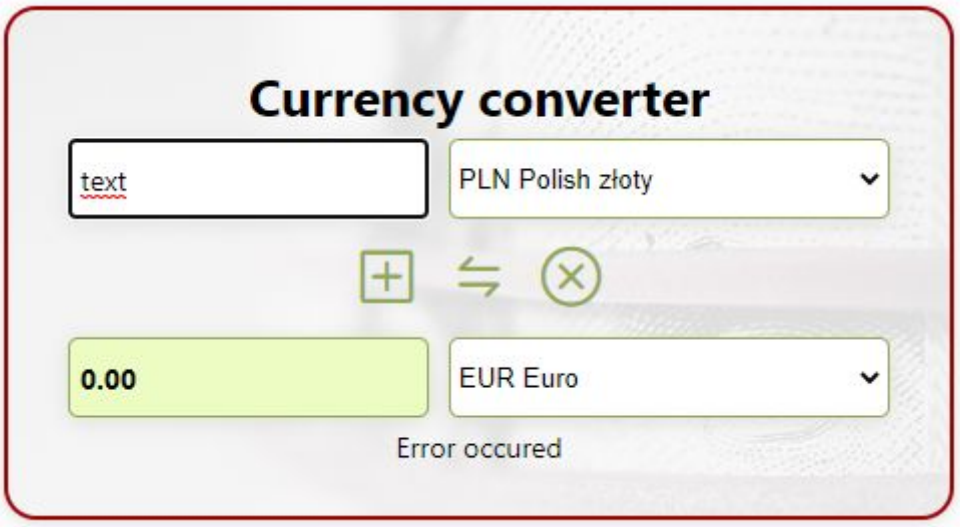
c. Ikona czyszczenia

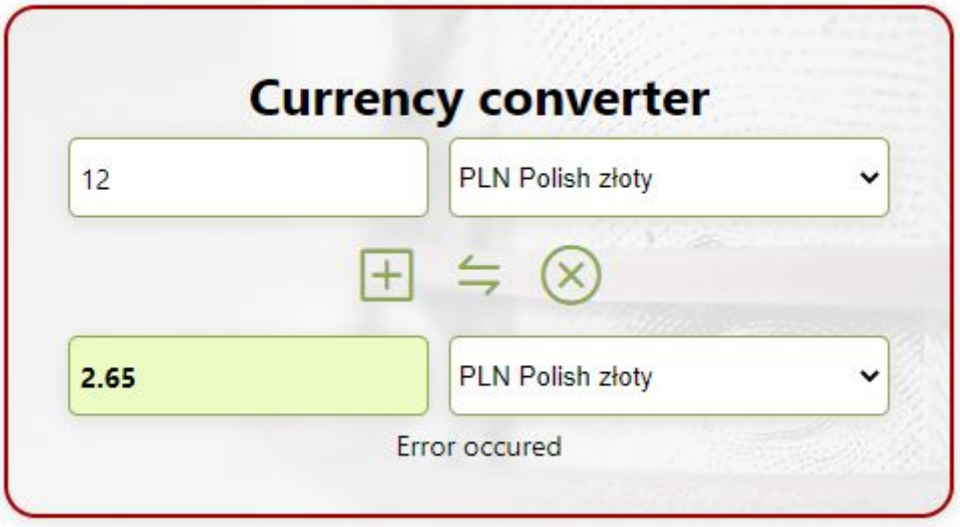
Przywraca wartości do wartości początkowych. Przydatna szczególnie w przypadku wystąpienia błędu. Funkcja clearInputs (./App.js)

```
  const clearInputs = () => {
    setCurrenciesNaming(currenciesNamingJSON)
    setSourceTextInput(0)
```

```
    setDestinationTextInput(0)
    setSourceSelect("PLN")
    setDestinationSelect("EUR")
    setErrorOccured(false)
  }
```

3. Karta główna w przypadku błędów takich jak podanie tekstu w polu waluty lub wybranie takiej samej waluty bazowej i destynacji. W dolnej części karty informacja o wystąpieniu błędu “Error occured”





Sprawdzanie błędu następuje przy każdej aktualizacji stanu w funkcji useEffect() (./App.js)

```
  useEffect(() => {
```

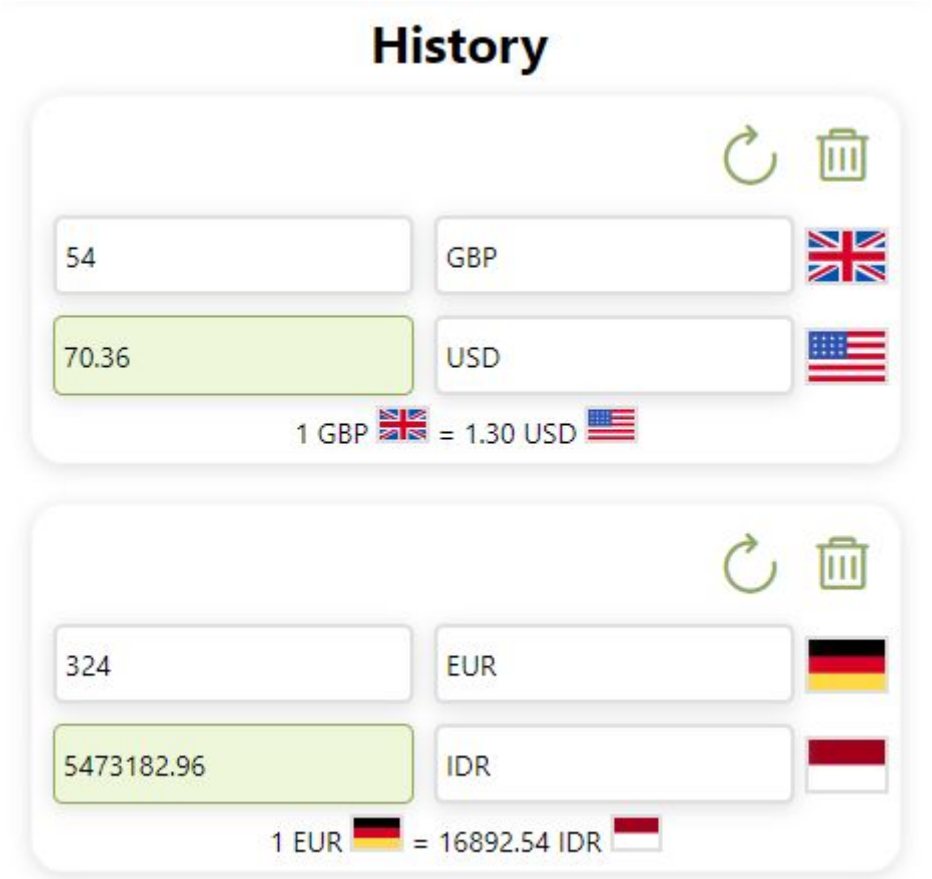
```
    if (
      (sourceSelect != destinationSelect)
      &&
      currenciesNaming.some(currency => currency.code ==
sourceSelect)
      &&
      currenciesNaming.some(currency => currency.code ==
destinationSelect)
      &&
      !isNaN(sourceTextInput)
      &&
      !isNaN(destinationTextInput)
    ) {
      setErrorOccured(false)

setSourceFlag(findTerritoryByCurrencyCode(sourceSelect))

setDestinationFlag(findTerritoryByCurrencyCode(destinatio
nSelect))
      refreshCalculation()
    } else {
      setErrorOccured(true)
    }


  }, [{ sourceTextInput, sourceSelect, destinationSelect
}])
```

4. Historia (./components/History.js)
   Wyświetla dane pobrane z bazy danych, które mogły zostać wcześniej zapisane przez użytkownika. Prezentacja następuje przez kompontent (./components/HistoryEntity.js).



Komponent pobiera dane z bazy danych poprzez funkcję (./components/History.js)

```
const loadHistory = () => {
  db.transaction(tx => {
    tx.executeSql(
      `SELECT * FROM HISTORY`,
      [],
      (success, result) => {
        setItems(Object.values(result.rows))
      },
      (error, result) => {

      })
  }, error => {

  }, (success, result) => {

  })
}
```

Następnie dane są mapowane ze stanu. Kolejność od najnowszych do najstarszych.

```
        items.slice(0).reverse().map(item => {
          return <HistoryEntity
            key={item.ID_entity}
            item={item}
            handleDelete={handleDelete}

findTerritoryByCurrencyCode={findTerritoryByCurrencyCode}
          />
        })
```

Komponent zawiera także funkcję, która może być wywołana z komponentu HistoryEntity i służy do usuwania danego obiektu z historii.

```
  const handleDelete = (idToDelete) => {
    db.transaction(tx => {
      tx.executeSql(
        `DELETE FROM history where ID_entity = ?`,
        [idToDelete],
        (success, result) => {
          loadHistory()
        },
        (error, result) => {

        })
    }, error => {

    }, (success, result) => {

    })
  }
```

5. Komponent HistoryEntity (./components/HistoryEntity.js)
   Reprezentuje pojedynczy moduł pobrany i wywołany w komponencie History.
   Posiada dwa przyciski:

Do odświeżenia modułu dzięki funkcji

```
  const handleRefresh = () => {

currenciesService.getByCurrencyValue(item.sourceCurrency)
.then((response) => {

setDestinationCurrencyValue(response.rates[item.destinati
onCurrency].toFixed(2))

setDestinationTextInput((response.rates[item.destinationC
urrency] * item.sourceValue).toFixed(2))
    })
  }
```

Oraz przycisk

Do usunięcia z bazy danych poprzez funkcję handleDelete komponentu History

6. Nagłówek (./components/Header.js)

Zawiera nazwę aplikacji i informację o autorze

7. Stopka (./components/Footer.js)

Zawiera informacje o wykorzystanych API.