Biblioteki SALESmanago Mobile dla React Native

1. Konfiguracja

Dla obu systemów należy dołączyć standardowe biblioteki SALESmanago Mobile i wykonać kroki opisane w instrukcjach:
https://pomoc.salesmanago.pl/developers

Android

Analogicznie tak jak plik appmanagolibrary-firebase-release.aar, do projektu należy dołączyć bibliotekę appmanagolibrary-react-adapter-release.aar i zaimportować w Gradle. Android wymaga dodatkowej rejestracji modułu (https://reactnative.dev/docs/native-modules-android#register-the-module-android-specific), dlatego w metodzie `getPackages()` klasy `MainApplication` należy dodać do listy pakietów instację pochodzącej z biblioteki klasy `AmPackage:`

```
import com.appmanago.lib.AmPackage;
//...

@Override
protected List<ReactPackage> getPackages() {
 @SuppressWarnings("UnnecessaryLocalVariable")
 List<ReactPackage> packages = new PackageList(this).getPackages();
 packages.add(new AmPackage());
 return packages;
}
```

iOS

Obok AmMonitor..framework należy dodać do projektu AppmanagoLibraryReactAdapter.framework. Moduł natywny powinień zostać zarejestrowany automatycznie, bez dodatkowej konfiguracji.

2. Użycie bibliotek w JS

Adapter do biblioteki można zaimportować do plików JS w następujący sposób:

```
import { NativeModules } from 'react-native';
const { AmMonitorAdapter } = NativeModules;
```

Wywołania poszczególnych metod zamieszczono poniżej. Opis ich działania można znaleźć w https://pomoc.salesmanago.pl/developers.

`create(), destroy(), eventStarted(), eventEnded()`:
Ze względu na implementację biblioteki dla Androida, ważne jest aby wywołania metod `eventStarted()` i `eventEnded()` znajdowały się pomiędzy wywołaniami `create()` i `destroy()` dla tego samego modułu. Zazwyczaj zapewnia się to przez wywoływanie `create()` podczas tworzenia ekranu / komponentu i `destroy()` przed zakończeniem jego działania.

```
AmMonitorAdapter.create('moduleSimpleId');
AmMonitorAdapter.eventStarted('moduleSimpleId');
AmMonitorAdapter.eventEnded('moduleSimpleId');
AmMonitorAdapter.create('moduleSimpleId');
```

`clicked()`:
Wywołanie metody powinno zawierać dodatkowo id modułu, do którego należy funkcja i podobnie jak poprzednie, powinno znajdować się między wywołaniami `create()` i `destroy()` dla tego samego modułu.

```
AmMonitorAdapter.clicked('moduleSimpleId', 'functionSimpleId');
```

`syncEmail(), syncMsisdn()`:

```
AmMonitorAdapter.syncEmail('mail@test.com');
AmMonitorAdapter.syncMsisdn('+48123456789');
```

`sendLocation()`:

```
Geolocation.getCurrentPosition(
 async(position) => {
   let coords = position.coords;
   AmMonitorAdapter.sendLocation(coords.latitude.toString(),
     coords.longitude.toString());
   //e. g.
   //AmMonitorAdapter.sendLocation('37.33233141', '-122.0312186');
 };
```

eventCustom():

```
AmMonitorAdapter.eventCustom('u_event_id', {
 "textParamterName": {
   "type": "text",
   "value": "Sample text"
 },
 "numberParamterName1": {
   "type": "integer",
   "value": 123
 },
 "numberParamterName2": {
   "type": "double",
   "value": 123.678
 },
 "dateParameterName": {
   "type": "date",
   "value": new Date().toISOString()
 },
 "booleanParameterName": {
   "type": "boolean",
   "value": true
}
});
```

sendUserProperties():

```
AmMonitorAdapter.sendUserProperties({
 "textPropertyName": {
   "type": "text",
   "value": "Sample text"
 },
 "numberPropertyName1": {
   "type": "123",
   "value": 666
 },
 "numberPropertyName2": {
   "type": "double",
   "value": 123.678
 },
 "datePropertyName": {
   "type": "date",
   "value": new Date().toISOString()
 },
```

```
 "booleanPropertyName": {
   "type": "boolean",
   "value": false
 }
});
```