Ocean.js integration with React Native

Overview

Currently we are coding the **ALGA App**, using React Native as the programming language.

We pretend to provide the **Add Liquidity** and **Swap** features which are actually working in **Ocean Protocol Marketplace** (https://market.oceanprotocol.com/)

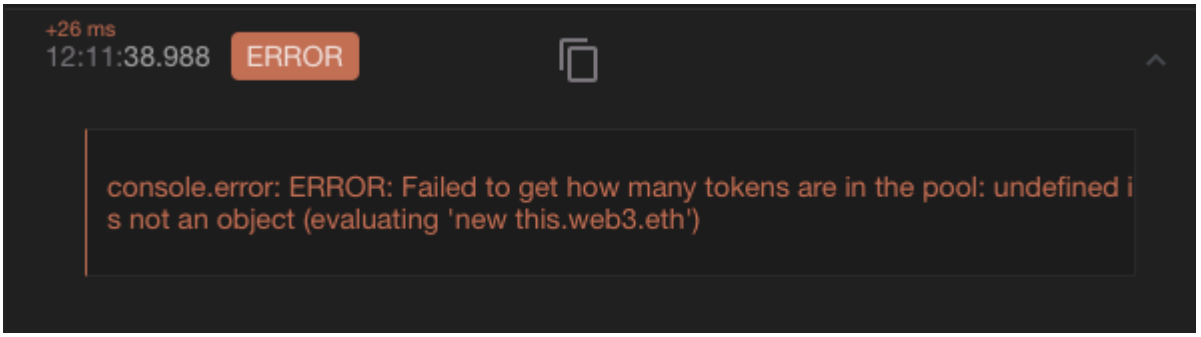
In order to achieve this, we installed the **ocean.js** library and we followed the steps described in the Github repo (https://github.com/oceanprotocol/ocean.js).

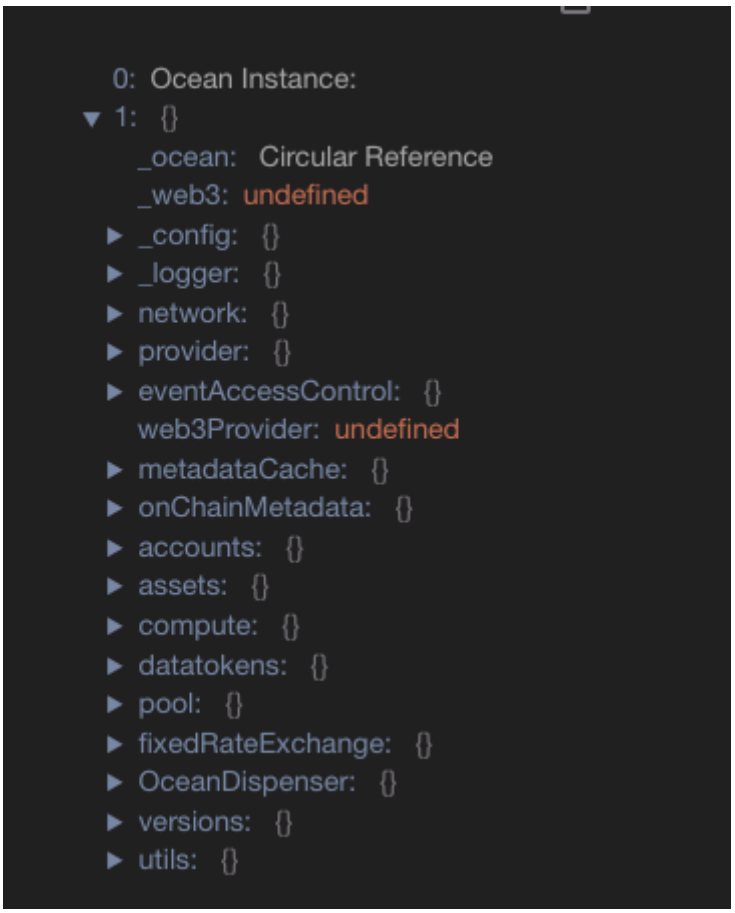
Note: We are sending the metadataCacheUri and providerUri fields with the default value provided by the ConfigHelper.

Below you can see the configuration that we are following per the Github repo.

```
import { Ocean, Config, ConfigHelper, Logger } from '@oceanprotocol/lib'

const defaultConfig: Config = new ConfigHelper().getConfig(
  'rinkeby',
  'YOUR_INFURA_PROJECT_ID'
)

const config = {
  ...defaultConfig,
  metadataCacheUri: 'https://your-metadata-cache.com',
  providerUri: 'https://your-provider.com'
}

async function init() {
  const ocean = await Ocean.getInstance(config)
  return ocean
}
```

When we tried to instantiate **ocean.js** with the previous configuration, we got this error, which is telling us that we need a **web3 instance.**





After getting this error, we made an analysis of how the Ocean Protocol Marketplace is working with **ocean.js**, in order to replicate the same behaviour in the **ALGA** App using React Native (mobile).

The conclusion is that the marketplace creates an instance of **ocean.js** sending a **web3provider** instance as a parameter. This **web3provider** instance is created when the user interacts with his wallet using the **web3modal** library**.**

Currently web3modal doesn't support React Native but we tried to install it anyway.

We tried to adapt the context in order to use it in React Native but we couldn't achieve the goal because web3modal uses the document object of the browser and it's not possible to get in mobile.

```
// ------------------------------------
// Helper: Create Ocean instance
// ------------------------------------
const connect = useCallback(
  async (config: ConfigHelperConfig) => {
    if (!web3) return

    const newConfig: ConfigHelperConfig = {
      ...config,
      web3Provider: web3
    }

    try {
      Logger.log('[ocean] Connecting Ocean...', newConfig)
      const newOcean = await Ocean.getInstance(newConfig)
      setOcean(newOcean)
      setConfig(newConfig)
      Logger.log('[ocean] Ocean instance created.', newOcean)
    } catch (error) {
      Logger.error('[ocean] Error: ', error.message)
    }
  },
  [web3]
)
```

(Ocean.tsx from market)

```
// ----------------------------------
// Helper: connect to web3
// ----------------------------------
const connect = useCallback(async () => {
  if (!web3Modal) {
    setWeb3Loading(false)
    return
  }
  try {
    setWeb3Loading(true)
    Logger.log('[web3] Connecting Web3...')

    const provider = await web3Modal?.connect()
    setWeb3Provider(provider)

    const web3 = new Web3(provider)
    setWeb3(web3)
    Logger.log('[web3] Web3 created.', web3)

    const networkId = await web3.eth.net.getId()
    setNetworkId(networkId)
    Logger.log('[web3] network id ', networkId)

    const chainId = await web3.eth.getChainId()
    setChainId(chainId)
    Logger.log('[web3] chain id ', chainId)

    const accountId = (await web3.eth.getAccounts())[0]
    setAccountId(accountId)
    Logger.log('[web3] account id', accountId)
  } catch (error) {
    Logger.error('[web3] Error: ', error.message)
  } finally {
    setWeb3Loading(false)
  }
}, [web3Modal])
```

(Web3.tsx from market)

React Native's Proofs of Concepts (all failed)

1. Use WalletConnect and ocean.js in React Native

We are using WalletConnect v1.0 - react-native-dapp library (https://github.com/cawfree/create-react-native-dapp) in order to make the connection between a pre-selected wallet and Alga App.

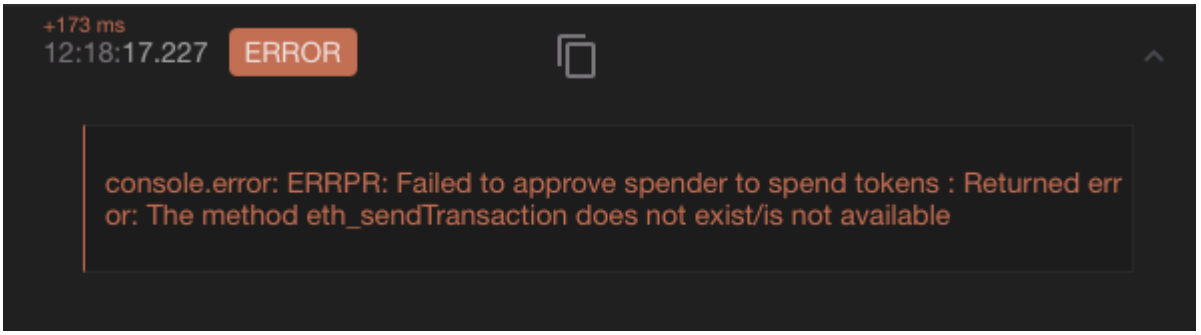
This integration was made successfully and after connecting to the wallet, we are getting a **WalletConnectClient** instance which provide us the methods to interact and send transactions in the Ethereum blockchain.

We couldn't find a way to integrate WalletConnect with ocean.js because as we described before, ocean.js expects a **web3provider** and we are getting a **WalletConnectClient**.

2. Create web3 instance using *infura's* + *id* in React Native

We tried to create an instance of web3 using infura's url + id and we tried to use it as the **web3provider** parameter expected by ocean.js.

We tried to execute an operation and we got this error:



We are not surprised that we got this error because we haven't an authorized session or provider, there is no integration with the wallet.

3. web3modal integration in React Native

As we described above, our first thought was to replicate how the Marketplace is working and we tried to integrate **web3modal** in React Native in order to get an instance of a **web3provider** which **ocean.js** uses.

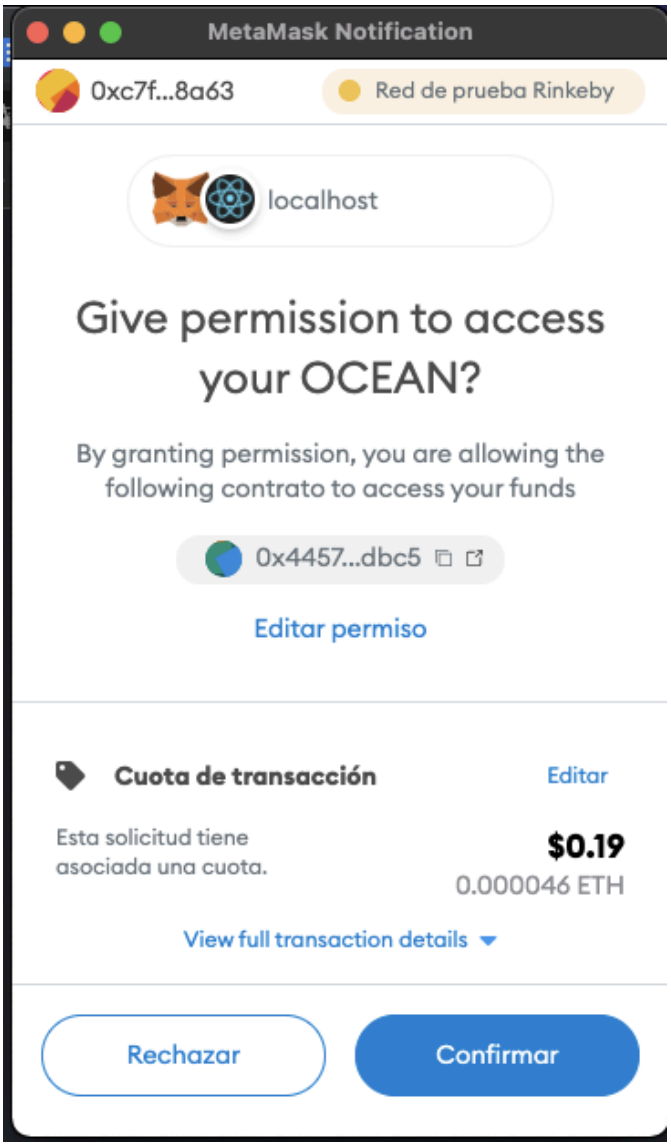
We couldn't achieve it because **web3modal** is a **web** library and it hasn't support in React Native. We tried to make some emulations or tricks without luck because it uses the document object of the browser which is not available in RN.

React *Web's* Proof of Concept (working as expected)

We wanted to be 100% sure that we were not misunderstanding anything so we made a POC using React **Web, web3 and Infura**.

We created a **web3provider** using **web3 with the infura id** and we used it as a parameter when we instantiated **ocean.js.**

It worked as expected, we created the instance successfully and we made an operation without problems. Below you can see a Metamask's screenshot asking the user authorization to execute the Add Liquidity operation (ocean.pools.addliquidity).



Conclusions - Technical limitations

Currently we haven't found a way to interact with **ocean.js** using **React Native** and we think that it’s crucial to get mobile support.

In order to achieve this, probably the best way is to use **WalletConnect** to interact with the wallets and **ocean.js** should provide a new constructor which receives a **WalletConnectClient** instead of a **web3provider** in order to execute all the operations provided by the library.

The other option, which in our opinion **is not possible**, due to technical limitations, is to create a **web3provider** instance in React Native in order to instantiate the **ocean.js**.