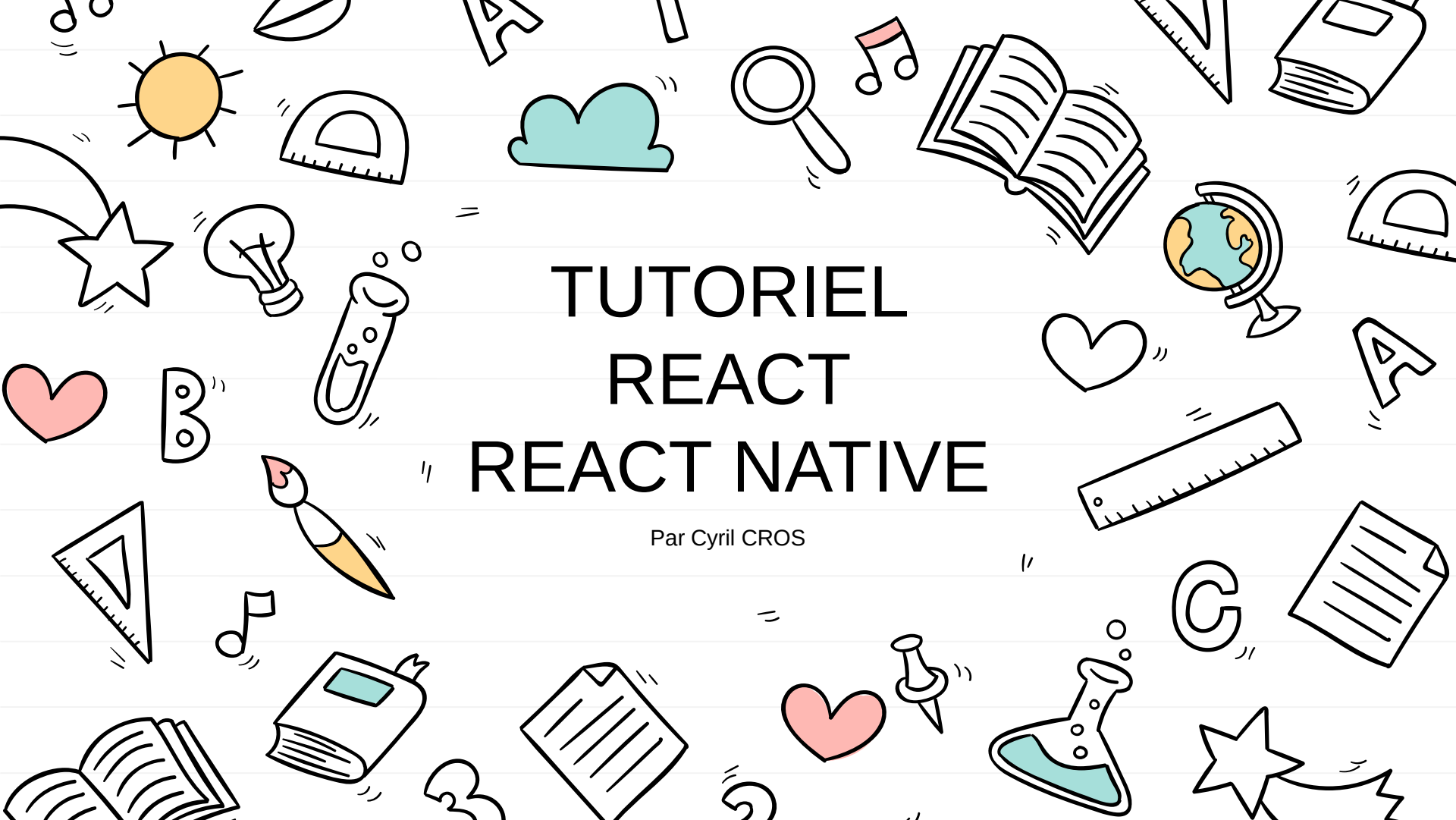
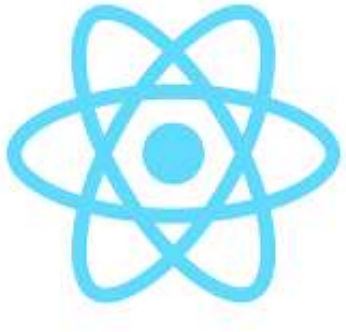


# TUTORIEL REACT REACT NATIVE

Par Cyril CROS

# Sommaire

# Historique de React et react native



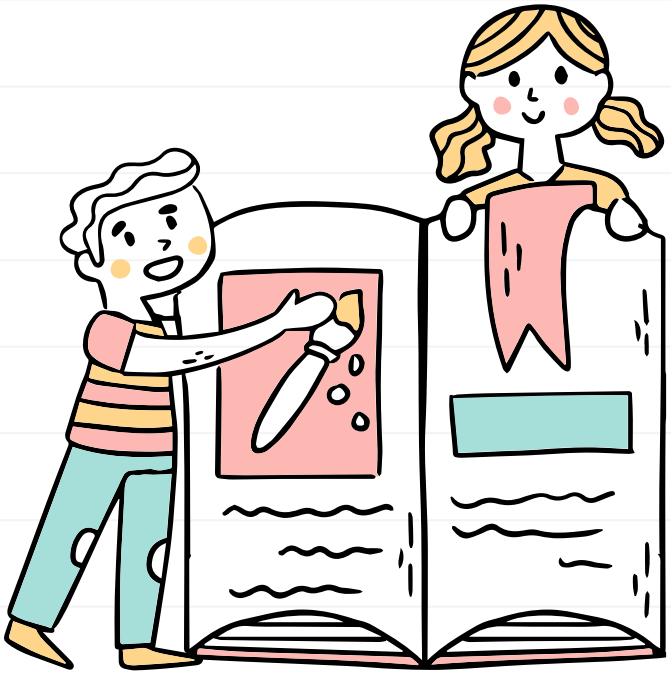
React (2011)

Jordan Walke


React Native (2015)

# But du tutoriel

- Etre capable de réaliser une vue
- Etre capable de naviguer entre les écrans
- Relier une API à l’application



# Configuration de l’environnement

- Visual studio Code : https://code.visualstudio.com/Download

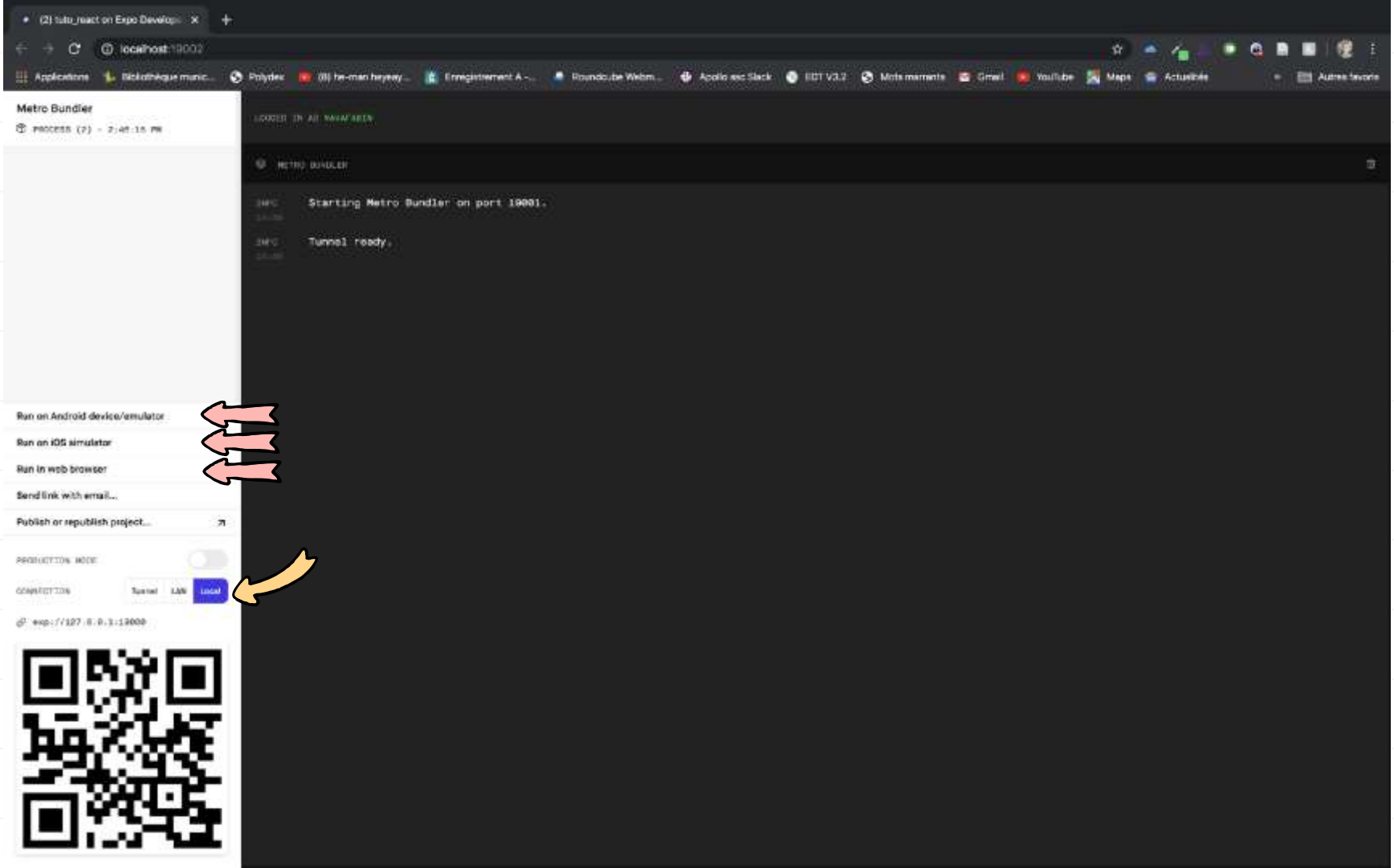
- NodeJS (versions LTS) : https://nodejs.org/en/

# Configuration de l’environnement

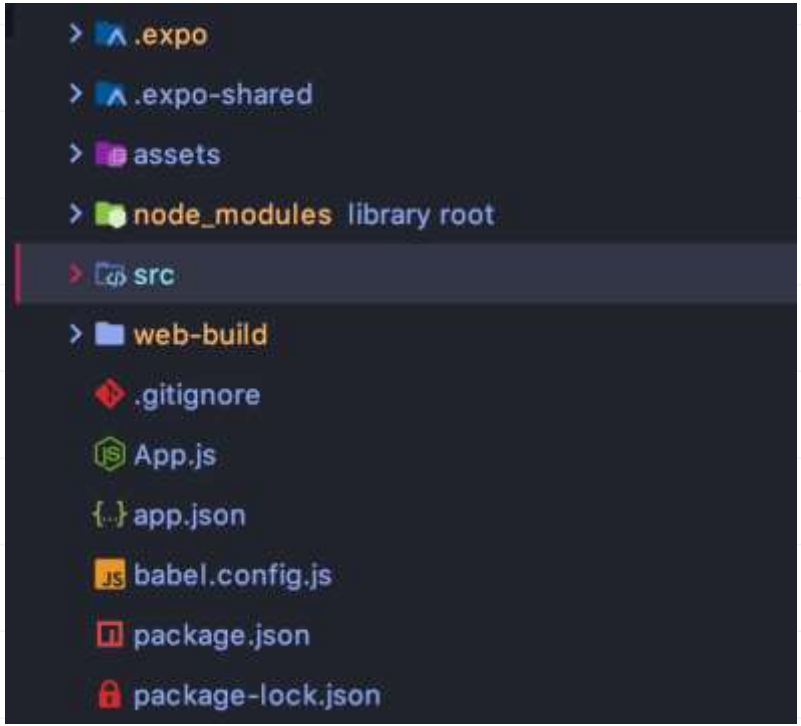
*Your best friend today*

Commandes :

- *npm install -g expo-cli*
- *expo init projectName*
- *cd projectName*
- *npm install*
- *npm start*


(2) tuto_react on Expo Develop
localhost:19002
Applications
Bibliothèque munic...
Polydev
(8) he-man heyway ...
Enregistrement A -...
Roundcube Webm...
Apollo asc Slack
EDT V3.2
Mots marrants
Gmail
YouTube
Maps
Actualités
Autres favoris
Metro Bundler
PROCESS (7) - 2:46:15 PM
LOADED IN 40 MANIFOLDS
METRO BUNDLER
INFO Starting Metro Bundler on port 19001.
INFO Tunnel ready.
Run on Android device/emulator
Run on iOS simulator
Run in web browser
Send link with email...
Publish or republish project...
PRODUCTION MODE
CONNECTION
Tunnel
LAN
Local
exp://127.0.0.1:19000

# Configuration de l’environnement




# Front –end : Vue d’accueil et vue principale

→ *loginScreen.js et mainScreen.js*

```
import React, { Component } from 'react';

export default class LoginScreen extends Component
{
    constructor(props) {
        super(props);
    }

    render() {
        return(
            //Code de la vue
        )
    }
}
```

# Front –end : Vue d’accueil et vue principale

```
render() {
        return(
                <View>
                <View>
                <Text>Tutorial React / React Native</Text>
            </View>
            <View>
                <Button
                        onPress={LoginScreen._LoginToAPI()}
                        title="Connect to Spotify"
                        color="#20D760"
                />
                </View>
        </View>
        )
}
```

```
static _LoginToAPI()
{

}
```

# Front –end : Vue d’accueil et vue principale

```
const styles = StyleSheet.create({
    container: {
        flex: 1,
        justifyContent: 'center',
    },
    buttonContainer: {
        marginLeft: 50,
        marginRight: 50,
        marginTop: 20,
        marginBottom: 20
    },
    titleContainer: {
        margin: 5,
        alignItems: 'center'
    },
    titleStyle: {
        fontSize: 30
    }

});
```

# Front –end : Vue d’accueil et vue principale

```
render() {
        return(
        <View style={styles.container}>
                <View style={styles.titleContainer}>
                <Text style={styles.titleStyle}>Tutorial React / React Native</Text>
            </View>
            <View style={styles.buttonContainer}>
                        <Button
                        onPress={LoginScreen._LoginToAPI()}
                    title="Connect to Spotify"
                    color="#20D760"
                />
            </View>
        </View>
        )
}
```

# Front –end : Vue d’accueil et vue principale

```
static loginScreenView () {
        return(
        <LoginScreen/>
    );
}
```


# Front –end : naviguer entre les ecrans

Commandes (à la racine du projet):

- *npm install @react-navigation/native*
- *expo install react-native-gesture-handler react-native-reanimated react-native-screens react-native-safe-area-context @react-native-community/masked-view*

# Front –end : naviguer entre les ecrans

→ *App.js*

```
const Stack = createStackNavigator();

function LoginScreen() {
    return (
        loginScreen.loginScreenView()
    );
}

function MainScreen() {
    return (
        mainScreen.mainScreenView()
    );
}
```

# Front –end : naviguer entre les ecrans

```
function App() {
  return (
      <NavigationContainer ref={navigationRef}>
        <Stack.Navigator headerMode="none">
          <Stack.Screen name="Login" component={LoginScreen}/>
          <Stack.Screen name="Home" component={MainScreen} />
        </Stack.Navigator>
      </NavigationContainer>
  );
}

export default App;
```

# Front –end : naviguer entre les ecrans

→ *NavigatorRef.js*

```
import { StackActions } from '@react-navigation/routers';

export const navigationRef = React.createRef();

export function navigate(name, params) {
    navigationRef.current?.navigate(name, params);
}

export function replace(name, params){
    const pushAction = StackActions.replace(name, params);

    navigationRef.current?.dispatch(pushAction);
}
```

# Front –end : naviguer entre les ecrans

→ *loginScreen.js*

```
import * as NavigatorRef from '../navigation/navigatorRef'
```

```
NavigatorRef.replace('Home');
```

# Front –end : Onglets vue principale


# Front –end : Onglets vue principale

Commande (à la racine du projet):

- npm install react-native-elements

*Cool UI Elements inside !*

# Front –end : Onglets vue principale

→ *playlistTab.js*

```
const playlists = [
    {
        name : 'Toad Party !',
        imageUrl : 'https://vignette.wikia.nocookie.net/mario/images/3/38/CTTTChampignon
    },
    {
        name : 'Mario Party !',
        imageUrl: 'https://upload.wikimedia.org/wikipedia/en/a/a9/MarioNSMBUDeluxe.png'
    }
];
```

# Front –end : Onglets vue principale

```
render() {
        return(
           <ScrollView style={stylePlaylist.container}>
       {
              playlists.map((playlist, i) =>{
                 return(
                <Card key={i}>
                   <View style={stylePlaylist.cardContainer}>
                          <View>
                        <Image
                                style={stylePlaylist.imageStyle}
                           resizeMode="cover"
                           source={{ uri: playlist.imageUrl }}
                       />
                       </View>
                       <View style={stylePlaylist.infoPlaylistContainer}>
                           <Text>{playlist.name}</Text>
                       </View>
                   </View>
                </Card>
               );
           })
        }
        </ScrollView>
    )
}
```

# Front –end : Onglets vue principale

→ *searchTab.js*

```
constructor(props) {
        super(props);
        this.state = {
          textSearch: ""
    }
}
```

# Front –end : Onglets vue principale

```
<View style={styleSearch.textInputContainer}>
        <TextInput
                style={styleSearch.textInput}
                placeholder="Search a song !"
        onChangeText={(textSearch) => this.setState({textSearch})}
        value={this.state.textSearch}
        onSubmitEditing={}
        />
</View>
```

# Front –end : Onglets vue principale

```
const result = [
    {
        name: 'GoGoToad',
        mainArtist : 'Toad',
        albumName: 'Toad Dance',
        imageUrl : 'https://vignette.wikia.nocookie.net/mario/images/3/38/CTTTChampignonD%2
    },
    {
        name : 'Marrrriiiooo',
        mainArtist: 'Mario',
        albumName: 'Mario and the Gambas',
        imageUrl: 'https://upload.wikimedia.org/wikipedia/en/a/a9/MarioNSMBUDeluxe.png'
    }
];
```

# Front –end : Onglets vue principale

Commande (à la racine du projet):

- npm i @expo/vector-icons

*All your favorites icons !*

# Front –end : Onglets vue principale

→ *mainScreen.js*

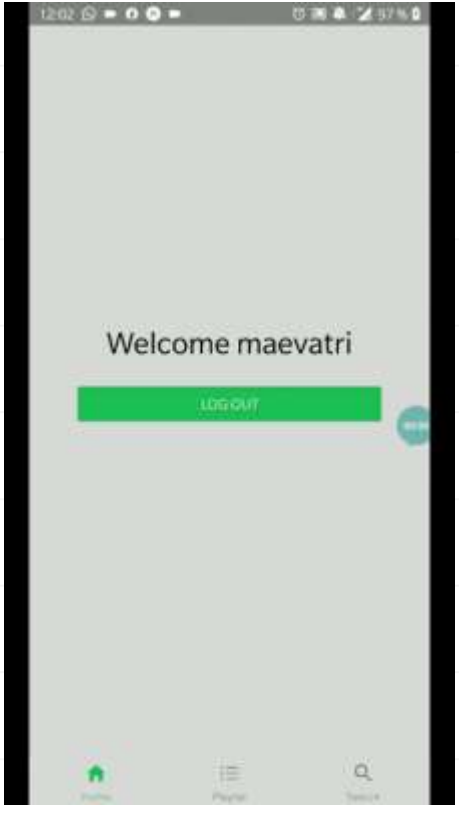
```
const Tab = createBottomTabNavigator();
```

```
import { Ionicons } from '@expo/vector-icons';
```

# Front –end : Onglets vue principale

```
render() {
        return(
                <NavigationContainer independent={true}>
                        <Tab.Navigator screenOptions={({route}) => ({
                                tabBarIcon: ({focused, color, size}) => {
                                        let iconName;

                                        if(route.name === 'Home') {
                                                iconName = 'md-home';
                                        }else if(route.name === 'Playlist'){
                                                iconName = 'ios-list';
                                        }else if(route.name === 'Search'){
                                                iconName = 'md-search';
                                        }

                                        return <Ionicons name={iconName} size={size} color={color}/>;
                                },
                                })}
                                tabBarOptions={{
                                        activeTintColor: '#20D760',
                                        inactiveTintColor: 'gray',
                                }}
                        >
                                <Tab.Screen name="Home" component={HomeView} />
                                <Tab.Screen name="Playlist" component={PlaylistView} />
                                <Tab.Screen name="Search" component={SearchView} />
                        </Tab.Navigator>
                </NavigationContainer>
        )
}
```

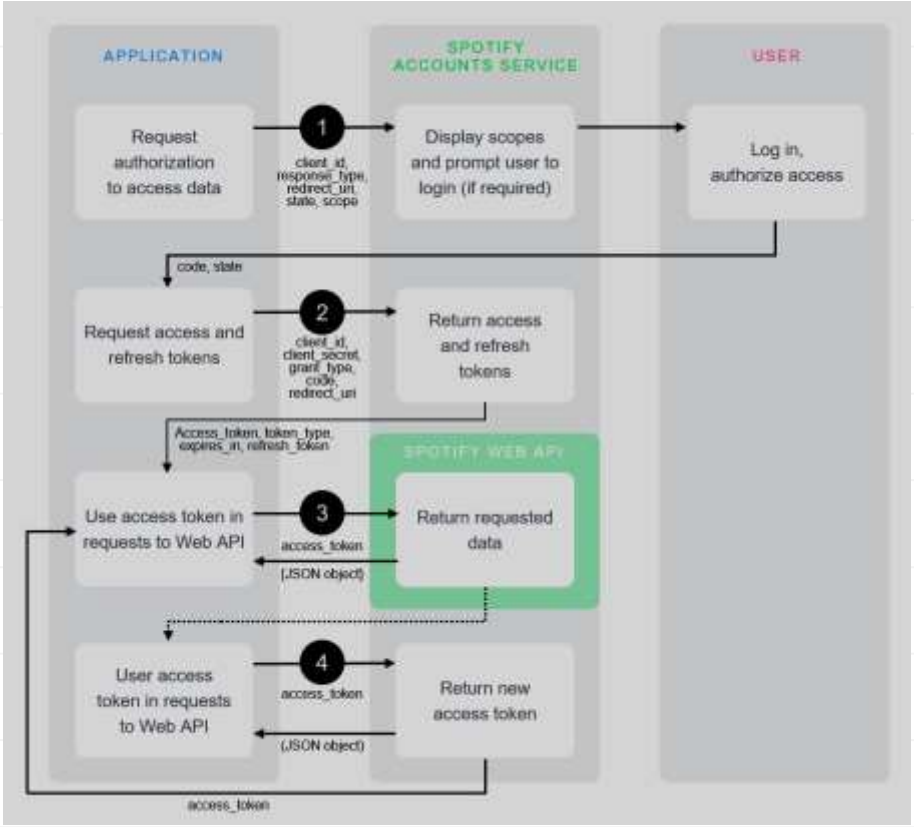
# Front –end : Onglets vue principale



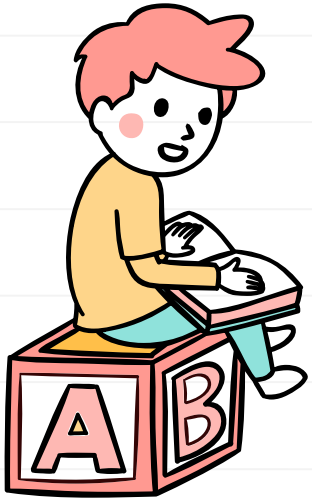
# Back-end : API Spotify

- API REST
- Plusieurs endpoints différents pour divers types de contenus
- Requiert une connexion de l’utilisateur pour accéder aux services

# Back-end : Connexion a l'API

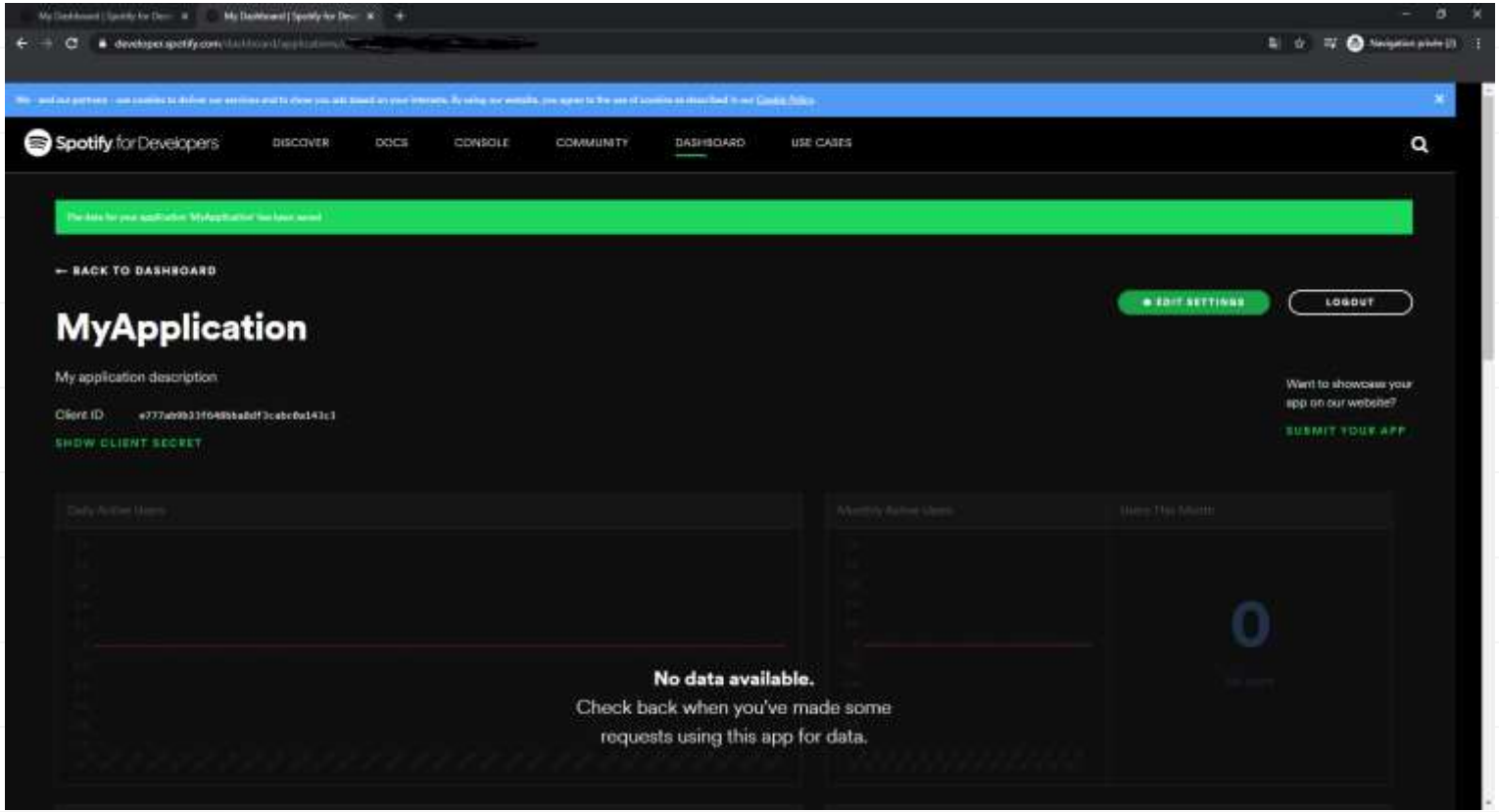


# Back-end : étapes préliminaires

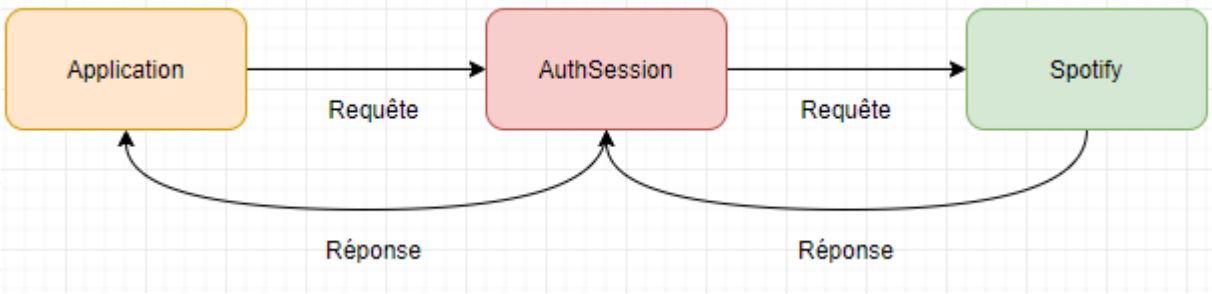



- Création d’une application sur Spotify
- Création d’un compte Expo et mise en place de l’auth Session
- Stockage des credentials

# Back-end : étapes préliminaires



# Back-end : étapes préliminaires


- *expo login*

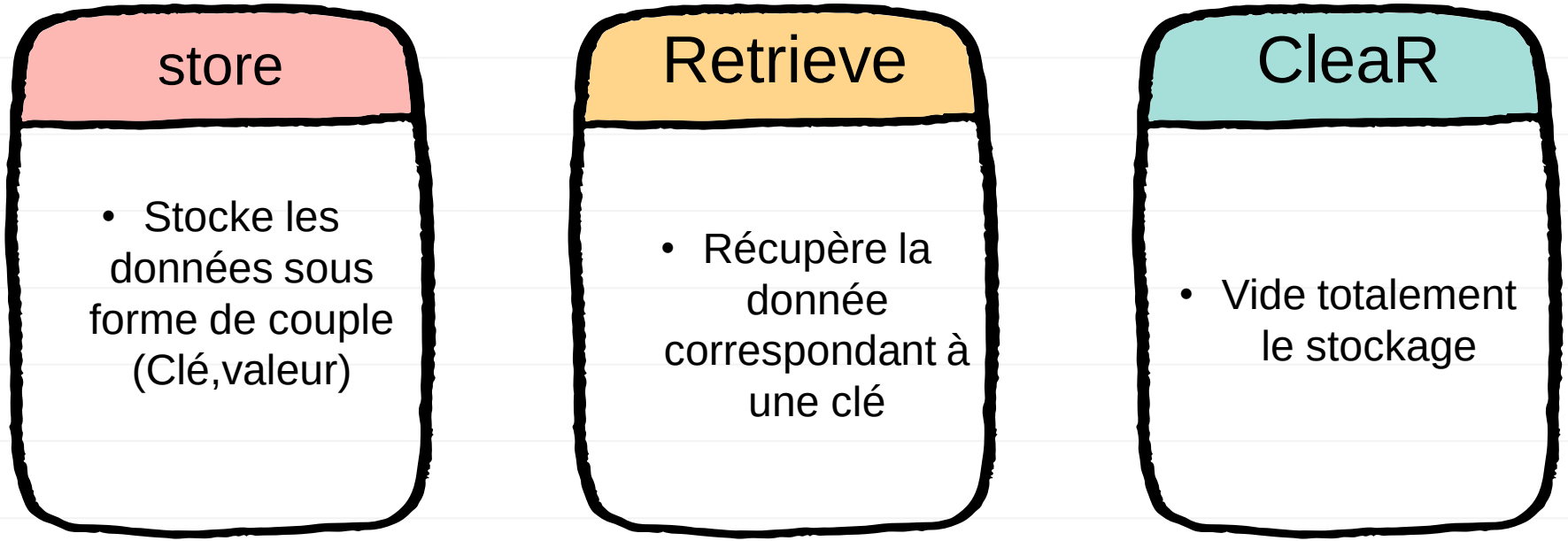
**https://auth.expo.io/@username/folder**

# Back-end : étapes préliminaires

```
export const spotifyCredentials = {
    clientId: 'votre clientId',
    clientSecret: 'votre clientSecret',
    redirectUri: 'votre redirectUri'
}
```

# Back-end : async storage

DataStore



# Back-end : connexion à l’API

- Récupération du code d’autorisation

```
//Récupère les credentials
const credentials = getSpotifyCredentials();

//Récupère l'URL AuthSession
const redirectUrl = AuthSession.getRedirectUrl();

/*
    Démarre le processus d'authentification avec AuthSession.
    AuthSession nous permet de gérer le processus
    comme une simple fonction asynchrone
*/
const result = await AuthSession.startAsync({
    authUrl:
    'https://accounts.spotify.com/authorize' +
    '?response_type=code' +
    '&client_id=' +
    credentials.clientId +
    (scopes ? '&scope=' + encodeURIComponent(scopes) : '') +
    '&redirect_uri=' +
    encodeURIComponent(redirectUrl),
});

//Retourne le code d'autorisation depuis la réponse
return result.params.code;
```

# Back-end : connexion à l'API

- Récupération des tokens

```
const getTokens = async () =>
{
    try {

    //Récupère les informations utiles
    const authorizationCode = await getAuthorizationCode();
    const credentials = getSpotifyCredentials();

    //Encode les credentials en base64
    const credsB64 = btoa(`${credentials.clientId}:${credentials.clientSecret}`);

    //Envoie la requête auprès des services spotify avec notre code d'autorisation
    const response = await fetch('https://accounts.spotify.com/api/token', {
        method: 'POST',
        headers: {
        Authorization: `Basic ${credsB64}`,
        'Content-Type': 'application/x-www-form-urlencoded',
        },
        body: `grant_type=authorization_code&code=${authorizationCode}&redirect_uri=${
        credentials.redirectUri
        }`,
    });
```

# Back-end : connexion à l’API

- Rafraichissement des tokens

```
export const refreshTokens = async () => {
    try
    {
        const credentials = getSpotifyCredentials();
        const credsB64 = btoa(`${credentials.clientId}:${credentials.clientSecret}`);
        const refreshToken = await retrieveData('refreshToken');

        //Envoi de la requête de rafraîchissement
        const response = await fetch('https://accounts.spotify.com/api/token', {
        method: 'POST',
        headers: {
            Authorization: `Basic ${credsB64}`,
            'Content-Type': 'application/x-www-form-urlencoded',
        },
        body: `grant_type=refresh_token&refresh_token=${refreshToken}`,
        });

        //Conversion en JSON
        const responseJson = await response.json();
```

# Back-end : connexion à l’API

- Connexion

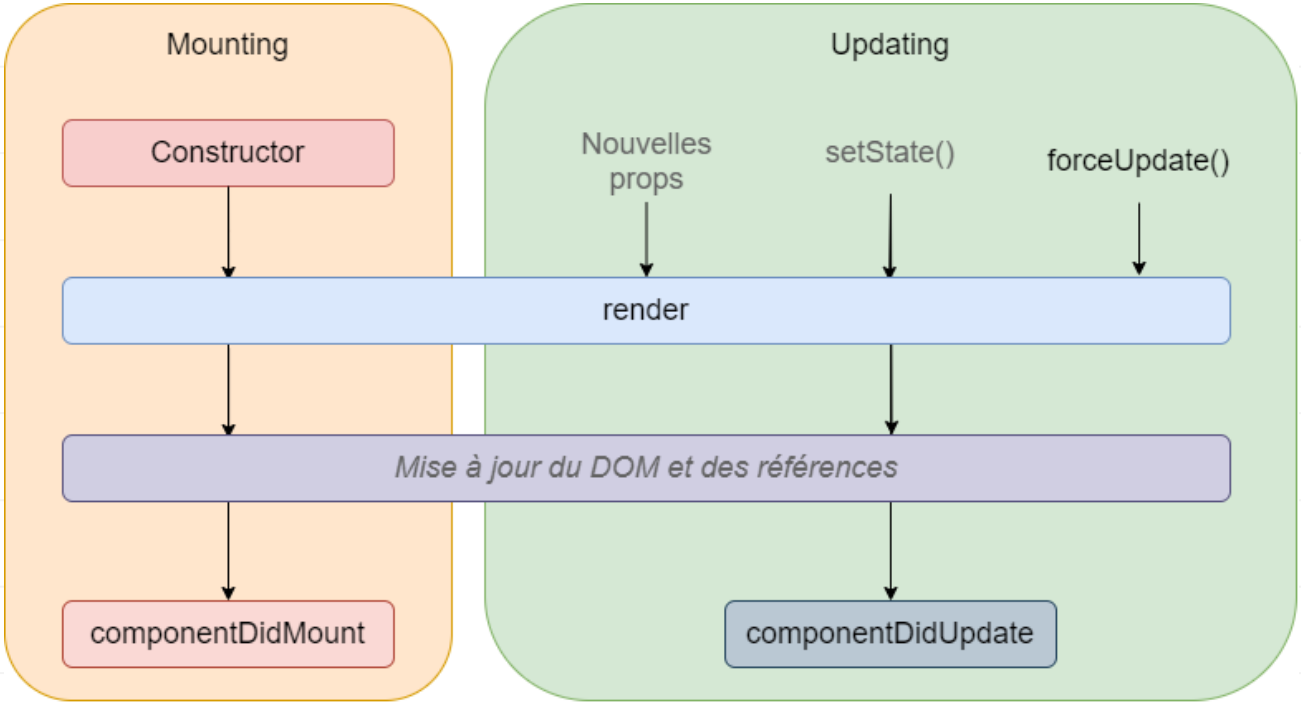
```
export const loginToSpotify = async () =>
{
    const result = await refreshTokens();
    if(result)
    {
        NavigatorRef.replace('Home');
    }
}
```

# Back-end : connexion à l'API

- Connexion

```
static checkIfConnected = async() =>
{
    if(await isAlreadyConnected())
    {
        await checkAndRefreshTokens();
        NavigatorRef.replace('Home');
    }
}

static _LoginToAPI = async() =>
{
    await loginToSpotify();
}

async componentDidMount()
{
    LoginScreen.checkIfConnected();
}
```

# Parenthèse : Le cycle de vie d'un composant




# Back-end : utilisation de l’API

Commande (à la racine du projet):

- npm i spotify-web-api-js

# Back-end : Utilisation DE l'API

- Récupération du wrapper

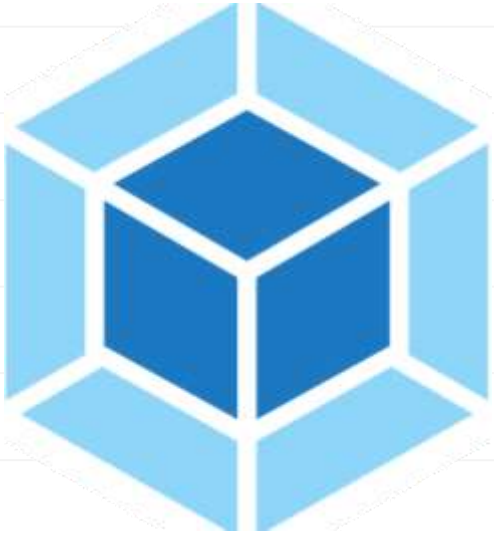
```
export const getAPIWrapper = async () => {
    var SpotifyWebApi = require('spotify-web-api-js');

    //On n'oublie pas de refresh les tokens si nécéssaires
    await checkAndRefreshTokens();
    const accessToken = await getAccessToken();

    //On crée une nouvelle instance du wrapper à laquelle on donne les tokens
    let sp = new SpotifyWebApi();
    await sp.setAccessToken(accessToken);
    return sp;
}
```

# Back-end : Utilisation DE l’API

- Récupération de l’utilisateur actuel

```
export const getCurrentUser = async() =>
{
    const apiWrapper = await getAPIWrapper();
    const apiResponse = await apiWrapper.getMe();
    return apiResponse;
}
```

# Adaptation au web : Comportement platform-specific



.native.js

Webpack

Metro

# Adaptation au web : Stockage

1. Création d’un nouveau dataStore.jsR
2. Installation de local-storage
3. Recréation des méthodes avec local storage

# Adaptation au web : Connexion à l'API

- Refonte du login

```
//Retrieves the authorization codes to have access to the spotify API
export const loginToSpotify = async () =>
{
    try
    {
        //Récupération des credentials
        const credentials = getSpotifyCredentials();

        //Création de l'url de requête
        let authUrl =   'https://accounts.spotify.com/authorize' +
                        '?response_type=code' +
                        '&client_id=' +
                        credentials.clientId +
                        (scopes ? '&scope=' + encodeURIComponent(scopes) : '') +
                        '&redirect_uri=' +
                        encodeURIComponent(credentials.redirectUri);

        //Redirection directe vers Spotify
        window.location.href = authUrl;
    }
    catch (err)
    {
        console.error(err)
```

# Adaptation au web : Connexion à l'API

- Vérification à la redirection

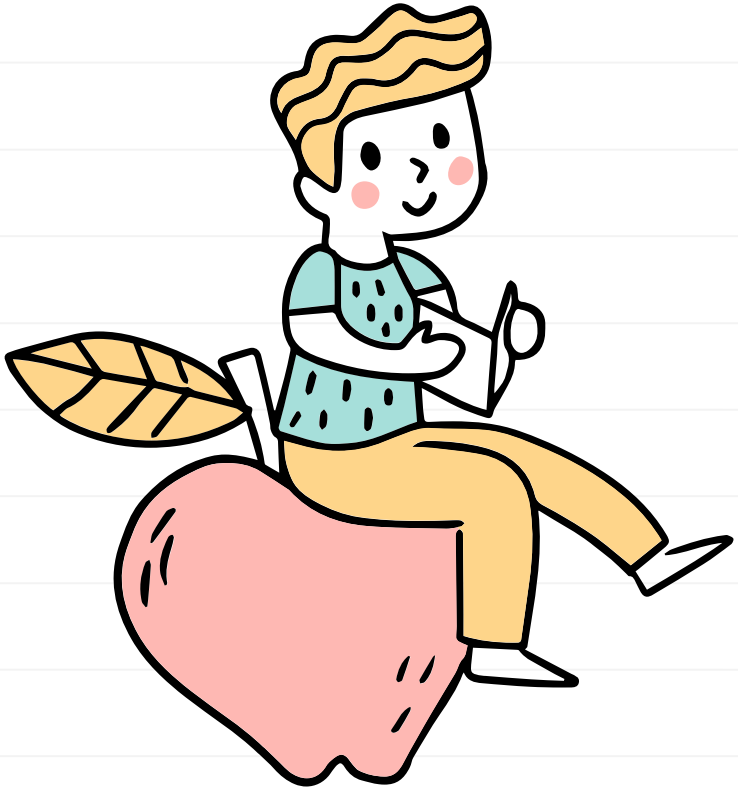
```
async componentDidMount()
{
    //Méthode de AuthUtils
    const checkResult = await loginScreenCheck();
    if(checkResult)
    {
        LoginScreen.checkIfConnected();
    }
}
```

# Adaptation au web : Connexion à l'API

- Vérification à la redirection

```
export const loginScreenCheck = async() => {
    //On récupère le code
    let code = window.location.search.substring(6);
    if (code) {
        storeData("authorization_code",code);
        const result = await refreshTokens();

        //On redirige vers la home
        if(result)
        {
            NavigatorRef.replace('Home');
        }
        return true;
    }

    return false;
}
```

# Merci pour votre écoute !



# Références


- [1] « @expo/vector-icons directory ». [En ligne]. Disponible sur: https://expo.github.io/vector-icons/. [Consulté le: 20-févr-2020].
- [2] « Getting Started · React Native ». [En ligne]. Disponible sur: https://facebook.github.io/react-native/. [Consulté le: 20-févr-2020].
- [3] « Getting Started · React Native Elements ». [En ligne]. Disponible sur: https://react-native-elements.github.io//react-native-elements/index.html. [Consulté le: 20-févr-2020].
- [4] « Getting started · React Navigation ». [En ligne]. Disponible sur: https://reactnavigation.org/index.html. [Consulté le: 20-févr-2020].
- [5] « Quelle sont les différences entre React Native et React.JS ? ⋆ Ambient Formations ». [En ligne]. Disponible sur: https://www.ambient-it.net/quelle-sont-les-differences-entre-react-native-et-reactjs/. [Consulté le: 20-févr-2020].
- [6] « AsyncStorage · React Native ». [En ligne]. Disponible sur:Disponible sur: https://facebook.github.io/react-native/. [Consulté le: 20-févr-2020].
- [7] « AuthSession - Expo Documentation ». [En ligne]. Disponible sur: https://docs.expo.io/versions/latest/sdk/auth-session/. [Consulté le: 20-févr-2020].
- [8] Bevacqua, Nicolás. bevacqua/local-storage. JavaScript, 2020.[En ligne]. Disponible sur: https://github.com/bevacqua/local-storage. [Consulté le: 20-févr-2020].
- [9] « Home | Spotify for Developers ». [En ligne]. Disponible sur: https://developer.spotify.com/. [Consulté le: 20-févr-2020].
- [10] Pérez, José M. JMPerez/spotify-web-api-js. TypeScript, 2020. [En ligne]. Disponible sur: https://github.com/JMPerez/spotify-web-api-js. [Consulté le: 20-févr-2020].