```
//Librairies
import React from 'react';
import { NavigationContainer } from '@react-navigation/native';
//Redux et Persist
import { Provider } from 'react-redux';
import { persistStore } from 'redux-persist';
import { PersistGate } from 'redux-persist/es/integration/react';
import Store from './Store/configureStore.js';
//Components perso
import Index from './Components/Index.js';

class App extends React.Component {
    render() {
        let persistor = persistStore(Store);
        return (
            <Provider store={Store}>
                <PersistGate persistor={persistor}>
                    <NavigationContainer>
                        <Index/>
                    </NavigationContainer>
                </PersistGate>
            </Provider>
        )
    }
}

//Export du component
export default App;
```

```
//Librairies
import { createStore } from 'redux';
import { persistCombineReducers } from 'redux-persist';
import AsyncStorage from '@react-native-community/async-storage';
//Components perso
import isConnected from './Reducers/connected.js';

//Configuration
const rootPersistConfig = {
    key: 'root',
    storage: AsyncStorage
}

//Export du component
export default createStore(persistCombineReducers(rootPersistConfig,
{isConnected}))
```

```
//State initial
const initialState = { idUser: false }

//Connecter ou déconnecter l'utilisateur
function isConnected(state = initialState, action) {
    let nextState;
    switch (action.type) {
        case 'IS_CONNECTED':
            //Déconnecter l'user
            if(action.value === false){
                nextState = {
                    ...state,
                    idUser: false
                }
            }else{
                //Connecter l'utilisateur
                nextState = {
                    ...state,
                    idUser: action.value
                }
            }
            return nextState || state
        default:
        return state
    }
}

//Export du component
export default isConnected
```

```
//Librairies
import React, { Component } from 'react';
import { View } from 'react-native';
//Components perso
import { BASE_STYLE, Loading, ImageList } from './MyComponent.js';
//Fonctions
import { getFav } from '../WS/functions.js';
//Redux
import { connect } from 'react-redux';

class Fav extends Component {
    constructor(props) {
        super(props);
        //State
        this.state = {
            wallpapers: undefined,
            isLoading: true,
            refreshing: false
        };
        //Récupérer les favoris
        getFav(this.props.idUser).then(data => this.setState({
            wallpapers: data,
            isLoading: false
        }));
    }

    //Quand la liste est rafraîchit
    _onRefresh = () => {
        this.setState({refreshing: true});
        //Récupérer les favoris
        getFav(this.props.idUser).then(data => this.setState({
            wallpapers: data,
            refreshing: false
        }));
    }

    render(){
        return (
            <View style={BASE_STYLE.container}>
                {/* Affichage des fonds d'écrans */}
                <ImageList
                data={this.state.wallpapers}
                onPress={(item) =>
this.props.navigation.navigate('WallpaperDetails', {wallpaper: item})}
                refreshing={this.state.refreshing}
                onRefresh={() => this._onRefresh()}
                />
                {/* Affichage du chargement */}
                {this.state.isLoading && <Loading/>}
            </View>
        );
    }
}

//Pour Redux
```

```
//Pour Redux
const mapStateToProps = (state) => {
    return {
        idUser: state.isConnected.idUser
    }
}

//Export du component
export default connect(mapStateToProps)(Fav);
```

```
//Librairies
import { Dimensions } from 'react-native';
import { createStackNavigator } from '@react-navigation/stack';
import { createBottomTabNavigator } from '@react-navigation/bottom-tabs';

//Constantes

//Liens
const URL_WS = 'https://flobeney.ch/wpheaven/ws/';
const URL_API = 'https://wallhaven.cc/api/v1/';

//Dimensions de l'appareil
export const HEIGHT_SCREEN = Dimensions.get("screen").height;
export const WIDTH_SCREEN = Dimensions.get("screen").width;
const PGCD_SCREEN_DIMENSIONS = pgcd(HEIGHT_SCREEN, WIDTH_SCREEN);
//Ratio de l'appareil
const RATIO = (WIDTH_SCREEN / PGCD_SCREEN_DIMENSIONS).toString() + 'x' +
(HEIGHT_SCREEN / PGCD_SCREEN_DIMENSIONS).toString();

//Constantes couleurs des onglets lors du focus ou non
export const COLOR_NOT_FOCUSED = '#b7b7b7';
export const COLOR_FOCUSED = '#000';

//Messages d'erreur
export const TEXT_FILL_ALL = 'Please fill in all values';

//Navigateur "Stack" (écran qui peuvent s'empiler)
export const Stack = createStackNavigator();
//Navigateur "Tab" (onglet en bas de l'application)
export const Tab = createBottomTabNavigator();

// Données de recherches
const CATEGORY_GENERAL = "General";
const CATEGORY_ANIME = "Anime";
const CATEGORY_PEOPLE = "People";
export const CATEGORIES = [
    {
        name: CATEGORY_GENERAL,
        value: 0
    },
    {
        name: CATEGORY_ANIME,
        value: 0
    },
    {
        name: CATEGORY_PEOPLE,
        value: 0
    }
];

const SORTING_DATE = 'date_added';
const SORTING_RELEVANCE = 'relevance';
const SORTING_RANDOM = 'random';
const SORTING_VIEWS = 'views';
const SORTING_FAVORITES = 'favorites';
```

```
const SORTING_TOPLIST = 'toplist';
export const SORTING = [
    SORTING_DATE,
    SORTING_RELEVANCE,
    SORTING_RANDOM,
    SORTING_VIEWS,
    SORTING_FAVORITES,
    SORTING_TOPLIST
];
export const ORDER_DESC = 'desc';
export const ORDER_ASC = 'asc';

//Fonctions

//Calcul du PGCD
function pgcd(nbOne, nbTwo){
    //Si le 2ème chiffre est 0 (division impossible)
    if(nbTwo == 0){
        return Math.abs(nbOne);
    }
    var temp = nbOne % nbTwo;
    if(temp == 0){
        //Un pgcd n'a pas de signe
        return Math.abs(nbTwo);
    }
    return pgcd(nbTwo, temp);
}

//Vérifier que tout les éléments d'un objet sont différents de null
export function nothingEmpty(obj){
    var res = true;
    Object.values(obj).forEach(function(item){
        if(item == ''){
            res = false;
        }
    })

    return res;
}

//Trimer toutes les valeurs d'un objet
export function trimValues(obj){
    const res = Object.fromEntries(
        Object.entries(obj)
        .map(([ key, val ]) => [ key, val.trim() ])
    );
    return res;
}

//Wallhaven

//Récupérer les derniers fonds d'écran en fonction du ratio de l'écran de
l'utilisateur
export function getLastWP(){
    //Lien
```

```
    //Lien
    var link = URL_API + 'search?ratios=' + RATIO;
    //Appel fetch
    return fetch(link, {
        method: 'GET',
        headers: {
            Accept: 'application/json', //type de retour
        }
    })
    .then((response) => response.json())
    .catch((error) => console.error(error));
}

//Récupérer les fonds d'écran similaires en fonction du ratio de l'écran de l'utilisateur
export function getSimilarWP(id){
    //Lien
    var link = URL_API + 'search?q=like:' + id + '&ratios=' + RATIO;
    //Appel fetch
    return fetch(link, {
        method: 'GET',
        headers: {
            Accept: 'application/json', //type de retour
        }
    })
    .then((response) => response.json())
    .catch((error) => console.error(error));
}

//Recherche de fond d'écran
export function searchWP(params, page = 1){
    //lien
    var link = setAPISearchLink(params, page);
    //Appel fetch
    return fetch(link, {
        method: 'GET',
        headers: {
            Accept: 'application/json', //type de retour
        }
    })
    .then((response) => response.json())
    .catch((error) => console.error(error));
}

//Return the link of the search
function setAPISearchLink(params, page){
    var link = URL_API + 'search?page=' + page;
    if(params.color != null){
        link += '&colors=' + params.color;
    }
    if(params.category != null){
        link += '&categories=' + params.category;
    }
    if(params.order != null){
        link+= '&order='+ params.order;
    }
```

```
    }
    if(params.sorting != null){
        link += '&sorting=' + params.sorting;
    }
    if(params.owner != null || params.tag != null){
        link += '&q=';
        if(params.tag != null){
            link += params.tag;
        }
        if(params.owner != null){
            link += '@' + params.owner;
        }
    }
    return link + '&ratios=' + RATIO;
}

//SQL

//Utilisateur (connexion, inscription)

//Connexion
export function login(user){
    //Lien
    var link = URL_WS + 'login.php';
    //FormData
    var formData = new FormData();
    //Ajouter les données
    formData.append('pseudo', user.pseudo);
    formData.append('pwd', user.pwd);
    //Appel fetch
    return fetch(link, {
        method: 'POST',
        headers: {
            Accept: 'application/json', //type de retour
            'Content-Type': 'multipart/form-data' //type d'envoie
        },
        body: formData, //Requete
    })
    .then((response) => response.json())
    .catch((error) => console.error(error));
}

//Inscription
export function signUp(user){
    //Lien
    var link = URL_WS + 'signUp.php';
    //FormData
    var formData = new FormData();
    //Ajouter les données
    formData.append('pseudo', user.pseudo);
    formData.append('email', user.email);
    formData.append('pwd', user.pwd);
    //Appel fetch
    return fetch(link, {
        method: 'POST',
```

```
        headers: {
            Accept: 'application/json', //type de retour
            'Content-Type': 'multipart/form-data' //type d'envoie
        },
        body: formData, //Requete
    })
    .then((response) => response.json())
    .catch((error) => console.error(error));
}

//Favoris

//Récupérer les favoris de l'utilisateur
export function getFav(idUser){
    //Lien
    var link = URL_WS + 'getFav.php';
    //FormData
    var formData = new FormData();
    //Ajouter les données
    formData.append('idUser', idUser);
    //Appel fetch
    return fetch(link, {
        method: 'POST',
        headers: {
            Accept: 'application/json', //type de retour
            'Content-Type': 'multipart/form-data' //type d'envoie
        },
        body: formData, //Requete
    })
    .then((response) => response.json())
    .catch((error) => console.error(error));
}

//Checker si le favoris existe
export function checkFav(fav){
    //Lien
    var link = URL_WS + 'checkFav.php';
    //FormData
    var formData = new FormData();
    //Ajouter les données
    formData.append('idUser', fav.idUser);
    formData.append('idImage', fav.idImage);
    //Appel fetch
    return fetch(link, {
        method: 'POST',
        headers: {
            Accept: 'application/json', //type de retour
            'Content-Type': 'multipart/form-data' //type d'envoie
        },
        body: formData, //Requete
    })
    .then((response) => response.json())
    .catch((error) => console.error(error));
}
```

```
//Toggle le favoris
export function toggleFav(fav){
    //Lien
    var link = URL_WS + 'toggleFav.php';
    //FormData
    var formData = new FormData();
    //Ajouter les données
    formData.append('idUser', fav.idUser);
    formData.append('idImage', fav.idImage);
    //Appel fetch
    return fetch(link, {
        method: 'POST',
        headers: {
            Accept: 'application/json', //type de retour
            'Content-Type': 'multipart/form-data' //type d'envoie
        },
        body: formData, //Requete
    })
    .then((response) => response.json())
    .catch((error) => console.error(error));
}
```

```
//Librairies
import React, { Component } from 'react';
import { View } from 'react-native';
//Components perso
import { BASE_STYLE, Loading, ImageList } from './MyComponent.js';
//Fonctions
import { getLastWP } from '../WS/functions.js';
//Redux
import { connect } from 'react-redux';

class Home extends Component {
    constructor(props) {
        super(props);
        //State
        this.state = {
            wallpapers: undefined,
            isLoading: true,
            refreshing: false
        };
        //Récupérer les images
        getLastWP().then(data => this.setState({
            wallpapers: data.data,
            isLoading: false
        }));
    }

    //Quand la liste est rafraîchit
    _onRefresh = () => {
        this.setState({refreshing: true});
        //Récupérer les images
        getLastWP().then(data => this.setState({
            wallpapers: data.data,
            refreshing: false
        }));
    }

    render(){
        return (
            <View style={BASE_STYLE.container}>
                {/* Affichage des fonds d'écrans */}
                <ImageList
                data={this.state.wallpapers}
                onPress={(item) =>
this.props.navigation.navigate('WallpaperDetails', {wallpaper: item})}
                refreshing={this.state.refreshing}
                onRefresh={() => this._onRefresh()}
                />
                {/* Affichage du chargement */}
                {this.state.isLoading && <Loading/>}
            </View>
        );
    }
}

//Pour Redux
```

```
//Pour Redux
const mapStateToProps = (state) => {
    return {
        idUser: state.isConnected.idUser
    }
}

//Export du component
export default connect(mapStateToProps)(Home);
```

```
//Librairies
import React from 'react';
//Components perso
import NavLogin from '../Navigation/NavLogin.js';
import NavLogged from '../Navigation/NavLogged.js';
//Redux
import { connect } from 'react-redux';

class Index extends React.Component {
    render() {
        return (
            (this.props.idUser === false ? <NavLogin/> : <NavLogged/>)
        )
    }
}

//Pour Redux
const mapStateToProps = (state) => {
    return {
        idUser: state.isConnected.idUser
    }
}

//Export du component
export default connect(mapStateToProps)(Index);

```

```
//Librairies
import React, { Component } from 'react';
import { View, Button, Alert } from 'react-native';
import { Input } from 'react-native-elements';
import { KeyboardAwareScrollView } from 'react-native-keyboard-aware-scroll-
view';
//Components perso
import { BASE_STYLE, Loading } from './MyComponent.js';
//Fonctions
import { login, trimValues, nothingEmpty, TEXT_FILL_ALL } from
'../WS/functions.js';
//Redux
import { connect } from 'react-redux';

class Login extends Component {
    constructor(props) {
        super(props);
        //Variables
        this.user = {
            pseudo: '',
            pwd: '',
        }
        //State
        this.state = {
            isLoading: false
        };
    }

    //Login
    _login(){
        //Trim
        this.user = trimValues(this.user);
        //Aucun champs vide
        if(nothingEmpty(this.user)){
            //Lancer le chargement
            this.setState({
                isLoading: true
            })
            //Login
            login(this.user).then((data) => {
                //Fin du chargement
                this.setState({
                    isLoading: false
                });
                //Si la connexion est valide
                if(data !== false){
                    const action = { type: "IS_CONNECTED", value: data };
                    this.props.dispatch(action);
                }else{ //Connexion invalide
                    //Envoie un message d'alerte
                    Alert.alert(
                        "Information",
                        "Invalid connection !",
                        [{text:"Ok"}]
                    );
```

```
                    '',

                }
            });
        }else{
            //Envoie un message d'alerte
            Alert.alert(
                "Information",
                TEXT_FILL_ALL,
                [{text:"Ok"}]
            );
        }

    }

    render(){
        return (
            <View style={BASE_STYLE.container}>
                <KeyboardAwareScrollView
                style={BASE_STYLE.scrollview_container}
                contentContainerStyle={BASE_STYLE.scrollview_content}
                >
                    {/* Pseudo */}
                    <Input
                    placeholder='Pseudo'
                    onChangeText={(value) => this.user.pseudo = value}
                    />
                    {/* Mot de passe */}
                    <Input
                    secureTextEntry={true}
                    placeholder='Password'
                    onChangeText={(value) => this.user.pwd = value}
                    />
                    <Button
                    title='Login'
                    onPress={() => this._login()}
                    />
                    <Button
                    title='Sign up'
                    onPress={() => this.props.navigation.navigate('SignUp')}
                    />
                </KeyboardAwareScrollView>
                {/* Affichage du chargement */}
                {this.state.isLoading && <Loading/>}
            </View>
        );
    }
}

//Pour Redux
const mapStateToProps = (state) => {
    return {
        idUser: state.isConnected.idUser
    }
}
```

```
109 //Export du component
110 export default connect(mapStateToProps)(Login);
```

```
//Librairies
import React, { Component } from 'react';
import { View, Button, Alert } from 'react-native';
//Components perso
import { BASE_STYLE } from './MyComponent.js';
//Redux
import { connect } from 'react-redux';

class Logout extends Component {
    constructor(props) {
        super(props);
        //State
        this.state = {
            isLoading: false
        };
    }

    //Avertir avant de déconnecter
    _logout(){
        Alert.alert(
            'Warning',
            'Are you sure you want to logout ?',
            [
                {
                    text: 'Cancel',
                    style: 'cancel'
                },
                {
                    text: 'Logout',
                    onPress: () => {
                        const action = { type: "IS_CONNECTED", value: false };
                        this.props.dispatch(action);
                    }
                }
            ]
        );
    }

    render(){
        return (
            <View style={BASE_STYLE.container}>
                <Button
                title='Logout'
                onPress={() => this._logout()}
                />
            </View>
        );
    }
}

//Pour Redux
const mapStateToProps = (state) => {
    return {
        idUser: state.isConnected.idUser
    }
```

```
}

//Export du component
export default connect(mapStateToProps)(Logout);
```

```
//Librairies
import React from 'react';
import { StyleSheet, ActivityIndicator, View, FlatList, TouchableOpacity,
Image, RefreshControl } from 'react-native';
//Constantes
import { WIDTH_SCREEN, HEIGHT_SCREEN } from '../WS/functions.js';

//Style utilisé sur plusieurs pages
export const BASE_STYLE = StyleSheet.create({
    //Container
    container: {
        flex: 1,
        backgroundColor: '#fff'
    },
    center_absolute: {
        flex: 1,
        position: 'absolute',
        alignItems: 'center',
        justifyContent: 'center',
        top: 0,
        left: 0,
        right: 0,
        bottom: 0,
    },
    scrollview_container: {
        flex: 1,
    },
    scrollview_content: {
        alignItems: 'center',
        margin: 15
    },
    //Images
    img_left: {
        width: (WIDTH_SCREEN / 2) - (WIDTH_SCREEN / 100),
        height: (HEIGHT_SCREEN / 2) - (HEIGHT_SCREEN / 100),
        marginRight: (WIDTH_SCREEN / 100),
        marginBottom: (HEIGHT_SCREEN / 100)
    },
    img_right: {
        width: (WIDTH_SCREEN / 2) - (WIDTH_SCREEN / 100),
        height: (HEIGHT_SCREEN / 2) - (HEIGHT_SCREEN / 100),
        marginLeft: (WIDTH_SCREEN / 100),
        marginBottom: (HEIGHT_SCREEN / 100)
    },
    img_full: {
        width: WIDTH_SCREEN,
        height: HEIGHT_SCREEN,
    },
    //Informations sur le wallpaper
    infos_wallpaper: {
        margin: 15
    },
    //Textes
    text_link: {
        fontWeight: 'bold',
```

```
        fontWeight: 'bold',
        textDecorationLine: 'underline'
    },
    text_subtitle: {
        margin: 15,
        fontSize: 20,
        fontWeight: 'bold'
    }
});

//Components

//Affichage d'une icône de chargement
export function Loading(){
    return(
        <View style={BASE_STYLE.center_absolute}>
            <ActivityIndicator size='large'/>
        </View>
    );
}

//Affichage de plusieurs images en liste
export function ImageList(props){
    //Si les données ne sont pas undefined
    if(props.data != undefined){
        return(
            <FlatList
            style={{flex: 1}}
            data={props.data}
            refreshControl={
                <RefreshControl
                refreshing={props.refreshing}
                onRefresh={props.onRefresh}
                />
            }
            renderItem={({item, index}) => (
                <TouchableOpacity
                onPress={() => props.onPress(item)}
                >
                    <Image
                    style={index % 2 == 0 ? BASE_STYLE.img_left :
BASE_STYLE.img_right}
                    source={{uri: item.thumbs.original}}
                    />
                </TouchableOpacity>
            )}
            numColumns={2}
            />
        );
    }else{
        return null;
    }
}
```

```
//Librairies
import React from 'react';
import { Entypo, MaterialCommunityIcons, Ionicons } from 'react-native-vector-icons';
//Components perso
import { HomeStack } from './SharedNav.js';
import Fav from '../Components/Fav.js';
import Search from '../Components/Search.js';
import SearchResult from '../Components/SearchResult.js';
import WallpaperDetails from '../Components/WallpaperDetails.js';
import Logout from '../Components/Logout.js';
//Constantes
import { COLOR_FOCUSED, COLOR_NOT_FOCUSED, Stack, Tab } from "../WS/functions.js";

//Favoris
function FavStack(){
    return(
        <Stack.Navigator>
            <Stack.Screen name="Favorites" component={Fav} />
            <Stack.Screen
            name="WallpaperDetails"
            options={{title: 'Details', headerBackTitle: 'Back'}}
            component={WallpaperDetails} />
        </Stack.Navigator>
    );
}

//Recherche
function SearchStack(){
    return(
        <Stack.Navigator>
            <Stack.Screen name="Search" component={Search} />
            <Stack.Screen
            name="SearchResult"
            options={{title: 'Search result', headerBackTitle: 'Back'}}
            component={SearchResult} />
            <Stack.Screen
            name="WallpaperDetails"
            options={{title: 'Details', headerBackTitle: 'Back'}}
            component={WallpaperDetails} />
        </Stack.Navigator>
    );
}

//Logout
function LogoutStack(){
    return(
        <Stack.Navigator>
            <Stack.Screen name="Logout" component={Logout} />
        </Stack.Navigator>
    );
}

//Navigation
```

```
//Navigation
function NavLogged(){
    return(
        <Tab.Navigator>
            {/* Page d'accueil */}
            <Tab.Screen
            name="Home"
            component={HomeStack}
            options={{
                tabBarIcon: ({focused}) => {
                    return(
                        <Entypo
                        name="home"
                        size={25}
                        color={focused ? COLOR_FOCUSED : COLOR_NOT_FOCUSED}
                        />
                    )
                }
            }}
            />
            {/* Favoris */}
            <Tab.Screen
            name="Favorites"
            component={FavStack}
            options={{
                tabBarIcon: ({focused}) => {
                    return(
                        <Entypo
                        name="star"
                        size={25}
                        color={focused ? COLOR_FOCUSED : COLOR_NOT_FOCUSED}
                        />
                    )
                }
            }}
            />
            {/* Recherche */}
            <Tab.Screen
            name="Search"
            component={SearchStack}
            options={{
                tabBarIcon: ({focused}) => {
                    return(
                        <Ionicons
                        name="ios-search"
                        size={25}
                        color={focused ? COLOR_FOCUSED : COLOR_NOT_FOCUSED}
                        />
                    )
                }
            }}
            />
            {/* Logout */}
            <Tab.Screen
            name="Logout"
            component={LogoutStack}
```

```
            component={LogoutStack}
            options={{
                tabBarIcon: ({focused}) => {
                    return(
                        <MaterialCommunityIcons
                        name="logout"
                        size={25}
                        color={focused ? COLOR_FOCUSED : COLOR_NOT_FOCUSED}
                        />
                    )
                }
            }}
            />
        </Tab.Navigator>
    );
}

//Export du component
export default NavLogged;
```

```
//Librairies
import React from 'react';
import { Entypo } from 'react-native-vector-icons';
//Components perso
import { HomeStack } from './SharedNav.js';
import Login from '../Components/Login.js';
import SignUp from '../Components/SignUp.js';
//Constantes
import { COLOR_FOCUSED, COLOR_NOT_FOCUSED, Stack, Tab } from
"../WS/functions.js";

//Login
function LoginStack(){
    return(
        <Stack.Navigator>
            <Stack.Screen name="Login" component={Login} />
            <Stack.Screen
            name="SignUp"
            options={{title: 'Sign Up'}}
            component={SignUp} />
        </Stack.Navigator>
    );
}

//Navigation
function NavLogin(){
    return(
        <Tab.Navigator>
            <Tab.Screen
            name="Home"
            component={HomeStack}
            options={{
                tabBarIcon: ({focused}) => {
                    return(
                        <Entypo
                        name="home"
                        size={25}
                        color={focused ? COLOR_FOCUSED : COLOR_NOT_FOCUSED}
                        />
                    )
                }
            }}
            />
            <Tab.Screen
            name="Login"
            component={LoginStack}
            options={{
                tabBarIcon: ({focused}) => {
                    return(
                        <Entypo
                        name="login"
                        size={25}
                        color={focused ? COLOR_FOCUSED : COLOR_NOT_FOCUSED}
                        />
                    )
```

```
                    }
                }}
                />
            </Tab.Navigator>
        );
}

//Export du component
export default NavLogin;
```

```
//Librairies
import React, { Component } from 'react';
import { Text, View, Button, TouchableOpacity } from 'react-native';
import { Input } from 'react-native-elements';
import { KeyboardAwareScrollView } from 'react-native-keyboard-aware-scroll-view';
//Components perso
import { BASE_STYLE, Loading } from './MyComponent.js';
//Fonctions
import { CATEGORIES, ORDER_DESC, ORDER_ASC, SORTING } from '../WS/functions.js';
//Redux
import { connect } from 'react-redux';

class Search extends Component {
    constructor(props) {
        super(props);
        //Recherche
        this.search = {
            tag: null,
            owner: null,
            color: null,
        }
        //State
        this.state = {
            isLoading: false,
            category: CATEGORIES,
            order: ORDER_DESC,
            sorting: null
        };
    }

    //Effectuer la recherche
    _search(){
        //Récupérer les valeurs des catégories
        var categories = '';
        this.state.category.map(categorie => categories += categorie.value);

        //Page suivante
        this.props.navigation.navigate('SearchResult', {search: {...this.search, category: categories, order: this.state.order, sorting: this.state.sorting}})
    }

    render(){
        return (
            <View style={BASE_STYLE.container}>
                <KeyboardAwareScrollView
                style={BASE_STYLE.scrollview_container}
                contentContainerStyle={BASE_STYLE.scrollview_content}
                >
                    {/* Catégories de recherches */}
                    <Text style={BASE_STYLE.text_subtitle}>Categories</Text>
                    <View style={{flexDirection: 'row'}}>
                        {this.state.category.map(categorie => (
```

```
                        {this.state.category.map(categorie => (
                            <TouchableOpacity
                            key={categorie.name}
                            style={{flex: 1}}
                            onPress={() => {
                                //Copier les catégories dans la variable de résultat
                                var res = [...this.state.category];
                                //Trouver l'élément clické
                                var indexElem = res.findIndex(elem => elem.name == categorie.name);
                                //Inverser sa valeur
                                res[indexElem].value = (res[indexElem].value == 0 ? 1 : 0);

                                //Mettre à jour la valeur
                                this.setState({
                                    category: res
                                });
                            }}
                            >
                                <Text style={{textAlign: 'center', fontWeight: (categorie.value == 1 ? 'bold' : 'normal')}}>{categorie.name}</Text>
                            </TouchableOpacity>
                        ))}
                    </View>
                    {/* Sorting */}
                    <Text style={BASE_STYLE.text_subtitle}>Sorting</Text>
                    {SORTING.map(typeSorting => (
                        <TouchableOpacity
                        key={typeSorting}
                        style={{marginVertical: 5}}
                        onPress={() => this.setState({sorting: typeSorting})}
                        >
                            <Text style={{textAlign: 'center', fontWeight: (typeSorting == this.state.sorting ? 'bold' : 'normal')}}>{typeSorting}</Text>
                        </TouchableOpacity>
                    ))}
                    {/* Order */}
                    <Text style={BASE_STYLE.text_subtitle}>Order</Text>
                    <TouchableOpacity
                    onPress={() => this.setState({
                        order: (this.state.order == ORDER_DESC ? ORDER_ASC : ORDER_DESC)
                    })}
                    >
                        <Text>{this.state.order}</Text>
                    </TouchableOpacity>

                    {/* Tag */}
                    <Input
                    placeholder='Tag'
                    onChangeText={(value) => this.search.tag = value}
                    autoCapitalize='none'
```

```
                    autoCapitalize='none'
                    autoCorrect={false}
                    />
                    <Text>Which tag are you looking for ?</Text>
                    {/* Owner */}
                    <Input
                    placeholder='Owner'
                    onChangeText={(value) => this.search.owner = value}
                    autoCapitalize='none'
                    autoCorrect={false}
                    />
                    <Text>Owner of the wallpaper</Text>
                    {/* Color */}
                    <Input
                    placeholder='Color'
                    onChangeText={(value) => this.search.color = value}
                    autoCapitalize='none'
                    autoCorrect={false}
                    />
                    <Text>Hexa value of the color without the #</Text>
                    {/* Recherche */}
                    <Button
                    title='Search'
                    onPress={() => this._search()}
                    />
                </KeyboardAwareScrollView>
                {/* Affichage du chargement */}
                {this.state.isLoading && <Loading/>}
            </View>
        );
    }
}

//Pour Redux
const mapStateToProps = (state) => {
    return {
        idUser: state.isConnected.idUser
    }
}

//Export du component
export default connect(mapStateToProps)(Search);
```

```
//Librairies
import React, { Component } from 'react';
import { Text, View, TouchableOpacity } from 'react-native';
import { Entypo } from 'react-native-vector-icons';
//Components perso
import { BASE_STYLE, Loading, ImageList } from './MyComponent.js';
//Fonctions
import { searchWP } from '../WS/functions.js';
//Redux
import { connect } from 'react-redux';

const SIZE_CHEVRON = 30;

class SearchResult extends Component {
    constructor(props) {
        super(props);
        //State
        this.state = {
            wallpapers: undefined,
            isLoading: true,
            refreshing: false,
            currentPage: 0,
            lastPage: 0,
        };
        //Récupérer les images
        searchWP(this.props.route.params.search).then(data => this.setState({
            wallpapers: data.data,
            currentPage: data.meta.current_page,
            lastPage: data.meta.last_page,
            isLoading: false
        }));
    }

    //Quand la liste est rafraîchit
    _onRefresh = () => {
        this.setState({refreshing: true});
        //Récupérer les images
        searchWP(this.props.route.params.search,
this.state.currentPage).then(data => this.setState({
            wallpapers: data.data,
            currentPage: data.meta.current_page,
            lastPage: data.meta.last_page,
            refreshing: false
        }));
    }

    //Changer de page
    _changePage(page){
        //Si la page demandé reste dans les limites
        if(page > 0 && page <= this.state.lastPage){
            this.setState({isLoading: true});
            //Récupérer les images
            searchWP(this.props.route.params.search, page).then(data =>
this.setState({
                wallpapers: data.data,
```

```
                wallpapers: data.data,
                currentPage: data.meta.current_page,
                lastPage: data.meta.last_page,
                isLoading: false
            }));
        }
    }

    render(){
        return (
            <View style={BASE_STYLE.container}>
                {/* Affichage des résultats, ou d'un message indiquant qu'il
n'y a pas de résultat */}
                {(this.state.wallpapers != undefined &&
this.state.wallpapers.length == 0) ?
                    <View style={{flex: 1, justifyContent: 'center',
alignItems: 'center'}}>
                        <Text>There's nothing here...</Text>
                    </View>
                    :
                    <ImageList
                    data={this.state.wallpapers}
                    onPress={(item) =>
this.props.navigation.navigate('WallpaperDetails', {wallpaper: item})}
                    refreshing={this.state.refreshing}
                    onRefresh={() => this._onRefresh()}
                    />
                }
                {/* Pagination */}
                <View style={{flexDirection: 'row', justifyContent: 'space-
around', alignItems: 'center'}}>
                    {/* Page précédente */}
                    <TouchableOpacity
                    onPress={() => this._changePage(this.state.currentPage -
1)}
                    >
                        <Entypo
                        name="chevron-thin-left"
                        size={SIZE_CHEVRON}
                        />
                    </TouchableOpacity>
                    {/* Page actuelle */}
                    <Text>
                        Current page:
{this.state.currentPage}/{this.state.lastPage}
                    </Text>
                    {/* Page suivante */}
                    <TouchableOpacity
                    onPress={() => this._changePage(this.state.currentPage +
1)}
                    >
                        <Entypo
                        name="chevron-thin-right"
                        size={SIZE_CHEVRON}
                        />
                    </TouchableOpacity>
```

```
                    </TouchableOpacity>
                </View>
                {/* Affichage du chargement */}
                {this.state.isLoading && <Loading/>}
            </View>
        );
    }
}

//Pour Redux
const mapStateToProps = (state) => {
    return {
        idUser: state.isConnected.idUser
    }
}

//Export du component
export default connect(mapStateToProps)(SearchResult);
```

```
//Librairies
import React from 'react';
//Components perso
import Home from '../Components/Home.js';
import WallpaperDetails from '../Components/WallpaperDetails.js';
//Constantes
import { Stack } from "../WS/functions.js";

//Home
export function HomeStack(){
    return(
        <Stack.Navigator>
            <Stack.Screen name="Home" component={Home} />
            <Stack.Screen
            name="WallpaperDetails"
            options={{title: 'Details', headerBackTitle: 'Back'}}
            component={WallpaperDetails} />
        </Stack.Navigator>
    );
}
```

```
//Librairies
import React, { Component } from 'react';
import { View, Button, Alert } from 'react-native';
import { Input } from 'react-native-elements';
import { KeyboardAwareScrollView } from 'react-native-keyboard-aware-scroll-
view';
//Components perso
import { BASE_STYLE, Loading } from './MyComponent.js';
//Fonctions
import { signUp, trimValues, nothingEmpty, TEXT_FILL_ALL } from
'../WS/functions.js';
//Redux
import { connect } from 'react-redux';

class SignUp extends Component {
    constructor(props) {
        super(props);
        //Variables
        this.user = {
            pseudo: '',
            email: '',
            pwd: '',
        }
        //State
        this.state = {
            isLoading: false
        };
    }

    //Inscription
    _signUp(){
        //Trim
        this.user = trimValues(this.user);
        //Aucun champs vide
        if(nothingEmpty(this.user)){
            //Lancer le chargement
            this.setState({
                isLoading: true
            });
            //Inscription
            signUp(this.user).then((data) => {
                //Fin du chargement
                this.setState({
                    isLoading: false
                });
                //Si l'inscription est valide
                if(data.done === true){
                    //Envoie un message d'alerte
                    Alert.alert(
                        "Information",
                        "Successful sign up",
                        [{
                            text:"Ok",
                            onPress: () => {
                                const action = { type: "IS_CONNECTED", value:
```

```
                                const action = { type: 'IS_CONNECTED', value:
data.msg };
                                this.props.dispatch(action);
                            }
                        }]
                    );
                }else{ //Inscription invalide
                    //Envoie un message d'alerte
                    Alert.alert(
                        "Information",
                        data.msg,
                        [{text:"Ok"}]
                    );
                }
            });
        }else{
            //Envoie un message d'alerte
            Alert.alert(
                "Information",
                TEXT_FILL_ALL,
                [{text:"Ok"}]
            );
        }
    }

    render(){
        return (
            <View style={BASE_STYLE.container}>
                <KeyboardAwareScrollView
                style={BASE_STYLE.scrollview_container}
                contentContainerStyle={BASE_STYLE.scrollview_content}
                >
                    {/* Pseudo */}
                    <Input
                    placeholder='Pseudo'
                    onChangeText={(value) => this.user.pseudo = value}
                    />

                    {/* Email */}
                    <Input
                    keyboardType='email-address'
                    placeholder='Email'
                    onChangeText={(value) => this.user.email = value}
                    />
                    {/* Mot de passe */}
                    <Input
                    secureTextEntry={true}
                    placeholder='Password'
                    onChangeText={(value) => this.user.pwd = value}
                    />
                    <Button
                    title='Sign up'
                    onPress={() => this._signUp()}
                    />
                </KeyboardAwareScrollView>
                {/* Affichage du chargement */}
```

```
                {/* Affichage du chargement */}
                {this.state.isLoading && <Loading/>}
            </View>
        );
    }
}

//Pour Redux
const mapStateToProps = (state) => {
    return {
        idUser: state.isConnected.idUser
    }
}

//Export du component
export default connect(mapStateToProps)(SignUp);
```

```
//Librairies
import React, { Component } from 'react';
import { StyleSheet, Text, View, ScrollView, Button, FlatList, Image,
TouchableOpacity, Share } from 'react-native';
import * as Linking from 'expo-linking';
import { Entypo } from 'react-native-vector-icons';
//Components perso
import { BASE_STYLE, Loading, ImageList } from './MyComponent.js';
//Fonctions
import { getSimilarWP, checkFav, toggleFav } from '../WS/functions.js';
//Redux
import { connect } from 'react-redux';

class WallpaperDetails extends Component {
    constructor(props) {
        super(props);
        //State
        this.state = {
            wallpaper: this.props.route.params.wallpaper,
            similars: undefined,
            fav: false
        },
        //Si l'user est connecté
        (this.props.idUser !== false &&
            //Checker si l'image est en favoris
            checkFav({idUser: this.props.idUser, idImage:
this.state.wallpaper.id}).then(data => {
                this.setState({fav: data});
            }),
            //Récupérer les images similaires
            getSimilarWP(this.state.wallpaper.id).then(data => {
                this.setState({similars: data.data});
            })
        );
    }

    //Partage du wallpaper
    _share(){
        Share.share({
            message: "Wallhaven wallpaper : " + this.state.wallpaper.url,
        });
    }

    //Toggle le favoris
    _toggleFav(){
        toggleFav({idUser: this.props.idUser, idImage:
this.state.wallpaper.id}).then(data => {
            this.setState({fav: data});
        });
    }

    //Ouvrir dans Wallhaven
    _openInWallhaven(){
        Linking.openURL(this.state.wallpaper.url);
    }
```

```
    }

    render(){
        return (
            <View style={BASE_STYLE.container}>
                <ScrollView>
                    {/* Image */}
                    <Image
                    style={BASE_STYLE.img_full}
                    source={{uri: this.state.wallpaper.path}}
                    />
                    {/* Informations sur le wallpaper */}
                    <View style={BASE_STYLE.infos_wallpaper}>
                        {/* Définition */}
                        <Text>Resolution: {this.state.wallpaper.resolution}
</Text>
                        {/* Ouvrir dans Wallhaven */}
                        <TouchableOpacity
                        onPress={() => this._openInWallhaven()}
                        >
                            <Text style={BASE_STYLE.text_link}>Open in
Wallhaven</Text>
                        </TouchableOpacity>
                        <View style={{flexDirection: 'row', justifyContent:
'space-between'}}>
                            {/* Partage */}
                            <TouchableOpacity
                            onPress={() => this._share()}
                            >
                                <Entypo
                                name="share"
                                size={25}
                                />
                            </TouchableOpacity>
                            {/* Étoile de favoris */}
                            {this.props.idUser !== false &&
                                <TouchableOpacity
                                onPress={() => this._toggleFav()}
                                >
                                    <Entypo
                                    name={this.state.fav ? "star" : "star-
outlined"}
                                    size={25}
                                    />
                                </TouchableOpacity>
                            }
                        </View>
                    </View>
                    {/* Affichage des fonds d'écrans similaire seulement si
l'utilisateur est connecté */}
                    {this.props.idUser !== false &&
                        <View>
                            <Text style={BASE_STYLE.text_subtitle}>Similar
wallpaper :</Text>
                            <ImageList
                            data={this.state.similars}
```

```
                                data={this.state.similars}
                                onPress={(item) =>
this.props.navigation.push('WallpaperDetails', {wallpaper: item})}
                                />
                            </View>
                        }

                    </ScrollView>
                    {/* Affichage du chargement */}
                    {this.state.isLoading && <Loading/>}
                </View>
        );
    }
}

//Pour Redux
const mapStateToProps = (state) => {
    return {
        idUser: state.isConnected.idUser
    }
}

//Export du component
export default connect(mapStateToProps)(WallpaperDetails);
```