Documentation projet LPO - Application Mobile

Ce projet utilise react-native comme base. Pour obtenir ce module et les autres, utilisez la commande

```
npm install
```

dans l'invité de commande au niveau `./lpo-mobapp` de vos fichiers.

npm est téléchargable avec node.js.

Structure de l'application

Jeu :

- Une page dans composant : `composant.page.js`
- Un composant correspondant au jeu dans un dossier de nom de la page : `composant.component.js`
- Un style donnant la mise en forme du component : `composant.component.style.js`

Style et charte graphique :

- Dans `theme.js` , on retrouve les constantes (notamment les couleurs) utilisées dans l'application
- Dans `commun.js` , on retrouve les styles css commun aux pages. Ils sont appelés dans les .style de tout les composants.

Requêtes : Elles se trouvent dans querry.js

Sauvegarde : Elles sont effectuées grâce à `loadParcours.js` et `saveParcours.js`

Ajouter un jeu côté app mobile

Un jeu côté application mobile est généralement organisé de manière suivante:

A 'pages' dans l'architecture: NomDuJeu.page.js qui contient:

```
import React, {Component} from 'react';
import NomDuJeu from './components/NomDuJeu/NomDuJeu.component'

class NomDuJeuPage extends Component {
constructor(props) {
    super(props);
}

    render () {
    return (
        <NomDuJeu
```

```
            parcours = {this.props.route.params.parcours}
            currentGame = {this.props.route.params.currentGame}
            />
        );
        }

    }

    export default NomDuJeuPage
```

Dans pages/components/NomDuJeu: un fichier NomDuJeu.component.js ainsi qu'un fichier associé NomDuJeu.component.style.js

Le fichier style contiendra les styles utilisés dans notre composant. La plupart proviennent de common de la manière suivante:

```
import { StyleSheet } from 'react-native';
import common from '../../../styles/common.style.js'
export default StyleSheet.create({
nomDuStyle: {
  ...common.nomDuStyle
},
});
```

Le fichier component contiendra: Notre classe :

```
class NomDuJeu extends Component{
    constructor(props){
        super(props);
        ...;
        this.handleBackButtonClick = this.handleBackButtonClick.bind(this);
    }
    componentDidMount() {
        BackHandler.addEventListener('hardwareBackPress', this.handleBackButtonClick);
    }
    componentWillUnmount() {
        BackHandler.removeEventListener('hardwareBackPress', this.handleBackButtonClick);
    }
    handleBackButtonClick() {
        return true;
    }

`
```

Cette partie a servi à récupérer les propiriétés passées en paramètres et à désactiver le bouton retour du téléphone

```
    ...;
    render() {
        ... (initialisation des const);
        var TopBarreName = "Étape " + this.props.currentGame.ordre + "/" + (this.props.currentGame
        return (
            <SafeAreaView style={styles.outsideSafeArea}>
                <View style={styles.globalContainer}>
                    <ScrollView contentContainerStyle={styles.scrollViewContainer} style={styles.sc
```

Ceci est la barre en haut de l'écran avec les logo et le nom de la page (ou la progression dans le parcours).

Cette vue de style card contient le jeu {VariableContenantLeTitreDuJeu} {VariableContenantLaQuestion} {VariableContenantLaDescription} Parfois un seul des deux champs précédent est présent, parfois il y a aussi les règles. C'est à adapter selon le jeu <Image source={{ uri: this.props.currentGame.image_url }} style={styles.areaImage} /> Il y a ensuite le champ même du jeu, parfois l'image n'est pas comme ici, par exemple pour trouver l'intrus.

Il y a la bouton pour avancer. Si le jeu nécessite une page réponse, il faut rediriger vers GameOutcomePage, avec les paramètres parcours; currentGame et win sinon on redirige vers GamePage avec parcours en paramètre. Ceci est présent dans la majorité des cas, mais par exemple le QCM ou trouver l'intrus permettent d'avancer sans bouton valider mais dès qu'on choisit une réponse. Dans ce cas, cette partie n'est pas présente.

```
                    <View style={styles.rightAlign}>
                        <NextPage pageName={"NomPageSuivante"}
                            parameters={{
                                // paramètres
                            }}
                            text="Valider" blockButton=
                            {true}
                        />
                    </View>
                </ScrollView>
            </View>
        </SafeAreaView>
    );
    }
}
export default function (props) {
    const navigation = useNavigation();
    return <NomDuJeu {...props} navigation={navigation} />
}
```

Il faut ensuite compléter si nécessaire le fichier GameOutcomePage.component.js, soit en ajoutant NomDuJeu à la liste des possibilités, soit, si cela nécessite un traitement particulier en rajoutant un

nouveau traitement, avant ceci:

```
this.props.parcours[this.props.parcours.length - 1].score_max++;
if (win) {
  this.props.parcours[this.props.parcours.length - 1].score++;
}
```

Dans GamePage.component.js, ajouter:

```
    case ("NomDuJeu""): {
      navigation.navigate("NomDuJeuPage",{parcours : parcours, currentGame : currentGame});
      break;
    }
```

Dans Navigation.js, dans HomeStack, ajouter:

```
        <Stack.Screen
            name="NomDuJeuPage"
            component={NomDuJeuPage}
            options={{ headerShown: false }}
        />
```

Fonctionnement de la sauvegarde locale

La sauvegarde locale se base sur le module AsyncStorage. La sauvegarde enregistre des informations à différents endroits :

- Le nom de la commune, dans un tableau récupérable avec la clé `commune` .
- Les informations pour la présentation du parcours (ainsi que l'id du parcours), dans un tableau récupérable avec la clé `commune.nomDeLaCommune` .
- Les informations générales du parcours (dont le nombre d'étapes dans le champ length) récupérables en utilisant l'id du parcours `idParcours` en temps que clé.
- Les étapes du parcours, récupérables avec la clé `idParcours.numeroEtape` (numéro entre 0 et length-1 inclus).
- Si l'étape possède un tableau d'images, elles sont stockées individuellement dans `idParcours.numeroEtape.numeroImage` , mais ces clés sont dans le tableau censé contenir les images (c'est à dire le champ `images_tab` de l'étape).

Le component `ParcoursCard` s'occupe de sauvegarder et supprimer les parcours

- `saveParcours` ; dans le même dossier; sauvegarde toutes les informations nécessaires pour réutiliser ce parcours hors-connexion.
- `loadParcoursLacally` dans le dossier `utils` possède des fonctions pour charger les données

d'un parcours
- `deleteLocalParcours` dans le dossier `utils` permet de supprimer un parcours (et si utile une commune)

Déploiement mobile

Pour déployer l'application, il est nécessaire de générer un .aab avec la même clé que le projet. Pour cela, nous utilisons un compte ExpoGo combiné avec eas.

Étapes à suivre :

- Installer le module eas.
- Créer d'un compte ExpoGo.
- Ajouter le compte dans la liste des comptes autorisés sur le compte responsable de la lpo (le propriétaire du compte de l'entreprise).
- Se connecter sur le compte expo par eas
- Exécuter la commande : `eas build`

L'utilisateur obtient alors un lien pour télécharger le .aab qu'il faudra mettre en ligne sur le compte Google PlayStore. Il est important que chaque version mise en ligne ai un numéro de build différent.

Sécurité et Firebase

Pour se connecter côté mobile, l'application utilise un compte commun d'authentification nécessaire pour accéder à la base (le mot de passe n'est pas en clair dans l'application mais peu être techniquement retrouvé grâce à une fonction, offuscate). Cependant, ce compte n'a que des droits de lecture dans la base. C'est une implémentation perfectible faites en urgence pour combler une faille de sécurité importante. Ensuite, le code est également offusqué à l'aide du fichier transformer.js à la racine.

Cette partie peut être rendu plus sûr par l'utilisation d'AppCheck ou en limitant l'accès à des applications avec la bonne signature. Cela empêchera cependant l'utilisation d'expo go.

Pistes d'amélioration

- Mettre en place une meilleure sécurité pour la Firebase
- Si les infos changent sur la base de la LPO, les données des jeux doivent être mise à jour automatiquement.
- Ajouter des éléments de géolocalisation. Penser à mettre à jour les CGU dans ce cas.
- Au démarrage, afficher une carte localisant tout les parcours.