Compte rendu - Agora Mobile Sprint 2

Projet réalisé : Création d’une application mobile en React

Objectifs de la mission

L'association Agora souhaitait disposer d'une application mobile permettant de gérer les informations relatives aux jeux vidéo : jeux, genres, marques, classifications pegi et plateformes. L'objectif de ce sprint est de sécuriser l’application via un formulaire de connexion.

Organisation

Nous avons utilisé Trello pour représenter les tâches.

Conception - diagrammes

Diagramme de cas d'utilisation

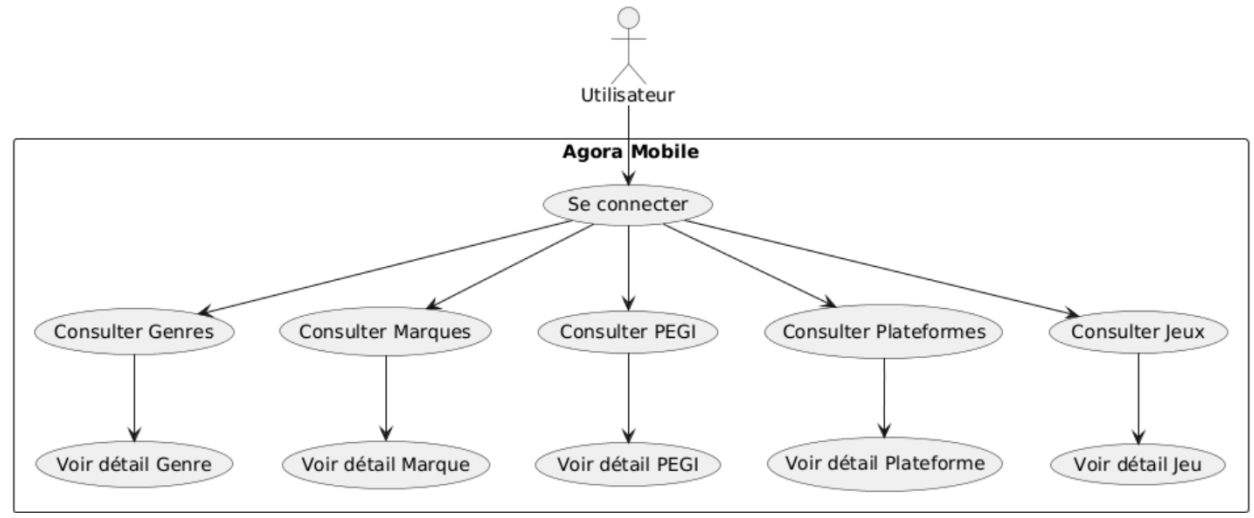
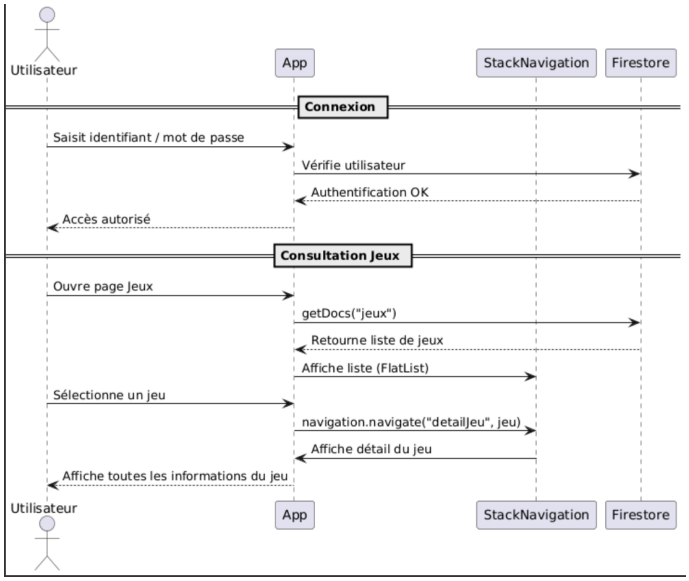


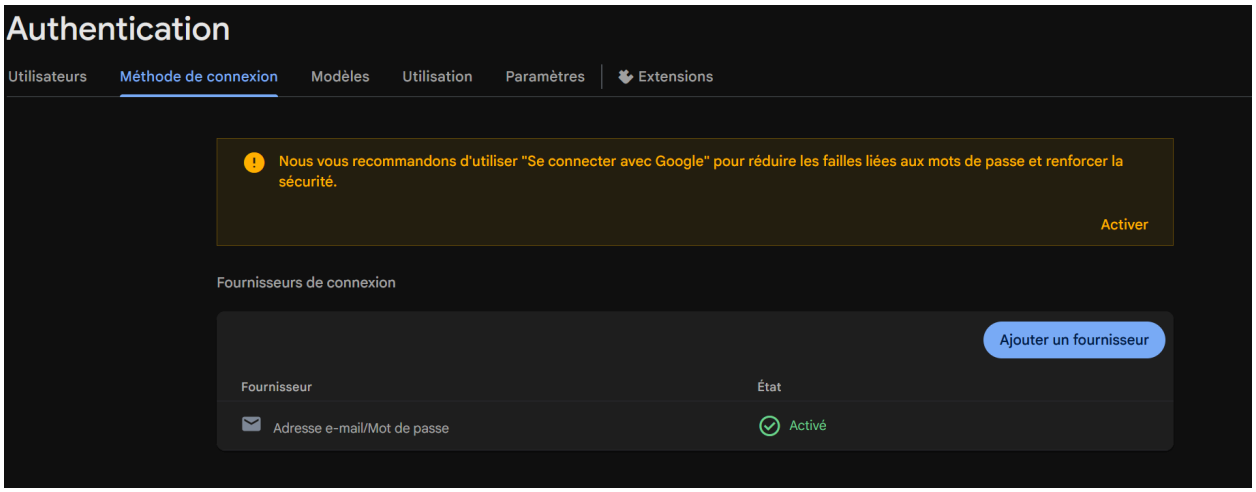
Diagramme de séquence



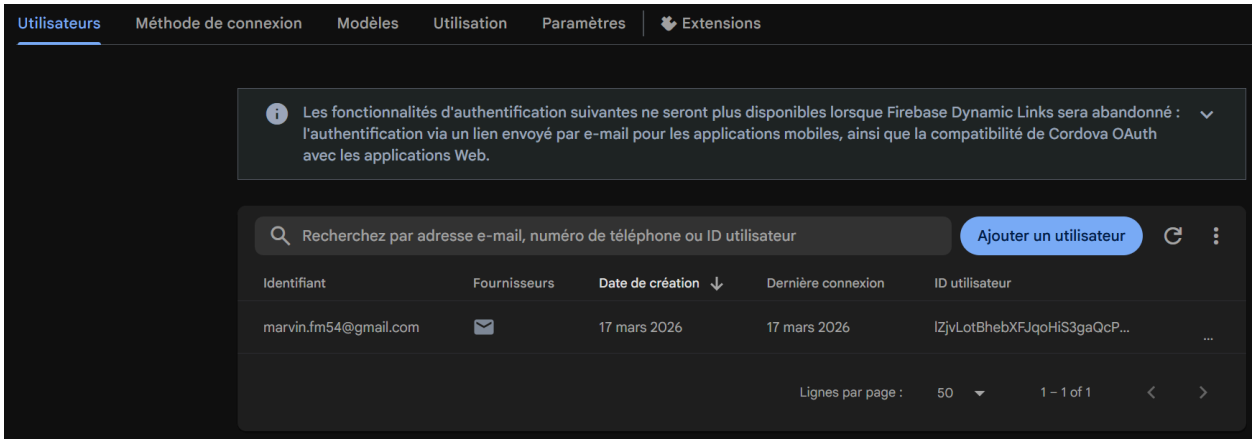
Grandes étapes

Ajout d’un dispositif d’authentification

Activation des emails / mdp dans la partie authentification de firebase



Création d’un utilisateur



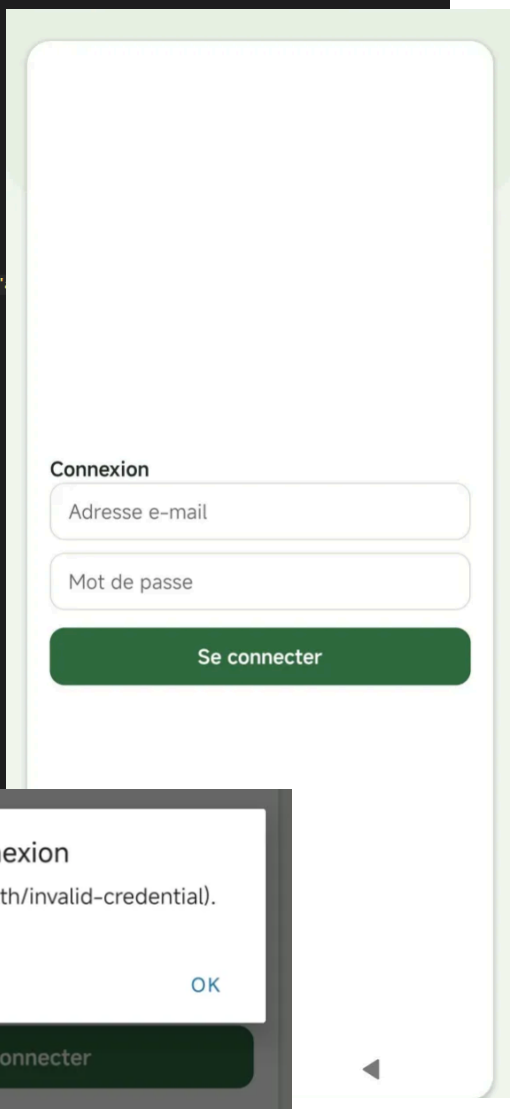
Installation de async-storage

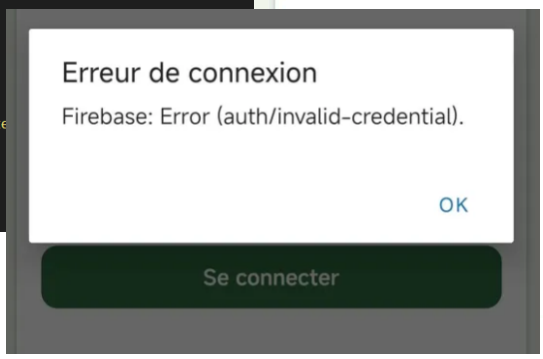
```
PS C:\wamp64\www\agoramobile> npm install @react-native-async-storage/async-storage

added 3 packages, and audited 796 packages in 40s

54 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Page de connexion fonctionnelle + gestion des erreurs de connexion

screens > connexion.tsx > Connexion > handleLogin > catch() callback

```tsx
import React, { useState } from "react";
import { TouchableOpacity, View, Text, Button, StyleSheet, TextInput, Alert } from "react-native";
import { auth } from "../services/firebaseConfig";
import { signInWithEmailAndPassword, AuthError } from "firebase/auth";
import { styles } from "../ui/commonStyles";

export default function Connexion({ navigation }) {

    const [email, setEmail] = useState("");
    const [password, setPassword] = useState("");

    const handleLogin = () => {
        signInWithEmailAndPassword(auth, email, password)
            .then((userCredential) => {
                navigation.navigate("pageMenu", { email });
            })
            .catch((error: AuthError) => {
                if (error.code === "auth/configuration-not-found" || error.code === "
                    Alert.alert(
                        "Configuration Firebase manquante",
                    );
                    return;
                }

                Alert.alert("Erreur de connexion", error.message);
            });
    };

    return (
        <View style={styles.screen}>
            <View style={styles.accentStrip} />
            <View style={[styles.containerCard, styles.centerCard]}>
                <Text style={styles.cardTitle}>Connexion</Text>
                <TextInput
                    placeholder="Adresse e-mail"
                    keyboardType="email-address"
                    value={email}
                    onChangeText={setEmail}
                    autoCapitalize="none"
                    style={styles.button}
                />
                <TextInput
                    placeholder="Mot de passe"
                    secureTextEntry
                    value={password}
                    onChangeText={setPassword}
                    style={styles.button}
                />
                <Button color="gray" title="Se connecte
            </View>
        </View>
    );
}
```



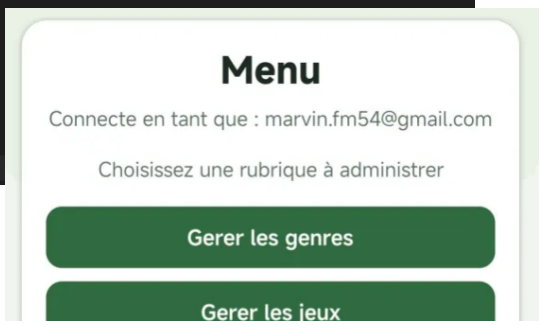


Vérification d’un utilisateur authentifié sur les pages de l’app

Vérifier que l’utilisateur est bien connecté, sinon le rediriger vers la page de connexion.

screens > menu.tsx > Menu

```tsx
import React, { useEffect } from "react";
import { TouchableOpacity, View, Text, ScrollView } from "react-native";
import { styles } from "../ui/commonStyles";
import { auth } from "../services/firebaseConfig";
import { onAuthStateChanged } from "firebase/auth";

export default function Menu({ navigation, route }) {
    const email = route?.params?.email || auth.currentUser?.email || "inconnu";

    useEffect(() => {
        const unsubscribe = onAuthStateChanged(auth, (user) => {
            if (!user) {
                navigation.replace("pageConnexion");
            }
        });

        return unsubscribe;
    }, []);
```



Déconnexion

Fonction pour se déconnecter

```tsx
    const handleLogout = async () => {
        try {
            await signOut(auth);
            navigation.replace("pageConnexion");
        } catch (error) {
            console.log("Erreur deconnexion :", error);
        }
    };
```