DOCUMENTATION PFE
Ma Borne - CATS

1. Front-end

a. Technologies utilisées

Android studio pour développer en KMP
Android -> UI avec Jetpack Compose avec Material Design
iOS -> Xcode et SwiftUI

b. Mise en place du projet

Nom du projet : MaBorneApp
Nom du package : com.pfe.maborne

Shared Code : Contient la logique partagée (module shared).
Android App : Contient la partie Android (module androidApp).
iOS App : Contient la partie iOS (visible dans Xcode).

Pour afficher des logs :

```
Log.e("[DEBUG]", "Button clicked. Email=$email, Password=$password")
```

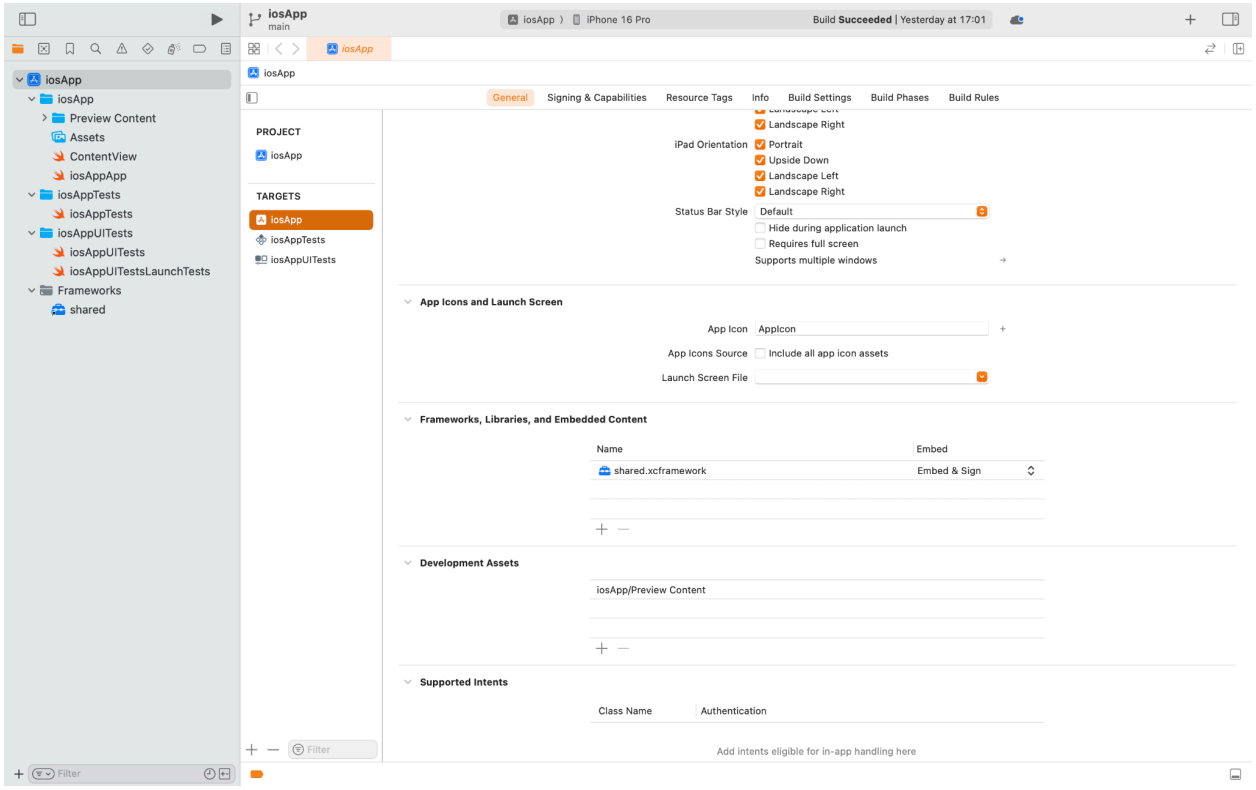
dans logcat, filtré avec `[DEBUG]` (erreur en rouge)

Côté ios :
ouvrir le fichier iosApp.xcodeproj dans XCode
erreur : ‘Process 'command 'codesign" finished with non-zero exit value 1 sur xcode, exécuter la commande dans le dossier contenant shared.xcframework :
codesign --verify --deep --verbose shared.xcframework
si la réponse est la suivante c’est bon :
shared.xcframework: valid on disk
shared.xcframework: satisfies its Designated Requirement
sinon -> *codesign --force --sign "Apple Development: poirierauriane@gmail.com (W946ZZATS8)" --timestamp --deep shared.xcframework*
(avec certificat dans trousseau d’accès)

Lorsque dossier shared modifié :

- ANDROID : cliquer sur "Sync Project with Gradle Files" dans Android Studio puis exécuter la commande *./gradlew build* à la racine du projet

- IOS : exécuter la commande *./gradlew shared:assembleXCFramework* à la racine du projet, puis aller sur XCode, supprimer le dossier shared.xcframework de la section **Frameworks, Libraries, and Embedded Content**, puis ajouter le nouveau (+, add file, sélectionner le dossier shared.xcframework dans le dossier shared/build/xcframework), vérfiier qu’il est bien en *‘Embed & Sign’*.



c. Architecture

Architecture en MVVM (Model-View-ViewModel)

i. Shared

```
shared/
    src/commonMain/kotlin/com.pfe.maborneapp/
        repository/
            LoginRepository.kt
        models/
            LoginRequest.kt
            User.kt
```

Le **LoginRepository** agit comme un intermédiaire entre :

- Les ViewModels (qui représentent la logique de présentation).
- Les sources de données (back-end via API, stockage local, etc.).

Ce qu'il contient :

1. **Logique d'accès aux données :**
    - Appels au back-end via des requêtes HTTP.
    - Gestion du cache local (si besoin).
    - Gestion des erreurs.

LoginRequest représente les données envoyées au back-end pour se connecter.

User représente les données d’un utilisateur connecté, telles que renvoyées par le back-end.

ii. Android

```
androidApp/
    src/main/java/com.pfe.maborneapp.android/
        view/
            LoginPage.kt
            admin/
                AdminHomePage.kt
            user/
                UserHomePage.kt
        viewmodel/
            admin/ user/
            factories/
                admin/
                user/
                LoginViewModelFactory.kt
            LoginViewModel.kt
        theme/
            MyApplicationTheme.kt  // Gestion des thèmes globaux
        MainActivity.kt
```

La **View** gère uniquement **l'affichage de l'interface utilisateur** et **les interactions utilisateur** (clics, saisie de texte, etc.).

- **Responsabilités de la View :**
  - Afficher les données reçues depuis le `ViewModel`.
  - Réagir aux événements utilisateur (boutons, champs de texte, navigation).
  - Observer l'état (via `StateFlow` ou `LiveData`) exposé par le `ViewModel`.

La **ViewModel** contient :

- **La logique de présentation :**
  - Les données observables exposées à la `View` (par exemple, via `StateFlow` ou `LiveData`).
  - Les méthodes appelées par la `View` pour effectuer des actions (par exemple, `login()`).
- **Les interactions avec le repository :**
  - Elle appelle le repository pour récupérer ou envoyer des données.

La **ViewModelFactory** est responsable :

- **De l’instanciation du ViewModel** : Elle crée des objets `ViewModel` en passant les dépendances nécessaires (comme le repository).
- De centraliser l’instanciation pour faciliter le passage de dépendances personnalisées, par exemple lors des tests.

La **MainActivity** est le **point d'entrée principal** de l'application Android. Dans ton cas, elle :

- Définit la **navigation** (entre les pages comme `LoginPage`, `AdminHomePage`, `UserHomePage`).
- Fournit les **dépendances communes** (par exemple, le `LoginRepository` à injecter dans le `ViewModel`).

iii. iOS

La structure et la logique sont les mêmes que pour android.

2. Back-end