Installation Quickstart

Dans cette page, nous allons procéder à une installation quickstart de l'application, c'est-à-dire que cette dernière…

- Fonctionnera de manière autonome avec des mock pas de connexion au SI
- Présentera la liste exhaustive des services qu'elle offre
- Ne proposera aucune personnalisation graphique

Cette installation vise plutôt le test de l'application ou la création d'un démonstrateur.

Récupération des sources

Cloner les sources disponibles sous **gitHub** https://github.com/univlorraine/esup-multi

```
git clone git@github.com:univlorraine/esup-multi.git
```

Le CMS Headless

Le CMS HeadLess va permettre de fournir du contenu dynamique au client mobile

Installation

Le CMS se lance avec Docker. Par défaut, la configuration proposée utilise MySQL

1. Se rendre dans le dossier **env/local/docker/directus**, le déploiement est décrit dans le fichier **docker-compose.yml**
2. Lancer Directus :
   ```
   $ docker compose up --build -d
   ```
3. Le CMS sera accessible sur http://localhost:8055, connectez-vous avec les identifiants renseignés plus tôt (`ADMIN_EMAIL` et `ADMIN_PASSWORD`).
4. Dans **Settings > Project Settings**, passez le CMS en français (optionnel, mais les explications suivantes se font avec l'interface en français).

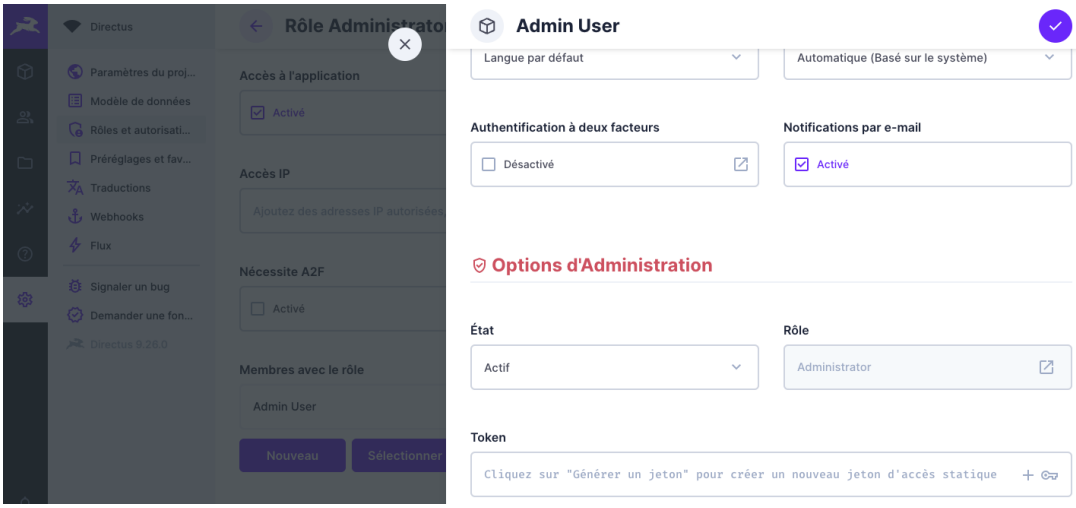
Import

1. Pour importer les collections :
   ```
   $ docker compose exec directus npx directus schema apply --yes ./snapshot/snapshot.yaml
   ```

2. Redémarrez le conteneur Docker de Directus pour que l'import soit bien pris en compte.
3. Dans **Réglages > Modèles de données**, pour la collection **Languages**, sélectionnez **Voir le contenu**.
4. Sélectionnez le fichier des langues (**languages.json**) et importez-le.
5. Pour chaque collection visible dans le panneau latéral (dans l'onglet **Contenu**), importez le contenu.
6. Saisir les informations demandées dans **Contact US** et **Login** (pour plus d'explications, consultez la page suivante)

Permissions

1. Dans **Réglages > Rôles et autorisations > Role Administrator > Membres avec ce rôle > Nouveau**
2. Indiquer uniquement un nom pour ce nouvel utilisateur
3. Générez un token pour cet utilisateur (on ne peut pas créer de token pour **Admin User** cela provoque une erreur à l'enregistrement du rôle)
4. Gardez-le, vous en aurez besoin pour la configuration du backend.



pour plus d'explication consultez la page suivante

Le Backend

Le backend doit disposer des briques suivantes (Voir l'installation des pré-requis) :

- Une base MongoDB
- Un serveur Nats
- Une base Redis
- Instance de Directus (cf. section Le CMS Headless)

Configuration des variables d'environnement

Esup-Multi s'accompagne d'un serveur de mocks visant à simuler les connecteurs qui devront être branchés sur le SI de l'établissement. Esup-multi est donc pré-paramétré pour fonctionner avec ces mocks et les paramètres par défaut des services à installer en pré-requis.

Dans **dev/user-backend-nest** copier et renommer chaque fichier *.env.dist* en *.env* pour :

- la gateway dans **/main**
- chaque µService dans **/microservices/***

Dans chaque µService utilisant directus positionner la variable <MICRO_SERVICE_NAME>_DIRECTUS_API_BEARER_TOKEN avec le token obtenu pour le user directus (cf paragraphe précédent) dans le fichier *.env*

Exemple :

```
SOCIAL_NETWORK_SERVICE_DIRECTUS_API_BEARER_TOKEN=azertyqsdfg123456
```

Dans renseigner **/main/.env** et dans **/microservices/auth/.env** un secret pour les JWT dans la variable `AUTH_SERVICE_JWT_SECRET`

```
AUTH_SERVICE_JWT_SECRET=secretsecret
```

Serveur de mock

L'application multi est fournie avec un serveur de mock. Il s'agit d'API qui renvoient des données statiques qui vous permettent d'émuler ce que retourneraient des connecteurs branchés sur votre système d'information.

L'URL de chaque mock est précisée par défaut dans le fichier *.env* du microservice sous la forme :

```
http://localhost:3099/mocking/<api_exemple>
```

Il n'y a donc rien de plus à configurer de ce côté-là.

Dans **dev/user-backend-mocks** copier et renommer chaque fichier *.env.dist* en *.env*

Installer le serveur depuis */dev/user-backend-mocks/* avec la commande :

```
npm install
```

Puis démarrer le serveur avec la commande :

```
npm start
```

⚠ Ces mocks ne servent qu'à des fins de test et ne doivent pas être déployés dans un environnement de production. Vous trouverez plus d'informations dans le fichier README situé à la racine du dossier */dev/user-backend-mocks/*.

Installer les dépendances et build

Placez-vous à la racine du projet et installez les dépendances :

```
$ npm ci
```

puis lancez (toujours depuis la racine du projet) :

```
$ npm run build:back
```

Cette commande va, pour la gateway et chaque microservice, installer les dépendances et build.

```
[map] npm run back:map -- run build exited with code 0
[static-pages] npm run back:static-pages -- run build exited with code 0
[chatbot] npm run back:chatbot -- run build exited with code 0
[social-network] npm run back:social-network -- run build exited with code 0
[cards] npm run back:cards -- run build exited with code 0
[rss] npm run back:rss -- run build exited with code 0
[important-news] npm run back:important-news -- run build exited with code 0
[statistics] npm run back:statistics -- run build exited with code 0
[features] npm run back:features -- run build exited with code 0
[notifications] npm run back:notifications -- run build exited with code 0
[contact-us] npm run back:contact-us -- run build exited with code 0
[contacts] npm run back:contacts -- run build exited with code 0
[main] npm run back:main -- run build exited with code 0
[restaurants] npm run back:restaurants -- run build exited with code 0
[mail-calendar] npm run back:mail-calendar -- run build exited with code 0
[schedule] npm run back:schedule -- run build exited with code 0
[clocking] npm run back:clocking -- run build exited with code 0
[auth] npm run back:auth -- run build exited with code 0
```

Démarrer la gateway et les microservices

Deux modes de fonctionnement sont possibles :

- Normal :

```
npm run start:back
```

- Dev (redémarrage après modification) :

```
npm run start:back:dev
```

Ces commandes vont lancer en parallèle la gateway ainsi que tous les microservices.

Le Client

Prérequis

- Android Studio + JDK pour l'app Android
- Xcode + CocoaPods à jour (MacOS uniquement) pour l'app iOS
- Avoir un backend fonctionnel, démarré et accessible depuis la machine locale ou le périphérique qui seront utilisés pour les tests
- Avoir une instance du CMS headless fonctionnelle, démarrée et accessible depuis le backend ET la machine locale ou le périphérique qui seront utilisés pour les tests

Paramétrages minimaux de l'application cliente

Dans **/dev/user-frontend-ionic** copier, coller et renommer le fichier **capacitor.ts.dist** en **capacitor.ts**. Ce fichier permet de paramétrer le nom de l'app et son identifiant, on laissera les valeurs par défaut pour le quickstart. Copier, coller et renommer également le fichier **angular.json.dist** en **angular.json**. Ce fichier permet de paramétrer les chemins vers les assets, on laissera les valeurs par défaut pour le quickstart.

Dans **/dev/user-frontend-ionic/src/environments** copier, coller et renommer le fichier **environment.ts.dist** en **environment.ts.** (Laisser les 2 lignes qui concernent firebase en commentaire)

Pour utiliser le thème par défaut disponible avec la démo, dans le sous-dossier **/src/theme** :

- copier et renommer le répertoire **/app-theme-dist** en **/app-theme**
- copier et renommer le fichier **theme.scss.dist** en **theme.scss**

Pour personnaliser tous ces éléments, voir la documentation

Préparation des fichiers Ionic

Depuis `dev/user-frontend-ionic/`,

installer les dépendances :

```
$ npm ci
```

puis, préparer l'app :

```
$ npm run build
```

Cette commande va compiler les assets et préparer les fichiers pour le build des applications natives.
Elle va en fait exécuter 2 commandes :

- prebuild (exécuté automatiquement avant build) : `npm run module:build-all` => va compiler les différents modules Angular présents dans le dossier `projects/` et les placer dans un dossier `dist/`
- build : `ng build` => va compiler l'app Angular et les assets associés présents dans le dossier `src/` et les placer dans un dossier `www/`

Build des apps

Build appli web

Il n'y a rien de plus à faire à ce niveau car les fichiers compilés nécessaires au fonctionnement de l'application web ont été buildés via la commande `npm run build` ci-dessus.

Build Android

Ajouter le support d'Android

Pour ajouter le support d'Android au projet, il est nécessaire d'exécuter la commande suivante :

```
$ npx capacitor add android
```

Cette commande va générer tous les fichiers nécessaires au build d'une application Android et les placer dans un dossier `android/`

Elle va également ajouter à ces fichiers, une copie des assets et des sources se trouvant dans `www/`

> ⚠ Cette commande n'est à exécuter qu'une seule fois sur le projet. Elle permet d'initier les fichiers nécessaires au build d'une app Android

Copier les sources

Pour ajouter les assets et sources du projet compilés, il est nécessaire d'exécuter la commande suivante :

```
$ npx capacitor sync android
```

Cette commande va copier les fichiers de l'app Angular compilés dans le dossier `android/` (équivalent de la commande `npx capacitor copy android`) en plus de faire correspondre les différents plugins Capacitor mis en place sur le projet avec les différentes fonctionnalités natives Android.

Build iOS

> ⛔ Les tests sur périphérique iOS externe ou même émulateur ne sont possibles que depuis une machine tournant sur macOS (nécessite l'utilisation de XCode)

Ajouter le support d'iOS

Pour ajouter le support d'iOS au projet, il est nécessaire d'exécuter la commande suivante :

```
$ npx capacitor add ios
```

Cette commande va générer tous les fichiers nécessaires au build d'une application iOS et les placer dans un dossier `ios/`

Elle va également ajouter à ces fichiers, une copie des assets et des sources se trouvant dans `www/`

> ⚠ Cette commande n'est à exécuter qu'une seule fois sur le projet. Elle permet d'initier les fichiers nécessaires au build d'une app iOS

Copier les sources

Pour ajouter les assets et sources du projet compilés, il est nécessaire d'exécuter la commande suivante :

```
$ npx capacitor sync ios
```

Cette commande va copier les fichiers de l'app Angular compilés dans le dossier `ios/` (équivalent de la commande `npx capacitor copy ios`) en plus de faire correspondre les différents plugins Capacitor mis en place sur le projet avec les différentes fonctionnalités natives iOS.

Générer les splashscreens et les icônes

Les icônes et le splashscreens se trouvent par défaut dans `/dev/user-frontend-ionic/src/theme/default/resources`

Pour les ajouter dans les app natives Android et iOS, il est nécessaire d'exécuter la commande suivante :

```
$ npx capacitor-assets generate --assetPath src/theme/default/resources
```

Configurer les autorisations sur les périphériques

Suivant les modules que vous souhaiterez utiliser dans votre application, il peut parfois être nécessaire d'ajouter les autorisations nécessaires d'accès aux fonctionnalités natives du périphérique (appareil photo, annuaire des contacts, etc...).

Android

Éditer le fichier `android/app/src/main/AndroidManifest.xml` et y coller le contenu suivant (remplacer l'existant) :

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools">

  <supports-screens android:smallScreens="false"
                    android:normalScreens="true"
                    android:largeScreens="true"
                    android:xlargeScreens="true" />

  <application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme"
    android:hardwareAccelerated="true">

    <activity
      android:configChanges="
orientation|keyboardHidden|keyboard|screenSize|locale|smallestScreenSize|screenLayout|uiMode"
      android:name="fr.esupportail.mobile.multi.MainActivity"
      android:label="@string/title_activity_main"
      android:theme="@style/AppTheme.NoActionBarLaunch"
      android:launchMode="singleTask"
      android:exported="true">

      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>

    <provider
      android:name="androidx.core.content.FileProvider"
      android:authorities="${applicationId}.fileprovider"
      android:exported="false"
      android:grantUriPermissions="true">
      <meta-data android:name="android.support.FILE_PROVIDER_PATHS" android:resource="@xml/file_paths" />
    </provider>

<!--    <meta-data-->
<!--      android:name="firebase_messaging_auto_init_enabled"-->
<!--      android:value="false" />-->
<!--    <meta-data-->
<!--      android:name="firebase_analytics_collection_enabled"-->
<!--      android:value="false" />-->
  </application>

  <queries>
    <intent>
      <action android:name="android.intent.action.SENDTO" />
      <data android:scheme="mailto" />
    </intent>
  </queries>

  <!-- Features -->
  <uses-feature android:name="android.hardware.camera" />
  <uses-feature android:name="android.hardware.location.gps" />

  <!-- Permissions -->
  <uses-permission android:name="android.permission.INTERNET" />
  <uses-permission android:name="android.permission.CAMERA" />
  <uses-permission android:name="android.permission.READ_CONTACTS" />
  <uses-permission android:name="android.permission.WRITE_CONTACTS" />
<!--  <uses-permission android:name="android.permission.POST_NOTIFICATIONS" />-->
  <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
  <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

```
  <!-- Override Libraries -->
  <uses-sdk tools:overrideLibrary="com.google.zxing.client.android" />

</manifest>
```

iOS

Éditer le fichier ios/App/App/Info.plist et y coller le contenu suivant (remplacer l'existant) :

```
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
        <key>CFBundleDevelopmentRegion</key>
        <string>fr</string>
        <key>CFBundleDisplayName</key>
  <string>Esup-Multi</string>
        <key>CFBundleExecutable</key>
        <string>$(EXECUTABLE_NAME)</string>
        <key>CFBundleIdentifier</key>
        <string>$(PRODUCT_BUNDLE_IDENTIFIER)</string>
        <key>CFBundleInfoDictionaryVersion</key>
        <string>6.0</string>
        <key>CFBundleName</key>
        <string>$(PRODUCT_NAME)</string>
        <key>CFBundlePackageType</key>
        <string>APPL</string>
        <key>CFBundleShortVersionString</key>
        <string>$(MARKETING_VERSION)</string>
        <key>CFBundleVersion</key>
        <string>$(CURRENT_PROJECT_VERSION)</string>
        <key>FirebaseMessagingAutoInitEnabled</key>
  <string>NO</string>
  <key>ITSAppUsesNonExemptEncryption</key>
  <false/>
  <key>LSApplicationQueriesSchemes</key>
  <array>
    <string>mailto</string>
  </array>
        <key>LSRequiresIPhoneOS</key>
        <true/>
        <key>NSCameraUsageDescription</key>
  <string>L'application a besoin de l'appareil photo pour permettre le scan de code-barre</string>
  <key>NSContactsUsageDescription</key>
  <string>L'application a besoin d'accéder à votre répertoire pour vous permettre d'ajouter un contact depuis
la recherche dans l'annuaire</string>
  <key>NSLocationWhenInUseUsageDescription</key>
  <string>L'application a besoin de connaitre votre position afin d'améliorer la présentation des services<
/string>
  <key>NSPhotoLibraryUsageDescription</key>
  <string>L'application a besoin de la bibliothèque de photos pour permettre de stocker temporairement le scan
de code-barre</string>
        <key>UILaunchStoryboardName</key>
        <string>LaunchScreen</string>
        <key>UIMainStoryboardFile</key>
        <string>Main</string>
        <key>UIRequiredDeviceCapabilities</key>
        <array>
                <string>armv7</string>
        </array>
        <key>UISupportedInterfaceOrientations</key>
        <array>
                <string>UIInterfaceOrientationPortrait</string>
                <string>UIInterfaceOrientationLandscapeLeft</string>
                <string>UIInterfaceOrientationLandscapeRight</string>
        </array>
        <key>UISupportedInterfaceOrientations~ipad</key>
        <array>
                <string>UIInterfaceOrientationPortrait</string>
                <string>UIInterfaceOrientationPortraitUpsideDown</string>
                <string>UIInterfaceOrientationLandscapeLeft</string>
                <string>UIInterfaceOrientationLandscapeRight</string>
        </array>
        <key>UIViewControllerBasedStatusBarAppearance</key>
        <true/>
</dict>
</plist>
```

Tester les apps

Tester l'app Web

Pour tester l'app en mode Web sur un navigateur, il est nécessaire d'avoir le client **ionic** d'installé sur la machine locale :

```
$ npm install -g @ionic/cli
```

Exécutez ensuite la commande suivante à la racine du client :

```
$ ionic serve
```

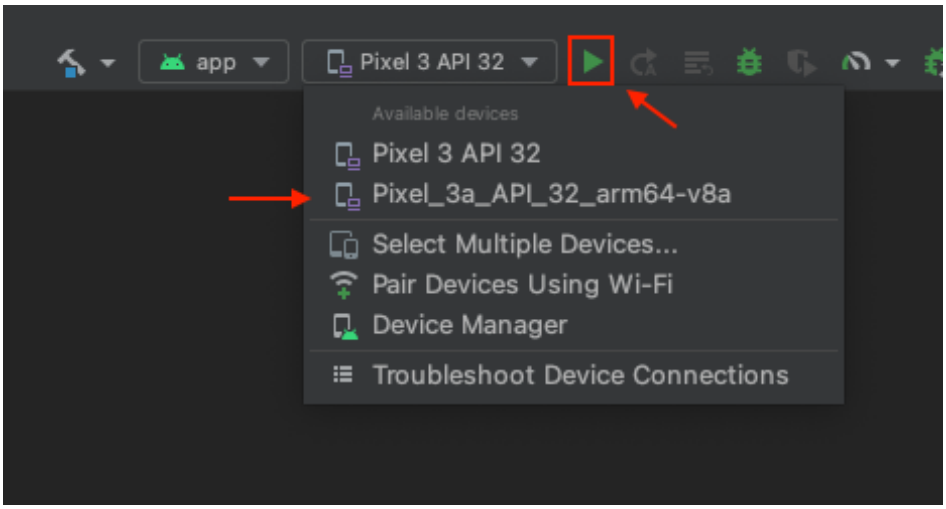
Le projet devrait alors se compiler et se lancer automatiquement sur une page de votre navigateur par défaut, joignable à l'adresse : http://localhost:8100

Tester l'app Android

Pour tester l'app sur Android, il faut ouvrir le projet dans Android Studio. Pour cela, exécutez la commande suivante :

```
$ npx capacitor open android
```

Une fois le projet ouvert dans Android Studio, il suffit de sélectionner le périphérique virtuel dans la barre de menu en haut, et de cliquer sur le bouton 'Run app' (triangle vert).
Android Studio va alors compiler le projet et le lancer dans un émulateur simulant le périphérique sélectionné.



Il est tout à fait possible de lancer le projet sur un périphérique réel connecté à la machine, mais cela nécessite un peu de conf réseau afin que le périphérique puisse accéder au backend et au CMS qui tourne localement

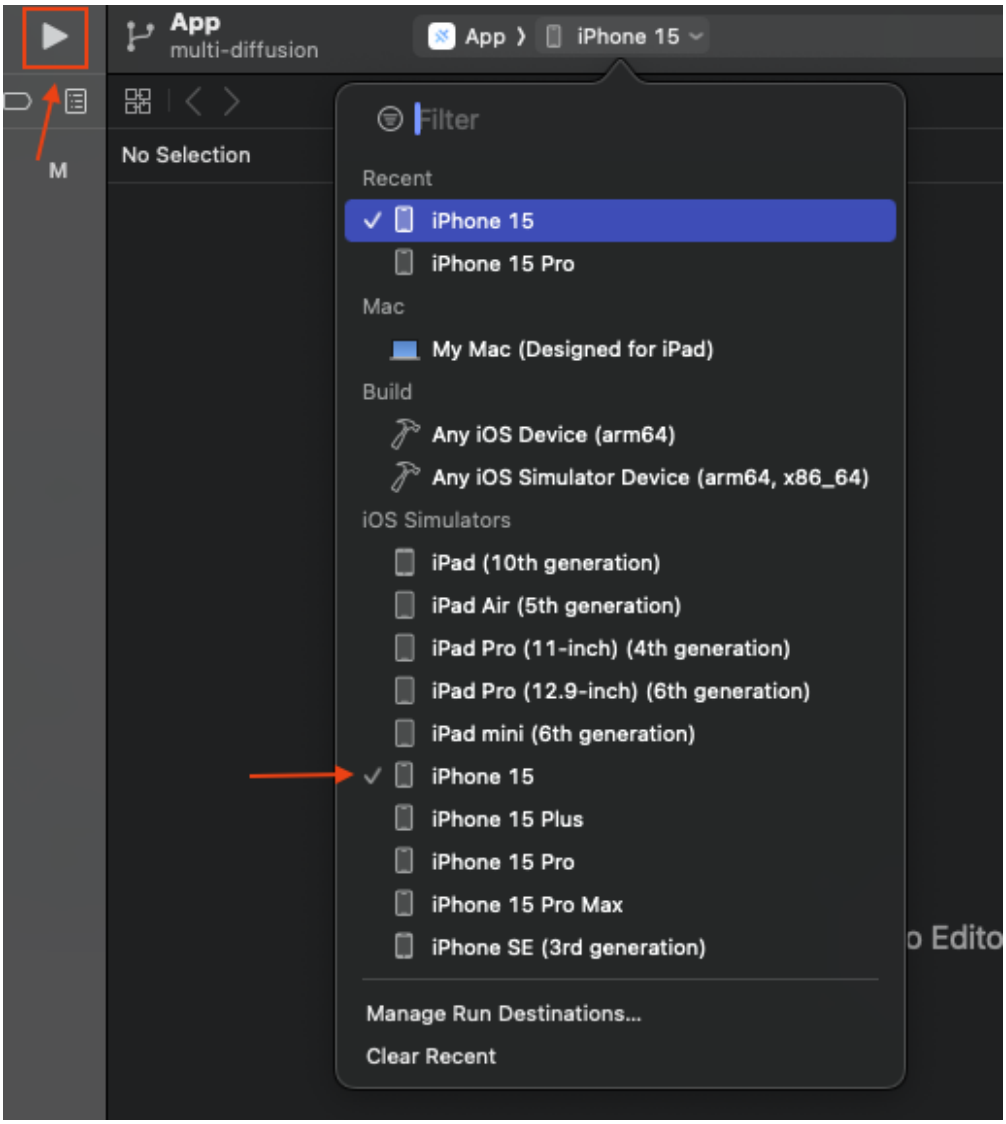
Tester l'app iOS

Les tests sur périphérique iOS ne sont possibles que depuis une machine tournant sur macOS (nécessite l'utilisation de XCode)

Pour tester l'app sur iOS, il faut ouvrir le projet dans XCode. Pour cela, exécutez la commande suivante :

```
$ npx capacitor open ios
```

Une fois le projet ouvert dans XCode, il suffit de sélectionner le périphérique virtuel dans la barre de menu en haut, et de cliquer sur le bouton 'Run' (triangle blanc).
XCode va alors compiler le projet et le lancer dans un émulateur simulant le périphérique sélectionné.



Utiliser les app

Une navigation anonyme offre quelques services.

S'authentifier avec les utilisateurs ayant pour *login* :

- "etu"
- "staff"
- "prof"

et ***mot de passe* identique au *login***, pour avoir des services supplémentaires et dédiés à chaque profil.