# Développement d'applications multi-plateforme : Mobioos & Ionic
# TD / TP 1

## Partie 1 - TD

### Exercice 1 : Le développement multi-plateforme

---

1. Quels sont les intérêts du développement multi plateforme ?
2. Quels en sont les inconvénients ?
3. Définissez les termes suivants :
    a. Application hybride
    b. Application native
    c. Application cross-platform

### Exercice 2 : Ionic

---

1. Quel type d'application Ionic permet il de construire ?
2. Qu'apporte Cordova à Ionic ?
3. Comment sont construites les vues d'une application Ionic ?
4. Expliquez ce qu'est un composant et ce qui le différencie d'une directive.
5. Qu'est ce qu'un "pipe" en Ionic ?
6. Expliquez ce qu'est un service et donnez ses cas d'utilisation.

### Exercice 3 : Mobioos

---

1. Qu'est ce qu'un "domaine" dans Mobioos Forge Studio ?
2. Que représente le domaine Concern ?
3. À quoi sert le domaine SmartApp ?
4. Quelle est la marche à suivre pour générer du code dans Forge ?
5. **À votre avis,** quels sont les intérêts de Mobioos Forge ?

# Partie 2 - TP - Prise en main

## Création d'une application de gestion de tâches

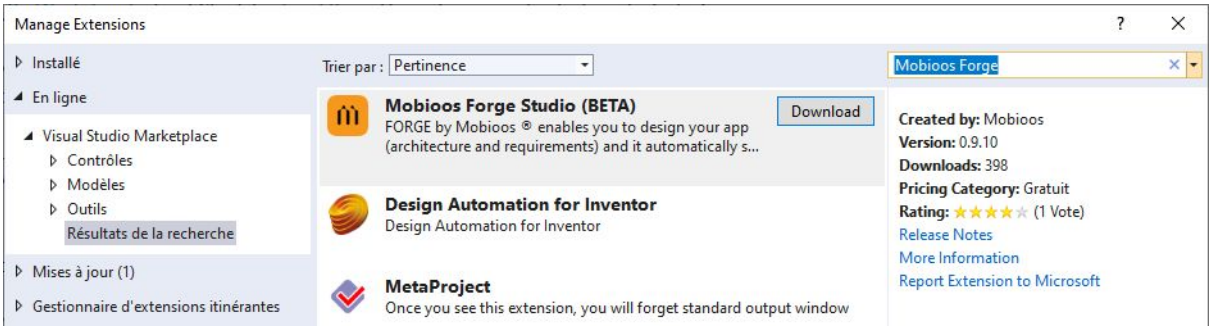
### Prérequis

Pour la modélisation :

- Visual Studio Community 2019
- Extension Visual Studio Mobioos Forge
- Un compte Mobioos (https://forge-app.redfabriq.com/#/account/signup)

Sur Visual Studio 2019 : Utilisez la boîte de dialogue "Gérer les extension" pour installer l'extension Mobioos Forge Studio

Extension > Gérer les extension > En ligne > Recherche : Tapez Mobioos Forge et téléchargez l'extension **Mobioos Forge Studio (BETA)**




Il faut ensuite quitter Visual Studio pour qu'il procède à l'installation de l'extension.

Pour l'implémentation :

- Node.js et NPM
- Ionic ( **npm i -g ionic** )

Pour tester sur Android :

- Au moins un SDK android (28), le plus simple est d'installer Android studio et d'utiliser le SDK manager.
- Cordova ( **npm i -g cordova** )
-

### Spécifications

---

Nous allons créer une petite application de gestion de tâches dans laquelle nous devons être en mesure de créer / supprimer / modifier des tâches, marquer ces tâches comme complétées et en afficher la liste.
Cette application ne contiendra donc que deux vues, la vue de liste et la vue de création / modification de tâche (celle ci n'a pas forcément besoin d'être une page à part entière et peut également être séparée en deux vues distinctes).

## Exercice 1 : Modélisation de l'application

---

1. Créer un nouveau projet
   Fichier > Nouveau > Projet > Visual c# > Mobioos > Forge Studio Empty Project
2. Modélisez les différents domaines.

Il est conseillé de commencer par la modélisation du domaine Concern et de modéliser les autres domaines lorsque le besoin apparaît. Pour une application plus lourde il sera préférable de réfléchir aux APIs nécessaires en amont pour éviter les redondances dans la modélisation (et donc dans le code généré).

La modélisation doit contenir une page racine, ce sera la page affichant la liste de tâches, pour ce faire il faudra changer la valeur de la propriété **isRoot** du layout en question à *true*.

3. Validez le modèle et procédez à la génération du code front end en Ionic.

Pour valider tous vos fichier d'un seul coup, depuis le diagram SmartApp, clic droit > Validate all.

Accédez à la fenêtre de génération en faisant un clic droit > Generate code.

Connectez vous en utilisant votre compte Mobioos (https://forge-app.redfabriq.com/#/account/signup) puis choisissez où télécharger le code généré.

Dans l'interview, choisissez la génération d'un Front end en Ionic et répondez aux autres questions.

4. Ouvrez le code généré dans votre IDE et testez l'application.

Il faut d'abord lancer la commande **npm i** pour procéder à l'installation des modules nécessaires au fonctionnement d'Ionic dans le projet.

La commande **ionic serve** permet ensuite de lancer l'application dans un navigateur.

## Exercice 2 : Implémentation de *l'exocode*

---

1. Implémentez la vue de liste.

Il s'agit d'une vue contenant une liste, il s'agit alors de créer une boucle (***ngFor**) permettant d'afficher la donnée provenant d'une liste gérée dans notre composant.

Il faut également un affichage différent en fonction de l'état de la tâche (si elle a été complétée ou non), ce genre de traitement peut être fait de plusieurs façon, en utilisant l'attribut ***ngIf** qui permet d'afficher l'élément concerné uniquement si la condition est remplie ou en créant une pipe par exemple.

Il faut ajouter un handler dans le cas du clique sur une tâche, on utilise pour cela l'attribut **(click)="myMethod()"** permettant d'exécuter une méthode au moment du clique sur un élément.

Il faudra aussi un bouton en marge de chaque tâche, on peut utiliser le composant **<ion-button>** ou une simple icône attachée à un handler de clique.

À vous de voir quel élément sert à quoi, l'un doit marquer la tâche comme complétée, l'autre doit permettre d'accéder à un menu pour supprimer ou modifier la tâche.

Manque un dernier bouton permettant d'accéder à la création de tâche.

2. Implémentez le composant associé à cette vue.

Il faut créer la liste correspondante à la liste de tâches contenant les données nécessaires à notre affichage. Cette liste doit être publique afin d'être accessible depuis le template.

Il faut ensuite implémenter le code métier des méthodes générées.

3. Implémentez la vue et le composant des autres pages
4. Modifier vos services pour sauvegarder les tâches localement et ne pas faire de requêtes http.

Un exemple de module permettant de manager une base de données locale : https://ionicframework.com/docs/native/sqlite

## Exercice 3 : Test de l'application sur un appareil

---

1. Ajouter la plateforme android

**ionic cordova platform add android**

2. Lancer l'application dans un appareil ou dans un emulateur

**ionic cordova run android**

Il faut alors avoir configuré les options de développement sur le mobile de test et accepter le débogage USB (https://www.frandroid.com/comment-faire/tutoriaux/229753_questcequelemodededebogageusb).

## Exercice 4 : Utiliser les ressources de l'appareil

---

1. Modifier la page de création de tâche de manière à :
    a. Pouvoir choisir un "type" de tâche (Sport, Loisir … ).
    b. Pouvoir choisir une date/heure et configurer une notification pour cette tâche.
    c. Pouvoir prendre une photo ou sélectionner une photo de la galerie à lier à cette tâche.

Module de gestion de notifications locales:
https://ionicframework.com/docs/native/local-notifications

Module de gestion de la Camera: https://ionicframework.com/docs/native/camera

2. Utilisez le CLI Ionic pour ajouter une page à l'application permettant d'afficher les détails d'une tâche, affichant ainsi le type de tâche et l'image sauvegardée, indiquant si une notification est prévue pour cette tâche et si oui à quelle date / heure.

## Aller plus loin

---

1. Rouvrez votre modélisation et générez cette fois un code back end en ASP.NET Core.
2. Ouvrez le code ainsi généré et implémentez les routes d'API permettant le fonctionnement de votre application sur la base du code généré.
3. Modifier le code front end pour permettre une sauvegarde des données sur serveur via les routes d'API plutôt que localement.
4. Implémentez un système permettant de sauvegarder ces données localement après les avoir récupérées du serveur pour palier à une potentielle perte de connexion.
5. Implémentez un système permettant, en cas de perte de connexion, de sauvegarder localement les tâches nouvellement créées jusqu'à retrouver la connexion au serveur et ainsi synchroniser les données.
6. Faites de même pour la modification des tâches hors connexion.

Module de gestion de connexion: https://ionicframework.com/docs/native/network