# Point évaluer

## Application, maquettes

Installation de l’environnement :   
aller installateur, modifier instal et choisir xamarin, être admin, librairie à instal, package, sdk.

Création du projet :   
choisir le bon projet avec filtre Xamarin, donner un nom explicite, endroit du projet, choisir un projet vide, version minimale de l’androïde (version haute = nouveauté mais non fonction sur ancien appareil)

Explication de l’environnement de développement :   
ressemble à C# dire quel fichier est outil

Emulateur (compilation exécution) :   
dire ce que c’est un émulateur (simuler un téléphone) avantage avoir plusieurs version, taille de l’écran etc. ou déployer sur élément physique. Donc décrire le gestionnaire d’émulateur

Activité et lien entre fichier .cs et fichier .xml :  
c’est quoi une activité (composer d’’un fichier .cs et .xml), une activité fonctionne pour elle-même. Quand on ouvre une activité, toute les autres ce ferment. Ou ce trouve le lien entre les fichiers

## Fichier de ressources :

C’est là ou sera stocker des « constantes » pour tout l’environnement (pas seulement une activité), pratique pour changer la langue ou le mode jour/nuit. Aussi faire des fichiers en fonction de la taille de l’appareille.

## Cycle de vie :

Différant état qu’un activité peut avoir : activée, en pause, arrêtée et redémarrée. Avec leur méthode (voire cachier) ! quand on est en OnPause() libérer les ressources et enregistrer (libérer le sensore) puis utiliser le OnResume()

## Création des composants dans un layout (xml) :

Comment on crée un composant dans un fichier xml avec balise. Description de quelque attribue (couleur, texte). Identifiants (@+id) unique dans le programme. Les différant layout. Drag and drop

## Lien entre un composant et le code (+ évènement) :

Récupération par id. classe correspondant pour chaque composant xml. Dire où on peut faire les choses. Décrire comment lien à une méthode événementielles (choix du nom de la méthode mais nom proposé) paramètre (sender objet qui à lancer la méthode + eventArgs donne info sur élément)

## Création des composants dynamiquement :

Comment créer l’élément dans le .cs puis l’afficher dans le layout puis les paramétrer. Certain paramètre sont des appels de méthode

## Lien entre deux activités :

Emulateur doit être éteint. Créer une deuxième activité. Conseil d’avoir la même nomenclature. Comment appeler cette deuxième activité.

## ListView :

Boîte qui va gérer les données et autre qu’on donne via un adaptateur, on donne les data et un layout/affichage. On doit donner les données et comment on veut afficher les string. Quelques layout sont à disposition. Le layout = une zone avec un texte. On dit comment on veut afficher les éléments à afficher. Le listview va ensuite afficher autant de données que ce qu’on lui à donner. Pourquoi donner un layout ? => pour dire comment mettre les données dans le xml et quelles données va où. Créer un répertoire model (donnée) et service (controlleur)

Model va contenir (une classe avec donnée)

Service va contenir (controleur, getTask, addTask(donnée), etc…)

Création d’une classe adapteur

Class AdapteurTask va contenir obligatoirement (constructeurm GetItemId, Task this[int position], Count, Java.Lang.Object GetItem, GetView)

## BD :

Ajouter la PCL (Portable class Librairies) SQLite-Net en utilisant le gestionnaire de packet NuGet intall « sqlite-net-pcl ». pour chaque table on crée un model et un service (repository). [Table(‘’nomtable‘’)] change le nom de la table sans pour autant changer le nom de la classe

Repository aura les méthodes pour récupérer les données.

## Senseur :

Il en existe plus différant. On travail avec un sensorManager qui va gérer tous les capteurs de notre appareil. Peut ressortir toute la liste de capteur que possède l’appareil. Avoir une classe sensor et sensormanager et mettre un listener sur le sensor. Ex : sensor = sensormanager.accelerometer ; listener(sensor). Utiliser une methode nommé OnSensorChanged qui va être appeler à chaque fois qu’il reçoit une information. À la fin, il faut désenregistrer le capteur lorsque l’activité est en pause. Il faut faire un héritage sur la classe activité. Ex : MainActivity : ISensorEventListener