Table des matières

[1 Introduction 1](#_Toc116481787)

[2 Application 1](#_Toc116481788)

[3 Maquette 2](#_Toc116481789)

[4 Installation de l’environnement 3](#_Toc116481790)

[5 Création du projet 3](#_Toc116481791)

[6 Création d’une première Activité 4](#_Toc116481792)

[6.1 Les fenêtres de l’environnement 4](#_Toc116481793)

[6.2 Interface utilisateur 4](#_Toc116481794)

[6.3 Layouts 5](#_Toc116481795)

[6.3.1 LinearLayout 5](#_Toc116481796)

[6.3.2 RelativeLayout 5](#_Toc116481797)

[6.3.3 TableLayout 6](#_Toc116481798)

[7 Fichier de ressources 6](#_Toc116481799)

[8 Lien entre deux Activités 7](#_Toc116481800)

[8.1 Création d’une deuxième Activité 7](#_Toc116481801)

[8.2 Lié les Activités 7](#_Toc116481802)

[9 Méthodes événementielles 8](#_Toc116481803)

[10 Layout dynamique 9](#_Toc116481804)

[11 Persistance de données 9](#_Toc116481805)

[12 Utilisation d’un sensor 9](#_Toc116481806)

# Introduction

Ce document a pour objectif de mettre en évidence les différentes étapes dans la réalisation d’une application Android.

Les explications qui seront précisé dans ce document seront pour des personnes déjà familiarisées avec C#.

Même si MS annonce la fin de cette plateforme avec l’arrivée de .Net Maui, nous utiliserons l’environnement Visual Studio C# avec Xamarin.

# Application

L’application sera un gestionnaire de tâches afin de passer par les différentes étapes de réalisation d’un projet sur mobile.

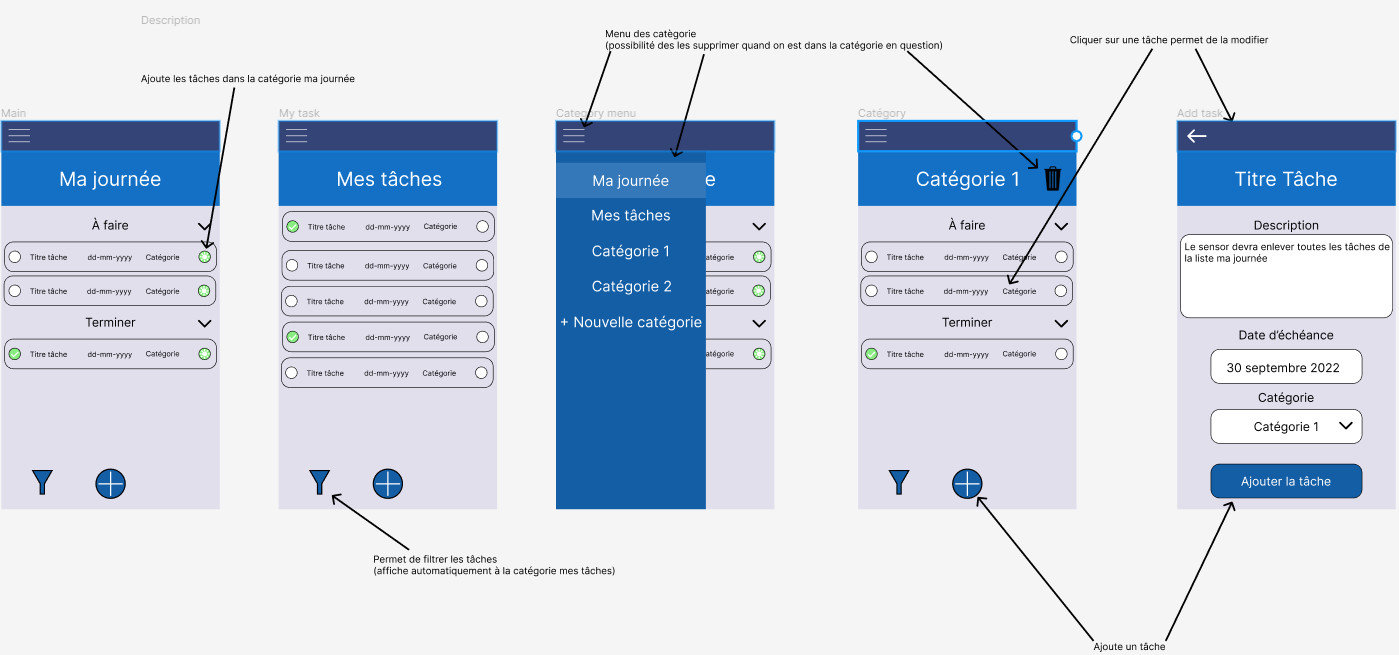
Cette application doit avoir au minimum les fonctionnalités suivantes :

* Pouvoir créer et visualiser des tâches.
* Une tâche est caractérisée au minimum par son titre, sa description et sa date d’échéance.
* Il doit être possible de mettre une tâche dans une catégorie.
* Il doit être possible de sélectionner des tâches pour en faire une liste « Ma journée », c’est-à-dire choisir les tâches à réaliser aujourd’hui.

De plus, cette application doit permettre d’aborder les thèmes suivants :

* Création d’une première Activité.
* Utilisation d’au moins un fichier de ressources, comme « Strings ».
* Lien entre deux Activités.
* Méthodes événementielles.
* Layout dynamique (c’est-à-dire création dans le code C#)
* Persistance de données.
* Utilisation d'un sensor.

# Maquette



Deux types interfaces sont prévues. Lors du lancement de l’application, on tombera sur la première interface qui affiche les tâches à faire de la journée. On y ajoute des tâches choisies en cliquant sur le cercle droit de la tâche (le cercle gauche permet de finir la tâche).

En cliquant sur les trois barres en haut à gauche, l’utilisateur aura accès aux autres sections. Il sera possible de créer, renommer et supprimer les sections.

En cliquant sur le + en bas de l’écran, on peut créer une nouvelle tâche. Avec un titre, une description, une date d’échéance et une catégorie (la catégorie de base est « Mes tâches »). Il est aussi possible de modifier une tâche en cliquant sur la tâche en question.

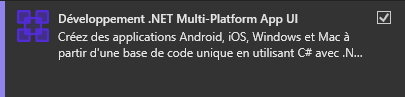
Si le temps le permet, on filtre sera ajouter permettant de trier par catégorie, date, terminée, non terminée.

# Installation de l’environnement

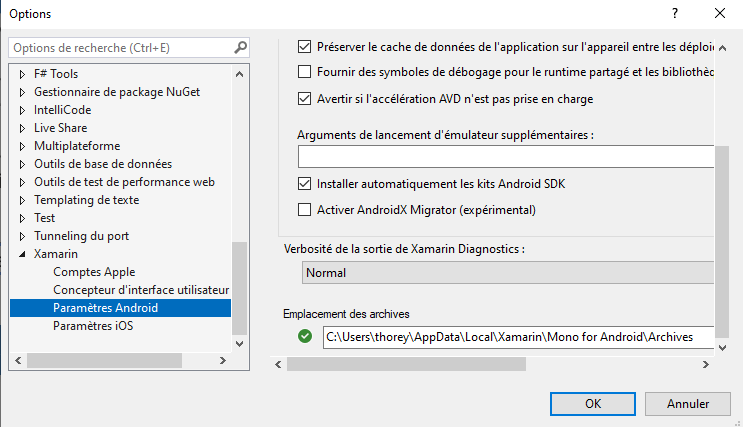
Une fois Visual Studio installé, il faut y intégrer l’environnement Xamarin. Pour cela, il faut lancer Visual Studio Installer et Modifier l’installation de Visual Studio 2022.



Le package « Développement mobile en .Net Multi-Platform App UI » doit être sélectionné.  
Attention : il faut le droit administrateur



Le développement pour Android reposant sur le SDK Java et sur le SDK Android, il est possible de le configurer dans Visual Studio. Menu Outils -> Options -> Xamarin -> Paramètres Android.



Après que le premier projet est créé, il faut aussi

# Création du projet

Lors de la création d’un projet, une étape supplémentaire sera ajoutée comparé à la création d’un projet normal C#.

Il s’agit d’un menu de sélection du modèle à utiliser, dans notre cas nous choisirons « Application vide » pour répondre à la contrainte de l’explication de la création d’une première Activité.

Choisir la version d’Android minimale. Pour cette partie, plus nous choisissons une récente, plus nous aurons des fonctionnalités récentes à disposition. La solution semble être de choisir la version la plus récente. Cependant cela aura comme conséquence que notre application ne pourra tourner que sur une portion minimale des supports Android existants sur le marché. En effet peu de personnes ont un support Android de dernière génération. Il faut donc choisir la version selon le publique cible.

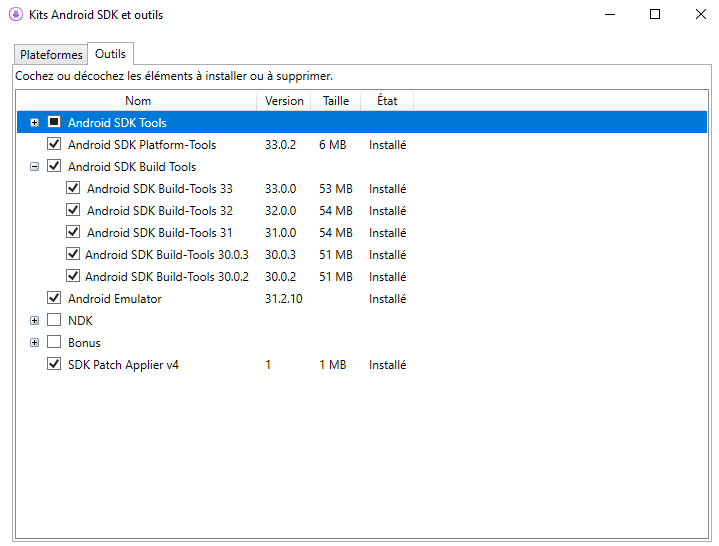


# Emulateur

Avant de regarder l’environnement, il faut configurer l’émulateur pour qu’il puisse faire tourner notre application. Un émulateur est un outil qui permet de simuler un outil informatique comme un smartphone ou une console de jeux sur ordinateur. On peut aussi utiliser un smartphone physique.

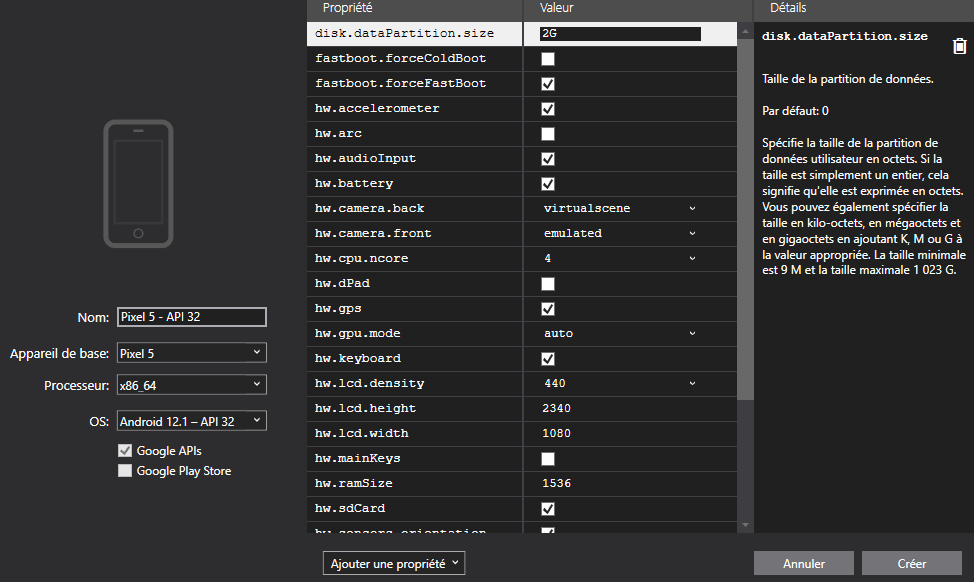
Ces deux méthodes on leurs avantages et leurs inconvénients. Le support physique nous permet de mieux se rendre compte de l’utilisateur du sensor, s’il est trop sensible ou non. Alors que l’émulateur lui nous offres une multitude de version, de taille ou d’appareille pour notre application. Il est donc conseiller d’utiliser les deux méthodes.

Pour utiliser l’émulateur, il faut aller dans Outils->Android->Gestionnaire SDK Android…  
Puis aller dans Outils de la fenêtre qui vient de s’ouvrir et cocher « Android SDK Build Tools » et « Android Emulator ». Puis appliquer les modifications.



Une fois fait, il faut créer notre émulateur. Pour cela, on clique sur le bouton  ou aller dans   
Outils->Androide-> Gestionnaire d’appareils Android

Après cela, on clique sur « Nouveauté » et on décide de ses propriété.

  
Et un fois cela fait, il ne manque plus que de le créer et de la démarrer.

# Environnement et première Activité

En créant le projet, la première activité (vide) est créée. Celle-ci est composée de deux fichiers principaux :

* MainActivity.cs
* Activity\_main.xml

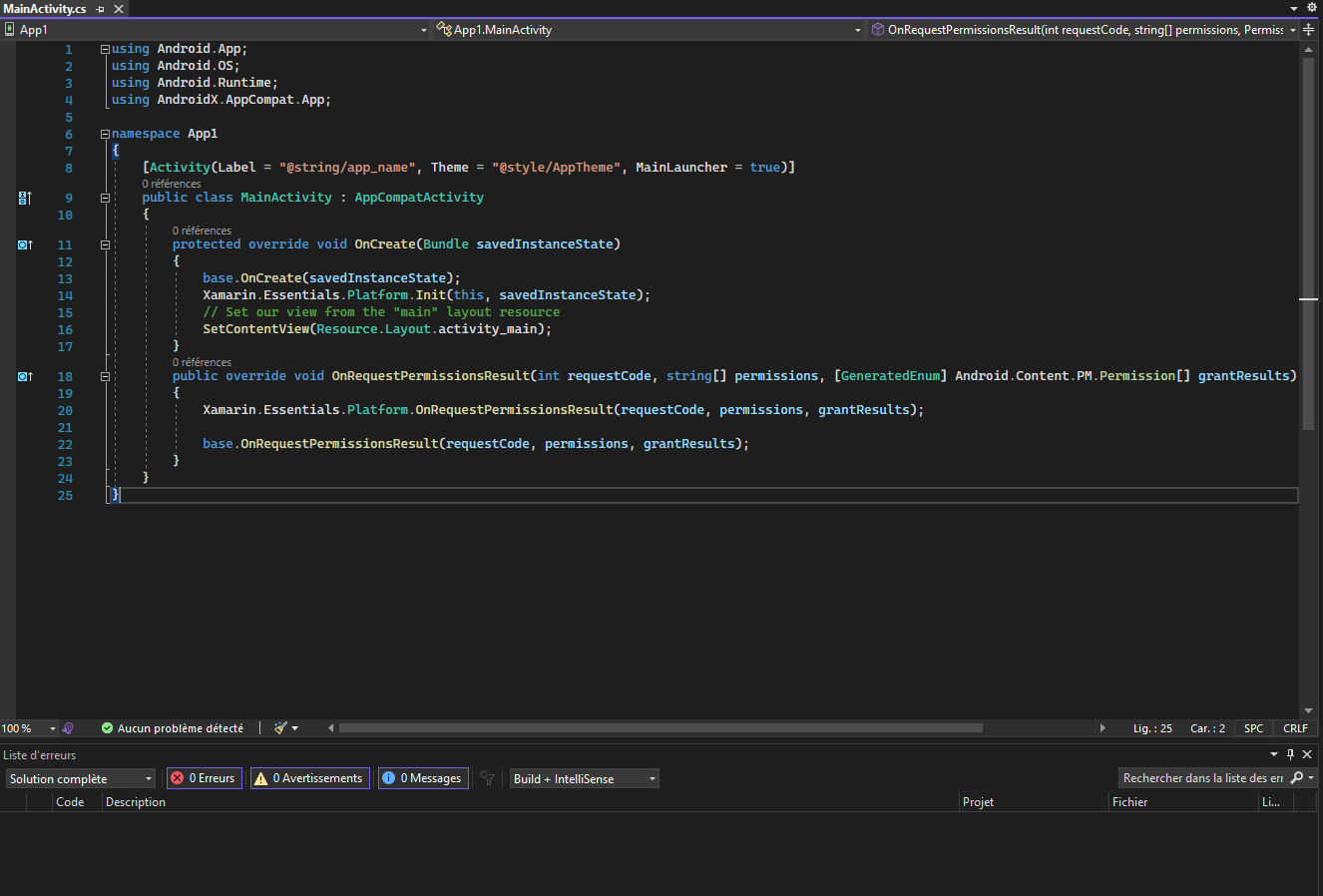
MainActivity.cs contient le code C# et les méthodes événementielles. Il s’agit de la partie C# de la première activité. C’est aussi l’activité principale puisque c’est celle qui est lancée à l’ouverture de l’application.

Activity\_main.xml contient les balises xml qui permettent de créer l’interface utilisateur qui sera affichée. C’est celle qui va contenir notre interface « ma journée ».

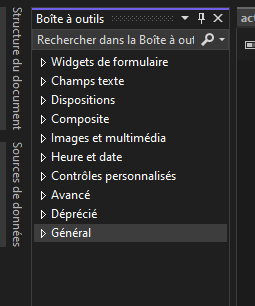
Une activité fonctionne pour elle-même, c’est-à-dire que quand on ouvre une activité, toutes les autres se ferment.

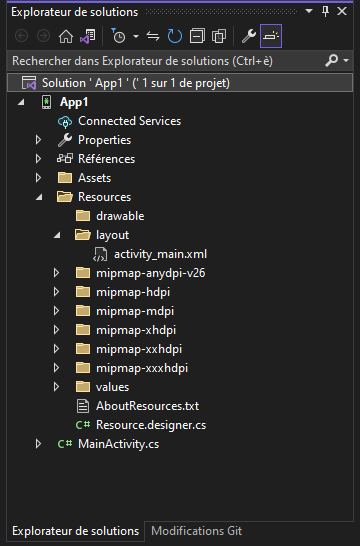
## Les fenêtres de l’environnement

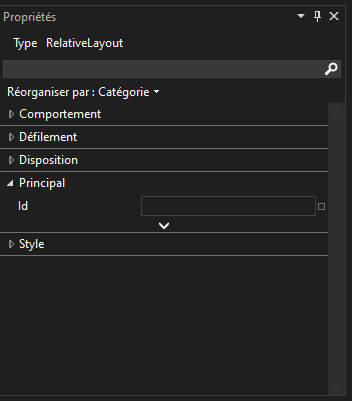
L’environnement s’ouvre avec différentes fenêtres à disposition.

La partie centrale contient la fenêtre de développement. C’est d’ailleurs « MainActivity.cs » qui est affiché et on peut voir que la liaison entre le MainActivity.cs et l’Activity\_main.xml grâce à ce bout de code :   
  
Comme pour un projet C# normal, le bas contient les erreurs et les avertissements que contient votre code.

A gauche, on peut trouver aussi la « Boîte à outils ». Elle contient le « Widgets » lorsque nous sommes sur la partie interface utilisateur. Donc il est normal que lorsque nous sur le MainActivity.cs, il soit vide.



A droite, nous trouvons « l’explorateur de solutions » qui nous permettra de sélectionner les différents fichiers comme par exemple, le code C#, les balises xml ou les fichiers de ressources.   


On peut trouver enfin la fenêtre des « propriétés ». Comme pour la fenêtre de la « Boite à outils », les propriétés des Widgets s’afficheront lorsque nous somme sur la partie interface utilisateur.   


En somme, l’environnement ressemble à celui d’un projet Windows Form.

## Interface utilisateur

Pour mettre en place l’interface utilisateur, il faut ouvrir le fichier xml correspondant. Pour cette première Activité, il s’agit du fichier « Activity\_main.xml ». Pour ceci. Il faut aller dans « l’explorateur de solutions » est aller dans le répertoire « Resources » et « Layout ».



Lorsque le fichier est sélectionné, la fenêtre correspondante est partagée en deux. A gauche se trouve une fenêtre de création graphique où il est possible de mettre des Widgets en Drag/Drop. A droite, nous trouvons le correspondant en balises xml.

## Layouts

Si l’on regarde du côté des balises, on peut voir que le premier Widget est un Relative Layout.

Les layouts sont des boîtes qui permettent de mettre en communs plusieurs Widgets et les traiter ensemble.

Il existe différents types de Layout, chacun ayant ses spécificités.

### LinearLayout

Ce type de mise en page permet d’organiser une liste d’éléments de façon horizontale ou verticale. Chaque nouvel élément venant automatiquement se placer en dessous ou à droite de l’élément précédent.



### RelativeLayout

Ce type de mise en page permet d’organiser les éléments les uns en fonction des autres, donc de façon relative. Il est ainsi possible de spécifier qu’un champ de texte soit au-dessus d’un champ de saisie ou qu’une image soit centrée dans son composant parent.

### GridLayout

# Fichier de ressources

Pour afficher les différents fichiers de références, il faut aller dans « l’explorateur de solutions » est aller dans le répertoire « Resources ».

Il y a trois dossiers importants dans le répertoire « Resources » : drawable, layout et values.



Le dossier « drawable » permet d’afficher des éléments qu’on ne peut pas reproduire dans les Layout, comme par exemple les images.  
Le dossier « layout » est l’endroit où sera contenu tous les fichier xml qui serviront à l’affichage de l’interface.  
Le dossier « values » quant à lui, sert à stocker toutes les données (constante) que nous allons utiliser pour les layouts ou la persistance des données. Les strings seront stockés dans strings.xml, les couleurs dans colors.xml et ainsi de suite. Pour faire appel à ces valeurs, il suffit de commencé par ‘’@leNomDuFichier/leNomDeLaValeurs‘’.  
Exemple de constante :   
Exemple d’appelle : 

Les fichiers resources sont très pratique car il permet de les utiliser pour changer la langues ou montrer un affichage selon la taille de l’écran.

# Cycle de vie

Une activité possède ce qu’on appelle un état et il y en a plusieurs.

## Active (Activée)

C’est l’état quand l’activité est au premier plan, c’est donc l’activité principale que l’utilisateur peut voir et avec laquelle il va interagir directement. L’OS considère alors que celle-ci est prioritaire sur toutes les autres et ne sera fermée que dans des cas très rares comme une consommation de mémoire trop importante.

## Paused (En pause)

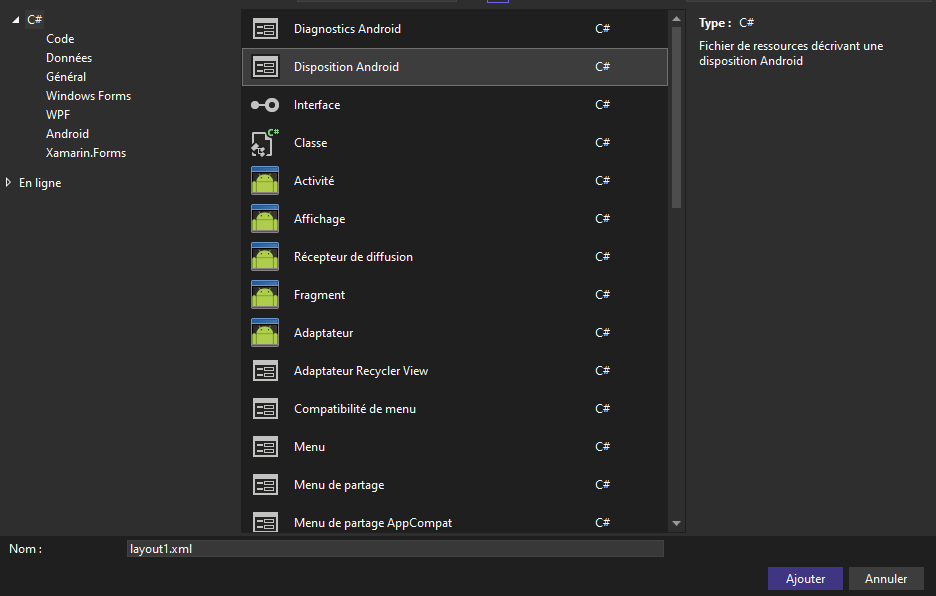
# Lien entre deux Activités

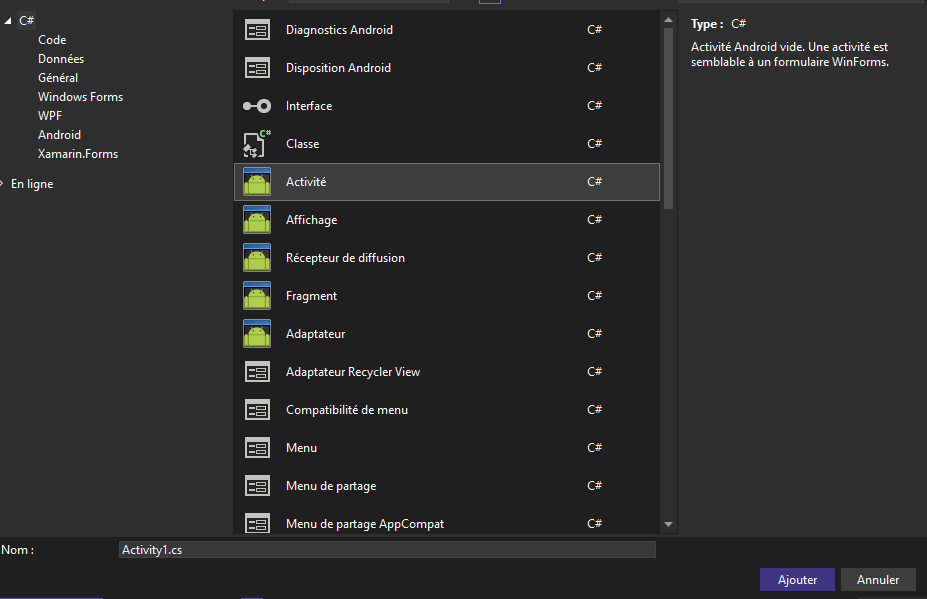
Pour créer un lien entre deux activités, il faut déjà en avoir deux, or nous en possédant qu’une qui est « Activity\_main.xml ». Il nous faut donc créer la deuxième.

## Création d’une deuxième Activité

Pour Créer une nouvelle Activité, il faut aller dans l’Explorateur de solution puis faire un clique droit sur le dossier layout et appuyer sur « Ajouter -> Nouvelle élément » ou alors faire Ctrl+Maj+A.  
Une image contenant texte, moniteur, capture d’écran, écran

Description générée automatiquement

Une fois cela fait, il faut sélectionner « Disposition Android » et nommé notre nouvelle Activité.  


Une fois le fichier .xml créer, il faut faire la même chose mais sur la solution pour créer un fichier .cs en sélectionnant « Activité » cette fois-ci.   


Le fichier .cs contiendra ceci :  
Une image contenant texte

Description générée automatiquement

Il ne manque plus qu’as lié le fichier .xml au .cs  en ajoutant ce bout de code:  


## Lié les Activités

Il y a deux manières pour lier des Activités entres elles. La première est la plus simple, il suffit d’appeler la méthode « StartActivity(typeof(MonActivité.cs))». De cette manière l’Activité appelée sera afficher sans problème :  


La deuxième manière est plus longue mais permet de faire passer des données entre les Activités.

Il suffit de mettre dans une variable « Intent » l’Activité à afficher est les données à passer  
Une image contenant texte

Description générée automatiquement

Une fois les données passées, il faut les récupérer dans la seconde Activité avec un TextView ou autre élément  
Une image contenant texte

Description générée automatiquement

# Méthodes événementielles

Les méthodes événementielles sont des méthodes qui sont appelées quand l’utilisateur fait une certaine action dans l’interface. Par exemple quand on appuie sur un bouton.

Pour cela il faut créer une méthode avec des paramètre « object sender » et « EventArgs e » puis mettre le code que vous voulez dans celle-ci.  
Une image contenant texte

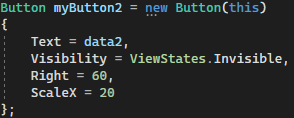
Description générée automatiquement

Puis lié cette méthode à un événement d’un élément.  

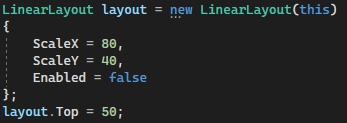

Si on ne sait pas quel événement sélectionner il faut prendre pour une action, on peut appuyer sur  lorsque VS nous donne des propositions.  
Une image contenant texte

Description générée automatiquement

# Layout dynamique

Le layout dynamique consiste d’ajouter des éléments dans le xml depuis le code. Pour créer un nouvel élément il suffit de mettre la variable, son nom et ses propriétés.  


Le bouton sera ajouté automatiquement au chargement de l’application.  
si on veut ajouter une propriété après avoir instancier l’élément, il suffit de faire comme ceci :  


Autre exemple :  


afficher l’élément sur le layout. Certain paramètre sont des appelles de méthode

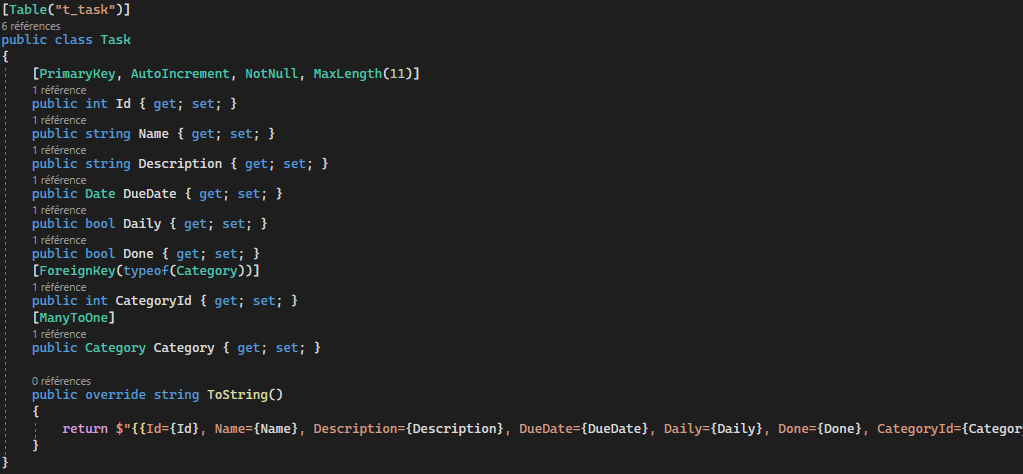
# Persistance de données

La persistance de données est l’équivalant de la base de données sur les smartphones. Pour faire une persistance de données, il faut crée trois fichiers CS : la base de données, le repository et l’adapter.

Pour les exemples qui vont suivre (et toujours en rapport avec notre projet), nous allons prendre les données pour les tâches.

## Base de données

Pour la base de données on peut créer soit un fichier



# Utilisation d’un sensor