

Table des matières

1	Introduction.....	1
2	Application.....	1
3	Maquette.....	2
4	Installation de l'environnement.....	3
5	Création du projet	3
6	Création d'une première Activité.....	4
6.1	Les fenêtres de l'environnement	4
6.2	Interface utilisateur	4
6.3	Layouts	5
6.3.1	LinearLayout	5
6.3.2	RelativeLayout	5
6.3.3	TableLayout	6
7	Fichier de ressources.....	6
8	Lien entre deux Activités.....	7
8.1	Création d'une deuxième Activité.....	7
8.2	Lié les Activités	7
9	Méthodes événementielles.....	8
10	Layout dynamique.....	9
11	Persistance de données.....	9
12	Utilisation d'un sensor.....	9

1 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.

2 Application

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

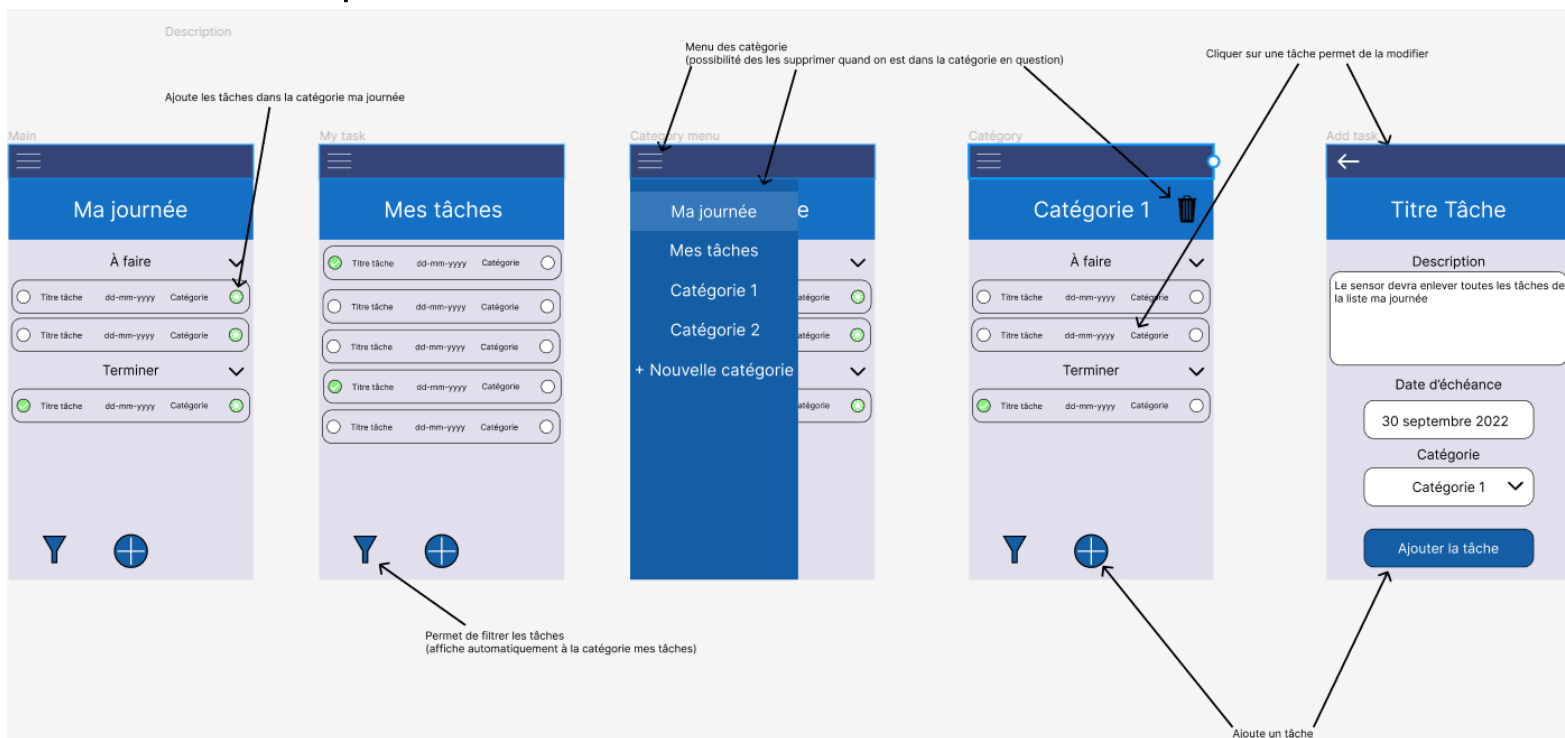
Cette application doit avoir au minimum les fonctionnalités suivantes :

- 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.

3 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.

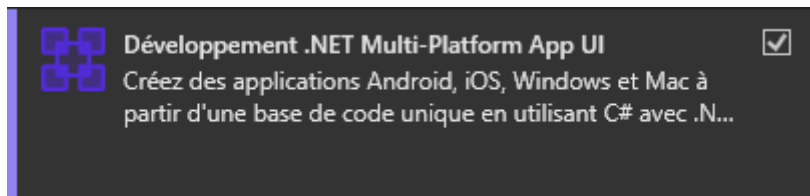
Si le temps le permet, on filtre sera ajouter permettant de trier par catégorie, date, terminée, non terminée.

4 Installation de l'environnement

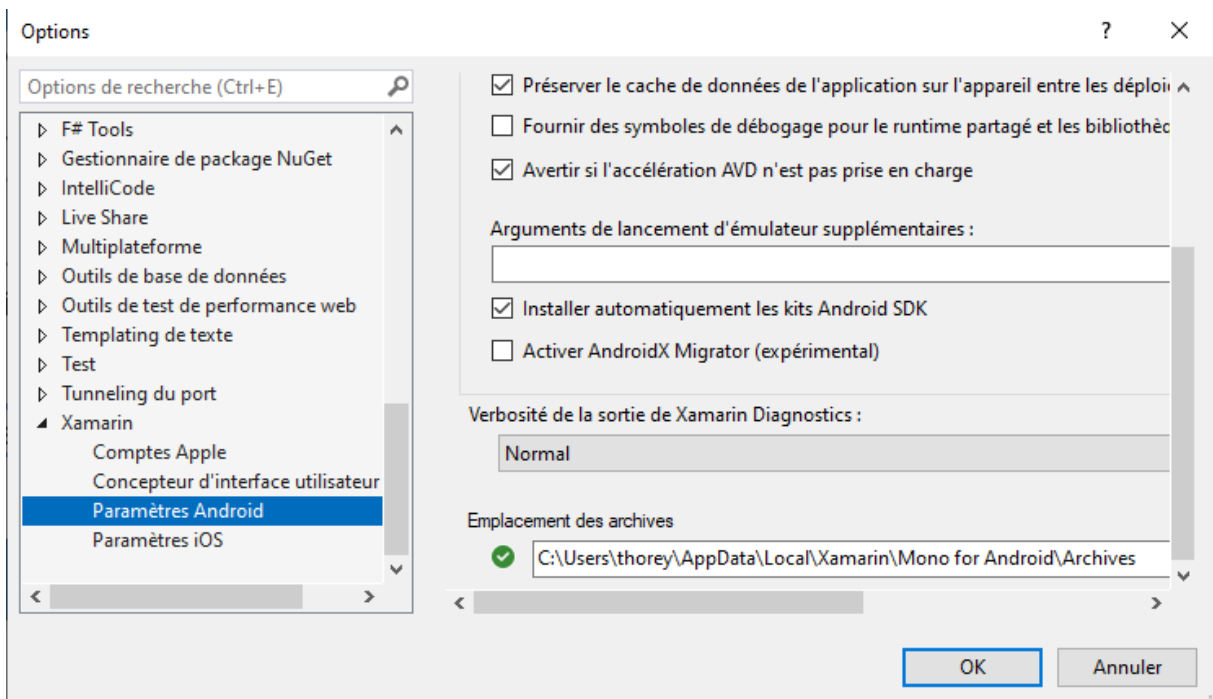
Une fois Visual Studio installé, il faut y intégrer l'environnement Xamarin. Pour cela, il faut lancer Visual Studio Installer et Modifier l'installation de Visual Studio 2022.



Le package « Développement mobile en .Net Multi-Platform App UI » doit être sélectionné.



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.

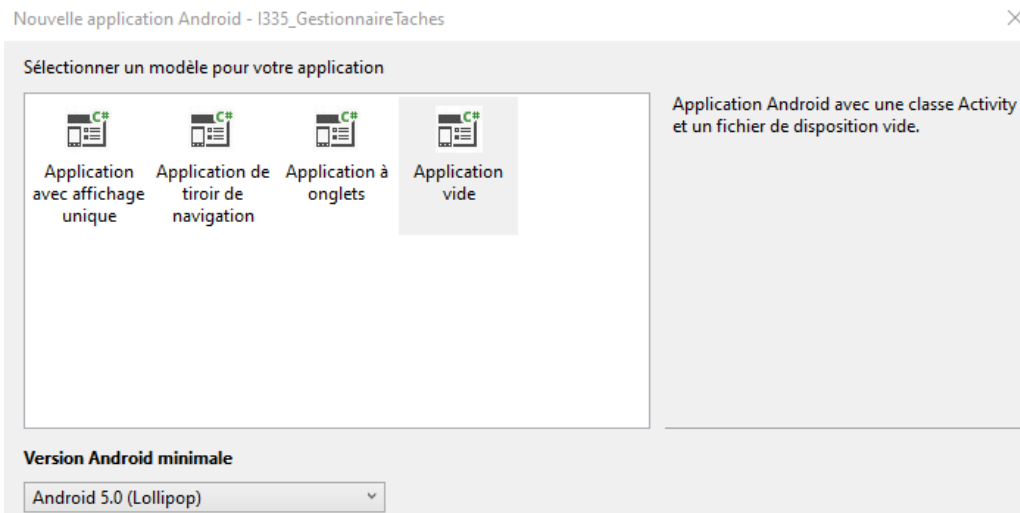


5 Création du projet

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.



6 Création d'une 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 ».

6.1 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.

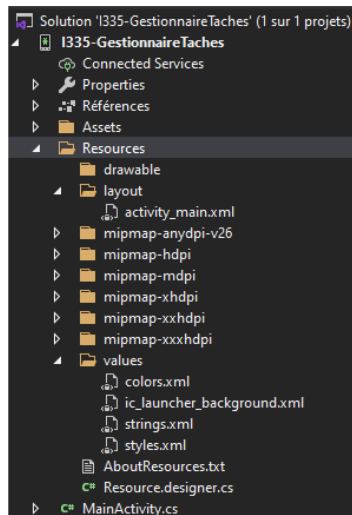
A gauche, on peut trouver aussi la « Boîte à outils ». Pour le moment elle est vide, mais elle contiendra le « Widgets » lorsque nous serons sur la partie interface utilisateur.

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 ». Elles aussi est vide pour le moment mais comme pour la fenêtre de la « Boîte à outils », les propriétés des Widgets s'afficheront lorsque nous serons sur la partie interface utilisateur.

6.2 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. À gauche se trouve une fenêtre de création graphique où il est possible de mettre des Widgets en Drag/Drop. À droite, nous trouvons le correspondant en balises xml.

6.3 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.

6.3.1 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.



6.3.2 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.

6.3.3 TableLayout

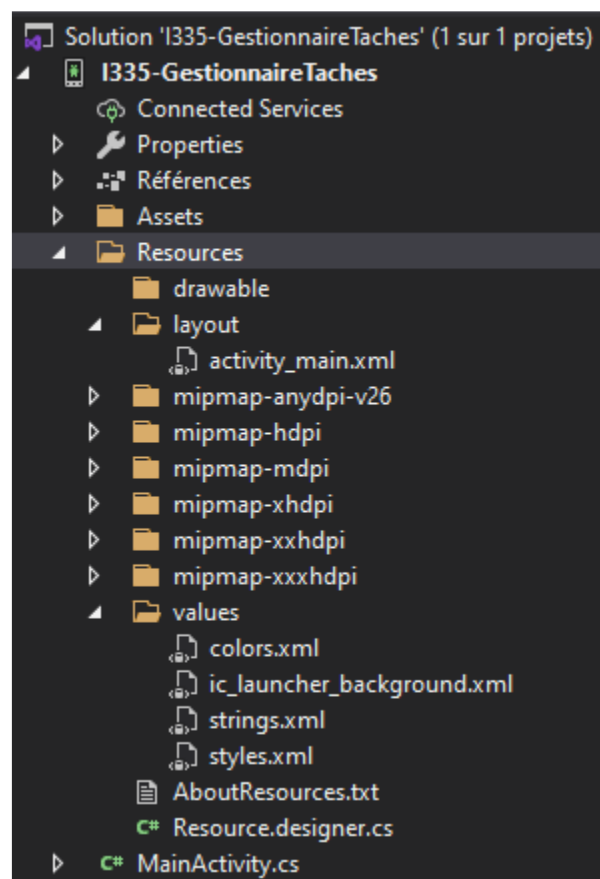
Lorsque l'on souhaite afficher des éléments sous la forme d'un tableau, avec lignes et colonnes, il faut utiliser la classe TableLayout. Elle se compose d'une liste de lignes (TableRow) pouvant avoir une ou plusieurs cellules associées.

Il est possible de rechercher ces layout ainsi que d'autre balise dans la « boîte à outils » qui se trouve à gauche de l'interface utilisateur. Si elle n'est pas présente, on peut la faire afficher en cliquant sur Affichage -> Boîte à outil.

7 Fichier de ressources

Pour afficher les différents fichiers de références, il faut aller dans « l'explorateur de solutions » et 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 fichiers xml qui serviront à l'affichage de l'interface.

Le dossier « values » quant à lui, sert à stocker toutes les données 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 commencer par "@leNomDuFichier/leNomDeLaValeurs".

Exemple de variable : `<string name="app_name">X_335_ThomasRey_Projet</string>`

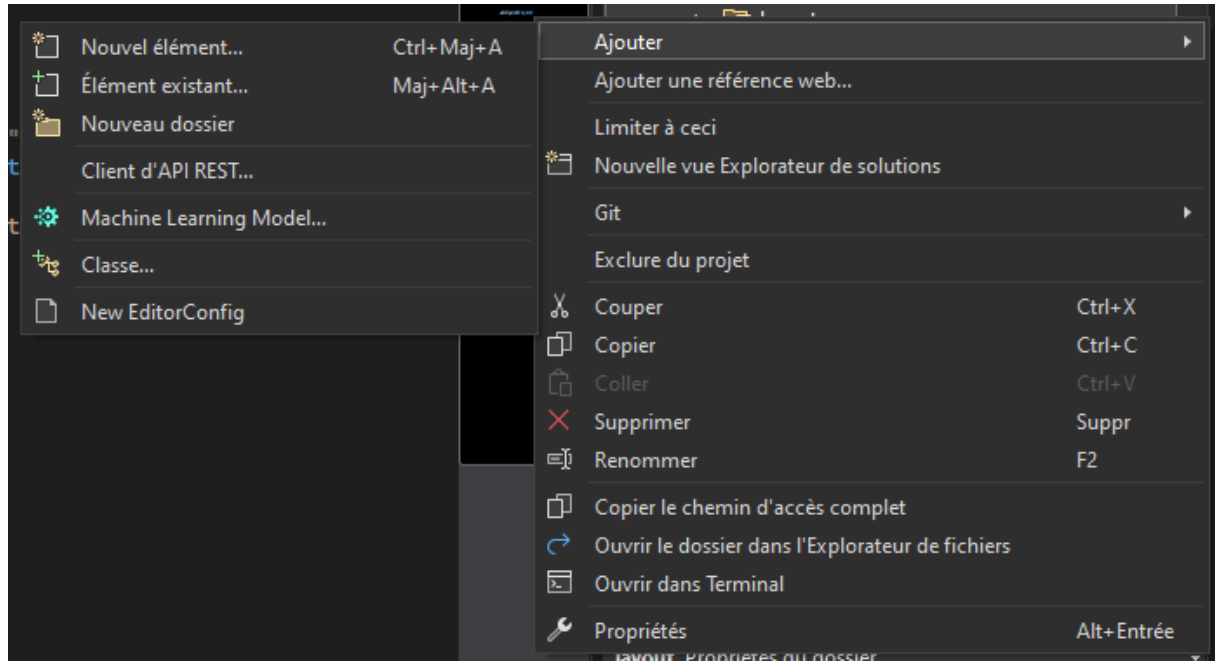
Exemple d'appel : `@string/app_name`

8 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.

8.1 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 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 le projet pour créer un fichier .cs en sélectionnant « Activité » cette fois-ci. Le fichier .cs contiendra ceci :

```
[Activity(Label = "Activity1")]
0 références
public class Activity1 : Activity
{
    0 références
    protected override void OnCreate(Bundle savedInstanceState)
    {
        base.OnCreate(savedInstanceState);

        // Create your application here
    }
}
```

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

```
Xamarin.Essentials.Platform.Init(this, savedInstanceState);
SetContentView(Resource.Layout.activity_task);
```

8.2 Lié les Activités

Il y a deux manières pour lier des Activités entre 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 :

```
StartActivity(typeof(AddTaskActivity));
```

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 et les données à passer

```
Intent addActivity = new Intent(this, typeof(AddTaskActivity));
addActivity.PutExtra("Data", "Some data from MainActivity");
addActivity.PutExtra("Age", 30);
addActivity.PutStringArrayListExtra("Names",
new List<string> { "Jerôme", "Michel", "Paul" });
StartActivity(addActivity);
```

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

```
TextView dataTextView = FindViewById<TextView>(Resource.Id.textView1);
string data = Intent.GetStringExtra("Data");
string data2 = Intent.GetStringExtra("Age");
if (!string.IsNullOrEmpty(data))
{
    dataTextView.Text = data;
}
if (!string.IsNullOrEmpty(data2))
{
    dataTextView.Text = data;
}

Button myButton2 = new Button(this)
{
    Text = data2
};
```

9 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 paramètres « object sender » et « EventArgs e » puis mettre le code que vous voulez dans celle-ci.


```
2 références
private void BackToMenu(object sender, EventArgs e)
{
    StartActivity(typeof(MainActivity));
}
```

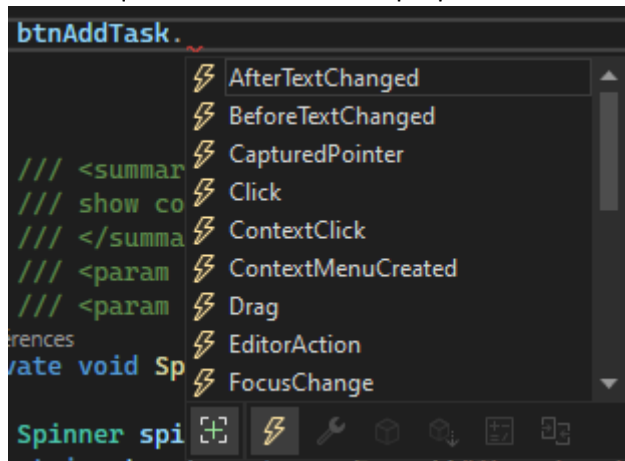
Puis lier cette méthode à un événement d'un élément.

```
Button btnAddTask = (Button)FindViewById(Resource.Id.btnAddTask);

btnAddTask.Click += BackToMenu;
```


Si on ne sait pas quel événement sélectionner il faut prendre pour une action, on peut appuyer sur

 lorsque VS nous donne des propositions.



10 Layout dynamique

11 Persistance de données

12 Utilisation d'un sensor