Hands-On Lab

Analyse des performances et publication

Version: 1.0.0

Dernière mise à jour : 1/10/2012

Sommaire

Vue d’ensemble

Cet atelier va vous permettre de comprendre comment tester les performances de votre application Windows Phone 7. Vous découvrirez également comment réussir la publication sur le MarketPlace.

Objectifs

A la fin de cet atelier, vous aurez appris :

- Les tests de performances des applications Windows Phone 7
- La publication d’une application Windows Phone 7 sur le MarketPlace

Pré requis

Afin de mener à bien cet atelier, vous devez installer les éléments suivants :

- Microsoft Visual Studio 2010 Express pour Windows Phone 7

Exercices

Cet atelier est divisé en plusieurs exercices. Ceux-ci sont listés ci-dessous :

1. Comment analyser les performances d’une application Windows Phone 7
2. Utilisation du MarketPlace Test Kit
3. Publier son application

Durée estimée pour compléter cet atelier: **60 minutes**.

Atelier : Analyses des performances et publication

Exercice 1 – Comment analyser les performances d'une application Windows Phone 7 ?

Les performances des applications sont de plus en plus importantes, surtout pour des périphériques tels que des téléphones portables, sur lesquels les ressources sont limitées. Ainsi, il est vital que les applications n'aient pas de fuite mémoire ou de consommation excessive de ressources. Depuis la version 7.1 du SDK de Windows Phone, il est possible d'utiliser l'analyseur de performances de Visual Studio 2010 pour investiguer sur les potentiels problèmes de votre application.

Capture et analyse de données

1. Dans Visual Studio 2010, compilez votre projet et assurez-vous de cibler le périphérique et non l'émulateur.

Remarque: Pour des raisons logiques, il est conseillé d'utiliser l'outil d'analyse des performances directement sur le téléphone et non sur l'émulateur, de part le fait que les 2 environnements n'ont pas les mêmes performances.

2. Dans le menu « *Debug* », sélectionnez « *Start Windows Phone Performances Analysis* » :

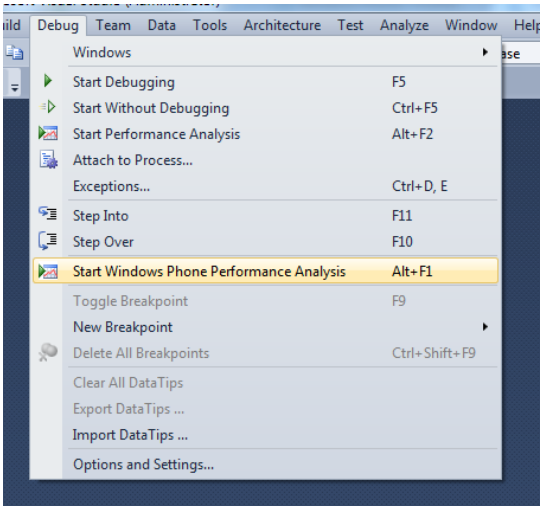


Figure 1 : Ouverture de l'outil d'analyse des performances

Remarque: Assurez-vous que le projet est bien un projet Windows Phone 7.1. Pour les applications Windows Phone 7.0, l'option est grisée dans le menu.

3. L'écran qui apparait vous permet alors de lancer l'analyseur pour récolter 2 types d'information :
    a. L'analyse des performances des éléments dessinés à l'écran et des appels de méthodes (option « *Execution* »)
    b. L'analyse des allocations d'objets (option « *Memory* »)

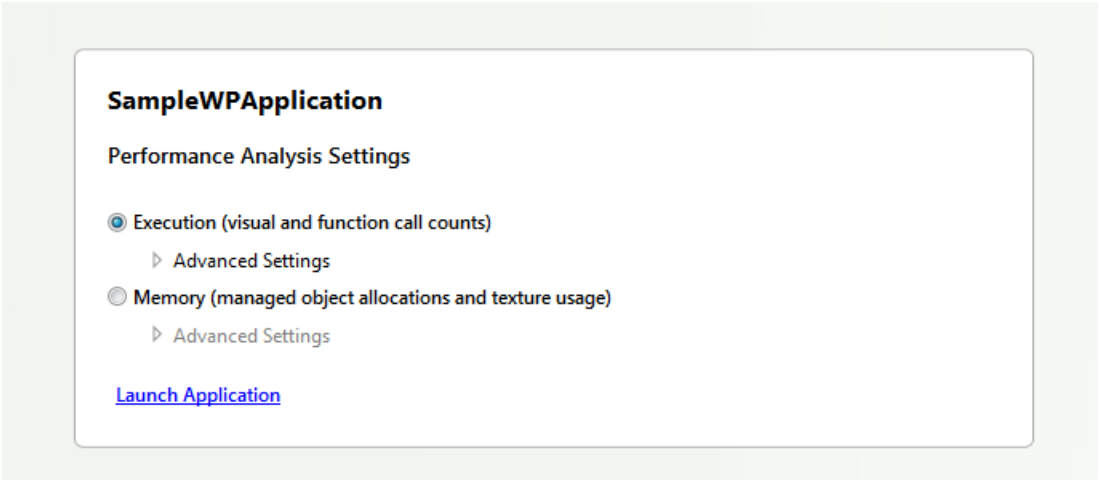


Figure 2 : Outil d'analyse des performances

4. Il est à noter que pour chacune des 2 options, vous pouvez spécifier des options avancées :

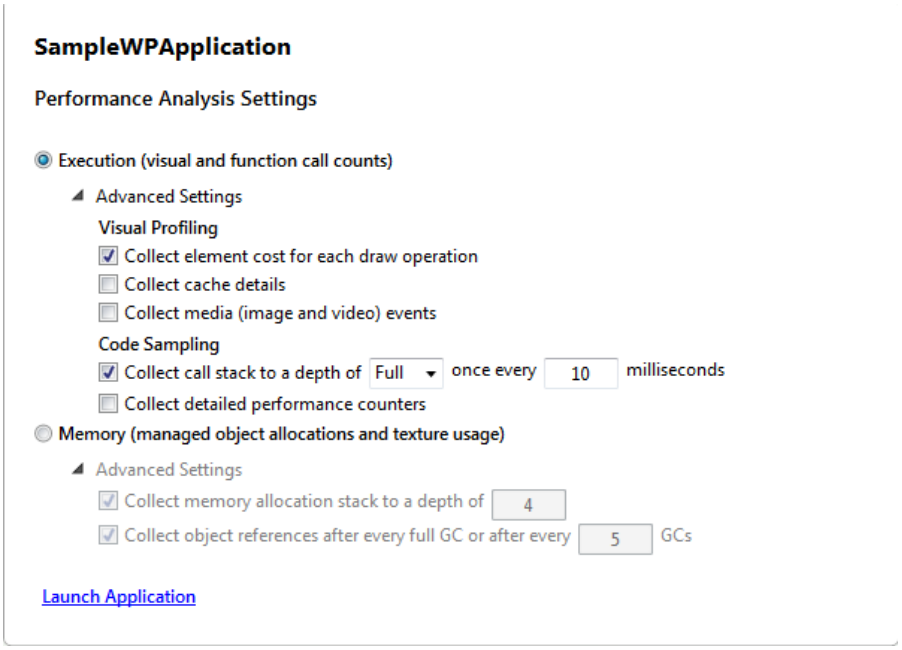


Figure 3 : Options avancées de l'outil d'analyse des performances

5. Sélectionnez l'option « *Memory* » et cliquez alors sur « *Launch Application* » pour démarrer la session de profiling, qui va exécuter votre application :

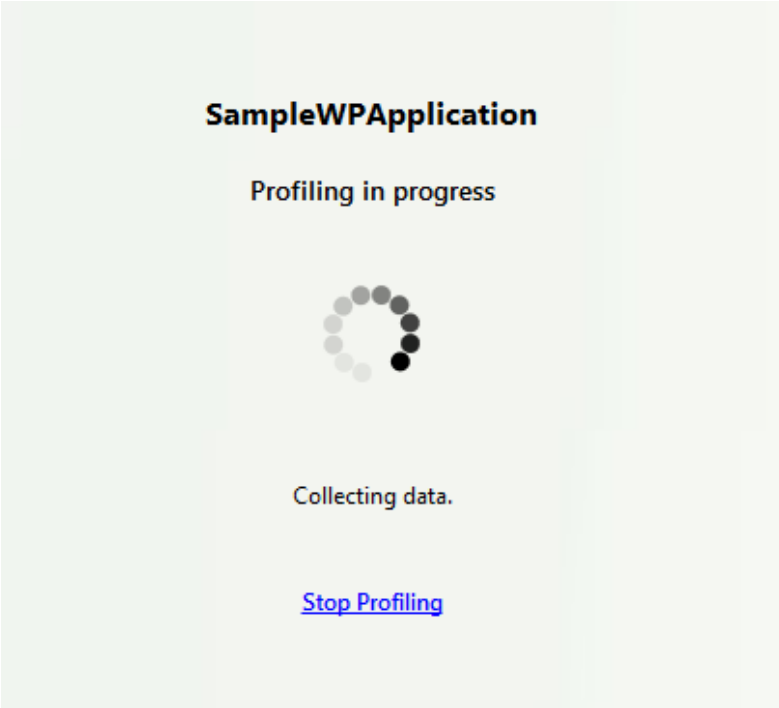


Figure 4 : Session de profiling en cours

6. Votre application démarée, utilisez-la comme n'importe quel utilisateur : testez différents scénarios, faîtes défiler les éléments, manipulez les Panorama/Pivot, etc. Une fois vos tests finis, appuyez sur le bouton « *Back* » pour terminer la session de profiling :

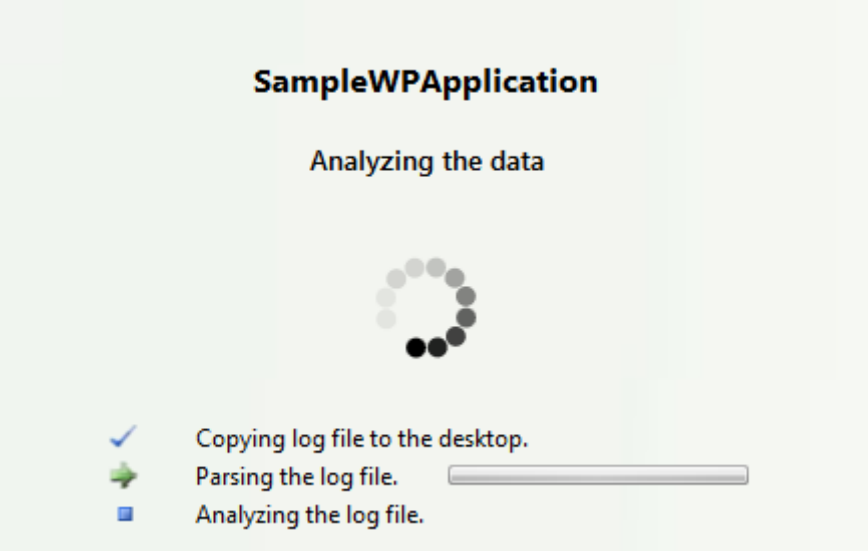


Figure 5 : Fin de la session de profiling

7. Visual Studio vous affiche alors les statistiques d'usage mémoire de votre application, vous laissant ainsi la possibilité d'identifier, entre autre, les problèmes de fuite mémoire (la consomation mémoire augmente sans jamais diminuer) :

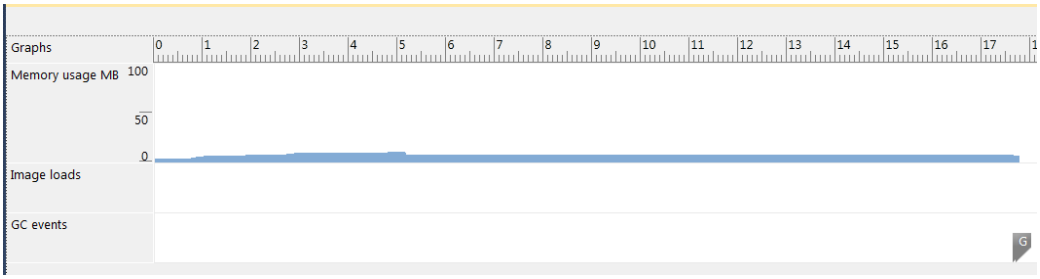


Figure 6 : Résultats de la session de profiling

8. Avec votre souris, sélectionnez une période de temps sur le graphique et, dans la fenêtre du bas, cliquez sur « Start Analysis » :

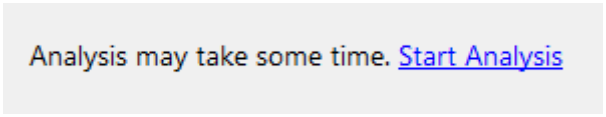


Figure 7 : Démarrer l'analyse des résultats de la session

9. Des informations détaillées sur la période de temps sélectionnée apparaissent alors dans la fenêtre inférieure :

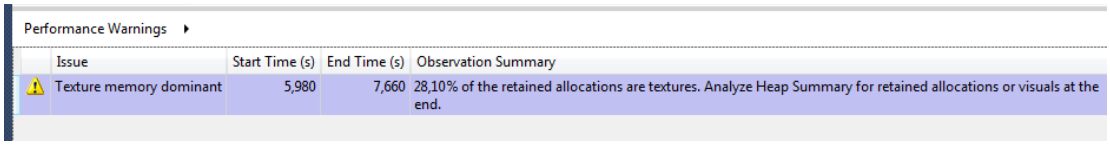


Figure 8 : Informations détaillées sur la période de temps sélectionnée

10. Si vous cliquez sur la flèche à droite de « *Performance Warnings* », vous pourrez afficher le « *Heap Summary* » qui vous permet d'afficher le contenu du tas managé :
    a. Les objets en cours d'exécution
    b. Le nombre de nouveaux objets
    c. Le nombre d'éléments visuels
    d. Etc.

Performance Warnings ▸ Heap Summary ▸

Category	Instances	Total Size (KB)
Retained Allocations at Start	7622	439,762
New Allocations	71	2,047
Collected Allocations	0	0,000
Retained Allocations at End	7693	441,809
Retained Silverlight Visuals at Start	24	9,063
Retained Silverlight Visuals at End	24	9,063

Figure 9 : Liste des éléments dans le tas managé

11. Là encore, si vous cliquez sur la flèche à droite de « *Heap Summary* », vous pourrez afficher des informations encore plus précises :

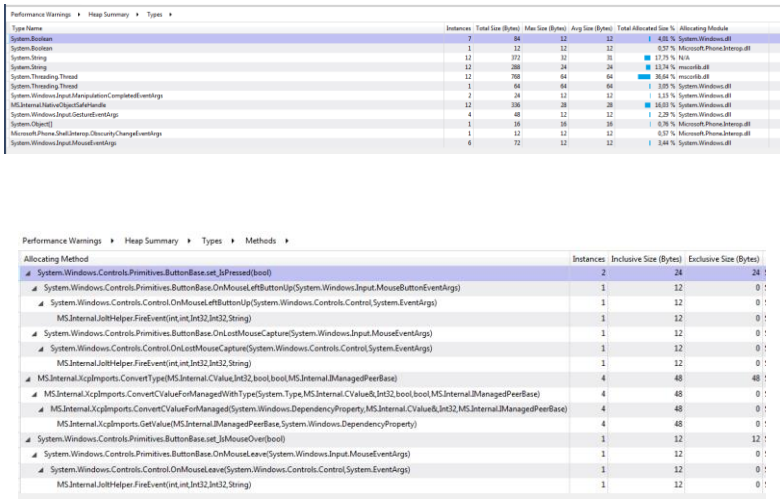


Figure 10 & 11 : Informations supplémentaires sur les objets en cours d'exécution

12. Vous pouvez ainsi avoir toutes les informations nécessaires pour débogger les problèmes mémoire qui peuvent survenir dans votre application.
13. Relancer une nouvelle session de profiling mais cette fois, choisissez « Execution » pour analyser la partie graphique de votre application.
14. Une fois les résultats affichés, il est possible d'observer, entre autre :
    a. « *Frame rate* » : le nombre de fois que l'écran est redessiné, en frame par secondes
    b. « CPU usage % » : affiche le pourcentage d'utilisation du processeur lorsque l'application est en cours d'exécution (la couleur verte symbolisant le thread graphique)
    c. « Memory usage Mb » : permet de connaitre la quantité de mémoire, en Megabyte, utilisée par l'application

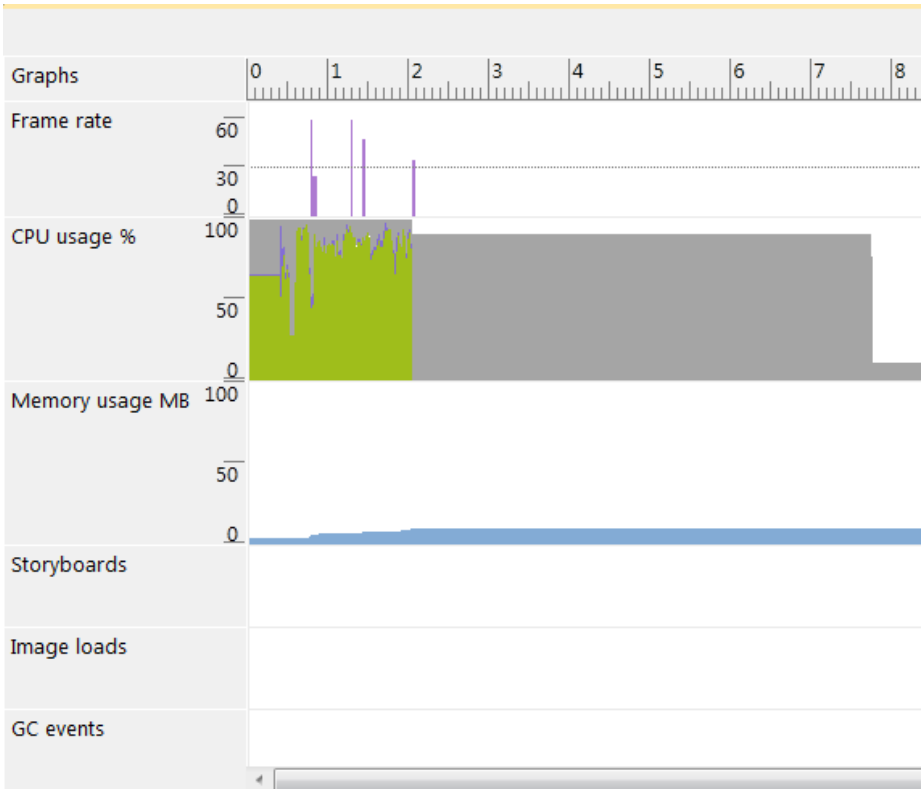


Figure 12 : Informations disponible lors de l'analyse de la couche graphique

15. Sélectionnez une tranche dans le temps afin d'obtenir, lors encore, différentes informations comme par exemple :

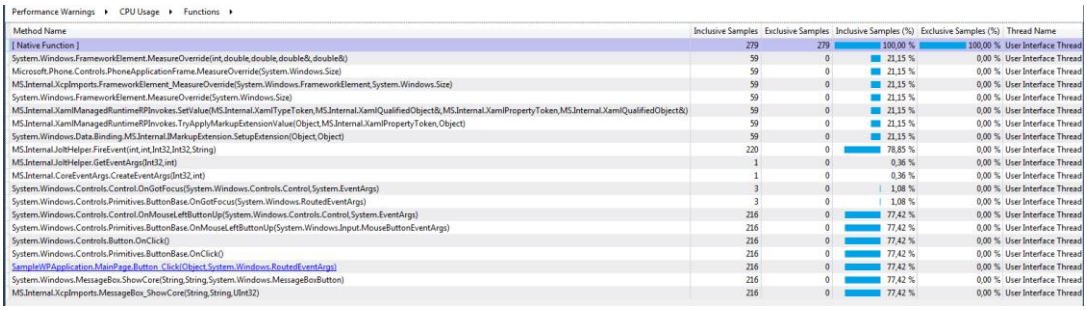
    a. Le pourcentage d'utilisation du processeur, en fonction des différents threads :

Performance Warnings ▸ CPU Usage ▸

Thread Name	Thread ID	CPU Time (ms)	CPU Time (%)
◢ User Interface Thread	602343042	82,303	2,52 %
Frame Operations	602343042	24,000	0,74 %
Other Tasks	602343042	58,303	1,79 %
Profiler Thread	572327870	4,032	0,12 %
Native Thread	577439274	1,151	0,04 %
Compositor Thread	584648562	4,000	0,12 %
Native Thread	579929882	1,000	0,03 %
Native Thread	586876694	1,000	0,03 %

Figure 13 : Pourcentage d'utilisateur du processeur

b. La liste des fonctions appelées :

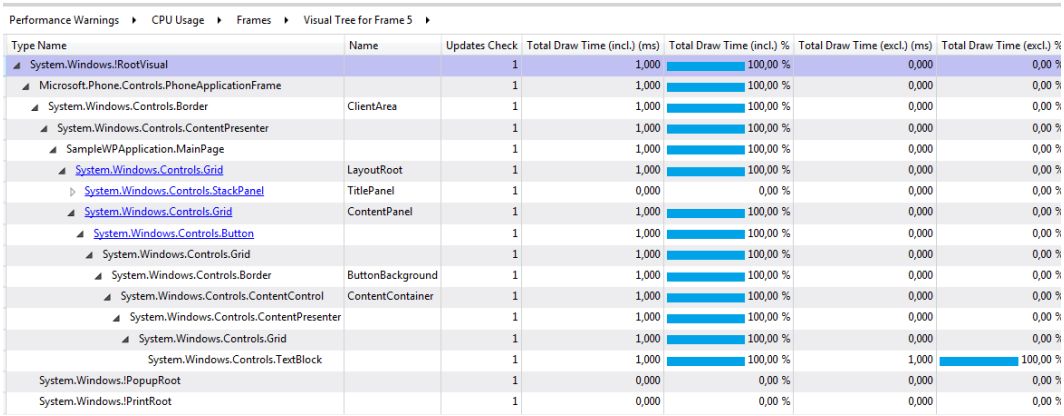


Performance Warnings ▸ CPU Usage ▸ Functions ▸

Method Name	Inclusive Samples	Exclusive Samples	Inclusive Samples (%)	Exclusive Samples (%)	Thread Name
[Native Function]	279	279	100,00 %	100,00 %	User Interface Thread
System.Windows.FrameworkElement.MeasureOverride(int,double,double,double&,double&)	59	0	21,15 %	0,00 %	User Interface Thread
Microsoft.Phone.Controls.PhoneApplicationFrame.MeasureOverride(System.Windows.Size)	59	0	21,15 %	0,00 %	User Interface Thread
MS.Internal.XcpImports.FrameworkElement_MeasureOverride(System.Windows.FrameworkElement,System.Windows.Size)	59	0	21,15 %	0,00 %	User Interface Thread
System.Windows.FrameworkElement.MeasureOverride(System.Windows.Size)	59	0	21,15 %	0,00 %	User Interface Thread
MS.Internal.XamlManagedRuntimeRPInvokes.SetValue(MS.Internal.XamlTypeToken,MS.Internal.XamlQualifiedObject&,MS.Internal.XamlPropertyToken,MS.Internal.XamlQualifiedObject&)	59	0	21,15 %	0,00 %	User Interface Thread
MS.Internal.XamlManagedRuntimeRPInvokes.TryApplyMarkupExtensionValue(Object,MS.Internal.XamlPropertyToken,Object)	59	0	21,15 %	0,00 %	User Interface Thread
System.Windows.Data.Binding.MS.Internal.IMarkupExtension.SetupExtension(Object,Object)	59	0	21,15 %	0,00 %	User Interface Thread
MS.Internal.JoltHelper.FireEvent(int,int,Int32,Int32,String)	220	0	78,85 %	0,00 %	User Interface Thread
MS.Internal.JoltHelper.GetEventArgs(Int32,int)	1	0	0,36 %	0,00 %	User Interface Thread
MS.Internal.CoreEventArgs.CreateEventArgs(Int32,int)	1	0	0,36 %	0,00 %	User Interface Thread
System.Windows.Controls.Control.OnGotFocus(System.Windows.Controls.Control,System.EventArgs)	3	0	1,08 %	0,00 %	User Interface Thread
System.Windows.Controls.Primitives.ButtonBase.OnGotFocus(System.Windows.RoutedEventArgs)	3	0	1,08 %	0,00 %	User Interface Thread
System.Windows.Controls.Control.OnMouseLeftButtonUp(System.Windows.Controls.Control,System.EventArgs)	216	0	77,42 %	0,00 %	User Interface Thread
System.Windows.Controls.Primitives.ButtonBase.OnMouseLeftButtonUp(System.Windows.Input.MouseButtonEventArgs)	216	0	77,42 %	0,00 %	User Interface Thread
System.Windows.Controls.Button.OnClick()	216	0	77,42 %	0,00 %	User Interface Thread
System.Windows.Controls.Primitives.ButtonBase.OnClick()	216	0	77,42 %	0,00 %	User Interface Thread
SampleWPApplication.MainPage.Button_Click(Object,System.Windows.RoutedEventArgs)	216	0	77,42 %	0,00 %	User Interface Thread
System.Windows.MessageBox.ShowCore(String,String,System.Windows.MessageBoxButton)	216	0	77,42 %	0,00 %	User Interface Thread
MS.Internal.XcpImports.MessageBox_ShowCore(String,String,UInt32)	216	0	77,42 %	0,00 %	User Interface Thread

Figure 14 : Liste des fonctions appelées

c. L'arbre visuel correspondant à chaque frame, avoir des liens hypertexte permettant de naviguer, dans votre code XAML, vers le contrôle correspondant :



Performance Warnings ▸ CPU Usage ▸ Frames ▸ Visual Tree for Frame 5 ▸

Type Name	Name	Updates Check	Total Draw Time (incl.) (ms)	Total Draw Time (incl.) %	Total Draw Time (excl.) (ms)	Total Draw Time (excl.) %
System.Windows.!RootVisual		1	1,000	100,00 %	0,000	0,00 %
Microsoft.Phone.Controls.PhoneApplicationFrame		1	1,000	100,00 %	0,000	0,00 %
System.Windows.Controls.Border	ClientArea	1	1,000	100,00 %	0,000	0,00 %
System.Windows.Controls.ContentPresenter		1	1,000	100,00 %	0,000	0,00 %
SampleWPApplication.MainPage		1	1,000	100,00 %	0,000	0,00 %
System.Windows.Controls.Grid	LayoutRoot	1	1,000	100,00 %	0,000	0,00 %
System.Windows.Controls.StackPanel	TitlePanel	1	0,000	0,00 %	0,000	0,00 %
System.Windows.Controls.Grid	ContentPanel	1	1,000	100,00 %	0,000	0,00 %
System.Windows.Controls.Button		1	1,000	100,00 %	0,000	0,00 %
System.Windows.Controls.Grid		1	1,000	100,00 %	0,000	0,00 %
System.Windows.Controls.Border	ButtonBackground	1	1,000	100,00 %	0,000	0,00 %
System.Windows.Controls.ContentControl	ContentContainer	1	1,000	100,00 %	0,000	0,00 %
System.Windows.Controls.ContentPresenter		1	1,000	100,00 %	0,000	0,00 %
System.Windows.Controls.Grid		1	1,000	100,00 %	0,000	0,00 %
System.Windows.Controls.TextBlock		1	1,000	100,00 %	1,000	100,00 %
System.Windows.!PopupRoot		1	0,000	0,00 %	0,000	0,00 %
System.Windows.!PrintRoot		1	0,000	0,00 %	0,000	0,00 %

Figure 15 : Arbre visuel correspondant à une frame spécifique

Ainsi, vous pouvez analyser toutes les parties de votre application, que ce soit aussi bien au niveau de la mémoire que tout ce qui touche à la partie graphique, dans le but de chercher à optimiser les ressources ou corriger un problème lié à une utilisation trop intense de votre application.

Comprendre les compteurs de performances

Une autre des fonctionnalités utilisables pour tenter de comprendre d'observer les performances d'une application est d'utiliser les « *Frame Rate Counters* » (traduit ici par « Compteur de performances »).

Pour les activer, c'est très simple : dans le fichier App.xaml.cs, décommenter la ligne suivante :

```
Application.Current.Host.Settings.EnableFrameRateCounter = true;
```

Ensuite, relancer l'application et vous pourrez observer que des indications supplémentaires sont présentes sur la partie droite de l'écran :

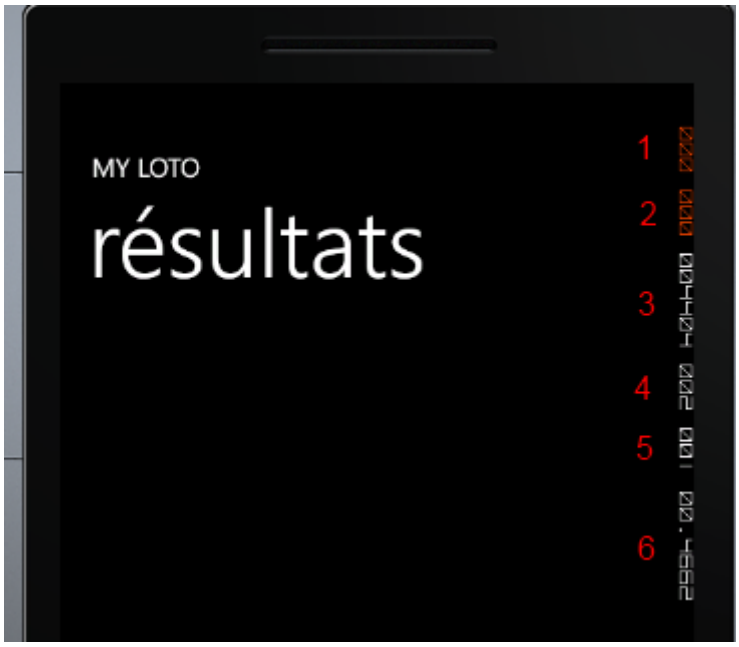


Figure 16 : Compteurs de performances d'une application Windows Phone

Il y a en tout 6 indicateurs d'affichés, chacun représentant une donnée particulière :

1. « *Render Thread FPS* » :
    a. Il s'agit du nombre de frame par seconde utilisé par le thread de rendu.
    b. C'est dans ce thread que sont exécutées les animations
    c. Une valeur autour de 60 permettra de proposer une bonne expérience utilisateur mais, à l'inverse, une valeur autour de 30 (ou moins) sera significatif d'une expérience déplaisante.
    d. A noter que sous une valeur de 30, le chiffre apparait en rouge.
2. « User Interface Thread FPS » :
    a. C'est le nombre de frame par seconde utilisé par le thread UI (thread lié à l'interface graphique)
    b. En dessous de 15, la valeur est affichée en rouge
3. « Texture Memory Usage » :
    a. Ce compteur permet d'afficher la taille de la mémoire vidéo utilisée pour stocker les textures de l'application
4. « Surface Counter » :
    a. C'est l'indicateur permettant de connaitre le nombre de surfaces qui sont passées à la carte graphique
5. « Intermediate Texture Count » :
    a. Permet de connaitre le nombre de textures intermédiaires utilisées pour la composition
6. « Screen Fill Rate » :

a. Cette métrique représente le nombre d'écrans qui sont redessinés à chaque fois, sur chaque frame.

Ainsi, avec ces différents compteurs, vous pouvez suivre en temps réel l'évolution des performances de l'application, détectant les points faibles et ceux nécessitant d'être améliorés.

Exercice 2 – Utilisation du MarketPlace Test Kit

Le MarketPkace Test Kit est un outil, fournit avec le SDK de Windows Phone, qui a pour objectif de vous permettre d'effectuer différents tests qui vous permettront de savoir si votre application est susceptible de passer la validation du MarketPlace.

C'est donc un outil à exécuter avant la publication, et qui évite de faire des allers-retours inutiles en les équipes de validation de Microsoft et vous-même.

1. Dans l'explorateur de solution, sélectionnez votre application Windows Phone 7 puis cliquez sur « *Project* » => « *Open MarketPlace Test Kit* »

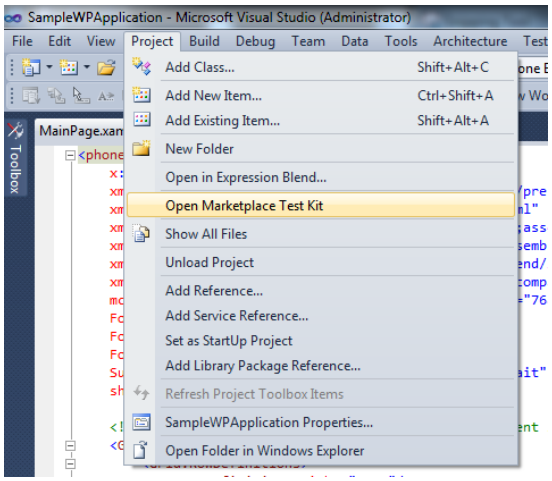


Figure 1 : Ouverture du MarketPlace Test Kit

2. L'écran qui apparait alors vous permet de rensigner les différents détails de l'application, à savoir les images, nécessaires pour la publication sur le Marketplace
3. Saisissez les valeurs obligatoires, afin de vous permettre de tester que les images que vous utilisées sont correctes :

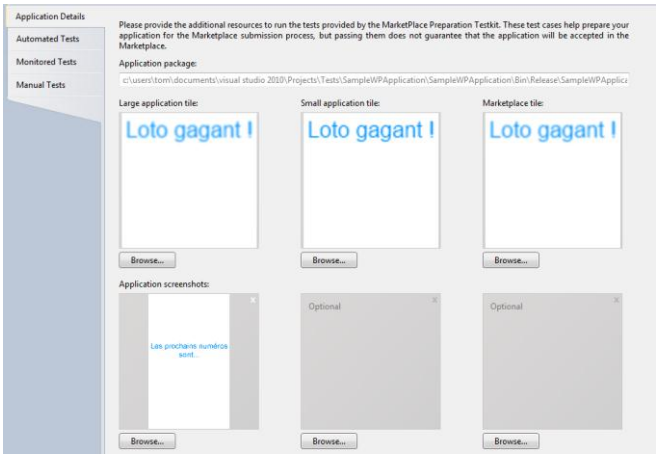


Figure 2 : Saisie des images utilisées lors de la publication

4. A présent, cliquez sur le 2ème onglet « *Automated Tests* » :

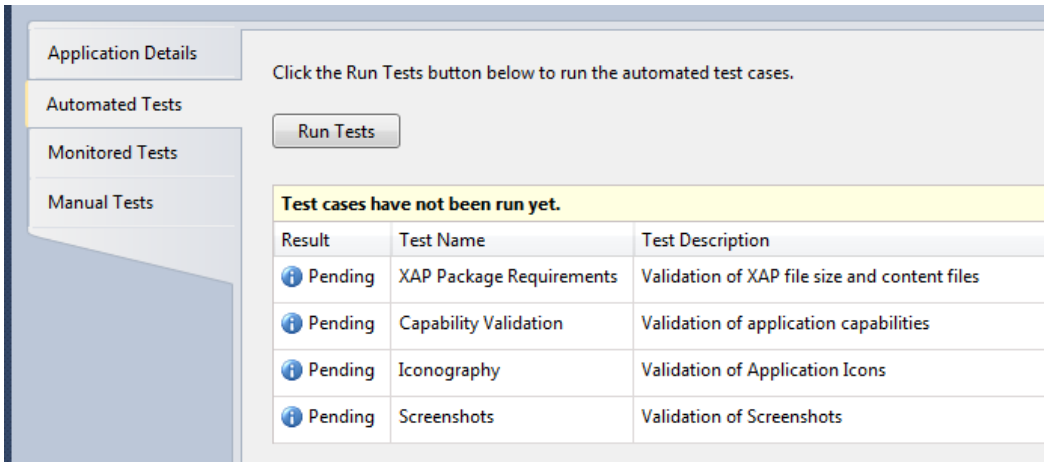


Figure 3 : Ecran des tests automatisés

5. Cet écran vous permet de lancer la première série de tests, qui va valider les éléments suivants :
    a. La taille du fichier XAP et son contenu
    b. Les fonctionnalités natives du téléphone
    c. Les différentes icônes/images utilisées
    d. Les screenshots de l'application
6. Cliquez sur le bouton « *Run Tests* » pour lancer l'exécution des tests. Si tout se passe bien, vous devriez voir apparaitre des icônes vertes, signe que les tests se sont bien déroulés. Dans le cas contraire, c'est une icône rouge en forme de croix qui sera affichée :

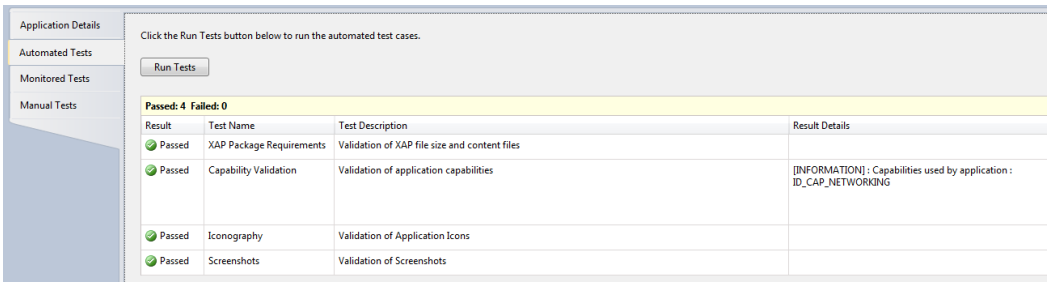


Figure 4 : Exécution réussie des tests automatisés

7. Vous pouvez également aperçevoir, lorsque cela est le cas, un message détaillant le message d'erreur ou le message informatif associé au test qui a été joué.

Remarque: Si une erreur est survenue lors de l'exécution des tests, il est conseillé de la corriger avant d'aller plus loin.

8. Cliquez à présent sur l'onget « *Monitored Tests* » pour afficher la page des tests monitorés. Ce type de test sert à valider que l'application se lance suffisament vite, qu'elle ne consomme pas trop de mémoire, qu'elle se termine correctement mais aussi que le bouton « Back » est bien utilisé :

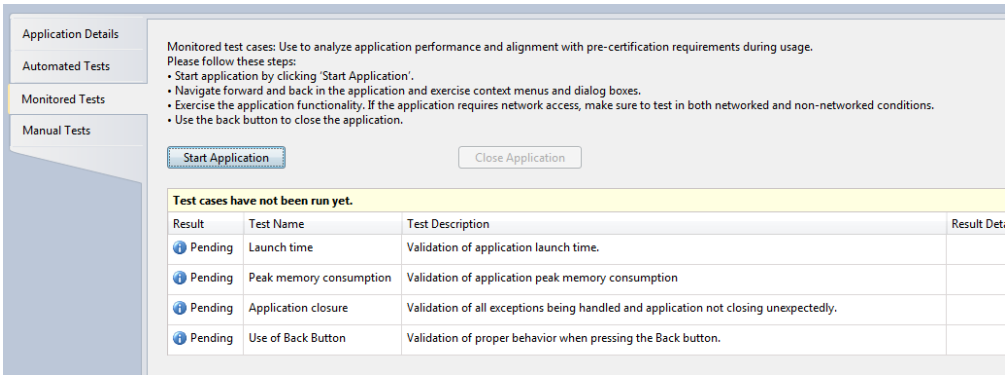


Figure 5 : La page des tests monitorés

Remarque: Pour exécuter les tests monitorés, il est impératif de les exécuter sur un téléphone et non pas dans l'émulateur, car les performances ne sont pas les mêmes dans les 2 environnements.

9. Pour exécuter ce test, il suffit de cliquer sur le bouton « *Start Application* » et d'utiliser l'application, en naviguant de pages en page, en retournant sur les pages précédentes en cliquant sur le bouton « *Back* », etc. Pour quitter l'application, cliquez à nouveau sur le bouton « *Back* ».
10. Une fois l'application quittée, l'outil analyse les résultats

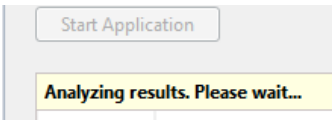


Figure 6 : Analyse des résultats des tests monitorés

11. Une fois calculés, les résultats sont affichés à l'écran, là encore avec un éventuel message d'informations ou d'erreur (et toujours au moyen d'une icône rouge ou verte) :

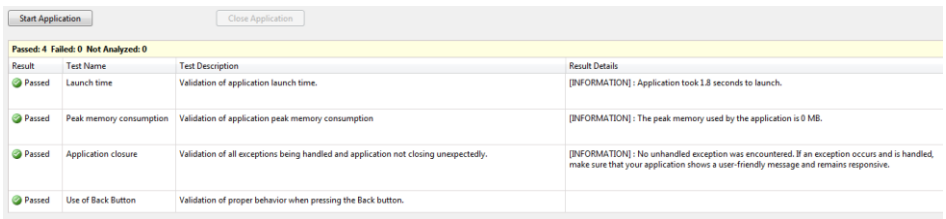


Figure 7 : Affichage des résultats des tests monitorés

12. Le dernier onglet, « *Manual Test* », vous permet de jouer tout un ensemble de tests, proposés par Microsoft pour valider la conformité de l'application. Ceux-ci ne sont pas automatique, ce sera donc à vous d'exécuter et d'indiquer si le test est concluant ou non :

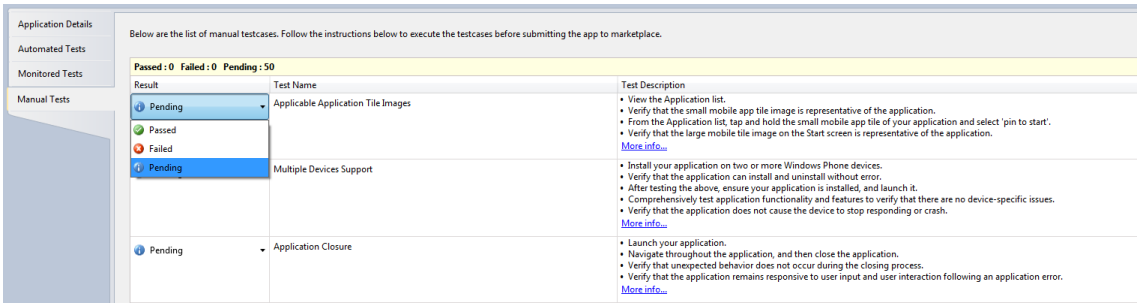


Figure 8 : Sélectiondu résultat d'un test manuel

13. Comme vous pouvez le contater, un grand nombre de test sont disponible (50 au total !), ceux-ci étant mis à jour, de temps en temps par Microsoft :

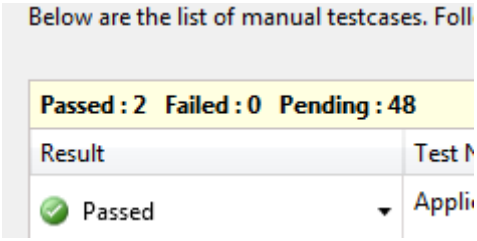


Figure 9 : Sélectiondu résultat d'un test manuel

14. Bien sûr, rien ne vous oblige à jouer tous les tests, certains étant en effet liés à des fonctionnalités bien spécifiques telles que les Background Agents, la gestion du mode « *Trial* », la manipulation de photo, etc.

Exercice 3 – Publier son application

Une fois votre application écrite et testée, il vous faut la publier pour la rendre accessible aux autres utilisateurs. Pour cela, plusieurs étapes sont nécessaires et nous allons les détailler par la suite.

1. Assurez-vous d'avoir un compte développeur. Pour cela, rendez-vous à l'adresse suivante : http://create.msdn.com/en-US/
2. Une fois votre compte créé, vous pouvez commencer la soumission de votre application, en vous rendant à cette adresse : https://windowsphone.create.msdn.com/AppSubmission#/PageUpload
3. Vous arrivez alors sur la première page du processus de soumission de l'application, sur laquelle vous devez renseigner :
    a. Le nom de l'application
    b. Le type de distribution :
        i. Publique : L'application sera visible pour tous les utililisateurs
        ii. Beta : L'application ne sera visible que pour les utilisateurs que vous avez sélectionnés
    c. Le fichier XAP de l'application
    d. Le numéro de version

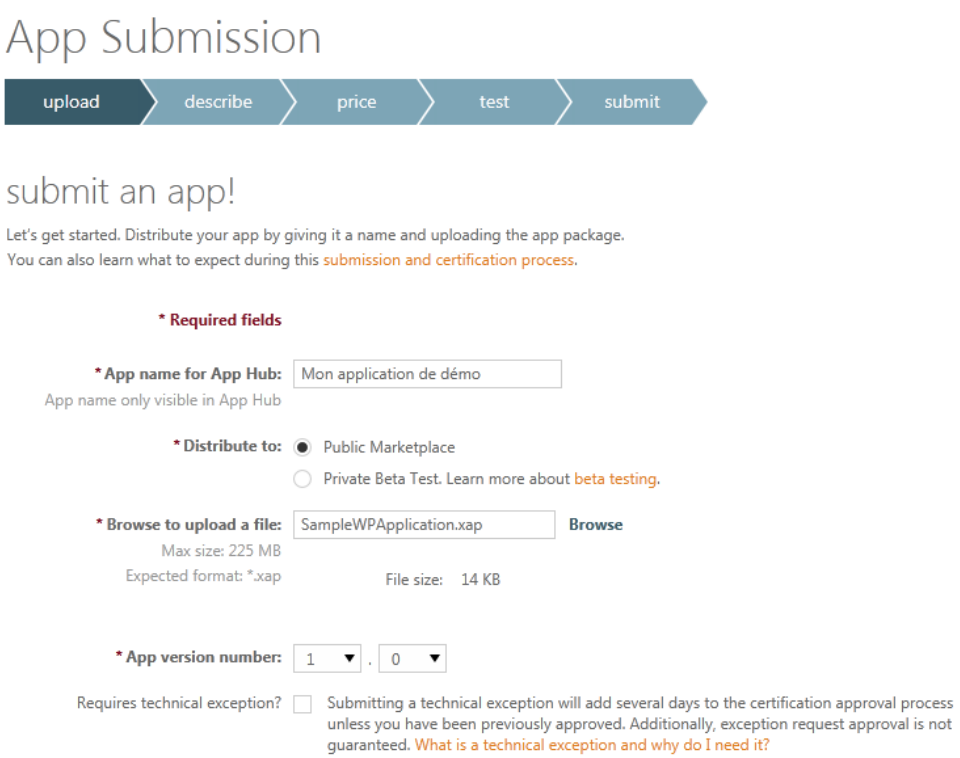


Figure 1 : Ecran de saisie des informations élémentaires

4. Lors du clic sur le bouton « Next », le fichier XAP de l'application est analysé pour tenter de détecter si vous n'utilisez pas du code qui ne sera pas supporté (pour le moment). Ensuite,

l'assistant vous affiche la page sur laquelle vous devez saisir différentes informations relatives à l'application :

a. Une description détaillée
b. Une liste de mots clé, qui permettront aux utilisateurs de trouver votre application lors d'une recherche (pensez-donc à bien choisir ces mots-clé !)
c. Les différentes images qui seront utilisées. Il est en effet nécessaire d'indiquer :
    i. Une petite image, de 99 pixels par 99 pixels
    ii. Une grande image, de 200 pixels par 200 pixels
    iii. Au moins 1 screenshot, de 480 pixels par 800 pixels (jusqu'à 8 screenshots peuvent être renseignés)

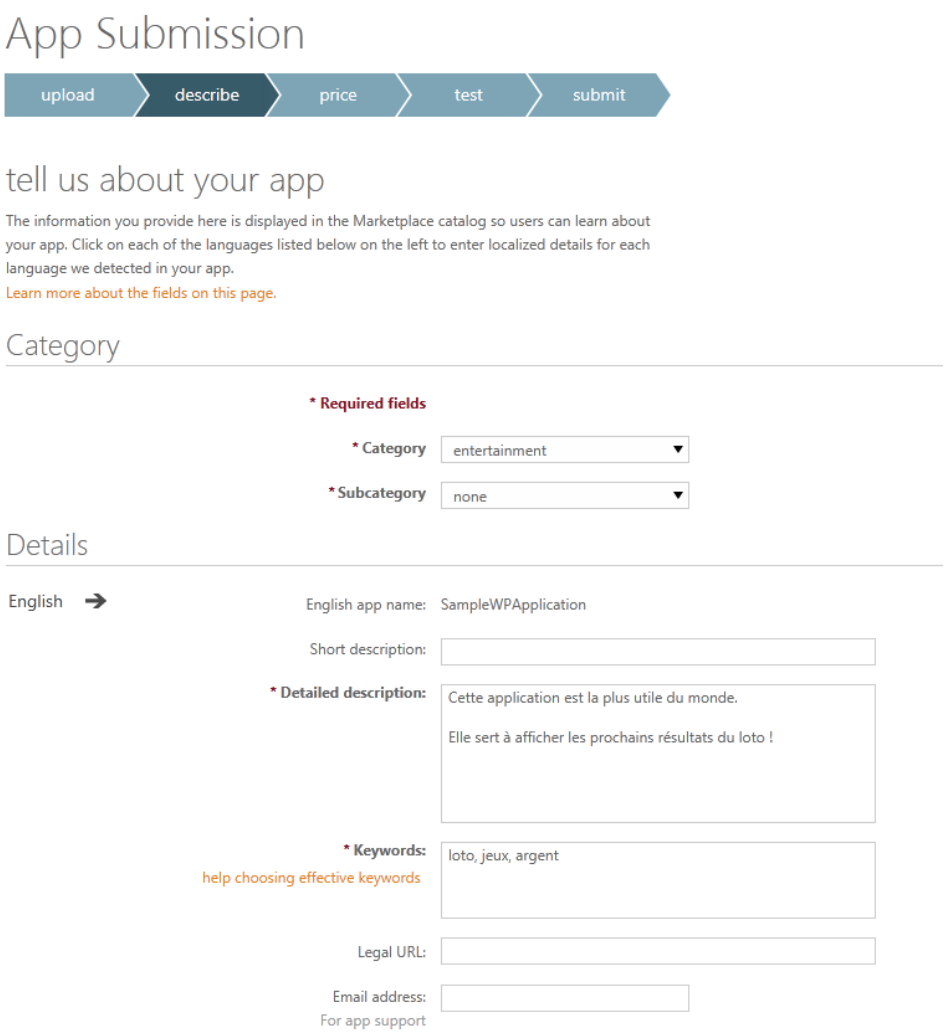


Figure 2 : Saisie de la description de l'application

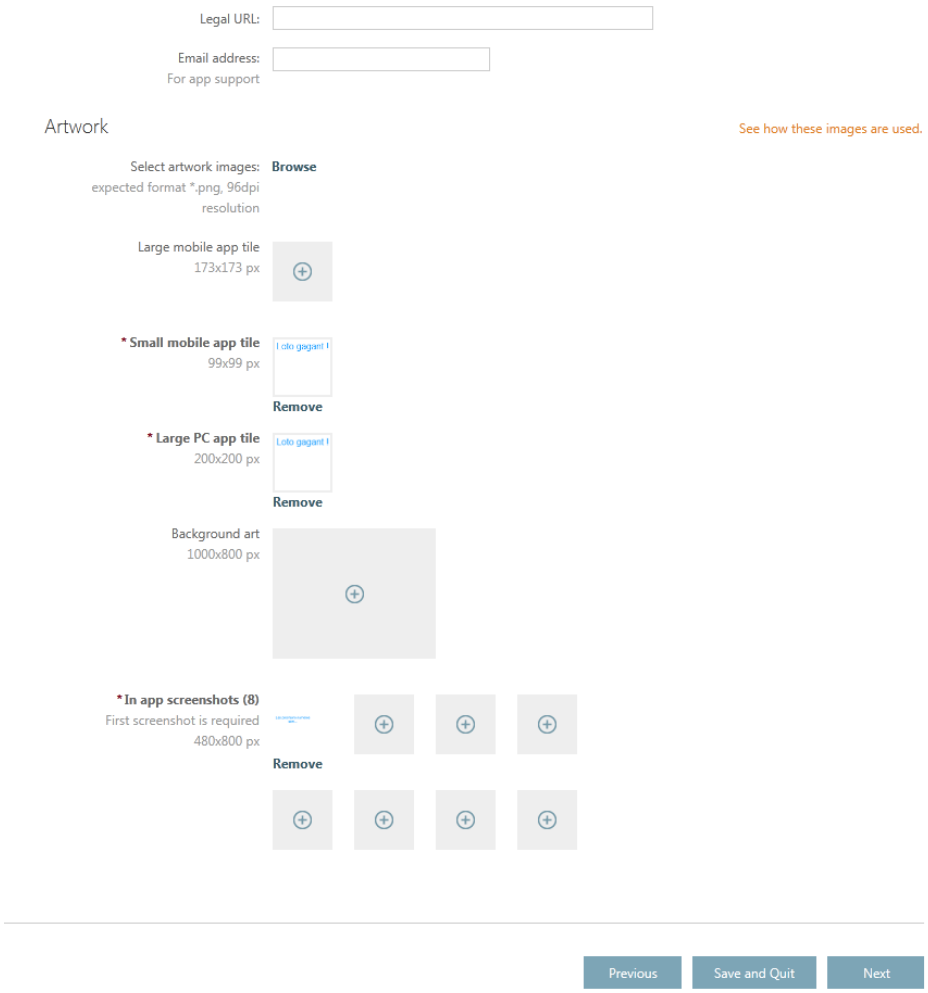


Figure 3 : Saisie des images de l'application

5. A présent, vous allez devoi renseigner le prix de votre application. En effet, vous avez plusieurs possibilités pour la distribuer :
    a. En faire une application totalement gratuite
    b. Faire une application payante avec un mode d'évaluation (trial)
    c. Faire une application totalement payante
6. Il est à noter que vous avez également la possibilité de spécifier qu'une application peut être disponible dans un pays mais pas dans un autre (pratique si vous n'avez pas encore géré la traduction). Vous pouvez aussi précisez que votre application supporte le mode « Trial », permettant aux utilisateurs de la tester avant d'envisager de l'acheter.

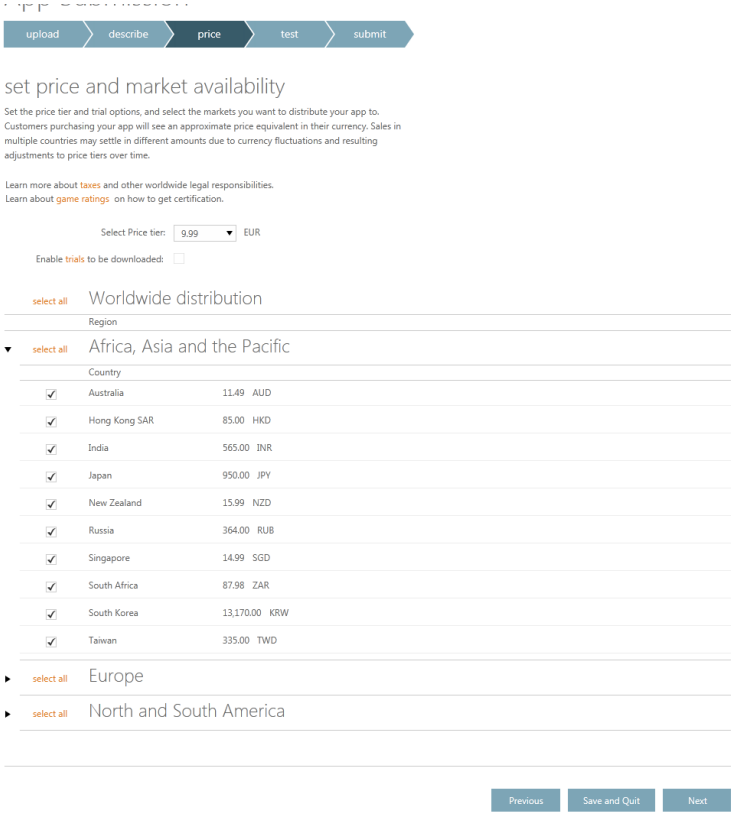


Figure 4 : Saisie des informations tarifaires

7. Enfin, le dernier écran vous permet d'indiquer des informations supplémentaires pour les testeurs (un compte utilisateur, une adresse de service, etc.).
8. Vous pouvez également spécifier quand vous voulez que l'application soit publiée sur le MarketPlace :
    a. Dès qu'elle aura été testée et certifiée par les équipes de Microsoft
    b. Une fois qu'elle aura été certifiée, mais de manière cachée
    c. A votre convenance : l'application est certifiée mais vous choisissez quand vous voulez qu'elle soit visible sur le MarketPlace

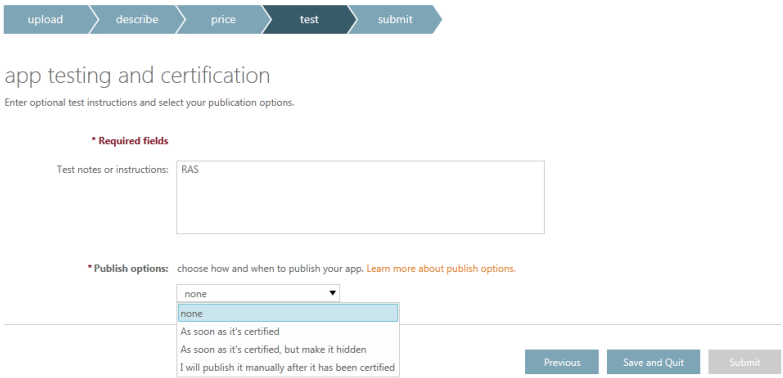


Figure 5 : Saisie des informations de publication

9. Il vous suffit alors de cliquer sur le bouton « Submit » pour soumettre votre application. Celle-ci sera alors testée par les équipes de Microsoft qui la valideront (ou non) et qui vous en informeront par email.
10. Si la validation est un succès et que vous avez choisit de publier automatiquement votre application alors **félicitations** : vous venez de publier avec succès votre première application Window Phone 7.
11. Si malheureusement votre application n'est pas validée par Microsoft, vous recevrez par email un rapport détaillé expliquant les raisons du refus de la validation. Il vous faudra alors corriger les points et re-soumettre l'application une nouvelle fois, en espérant que cela passe cette fois-ci !

Résumé

Cet atelier a introduit les différents outils utilisés pour tester les performances des applications Windows Phone 7 et réussir, au mieux, la publication et validation sur le MarketPlace.