Introduction à Xamarin

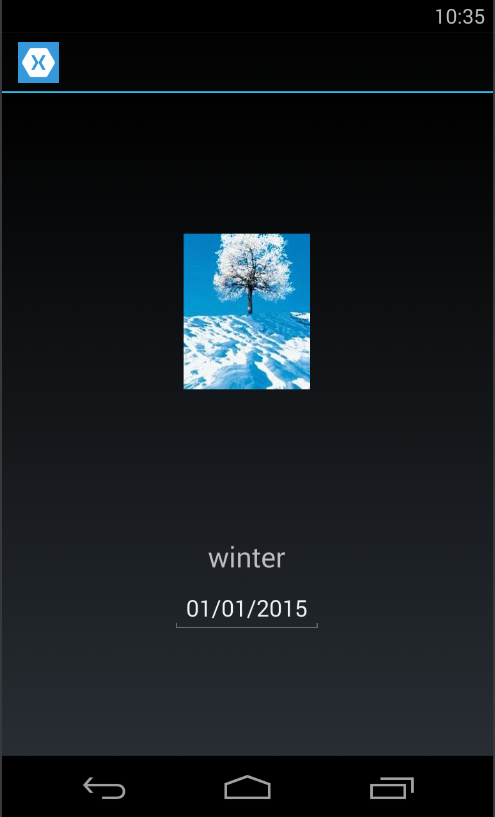
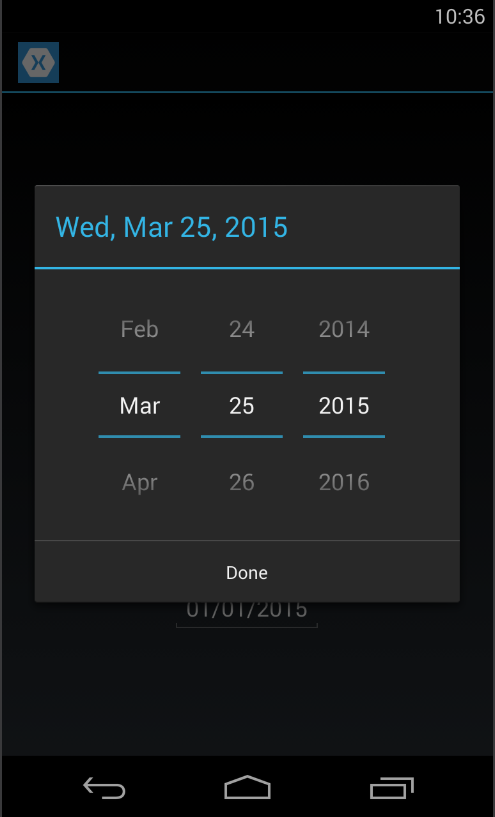
## Seasons

Dans cette application, l’utilisateur choisit une date et l’application affiche la saison associée à cette date. Pour cette application, on se limitera volontairement à gérer des dates en 2016. Sur cette période :

* Le printemps démarre le 20 mars,
* L’été démarre le 21 juin,
* L’automne démarre le 22 septembre,
* L’hiver démarre le 22 décembre.

Vous aurez besoin d’utiliser :

* DatePicker,
* Label,
* Image.

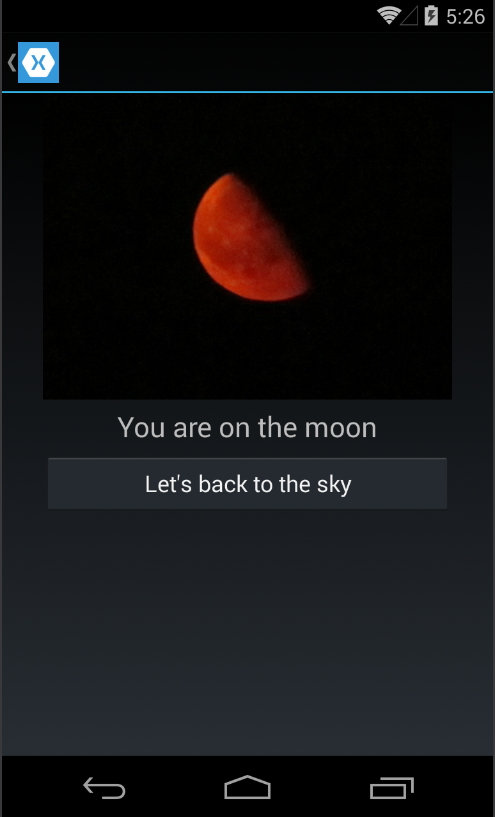
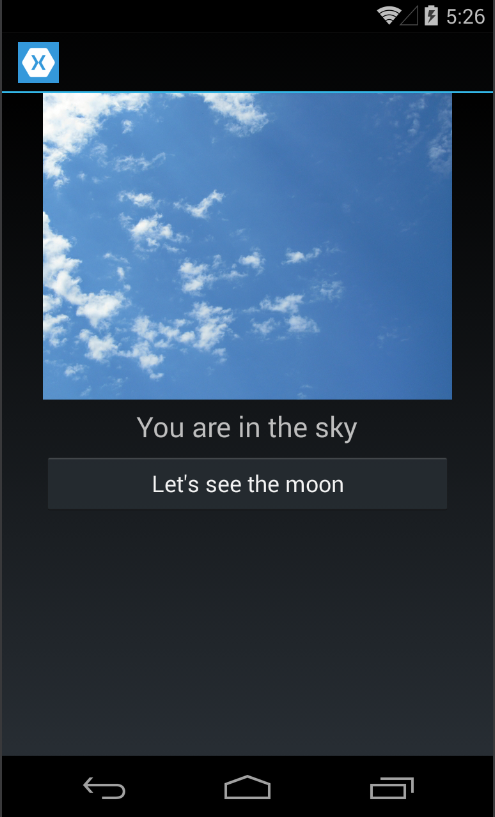
 

## From the sky to the moon

Afin de découvrir la navigation entre page, vous devez implémenter l’application permettant de naviguer entre deux pages (le ciel et la lune).

Vous aurez besoin d’utiliser :

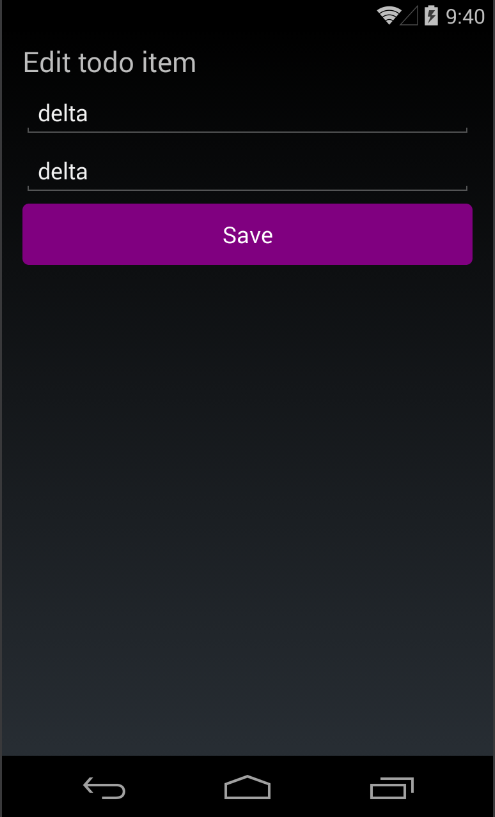
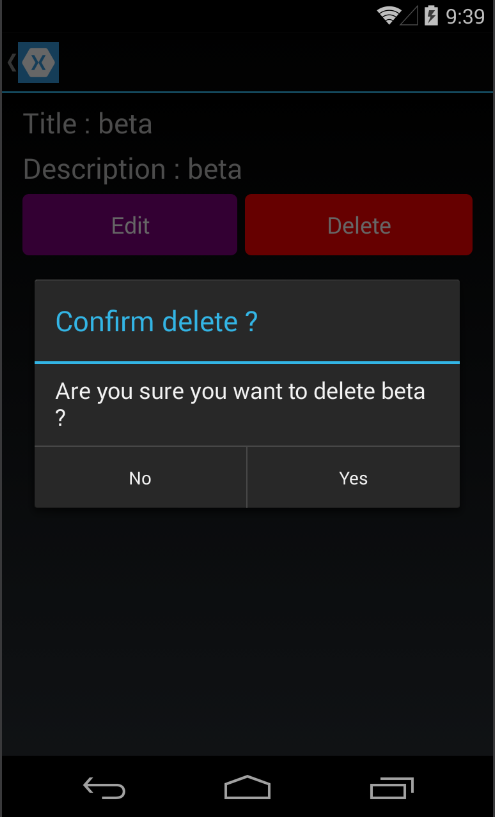
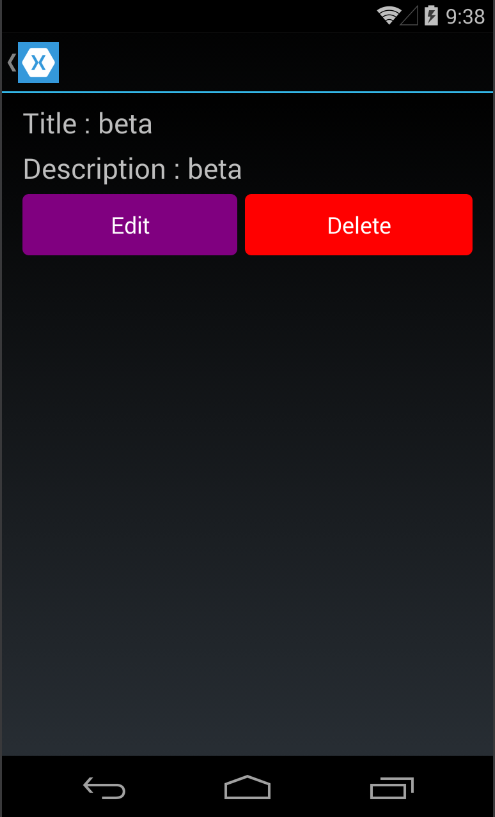
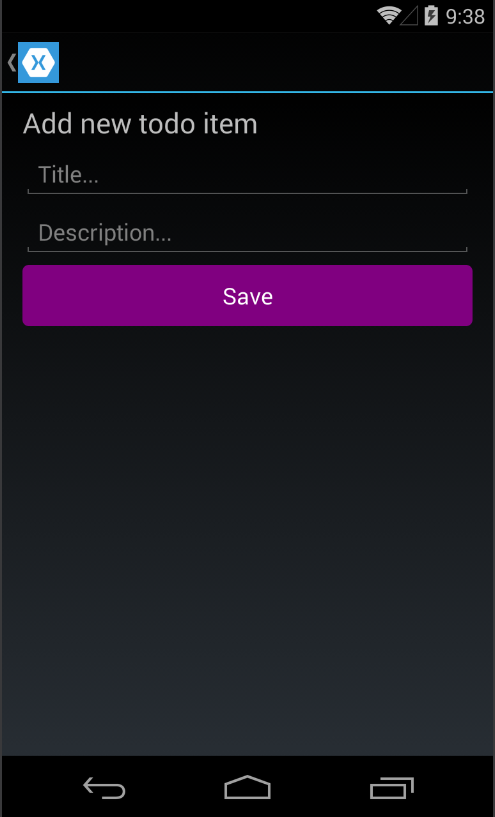
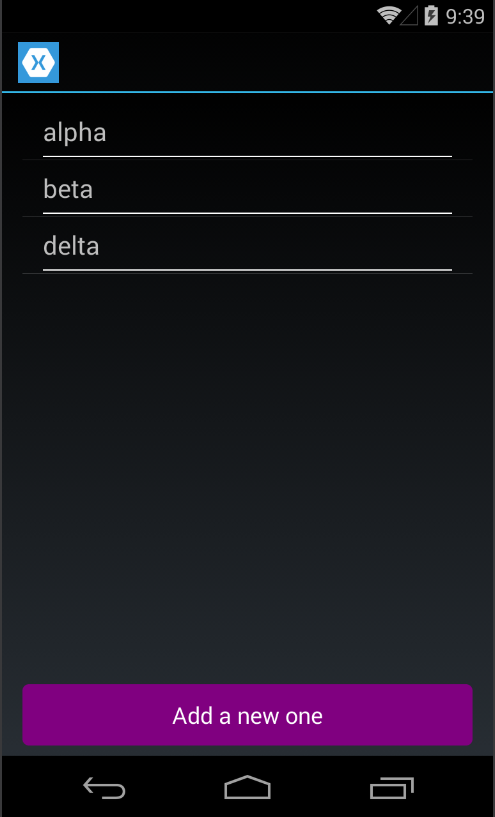
* Image,
* Label,
* Button,
* NavigationService.



## La TodoList, le retour

Le but est de réaliser une application pour gérer une liste de tâches, avec ajout, édition, suppressions, …

Vous aurez besoin d’utiliser une ObservableCollection<T> pour stocker vos Todos et qu’un ajout/suppression dans cette collection mette à jour l’UI.



## Service de gestion de vos Todos

Vous allez ici reprendre l’exercice précédent en y ajoutant un service vous permettant de gérer votre collection de Todos. De cette manière, vous pourrez très simplement mutualiser votre collection tout au long de l’application. Vous résoudrez ainsi le problème que vous avez du rencontrer lors de l’implémentation de la TodoList simple. Pensez à regarder du côté de l’injection de dépendance.

## Synchronisons les Todos

Dans cette seconde partie, vous avez pour but de relier votre application en fonctionnant qu’en local jusqu’ici à un web service (la documentation est disponible plus bas) avec authentification des utilisateurs.

Vous penserez à utiliser les packages nuget suivant si vous souhaitez vous simplifier la vie :

* Microsoft HTTP Client Libraries
* Json.NET
* PCL Crypto – Portable Crypto APIs
* Modern http client pour l’implémentation Android

Bien évidemment, vous devrez rajouter à votre application un écran de login + d’enregistrement de l’utilisateur.

Vous ferez bien entendu attention à la sécurité de vos utilisateurs et penserez à sécuriser vos mots de passe avant de les envoyer. Un compte de test (login user, mdp user) est disponible.

Cf : la description de l’api ci-dessous.

## Remember me ?

Dans cette troisième partie, votre application doit permettre à l’utilisateur de cocher une case « remember me » lors du login afin de permettre par la suite de l’authentifier automatiquement au lancement de l’application. Vous enregistrerez les identifiants dans un fichier JSON.

Vous aurez probablement besoin du package nuget « PCL Storage », pensez donc à l’importer. Vous prendrez bien évidemment soin de ne pas enregistrer le mot de passe de l’utilisateur en clair ☺

## Share your Todos

Maintenant, nous allons ajouter un peu de « social » dans votre application via le partage de vos Todos par email ou sms. Le but de cette dernière partie est de vous faire implémenter un service spécifique pour une plateforme et de l’utiliser afin de partager vos Todos. (On peut également imaginer du partage par bluetooth ou NFC ☺)

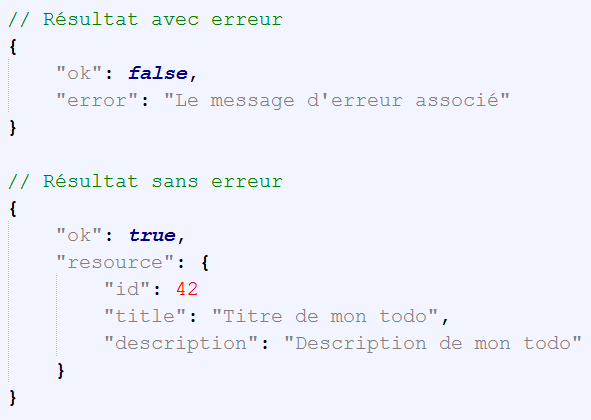
### Description du WebService

L’adresse de base du WS est : http://storm-project.fr /ios/api

Toutes les réponses sont envoyées au format JSON, chaque réponse est contenue dans un objet JSON avec les champs suivants :

* ok : booléen à true si il n’y a pas eu d’erreur, false sinon.
* error : présent si ok est à false, donne le texte de l’erreur
* resource : présent si ok est à true et que la requête renvoie un résultat, il contient ce résultat.

Exemple :



#### Description des requêtes pour l’utilisateur

POST /register : **inscrit un utilisateur** (et le connecte si tout se passe bien)

*Requête :*

login : login de l'utilisateur

password : password de l'utilisateur

*Résultat :*

Pas de résultat

POST /login : **connecte un utilisateur**

*Requête :*

login : login de l'utilisateur

password : password de l'utilisateur

*Résultat :*

Pas de résultat, ok est à true si l’utilisateur est connecté

***Les mots de passe sont à hasher avec un algorithme SHA-1 et à convertir en hexadécimal***

#### Description des requêtes pour les Todos

GET /list : **retourne la liste des Todos de l'utilisateur**

*Requête :*

Aucun paramètre

*Résultat :*

Array of TodoItem

POST /add : **ajoute un Todo**

*Requête :*

title : titre du todo

description : description du todo

*Résultat :*

Objet TodoItem

POST /edit : **update un todo**

*Requête :*

id : id du todo

title : titre du todo

description : description du todo

*Résultat :*

Objet TodoItem

POST /delete : **supprime un todo**

*Requête :*

id : id du todo

*Résultat :*

Pas de résultat

Dans les réponses de requête, un TodoItem est un objet avec trois champs (cf. exemple plus haut) :

* id : l’identifiant unique du Todo
* title : le titre du Todo
* description : la description qui lui est associée