

Introduction au test d'application Xamarin

Aloïs Deniel, à Paris, le 22 février 2017

Qui suis-je ?



Aloïs Deniel

Référent technique Xamarin & UWP
Orange Applications for Business
Rennes, Bretagne



**Business
Services**

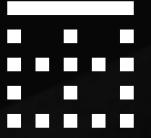
@aloisdeniel 
aloisdeniel 

Le test logiciel

Déetecter les comportements
anormaux d'un logiciel afin
d'en améliorer la qualité

Comprendre les différentes
natures de tests





Architecture de développement

Structure adaptée au test

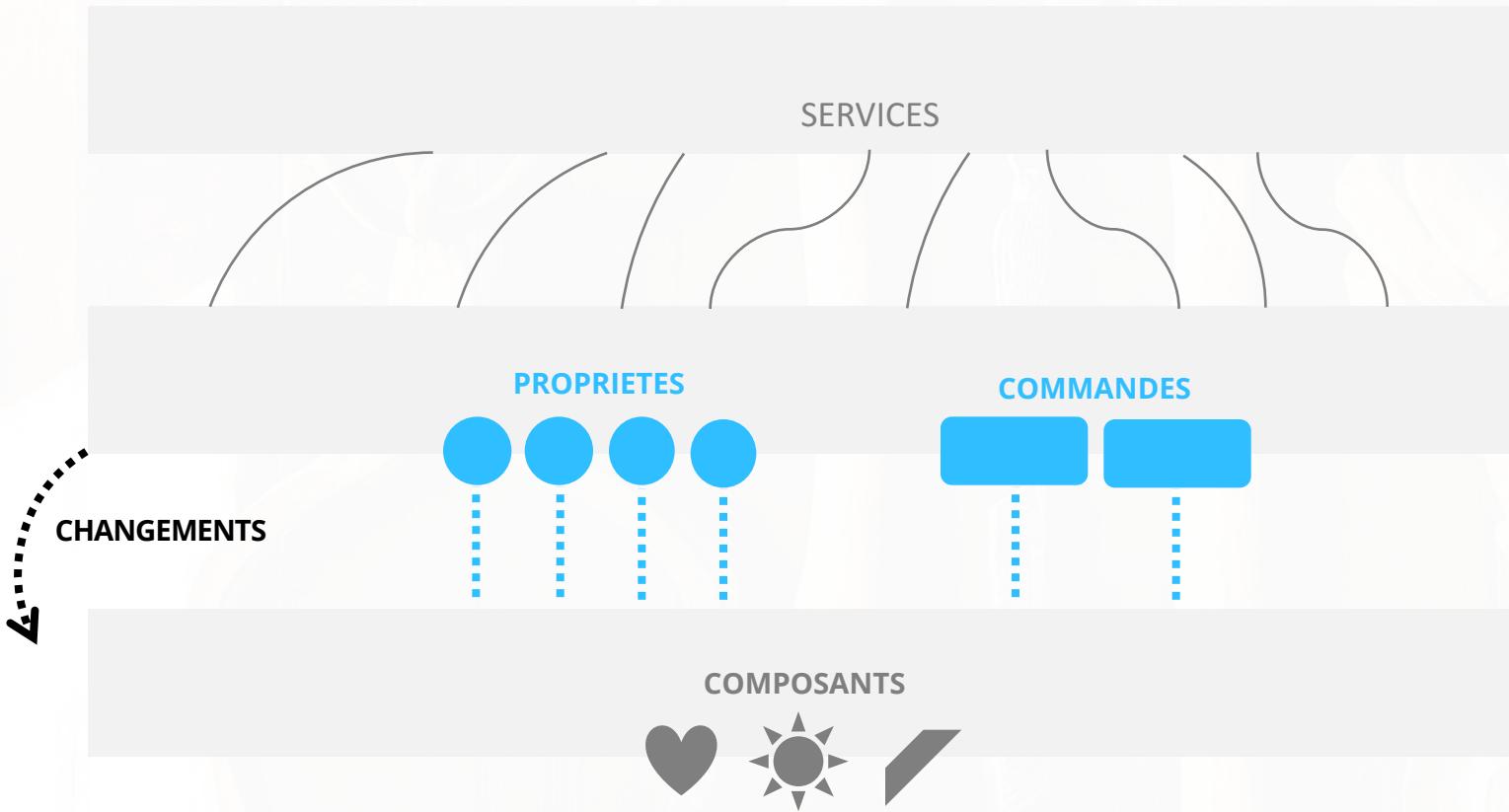
La clé d'une bonne testabilité
passe par une bonne
architecture logicielle

Couches avec des
responsabilités spécifiques



MVVM

Architecture au centre des frameworks de développement de clients lourds de Microsoft depuis de nombreuses années



Model

Logique métier

ViewModel

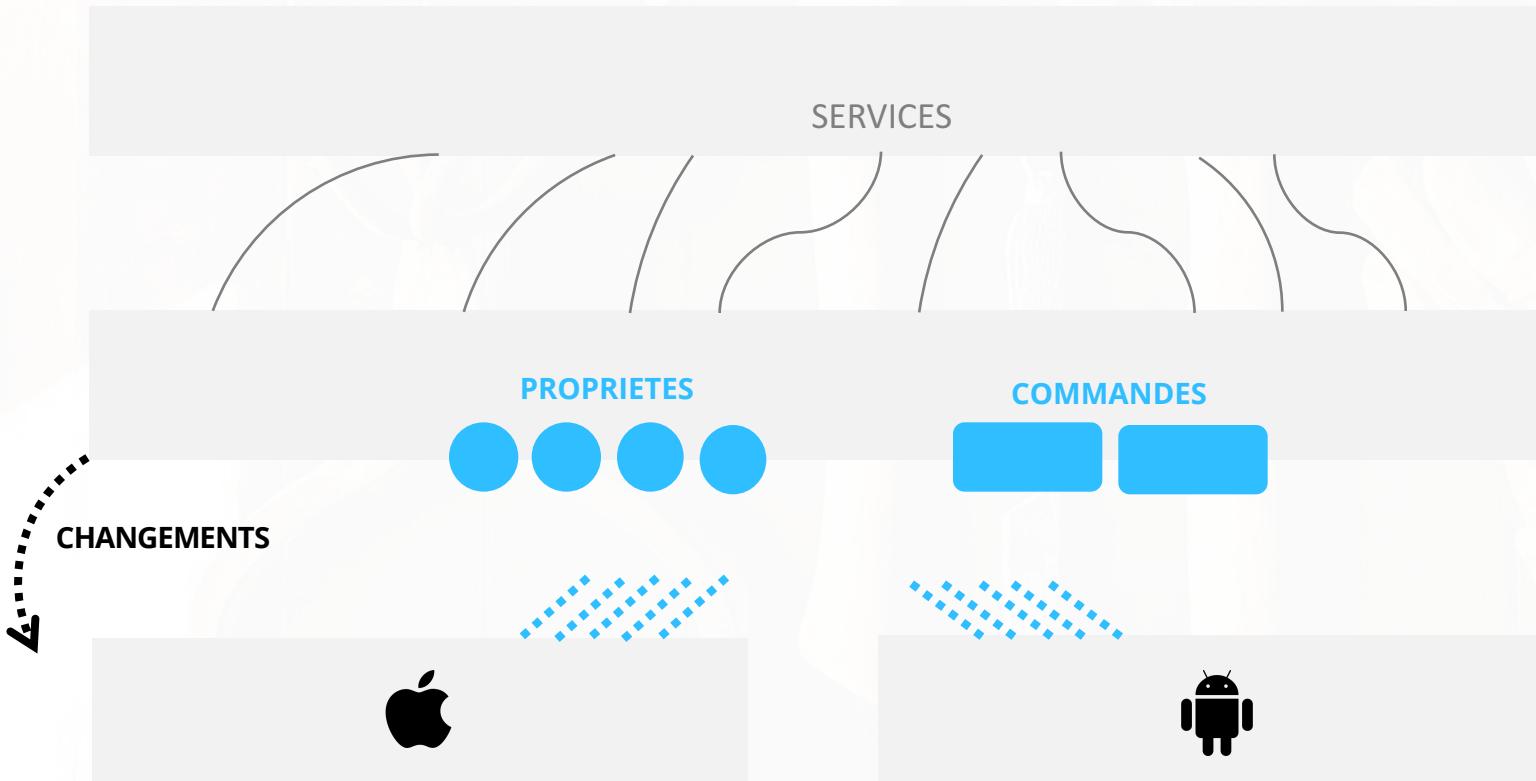
Abstraction de l'interface utilisateur

View

Interface utilisateur

MVVM

Architecture au centre des frameworks de développement de clients lourds de Microsoft depuis de nombreuses années



Model

Logique métier

ViewModel

Abstraction de l'interface utilisateur

Views

Interfaces utilisateurs

MVVM

Architecture au centre des frameworks de développement de clients lourds de Microsoft depuis de nombreuses années

Model

Logique métier

Test unitaire

Valider la logique métier

ViewModel

Abstraction de l'interface utilisateur

Test d'intégration

Valider l'intégration des différents systèmes

View

Interface utilisateur

Test fonctionnel

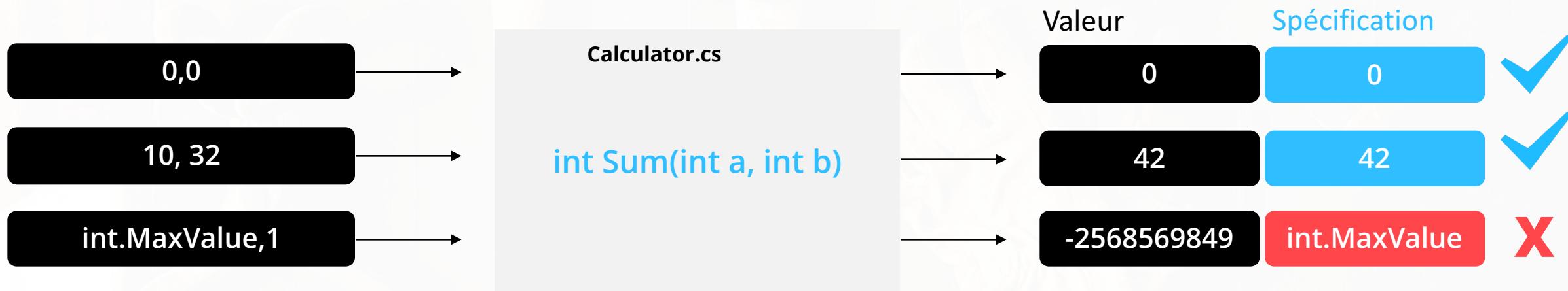
Valider le fonctionnement l'application



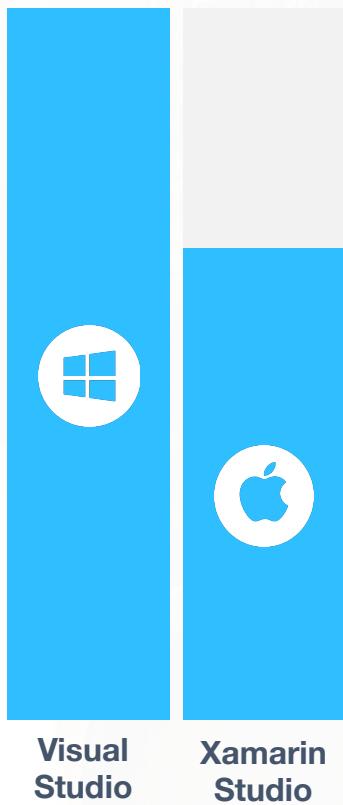
Test unitaire

Valider les règles métier

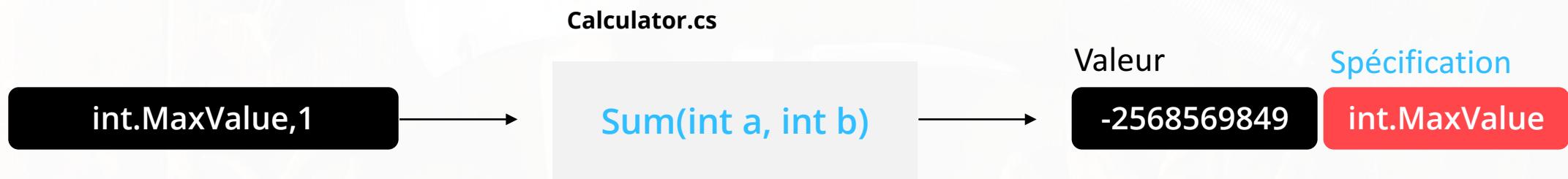
Le test unitaire est un programme qui valide le bon fonctionnement d'une fonction isolée du cœur de votre logique métier en fonction de certains paramètres d'entrée et d'une spécification établie



Frameworks



Framework de test automatisé est intégré aux différents templates Xamarin.

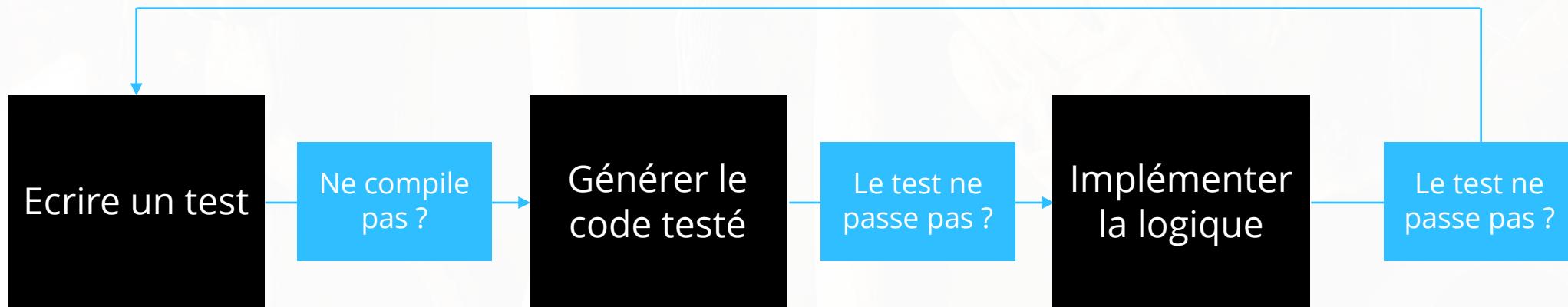


CalculatorTests.cs

```
[Test]
public void Addition_MaxValueAndOne_MaxValue()
{
    const int expected = int.MaxValue;
    var value = this.calculator.Sum(int.MaxValue,1);
    Assert.AreEqual(expected,value);
}
```

Test Driven Development

Une méthode de développement : commencer par écrire les tests unitaires avant de coder vos fonctionnalités. Le code de vos tests ne compile donc pas lorsque vous l'écrivez, et vous allez générer le code métier à partir de ces derniers. Une fois générés, il faudra les implémenter et vérifier que le résultat des tests est valide.



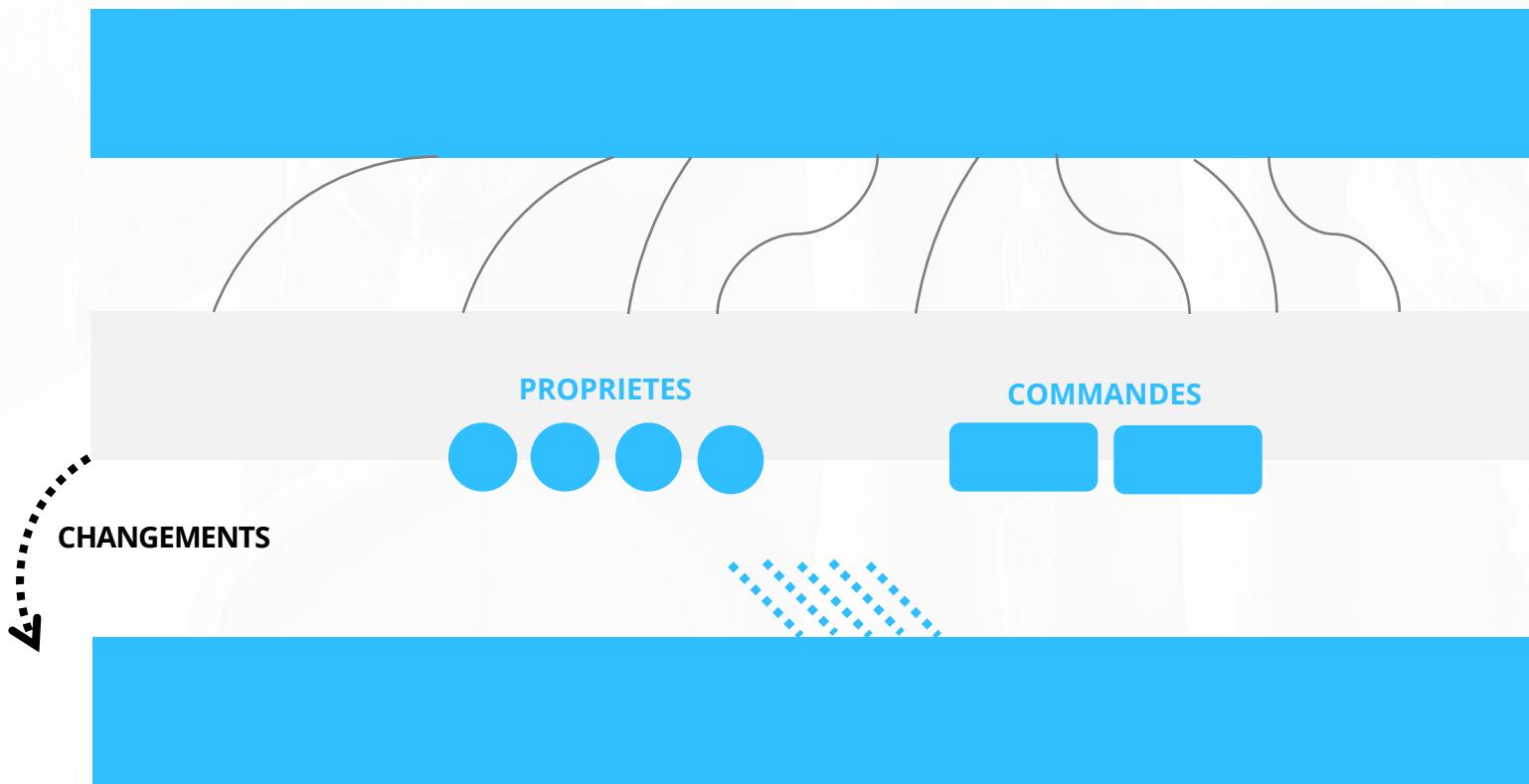


Test d'intégration
Valider l'intégration des différents systèmes

Pour le test d'intégration, nous allons vérifier que nos différents systèmes sont correctement coordonnés.

Isoler le système testé et simuler les systèmes dépendants pour vérifier toutes les situations.

Isolation



Mocks

Simuler la logique métier

ViewModel

Abstraction de l'interface utilisateur

Test

Simuler le parcours utilisateur

Séparer les responsabilités en modules indépendants pour une meilleure testabilité.

Maitriser chaque module pour mieux prédire le comportement de ceux qui en dépendent



Ne pas laisser les systèmes
instancier l'implémentation de
leurs dépendances

Inversion of Control

IoC

```
public class HomeViewModel
{
    public HomeViewModel()
    {
        this.api = new HttpWebApi("http://www.host.com/api");
    }

    private HttpWebApi api;

    private Task UpdateAsync()
    {
        var result = this.api.GetMembersAsync();
        // ...
    }

}
```

Ne pas laisser les systèmes
instancier l'implémentation de
leurs dépendances

```
public class HomeViewModel
{
    public HomeViewModel(IWebApi api)
    {
        this.api = api;
    }

    private IWebApi api;

    private Task UpdateAsync()
    {
        var result = this.api.GetMembersAsync();
        // ...
    }
}
```

Inversion of Control

IoC

```
public interface IWebApi
{
    Task<Member[]> GetMembersAsync();
}

public class HttpWebApi : IWebApi
{
    public HttpWebApi(string host) { ... }
    public Task<Member[]> GetMembersAsync() { ... }
}
```

Ne pas laisser les systèmes
instancier l'implémentation de
leurs dépendances

```
public class HomeViewModel
{
    public HomeViewModel(IWepApi api)
    {
        this.api = api;
    }

    private IWebApi api;

    private Task UpdateAsync()
    {
        var result = this.api.GetMembersAsync();
        // ...
    }
}
```

Inversion of Control

IoC

```
public class DemoWebApi : IWebApi
{
    public DemoWebApi() { }

    public async Task<Member[]> GetMembersAsync()
    {
        await Task.Delay(1000);
        return new Member[]
        {
            new Member("Miguel De Icaza"),
            new Member("Satya Nadela"),
            ...
        };
    }
}
```

Création d'implémentations « mocks »
dynamiquement pour accélérer la rédaction de
tests.

```
this.api = Substitute.For<IWebApi>();  
  
this.api.GetMembersAsync().Returns( new Member[] { ... });  
  
this.api.GetMembersAsync().Returns(x => { throw new Exception(); });  
  
...
```

NSubsitute



Test automatisé d'interface utilisateur
Valider le fonctionnel de l'application

Le test automatisé d'interface utilisateur va nous permettre de simuler des interactions humaines au travers d'un programme.

Facilité de répétition des scénarios afin de s'assurer de la non régression de la qualité de notre programme

Xamarin.UITest

Framework proposant des interactions avec une application iOS/Android.
Combiné à nUnit, il permet cette validation fonctionnelle automatisée.

```
var app = ConfigureApp.iOS.AppBundle("PACKAGE").StartApp();  
  
app.Tap(x => x.Marked("Membres"));  
app.ScrollDownTo("Mon élément");  
  
Assert.IsTrue(app.Query("Indication").Any());
```

Service proposant l'exécution de tests UITests sur une ferme de nombreux terminaux iOS et Android physique, hébergés par Xamarin.

Xamarin Test Cloud

 **Xamarin test cloud** >  Business Lou... > master > Aug 24, 2016 6:34 PM

New Test Run | Support | Docs |

Overview

ALL RESULTS

01 Suivi Conso

01 Lancement Application 9⚡

02 Connexion 9⚡

- fermeture pop up 9⚡
- Affichage vue connexion
- Insertion login
- Insertion mot de passe
- Clique bouton connexion
- redirection vue accueil
- ---

03 Verification Connexion 9⚡

01 SUIVI CONSO
✓ Affichage vue connexion

Spice Dream Uno H
Android 6.0.1

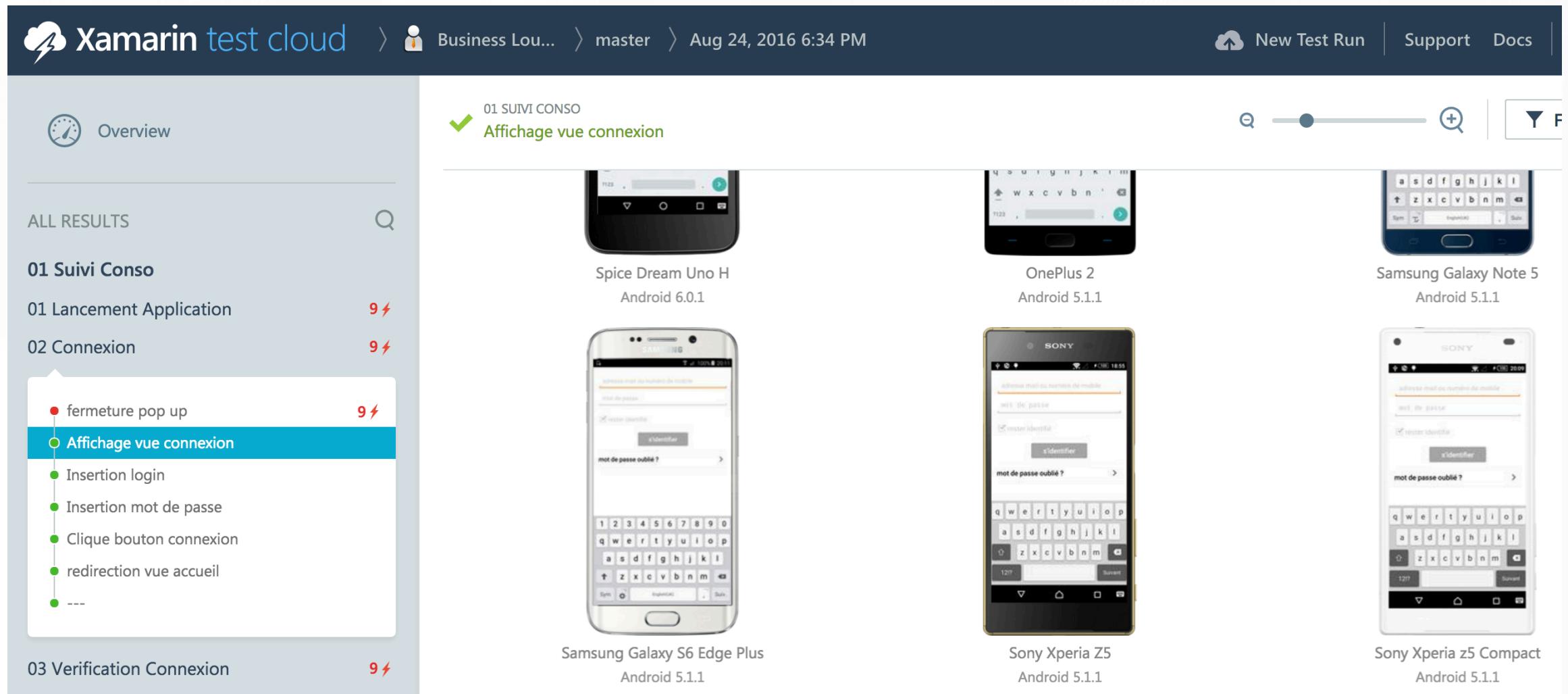
OnePlus 2
Android 5.1.1

Samsung Galaxy Note 5
Android 5.1.1

Samsung Galaxy S6 Edge Plus
Android 5.1.1

Sony Xperia Z5
Android 5.1.1

Sony Xperia z5 Compact
Android 5.1.1





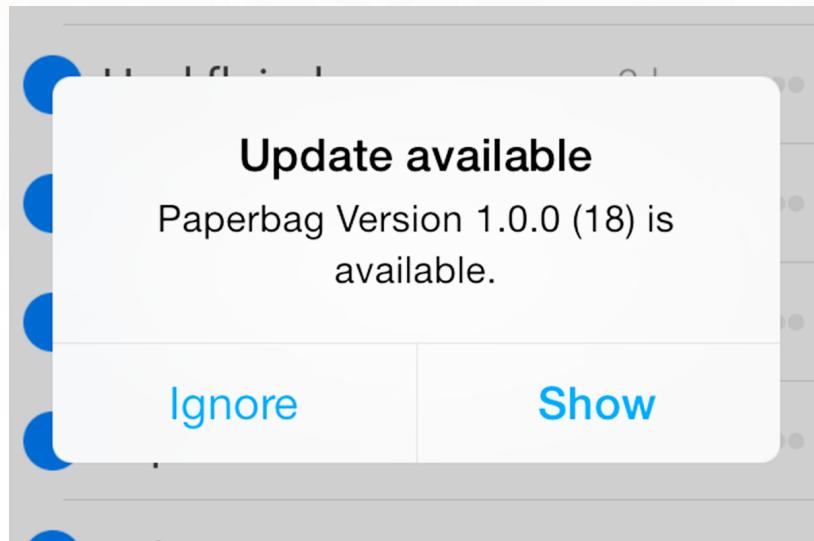
Test humain
Valider la cohérence générale

Pour le test humain, la mise à disposition de l'application à un panel d'utilisateurs va nous permettre d'obtenir un avis plus subjectif sur la qualité du produit.

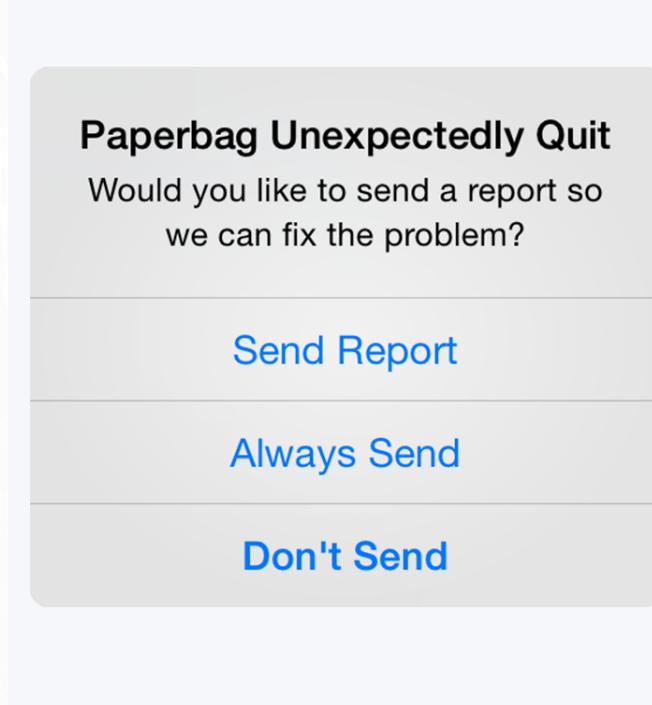
Il sera alors simple de reproduire des scénarios de tests afin de s'assurer de la non régression de la qualité de notre programme

HockeyApp

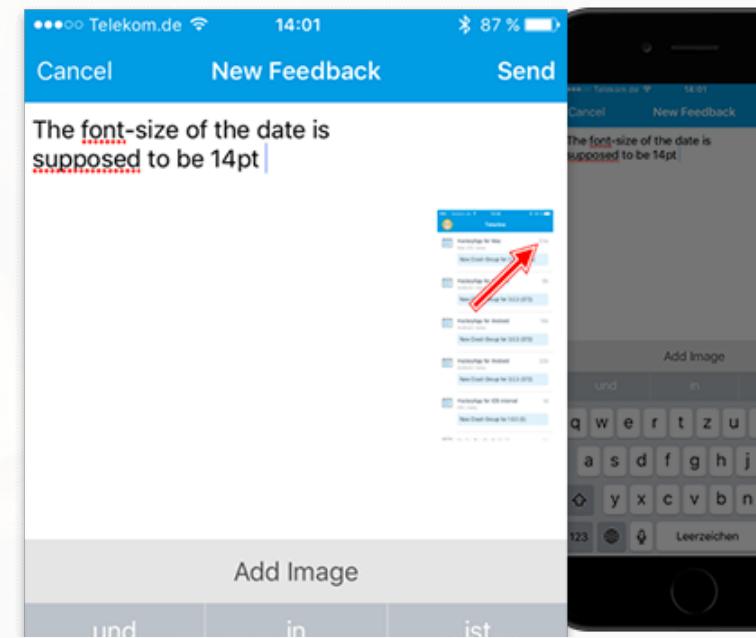
Distribution d'application



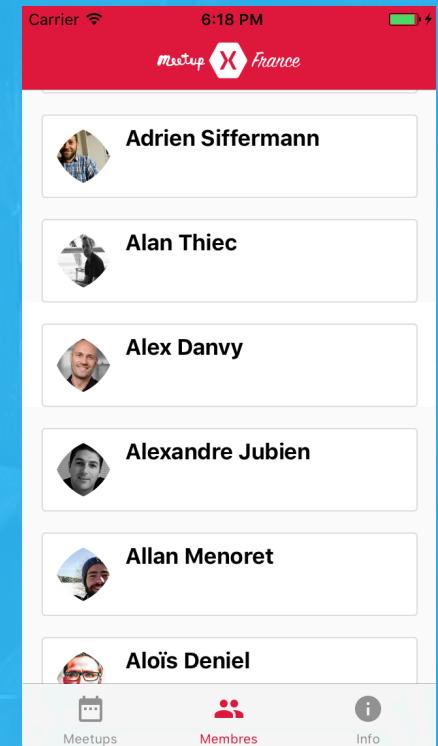
Crashes



Retours



Démonstration



<http://www.github.com/aloisdeniel/Meetup.Xamarin.France.Demo>



Merci
Des question ?