



Présentation de Xamarin



Yan Maniez
@nioux59



Yassine BENABBAS
@yostane

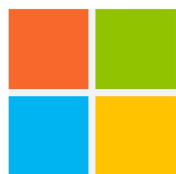
Meetup .Net Lille 30/10/2018



Présentation de Xamarin



Yassine BENABBAS
@yostane



Plan

- Introduction
- Principe
- Démo + étude de cas
- Plus et moins
- Conclusion



“Xamarin permet de créer des applications Android, iOS et Windows **natives**, avec une seule base de code .NET partagée.”

Historique

- Avant 2000: l'ère "avant .net"
 - Win32 / COM, langages C++ et VB
 - Arrivée de Java, MS adopte et sort sa MSVM, puis procès avec Sun, abandon de Java,



ère "avant .net"

2000



Historique

- 2000: lancement de .NET 1.0, C#, VB.NET, .NET Framework



ère "avant .net"



2000



Historique

- 2007: Avec Vista 🖱️, arrivée de WPF, XAML
- En parallèle, en 2004, création de Mono, portage de .NET pour Linux



ère "avant .net"



2000



2004



2007

Arrivée de Xamarin



2011: Xamarin.iOS et Xamarin.Android: .NET pour Android et iOS

Xamarin se base sur Mono

Produit payant 🖱️

Le compilateur transforme du code .NET en code natif selon la plateforme

+ libraries qui proposent des fonctions de base de .NET

2016: Rachat par MS en pleine déconfiture de Windows Mobile / Windows Phone, changement de stratégie mobile

Maintenant gratuit à l'usage. Inclus dans la licence Visual Studio de base

Plateforme .NET

.netstandard: ensemble d'API

.Net Framework: implémentation sous windows

.Net core: implémentation macOS, Linux et windows

Mono: implémentation sous linux

Depuis l'arrivée de .netstandard. Le framework .net est multiplateformes

Quelques applications Xamarin

Locate the restaurant near by you



RAPIDE ET CLAIR

Recherchez et réservez des vols rapidement



low frank

Hanx Writer

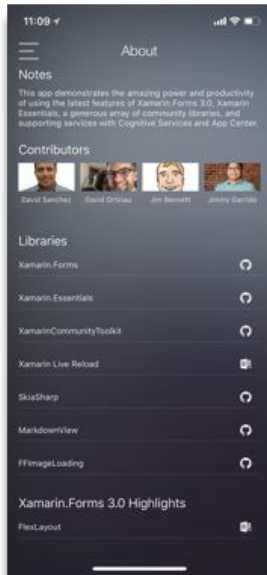


Xamarin Forms

Framework graphique pour Xamarin, début en 2014

Permet de développer l'interface graphique avec un seul code XAML / C#

Exemples avec src: <https://github.com/jsuarezruiz/xamarin-forms-goodlooking-UI>



Plateformes supportées par Xamarin Forms

Au début : Choix limité à Xamarin.iOS / Xamarin.Android / WinPhone 8

+ code commun en “shared code” ou en PCL

Evolution vers Windows 10

Remplacement du PCL (compliqué) par du .NET Standard -> perte de Windows Phone

Support de Tizen ajouté, macOS en beta, WPF en alpha, GTK, “web” en third party (<https://github.com/praeclarum/Ooui>), ...



Plusieurs possibilités

- 0% de code commun
 - Vue avec technologies natives
 - Code C#. Toutes les API ont un équivalent en C#.
- 100% de code commun
 - Vues codées avec Xamarin Forms
 - Pas accès aux fonctionnalités spécifiques
- Code et vues communes avec fonctionnalités spécifiques
 - Le meilleur des deux mondes
- Ou toute autre combinaison possible



**Possibilité d'appeler des bibliothèques écrites en
Java/C++/Objective-C/swift/Kotlin**

Architecture



Xamarin project written in C#



iOS

Platform specific features

Platform specific UI

Platform specific code



Android

Platform specific features

Platform specific UI

Platform specific code



UWP

Platform specific features

Platform specific UI

Platform specific code

Other targets:
macOS, WPF, ...

Platform specific features

Platform specific UI

Platform specific code

Common code via **.Net standard** or *shared project*

Autofac, MVVM Light

XAMARIN.FORMS

JSON.Net

Entity Framework Core

.NET Core

Other Libraries

Outils de développement



- Windows: Visual Studio 2017
 - Nécessite un mac configuré avec les outils de dev dans le réseau pour bénéficier de la partie iOS
 - Version la plus à jour et complète
- macOS: Visual Studio Mac
 - Basé sur Xamarin IDE mais a bien évolué depuis
 - Ne permet pas de cibler UWP
 - Moins à jour que son pendant windows
 - Moins de fonctionnalités
- Nécessite l'installation des SDK ou outils natifs (Android SDK, Xcode, etc.)
- Un projet Xamarin peut être fait de A à Z dans Visual Studio

VS 2019 prévue pour très bientôt

Librairies et API courantes

- .netstandard 2
 - HTTPClient
 - JSON, XML
 - Accès aux fichiers
 - etc.
- Persistance / ORM: Entity Framework Core
- Injection de dépendances
 - MVVM Light, Autofac
- Xamarin Essentials: API X platform
 - <https://github.com/xamarin/Essentials>
- Et plein d'autres

Entity Framework



MVVM
toolkit



Json.NET



Use Case : “HACHES & DÉS”

“LE PLUS GRAND JEU DE RÔLE AU MONDE”™

Depuis la version 3.0, publication d'un **SRD** (Systems Reference Document), utilisable par d'autres acteurs sous licence **OGL** (Open Game License).

Le SRD contient ce qui est publié en “Open Game Content”, le reste en “Product Identity”.

Pour sa dernière version 5.1 publiée en juillet 2014 en VO et en avril 2017 en VF, aucune app disponible sur les stores mobiles proposant le contenu du SRD français.

“Fork” français Héros & Dragons (ou H&D) offrant davantage de contenu open content.

Données “libres”

+ Xamarin.Forms à tester

= “HACHES & DÉS”



Structure du projet

- Utilisation de **Visual Studio 2017 Community**
 - gratuit pour les particuliers et petites entreprises
 - gratuit en entreprise pour les projets open source ou les formations
 - <https://www.visualstudio.com/fr/license-terms/mlt553321/>
- Projet Xamarin.Forms “netstandard” + ASP.NET Core + console .NET Core
- Sources du projet sur **GitHub** <https://github.com/Nioux/AideDeJeu>
- Publication sur **Google Play**
<https://play.google.com/store/apps/details?id=com.nioux.aidedejeu>
- Publication sur **Windows Store**
<https://www.microsoft.com/fr-fr/p/aide-de-jeu/9nvns0j25ct7>
- Version tournant sur **GTK (win + lin + mac)**
- Publication sur **Azure** d’un projet secondaire

Projet principal

Listes filtrées et pages de contenu du SRD avec interface en Xamarin.Forms

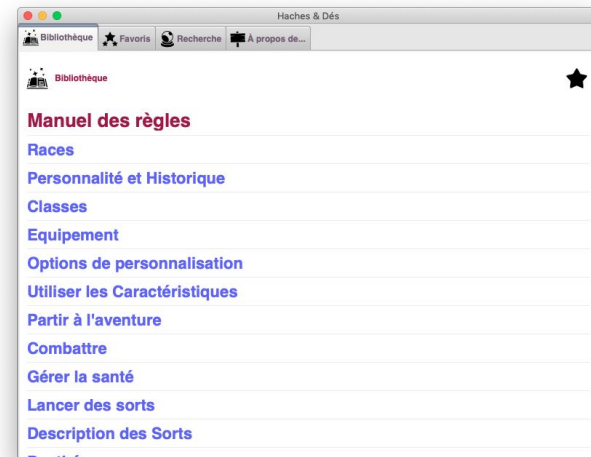
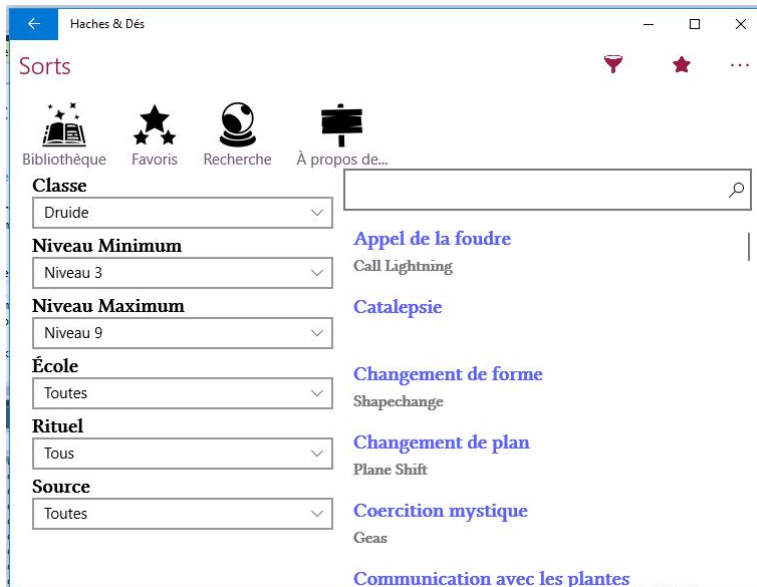


← Archimage ★

avec les attaques de sort).

L'archimage peut lancer déguisement et invisibilité à volonté et dispose des sorts de magicien préparés suivants :

- **Tours de magie (à volonté)** : [trait de feu](#), [lumièrre, main du mage](#), [poigne électrique](#), [prestidigitation](#)
- **1er niveau (4 emplacements)** : [armure du mage*](#), [détection de la magie](#), [identification](#), [projectile magique](#)
- **2e niveau (3 emplacements)** : [détection des pensées](#), [image miroir](#), [pas brumeux](#)
- **3e niveau (3 emplacements)** : [contresort](#), [éclair](#), [vol](#)
- **4e niveau (3 emplacements)** : [bannissement](#), [bouclier de feu](#), [peau de pierre*](#)
- **5e niveau (3 emplacements)** : [cône de froid](#), [mur de force](#), [scrutation](#)
- **6e niveau (1 emplacement)** : [globe d'invulnérabilité](#)
- **7e niveau (1 emplacement)** : [téléportation](#)



✓ Android

✓ iOS

✓ UWP

✓ GTK

30 derniers jours 24 mai 2018 – 22 juin 2018

143

Au 22 juin

+60,7 %

par rapport au 23 mai



Intervalles d'une journée

Par version de l'application

Toutes les versions de l'application 143 (100,0 %)

15 137 (95,8 %)

14 3 (2,1 %)

5,0



11 au total

5

4

3

2

1



Guillaume LEFRANC

★★★★★ 22 mai 2018

Super appli, bien pratique. Mes lanceurs de sorts te remercient!



Yan Maniez 22 mai 2018

Merci ;)



genesteal

★★★★★ 19 mai 2018

Une application indispensable pour tout les MJ de DD5 francophones. Tout vos monstres, sorts SRD sous la main en un clin d'oeil. Merci, à l'auteur, un passionné de nous offrir cet outil pour les francophones.



Yan Maniez 18 mai 2018

Merci ;)



Marc Romanini

★★★★★ 8 juin 2018




Projet secondaire

API REST basée sur **ASP.NET Core** de transformation des données à la volée vers un json utilisé par le “rpg card maker” crobi.github.io/rpg-cards/

aidedejeuweb.azurewebsites.net/api/values/Clerc



BURNING HANDS 

1st level evocation

Casting time 1 action
Range Self (15ft cone)
Components V, S

Each creature in a 15-foot cone takes **3d6 fire** damage.

The fire ignites any flammable objects in the area that aren't being worn or carried.

AT HIGHER LEVELS

+1d6 damage for each slot above 1st

Projet utilitaire annexe

“moulinettes” dans une application console .NET Core pour transformer les données dans les formats utilisés et à remplir la base de données

```
C:\Program Files\dotnet\dotnet.exe
* [Liane chasserresse](spells_hd.md#liane-chasserresse)
* [Liberté de mouvement](spells_hd.md#liberté-de-mouvement)
* [Localiser une créature](spells_hd.md#localiser-une-créature)
* [Peau de pierre](spells_hd.md#peau-de-pierre)

### Niveau 5
* [Arc enchanté](spells_hd.md#arc-enchanté)
* [Communion avec la nature](spells_hd.md#communion-avec-la-nature)
* [Nuée de projectiles](spells_hd.md#nuée-de-projectiles)
* [Passage par les arbres](spells_hd.md#passage-par-les-arbres)

## Sorcier

### Tours de magie
* [Aura du héros](spells_hd.md#aura-du-héros)
* [Bouffée de poison](spells_hd.md#bouffée-de-poison)
* [Contact glacial](spells_hd.md#contact-glacial)
* [Explosion occulte](spells_hd.md#explosion-occulte)
* [Illusion mineure](spells_hd.md#illusion-mineure)
* [Main du mage](spells_hd.md#main-du-mage)
* [Porte-bonheur](spells_hd.md#porte-bonheur)
* [Prestidigitation](spells_hd.md#prestidigitation)
* [Viser juste](spells_hd.md#viser-juste)

### Niveau 1
* [Charme-personne](spells_hd.md#charme-personne)
* [Compréhension des langues](spells_hd.md#compréhension-des-langues)
* [Flamboiemment funeste](spells_hd.md#flamboiemment-funeste)
* [Manteau de givre](spells_hd.md#manteau-de-givre)
* [Protection contre le mal et le bien](spells_hd.md#protection-contre-le-mal-et-le-bien)
* [Putréfaction](spells_hd.md#putréfaction)
* [Repli expéditif](spells_hd.md#repli-expéditif)
* [Représailles infernales](spells_hd.md#représailles-infernales)
* [Serveur invisible](spells_hd.md#serveur-invisible)
* [Strangulation](spells_hd.md#strangulation)
* [Texte illusoire](spells_hd.md#texte-illusoire)

### Niveau 2
* [Briser](spells_hd.md#briser)
```


CH'TITE DÉMO

Caractéristiques techniques

1. MVVM
2. Base de données EFCore sur SQLite
3. Parsing et affichage de markdown
4. Production de JSON



... et tout ça en factorisant / “déplaçant” le 🐙 même code 🐙 entre les projets mobile, serveur et ligne de commande, grâce à netstandard

Avantages de Xamarin Forms

- Très très souple de jongler entre projets Xamarin Forms, ASP.NET et Windows classique
- Binding XAML !
- Sous Android, pas de gestion des saved states car une seule activity
- Possible de n'avoir qu'un Mac pour compiler sur plusieurs PC la version iOS
- Support Microsoft, open source, évolution très rapide
- Langage C#



Avantage de C#

- Langage qui évolue régulièrement
- Compilateur open source
- Support natif de `async / await`
- Facile à prendre en main pour les développeurs Java
- LINQ
- Expressions lambda
- Et autres:
 - Namespace, Tuples, Propriétés, Indexeurs, Surcharge d'opérateurs
 - Delegates et évènements, Extension de classes



Inconvénients de Xamarin Forms

- Encore un peu jeune, parfois des couacs d'upgrade
 - ex : SkiaSharp 1.60.1 empêche déploiement sur Windows Store alors que 1.60.0 fonctionne
- Démarrage d'une app encore assez lent, fait partie de la roadmap
- XAML pas encore standardisé



Différentes sauces de XAML

Multiplés versions de XAML : WPF != UWP != Xamarin Forms

Note : étrangement une "ListView" mais pas de "GridView" proposée de base

Futur -> XAML standard ?

```
<!-- Xamarin.Forms XAML -->
<ContentView>
  <StackLayout Orientation="Horizontal">
    <Label Text="Work Order #"
      VerticalOptions="Center"/>
    <Entry Placeholder="Enter your work order"
      Text="{Binding WONumber, Mode=TwoWay}"/>
    <Button Text="Save"
      TextColor="White"
      BackgroundColor="#77D065"
      Command="{Binding SaveCommand}"/>
  </StackLayout>
</ContentView>
```


```
<!-- UWP XAML -->
<UserControl>
  <StackPanel Orientation="Horizontal">
    <TextBlock Text="Work Order #"
      VerticalAlignment="Center" />
    <TextBox PlaceholderText="Enter your work order"
      Text="{Binding WONumber, Mode=TwoWay}" />
    <Button Content="Save"
      Foreground="White"
      Background="#77D065"
      Command="{Binding SaveCommand}" />
  </StackPanel>
</UserControl>
```

Ce que ne fait pas Xamarin

- Pas de miracle pour les assets
 - Nécessaire de les préparer pour chaque plateforme, même si possible de les incorporer dans lib commune
 - Pas possible d'utiliser du SVG partout (pour l'instant)
- Ne divise pas le temps de test par 2
 - Indispensable de tester chaque changement important sur toutes les plateformes cibles
- Ne fait pas plus que ce que proposent les SDK natifs
- Il faut mettre à jour Xamarin pour suivre les MAJ des SDK des plateformes natives

Conclusion

- Technologie Xamarin / Xamarin Forms prometteuse et déjà exploitable en prod
- Supportée par Microsoft
- Supporte plusieurs OS
- Concurrence rude: Flutter, React native, Native Script, etc.

A group of people are shown in a medium shot. In the foreground, a man with dark hair, wearing a teal polo shirt, is looking down and slightly to the left. To his right, a woman with short dark hair, wearing a white turtleneck sweater, is looking directly at the camera with a neutral expression. In the background, a man with glasses and a blue button-down shirt is also looking towards the camera. The background is a plain, light-colored wall.

**MERCI ♥
A Vos questions !**

Roadmap / futur

<https://github.com/xamarin/Xamarin.Forms/wiki/Feature-Roadmap>

<https://blog.xamarin.com/glimpse-future-xamarin-forms-3-0/>

<https://github.com/jsuarezruiz/xamarin-forms-goodlooking-UI>

embedding, perfs

macOS, WPF, Linux, GTK#

XAML Standard