GRENOBLE INP UGA Ensimag

École nationale supérieure d'informatique et de mathématiques appliquées

Présentation de la plateforme .NET

Mnacho Echenim

Présentation des encadrants

Mnacho Echenim

- Enseignant-chercheur Ensimag
- mnacho.echenim@univ-grenoble-alpes.fr

Jonas Maubert

- Ingénieur développement logiciel Raise Partner
- jonas.maubert@raisepartner.com

Pourquoi utiliser cette plateforme?

- Très demandé par les institutions financières
  - **PFE 2022**: 22 stages sur 38
- Utilisé dans d'autres enseignements
  - Projet de Couverture Multiflux
  - Projet Evaluation de Produits Structurés
- **Nb**: vous ne serez pas des experts à la fin du projet
  - Il y a trop de sujets à couvrir et pas suffisamment de temps
  - La formation se poursuivra pendant les autres TP et projets

Présentation de la plateforme

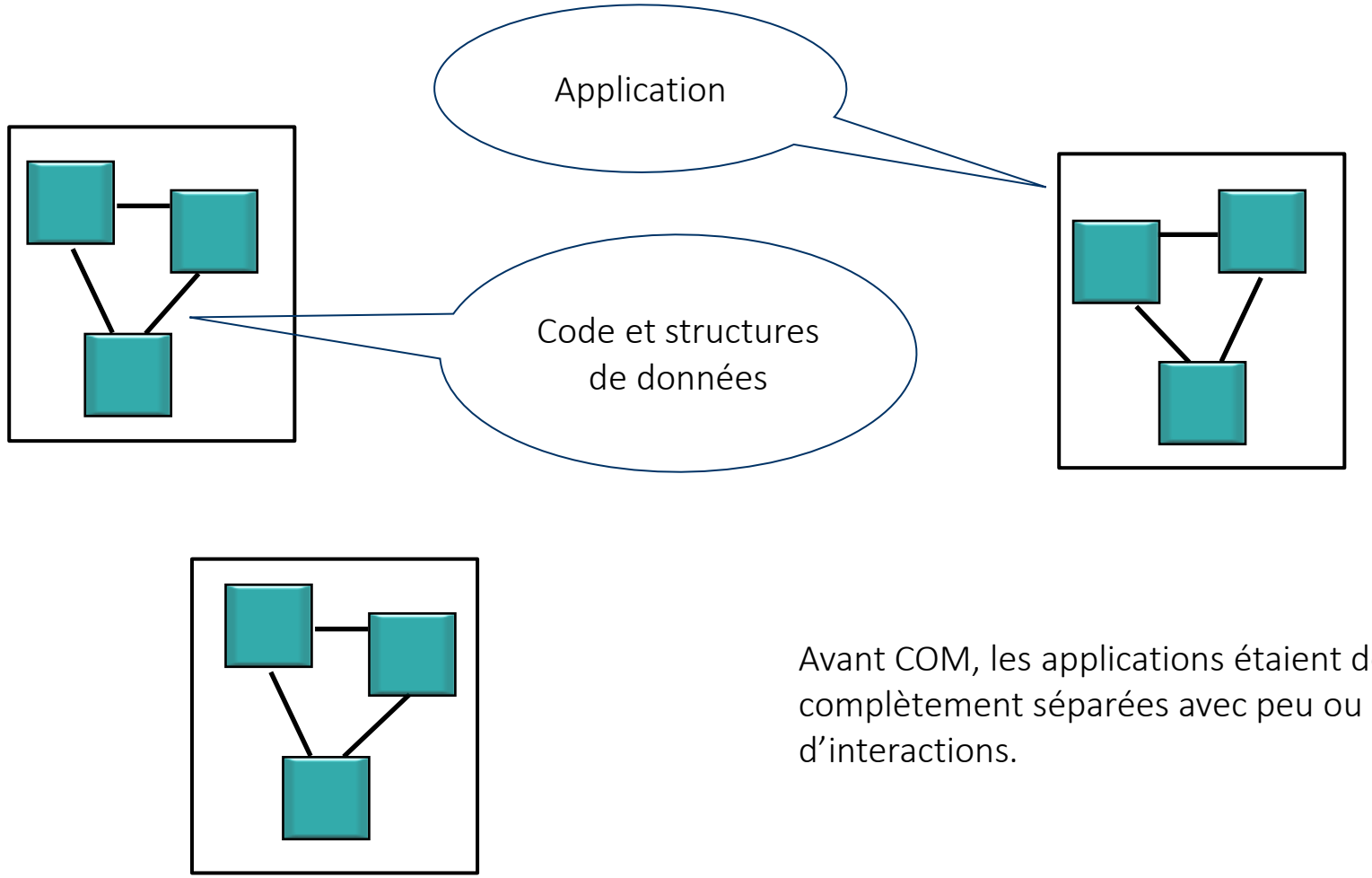
Qu’est-ce que la plateforme .NET?

- Une plateforme de développement logiciel
- Développement simple d’applications distribuées
  - Compatibilité inter-langages
  - Facilité de communication
- Open-source et cross-platform

Pourquoi une nouvelle plateforme?

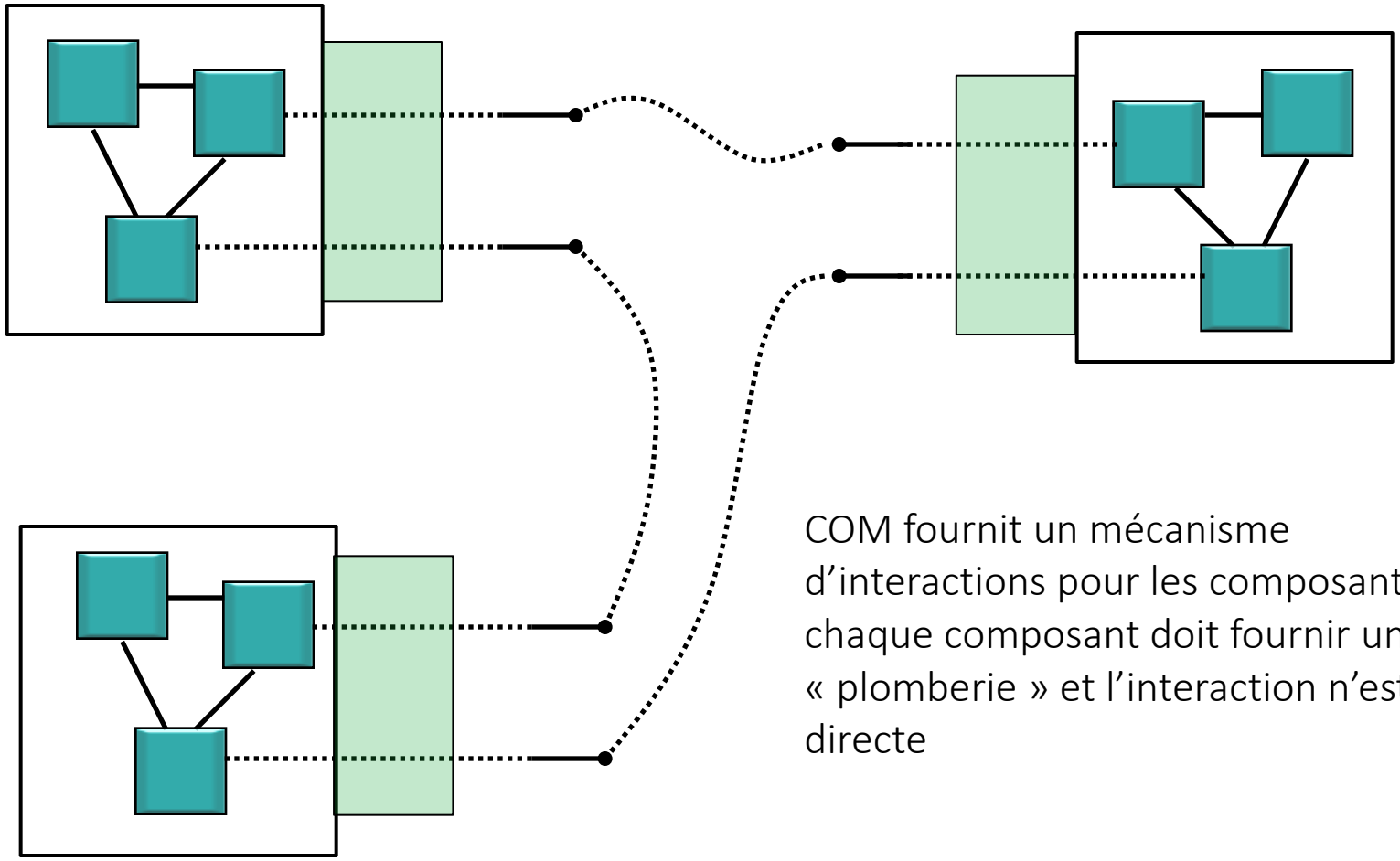
- Résoudre ‘l’Enfer des dlls’ sur Windows
- Déploiement simplifié
- Réutilisation simplifiée de code
- Intégration de nombreux langages

L’évolution vers .NET



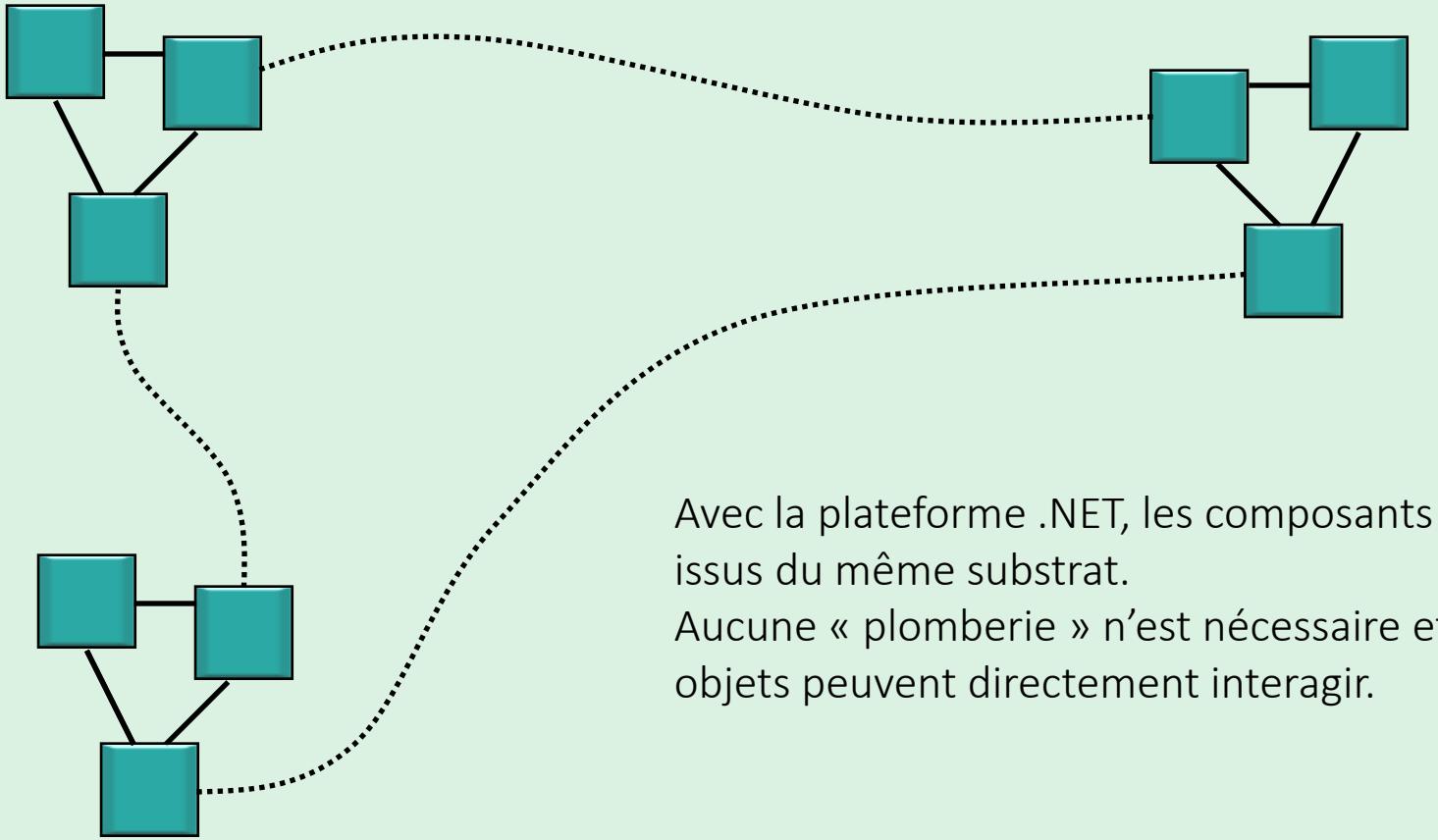
Avant COM, les applications étaient des entités complètement séparées avec peu ou pas d’interactions.

L’évolution vers .NET



COM fournit un mécanisme d’interactions pour les composants. Mais chaque composant doit fournir une « plomberie » et l’interaction n’est pas directe

L’évolution vers .NET (suite)



Avec la plateforme .NET, les composants sont issus du même substrat.
Aucune « plomberie » n’est nécessaire et les objets peuvent directement interagir.

Démo multilangages

Pourquoi ça marche?

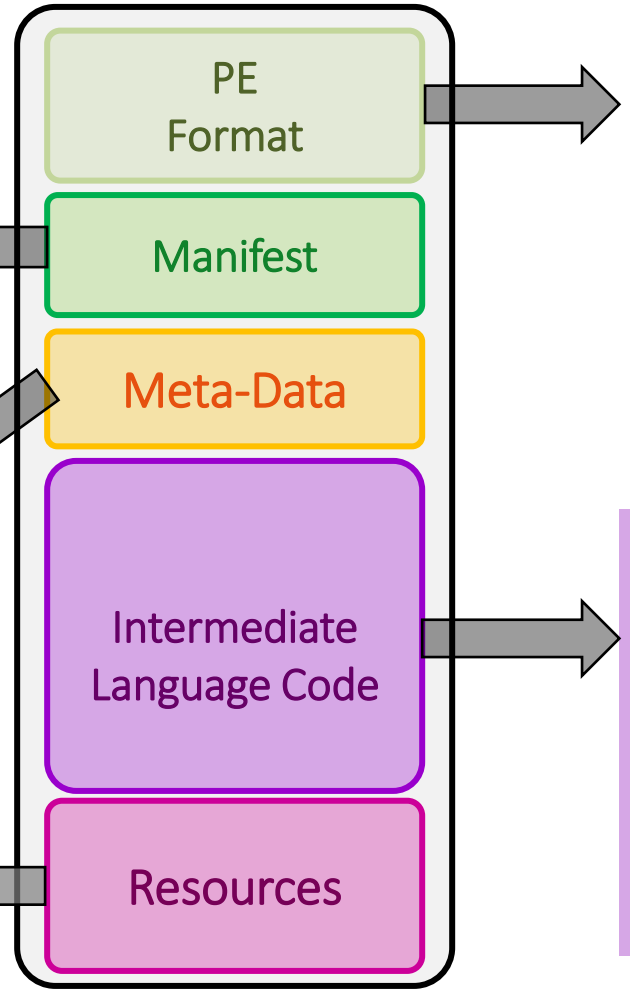
- Trois compilateurs sont évoqués:
  - Instrument.vb => *vbc.exe* => Instrument.dll
  - Option.cpp => *cl.exe* => Option.dll
  - PortefeuilleOptions.cs => *csc.exe* => PortefeuilleOptions.dll
- Ces trois fichiers contiennent **exactement** le même type d'information.
- Ce sont des *assemblies .NET*

Les assemblies .NET (*.dll, .exe*)

- Nom de l'assembly
- Version
- Culture
- Strong Name
- Assemblies référencés
- Fichiers de l'assembly

Description des types exposés (classes, methodes, propriétés et prototypages)

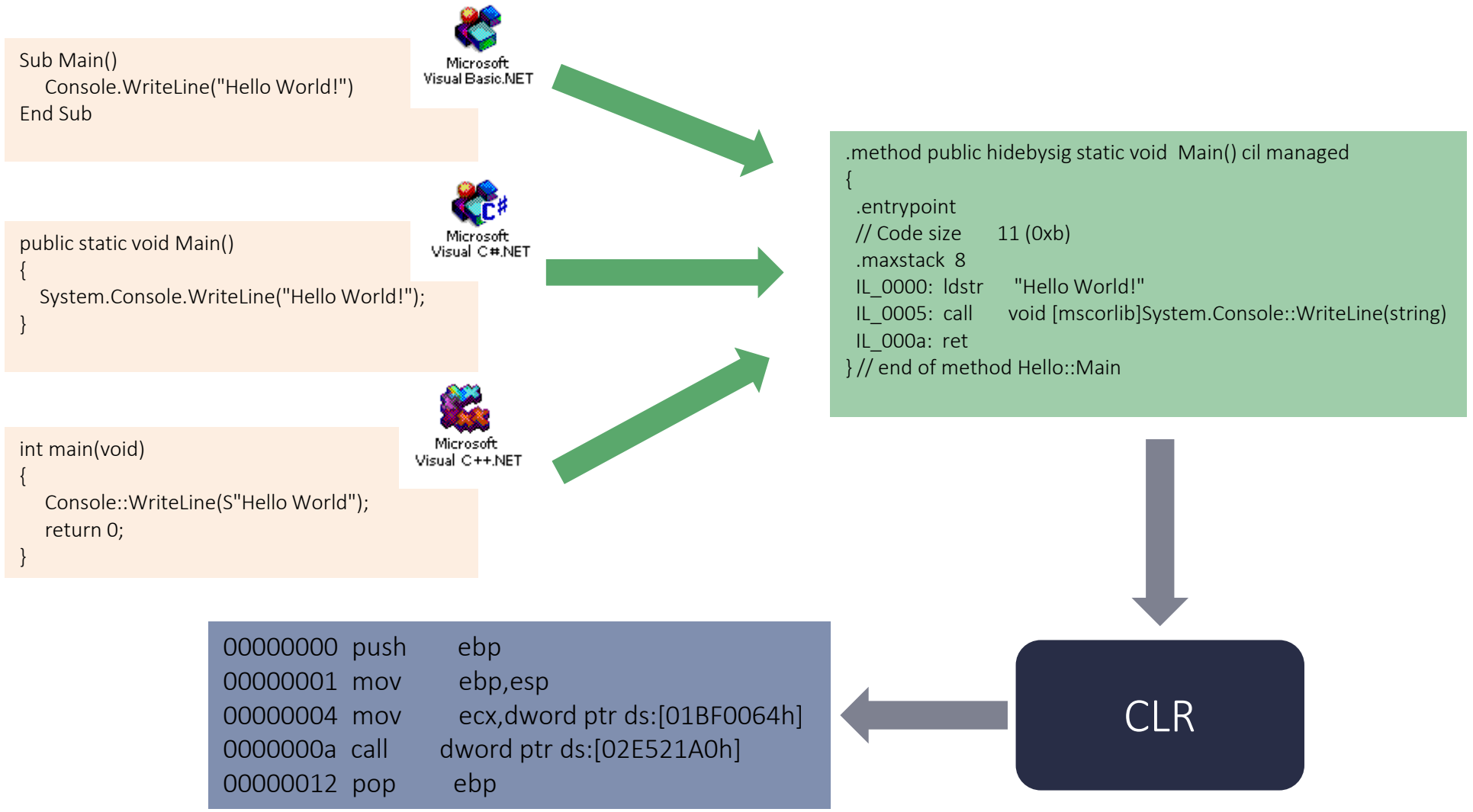
Fichiers texte, vidéos, images…



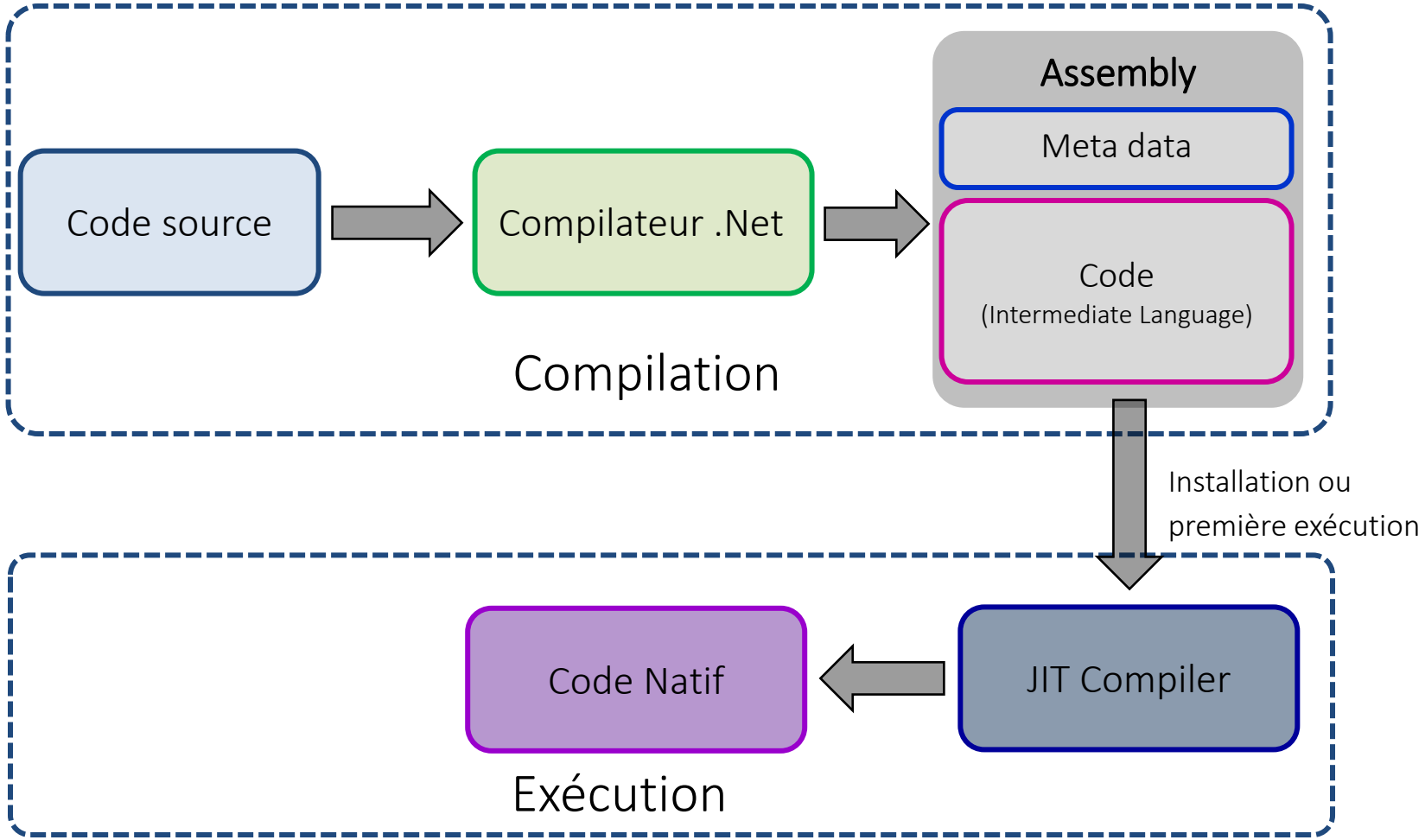

```
File Type: EXECUTABLE IMAGE

FILE HEADER VALUES
            14C machine (x86)
              3 number of sections
        3E78E307 time date stamp Wed Mar 19 22:37:11 2003
              0 file pointer to symbol table
              0 number of symbols
             E0 size of optional header
             10E characteristics
                  Executable
                  Line numbers stripped
                  Symbols stripped
                  32 bit word machine
```

```
.method public hidebysig specialname rtspecialname
      instance void  .ctor() cil managed
{
                  .maxstack  2
                  ldarg.0
                  call       class [mscorlib]System.IO.TextWriter
[mscorlib]System.Console::get_Out()
                  call       instance void Dumper.Dumper::.ctor(class
[mscorlib]System.IO.TextWriter)
                  ret
} // end of method Dumper::.ctor
```

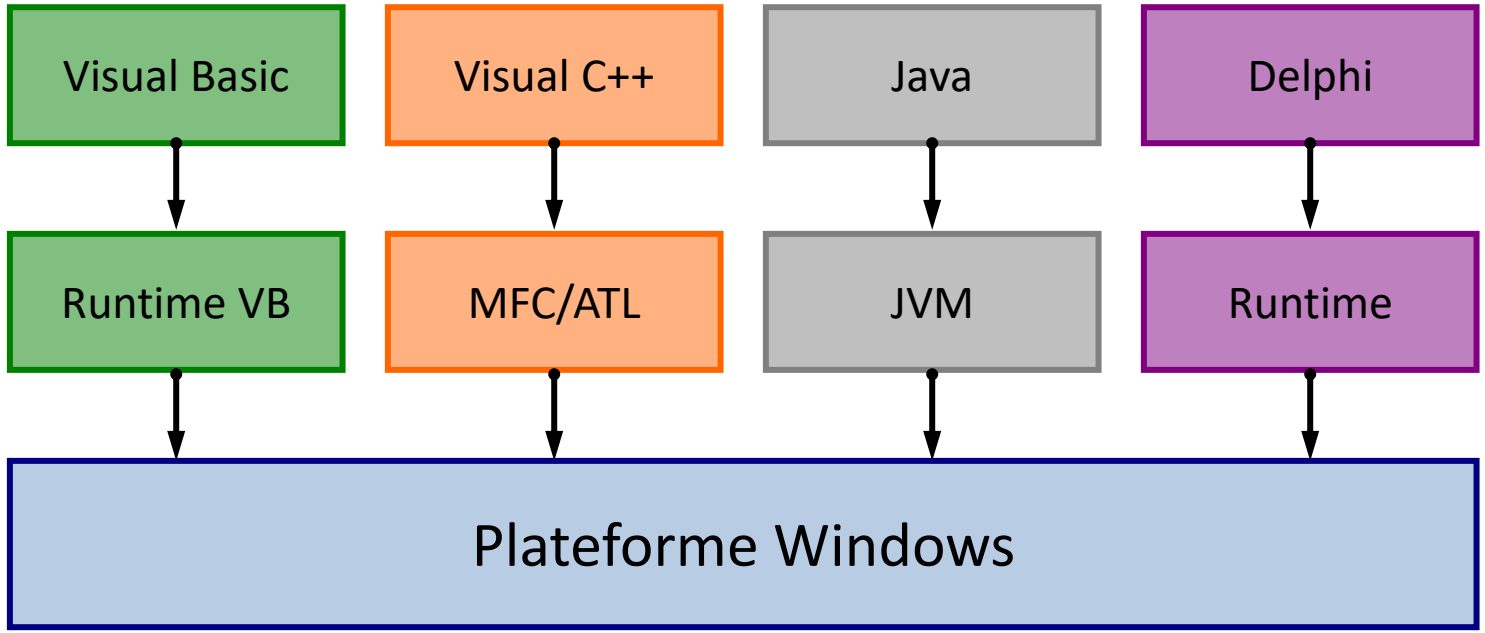
Compilation et exécution



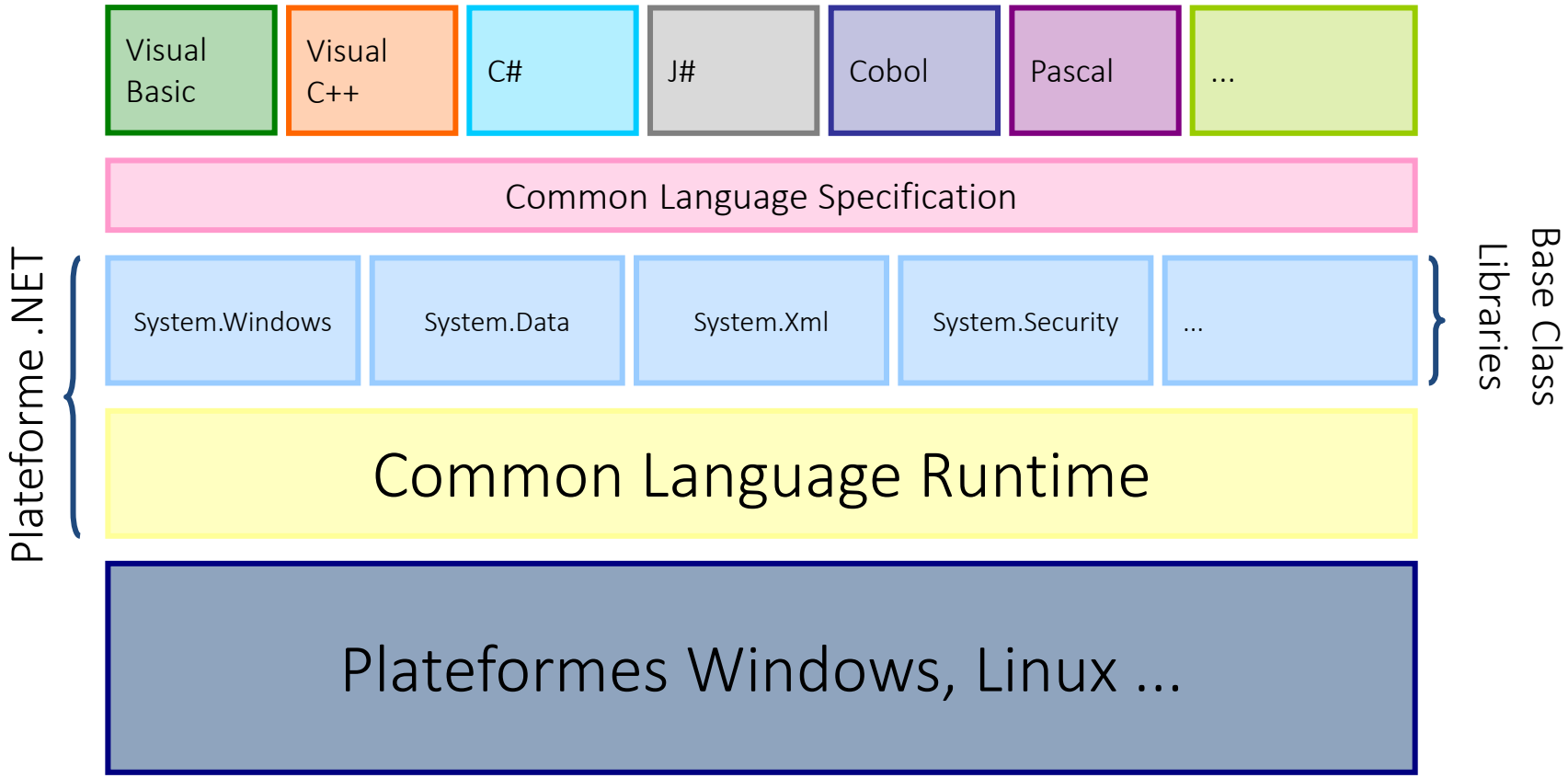
Compilation et exécution (suite)



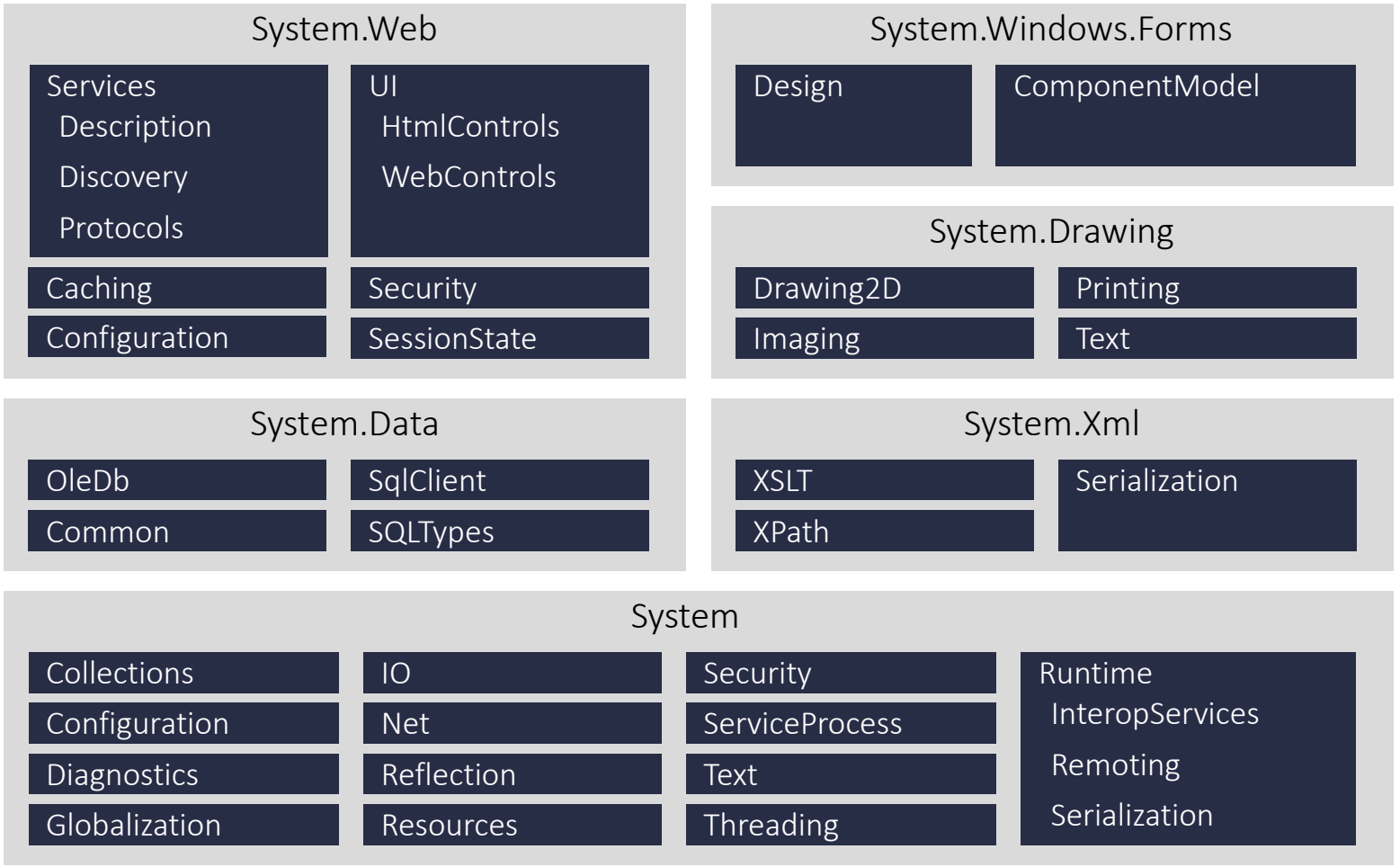
Modèle avant .NET



Modèle .NET


Visual Basic
Visual C++
C#
J#
Cobol
Pascal
...
Common Language Specification
Plateforme .NET
System.Windows
System.Data
System.Xml
System.Security
...
Base Class Libraries
Common Language Runtime
Plateformes Windows, Linux ...

Base Class Libraries



Compléments sur la CLR

- Prend du MSIL en entrée
- Compilation JIT: produit des instructions binaires qui dépendent de la plateforme, du processeur
  - Code compilé à l'exécution
  - Le code n'est pas interprété
- Quelques fonctionnalités:
  - Types unifiés
  - Debug
  - Ramasse-miette…

Ramasse-miettes

- Objets sont alloués sur un tas managé
- Collection effectuée quand pointeur interne passe la fin de l'espace d'adressage
  - Le GC vérifie si certains objets du tas ne sont plus utilisés
  - Destruction non-déterministe des objets pour améliorer les performances
  - Heuristiques puissantes pour lancement du ramasse-miette
  - On peut l'appeler explicitement mais c'est fortement déconseillé
- La durée de vie des objets peut être importante quand on cherche à améliorer la performance d'une application

Cross-platform

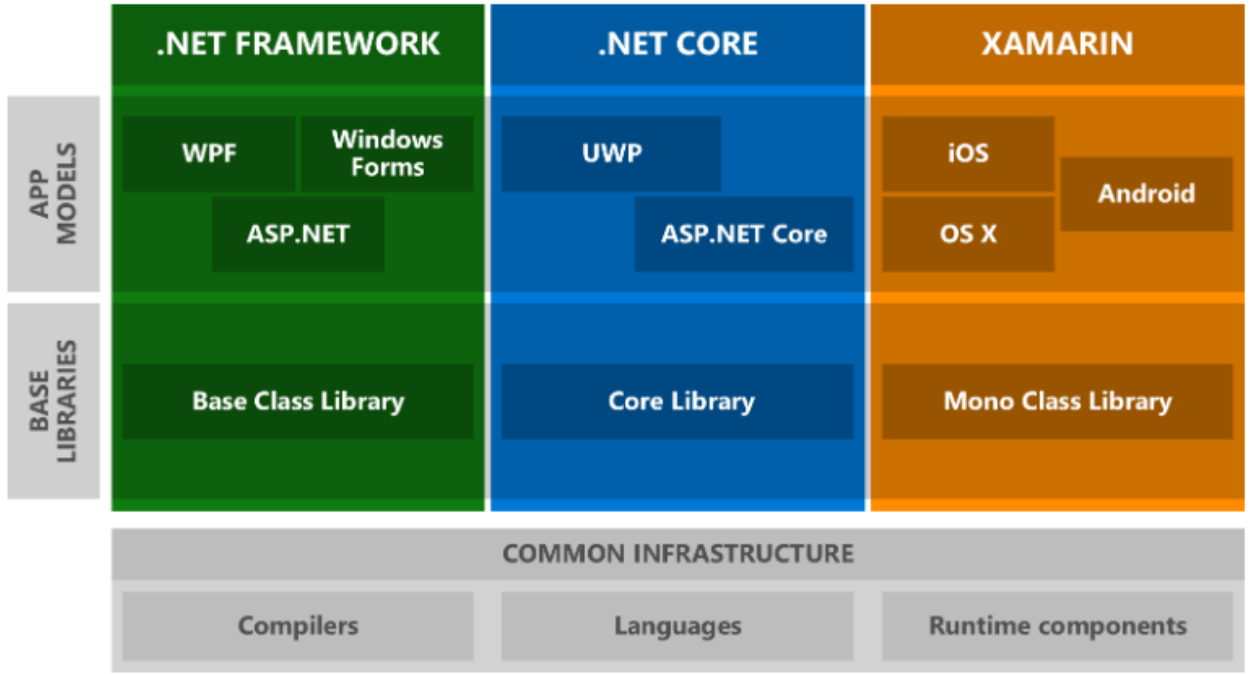
Historique

- .NET Framework
  - Plateforme d'origine (2002)
- .NET Compact Framework
  - Sous-ensemble de .NET Framework à l'origine
  - Destiné aux téléphones Windows Mobile
- ASP.NET 4
- ASP.NET 5
- Windows Store

Chaque plateforme évoluait de façon globalement indépendante

Une solution cross-platform

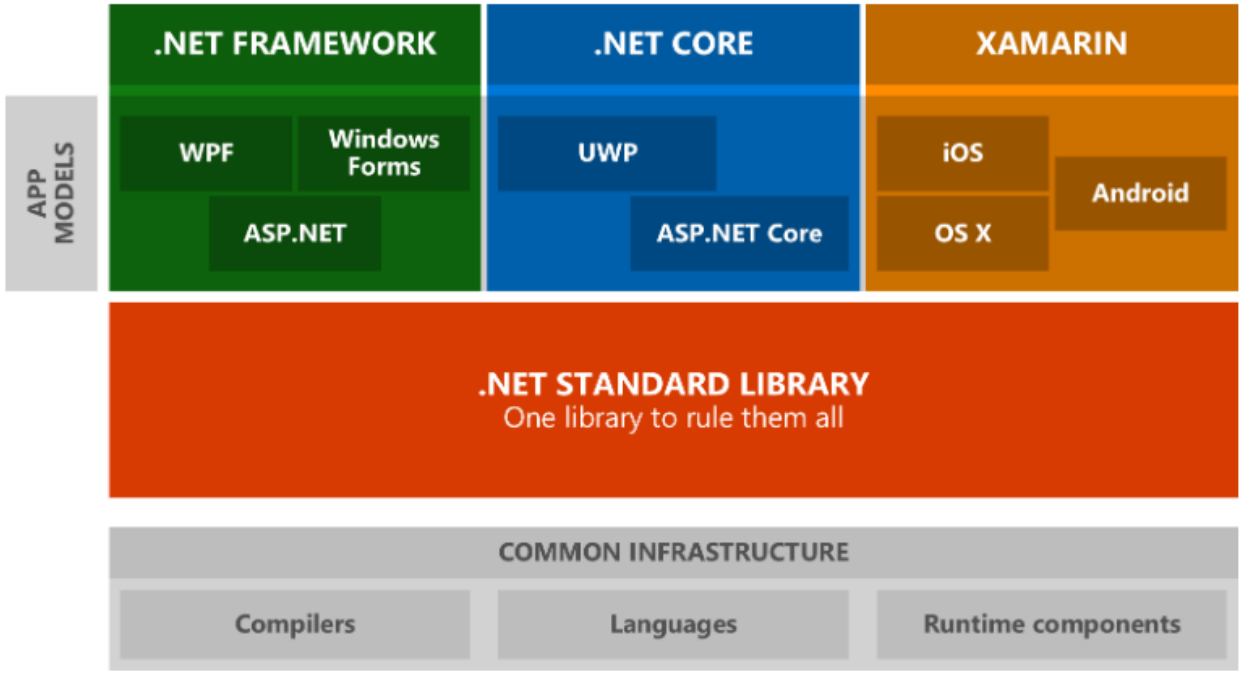
- .NET Core (2016): open-source et cross-platform



Source: https://devblogs.microsoft.com/dotnet/introducing-net-standard/

.NET standard

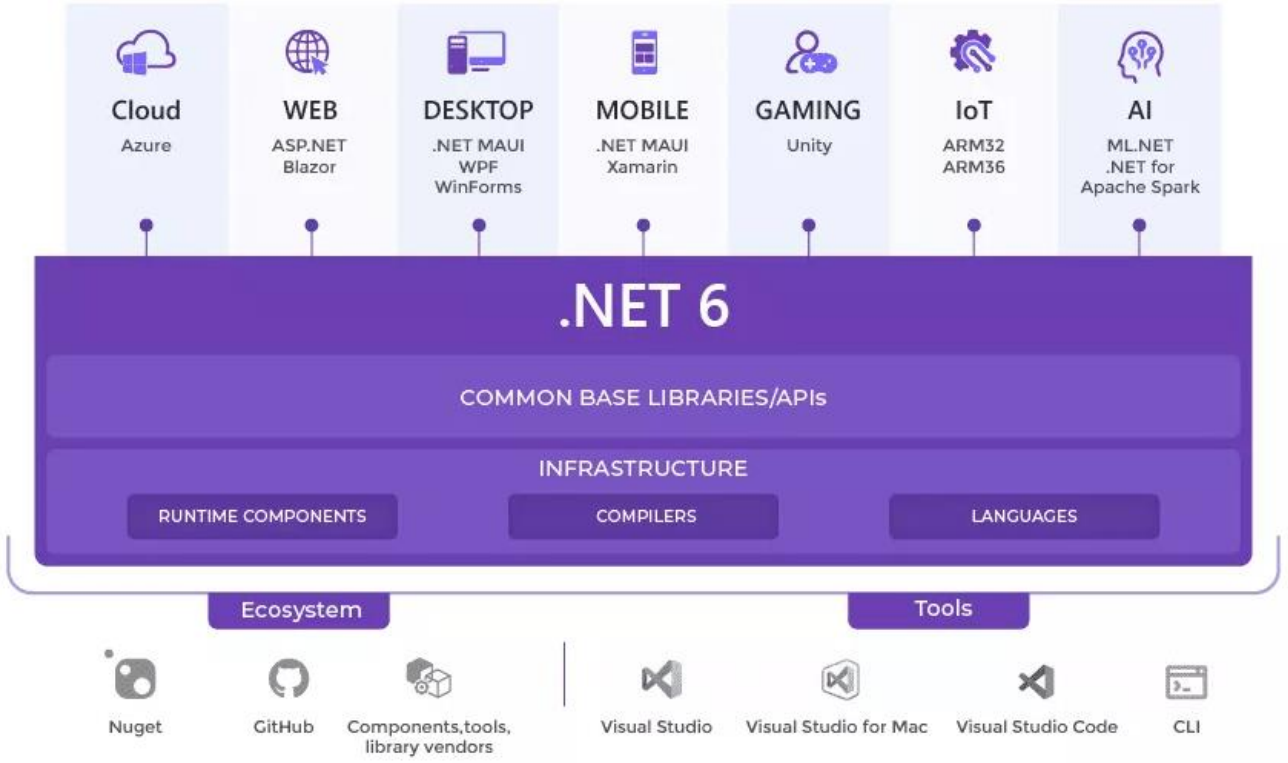
- Objectif: des librairies de base unifiées



Source: https://devblogs.microsoft.com/dotnet/introducing-net-standard/

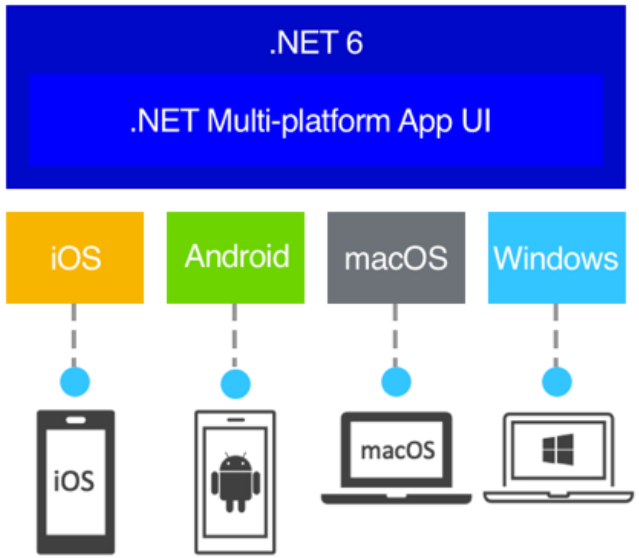
Des plateformes complètements unifiées

- .NET 5 (Nov. 2020) -- .NET 8 (Nov. 2023)

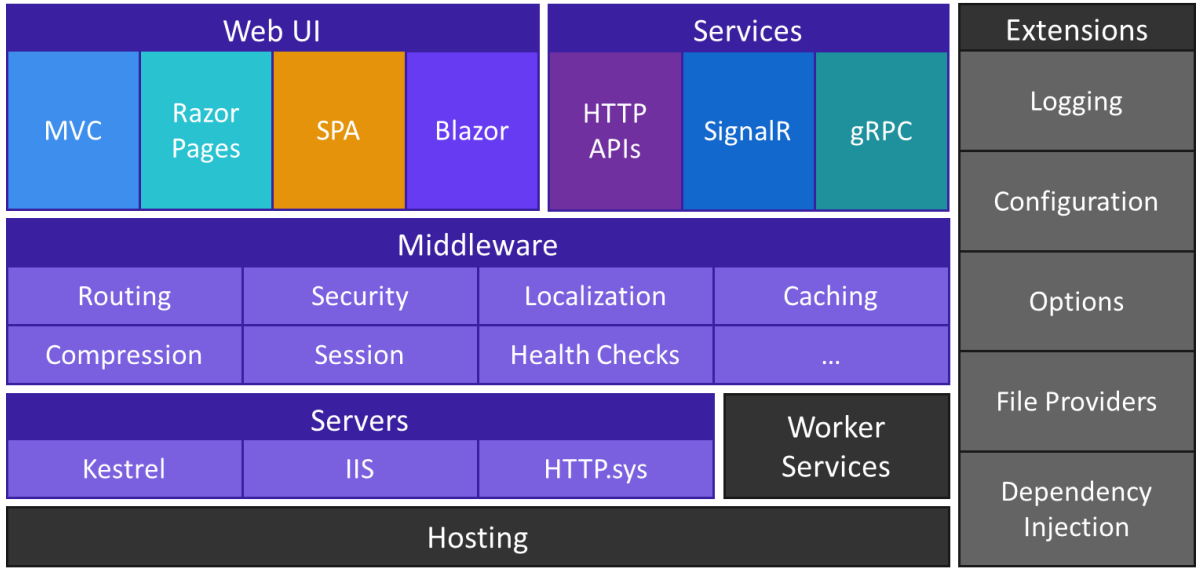


Source: https://www.rishabhsoft.com/blog/dotnet-6-features

Interfaces utilisateur

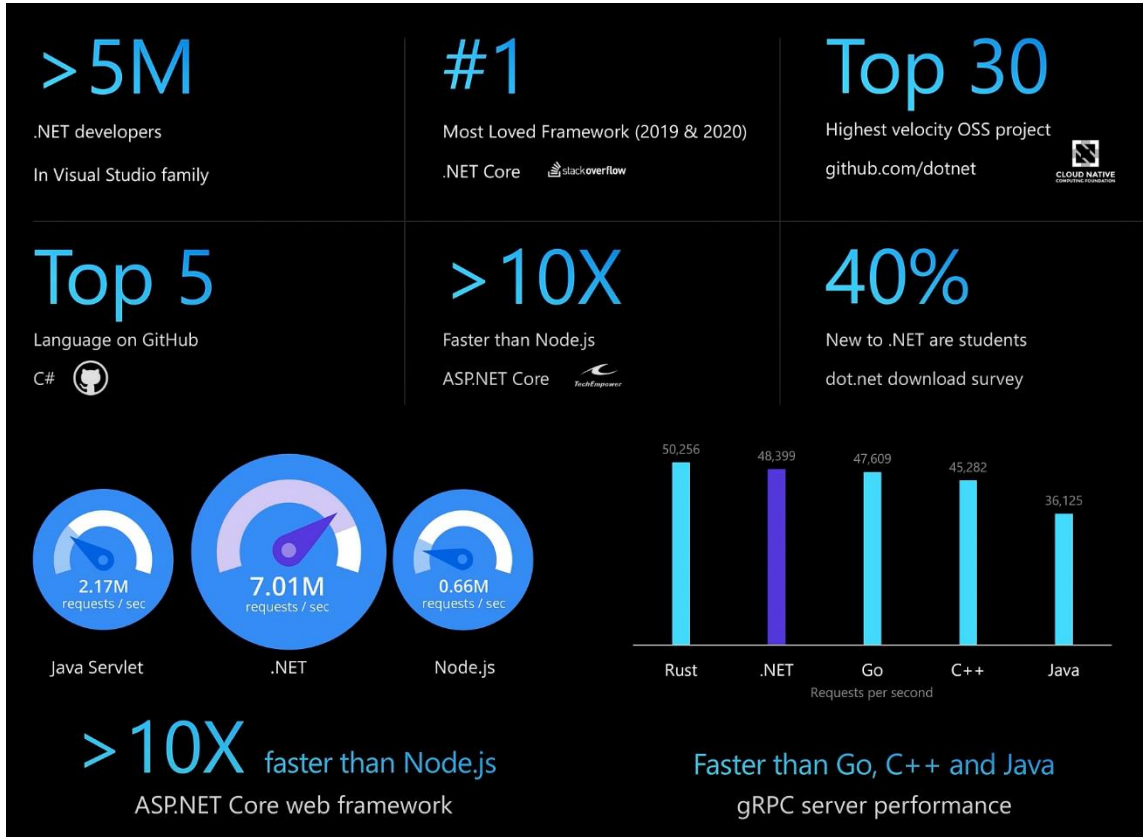


Source: https://docs.microsoft.com/fr-fr/dotnet/maui/what-is-maui



Source: https://www.codemag.com/Article/2111062/What%E2%80%99s-New-in-ASP.NET-Core-in-.NET-6

Etat des lieux



Source: ***.NET 6 deep dive.*** *https://docs.microsoft.com/en-us/events/build-may-2021/azure/breakouts/od485/*

Eléments du langage C#

Structure d'un programme C#

- L'exécution d'un programme commence avec Main()
- Le mot clé using fait référence à une ressource du framework .NET
- Premier exemple

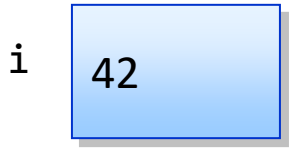
```
using System;
namespace MyHelloWorld
{
    class Program
    {
        static void Main()
        {
            Console.WriteLine("Hello, World!");
        }
    }
}
```

- **Rq**: la nouvelle version du langage nécessite encore moins de code

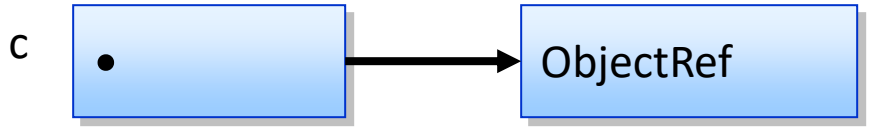
Types valués, de référence

- Type valué
  - Contient directement les données
  - Stocké sur la pile
  - Doit être initialisé
  - Ne peut pas être nul
- Type référence
  - Contient une référence aux données
  - Stocké sur le tas (managé)
  - Déclaré avec le mot clé new
  - Destruction gérée par le ramasse-miette

```
int i;
i = 42;
```




```
ObjectRef c;
```



Les espaces de nommage

- Espaces de nommage (imbriqués)

```
namespace NomDeLaCompagnie
{
    namespace Ventes
    {
        public class Client { }
    }
}
// ou bien

namespace NomDeLaCompagnie.Achats { ... }
```

- L'instruction using

```
using System;
using NomDeLaCompagnie.Ventes;
```

Accessibilité

- Mots-clés utilisés pour définir le niveau d'accessibilité des membres d'une classe

Déclaration	Définition
public	L'accès n'est pas limité
private	Accès limité à la classe même
internal	Accès limité à l'Assembly
protected	Accès limité à la classe même ainsi qu'aux classes dérivées
protected internal	Accès limité à la classe, aux classes dérivées ou aux membres de l'Assembly

Propriétés

- Méthodes qui contrôlent l'accès aux membres d'une classe

```
class Animal
{
    private int poidsAnimal;
    public int PoidsAnimal
    {
        get
        {
            return poidsAnimal;
        }
        set
        {
            poidsAnimal = value;
        }
    }
}
```

Propriétés en lecture seule

- Principalement utilisées pour les classes immuables

```
class Etudiant
{
    public string Prenom { get; }
    public string Nom { get; }
    public Etudiant(string prenom, string nom)
    {
        Prenom = prenom;
        Nom = nom;
    }
    public MajPrenom(string nouveauPrenom)
    {
        Prenom = nouveauPrenom; // Erreur
    }
}
```

Let mot-clé *var*

- Inférence de type: le compilateur détecte automatiquement le type de la variable
- La lecture du code est simplifiée
- Nb: la variable doit être déclarée et instanciée dans la même instruction
- Rq: le concept a été repris en C++ (mot-clé *auto*)

```
var i = 5; // le compilateur détecte un entier
var s = "Hello world"; // chaîne de caractères
var a = new[] { 1, 2, 3 }; // tableau d'entiers
```

Les délégués

- Permet d'invoquer une méthode qu'on ne connaît pas au moment de la compilation
  - Gestion d'évènements
  - Critères de recherche
- Très proches des pointeurs de fonctions

```
static int[] tabInt = new int[] { 1, 5, 3, 6, 10, 12, 13, 15 };
static bool infDix(int i)
{
    return i <= 10;
}
static void Main(string[] args)
{
    int[] tabInf = Array.FindAll(tabInt, infDix); // renvoie [1, 5, 3, 6, 10]
}
```

Méthodes anonymes, lambda expressions

- Pour rendre le code plus lisible
- Le compilateur s'occupe de (presque) tout

```
    static int[] tabInt = new int[] { 1, 5, 3, 6, 10, 12, 13, 15 };
    static void Main(string[] args)
    {
        int[] tab1 = Array.FindAll(tabInt, delegate (int i) { return i % 3 == 0; });
        // renvoie [3, 6, 12, 15]
        int[] tab2 = Array.FindAll(tabInt, i => (i % 5 == 0));
        // renvoie [5, 10, 15]
    }
```

Mémoire non gérée

- Exemple: lecture de fichiers
- Il faut garantir que la mémoire est désallouée à la fin de vie des objets
  - => Implémentation de l'interface *IDisposable*

```
void TreatTextFile()
{
    Char[] buffer = new Char[50];
    using (var s = new StreamReader("File1.txt"))
    {
        int charsRead = 0;
        while (s.Peek() != -1)
        {
            charsRead = s.Read(buffer, 0, buffer.Length);
            // Faire quelque chose //
        }
    }
}
```

Les données (ADO.NET)

Active X Data Objects

Sources de données

- Bases de données
  - Sql Server
  - Oracle
  - ODBC
  - …
- Excel
- Fichiers XML
- Fichiers texte…

Qu’est-ce que LINQ?

- Un modèle de programmation qui permet un accès unifié à n’importe quelle source de données.
- Implémentations disponibles:
  - LINQ to Objects
  - LINQ to ADO.NET
    - LINQ to Entities
    - LINQ to SQL
    - LINQ to DataSet
  - LINQ to XML

Un exemple

```
var req =
    from c in Clients
    where c.Pays == "Italie"
    select c.NomCompagnie;
```

La variable *req* contient le résultat de la requête, on peut parcourir les valeurs renvoyées:

```
foreach (string name in req)
{
    Console.WriteLine(name);
}
```

Un exemple (suite)

Où les données étaient-elles stockées?

```
Client[] Clients;

DataSet ds = retrieveDS();
DataTable Clients = ds.Tables["Clients"];

DataContext db = new DataContext(ConnectionString);
Table<Client> Clients = db.GetTable<Client>();

…
```

L’essentiel

- Facilité d’installation
- Compatibilité inter-langages
- La plateforme s’occupe de tout
  - Base class library
  - Ramasse-miette
  - Ne pas réinventer la roue
  - **Privilégier la propreté sur l’efficacité**