UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

# Introduction Framework .NET

# Sommaire

UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

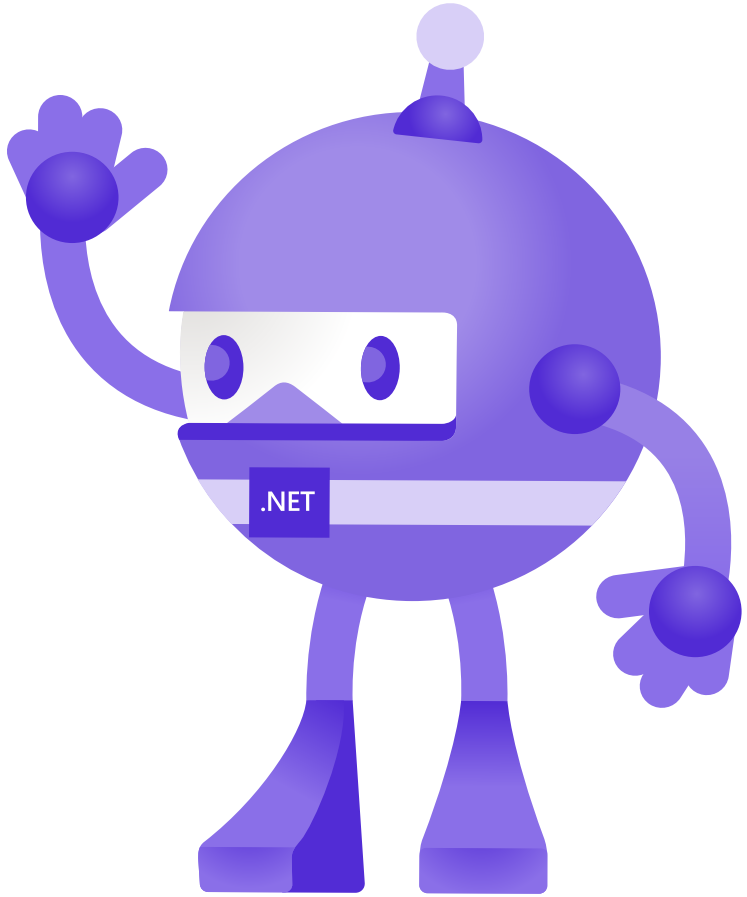
# Découverte du .NET

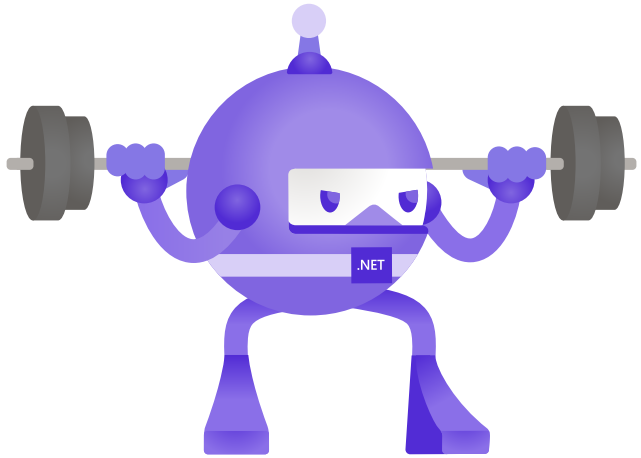
# Qu'est-ce que le .NET ?

- **.NET** est un framework de développement open-source et multiplateforme créé par Microsoft.
- Initialement, il y avait le **.NET Framework** conçu pour les applications Windows.
- Par la suite, **.NET Core** a été développé pour être multiplateforme (Windows, macOS, Linux).
- Avec **.NET 5**, Microsoft a unifié .NET Framework et .NET Core pour créer une seule plateforme : **.NET**.
- Les versions récentes incluent chacune des améliorations et de nouvelles fonctionnalités.

# .NET Bot, la mascotte du .NET






# .NET - Une plateforme de développement Unifiée

| Cloud | Web | Bureau | Mobile | Jeux Vidéos | IoT | IA |
|---|---|---|---|---|---|---|
| Azure | ASP.NET Blazor | .NET MAUI WPF WinForms | .NET MAUI Xamarin | Unity MonoGame Godot | ARM32 ARM64 | ML.NET .NET pour Spark |

## .NET 5+

Bibliothèques et APIs de Base Communes

Infrastructure

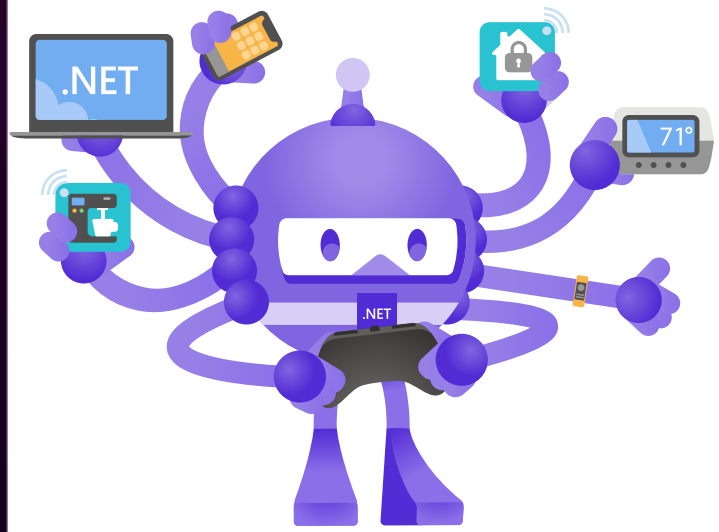
- Composant d'Exécution (Runtime)
- Compilateurs
- Langages

**Nuget**

Bibliothèques
Composants
Outils

## Outils / IDE

- Visual Studio 2022
- Visual Studio 2022 pour Mac
- Visual Studio Code
- CLI



# Objectifs du .NET

- **Standardisation des langages et protocoles** : Uniformiser les langages de programmation et les rendre interopérables.
- **Unification des technologies** : Permettre le développement d'applications Windows, Web, mobiles, et cloud avec une seule plateforme.
- **Open source et multiplateforme** : Assurer que .NET peut fonctionner sur différents systèmes d'exploitation et est accessible à tous.

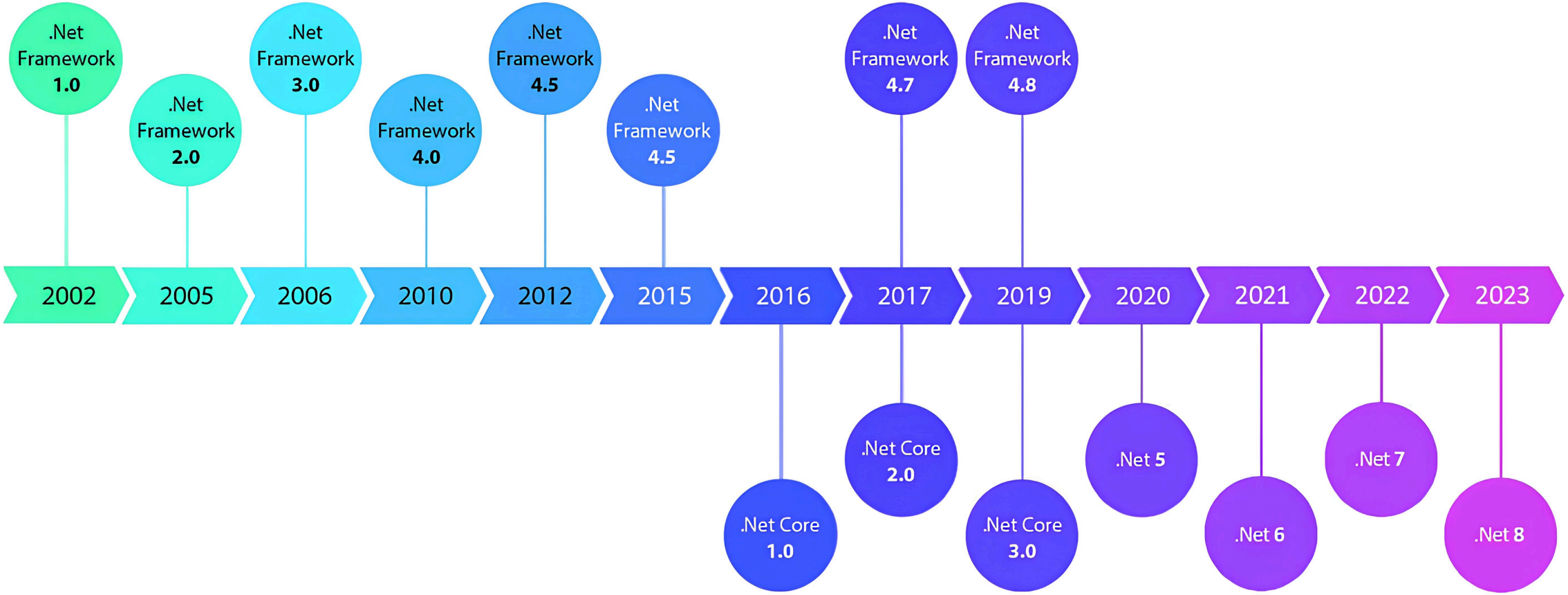
# Rapide Histoire de .NET

- En 2000, Microsoft a annoncé le langage de programmation C# et le framework .NET.
- **.NET Framework** a été conçu pour les applications Windows, offrant un environnement de développement et d'exécution pour les services web.
- En 2014, Microsoft a introduit **.NET Core**, une version open-source et multiplateforme de .NET Framework.
- En 2020, **.NET 5** a été lancé, unifiant .NET Framework et .NET Core sous une seule plateforme.

# Évolution de .NET

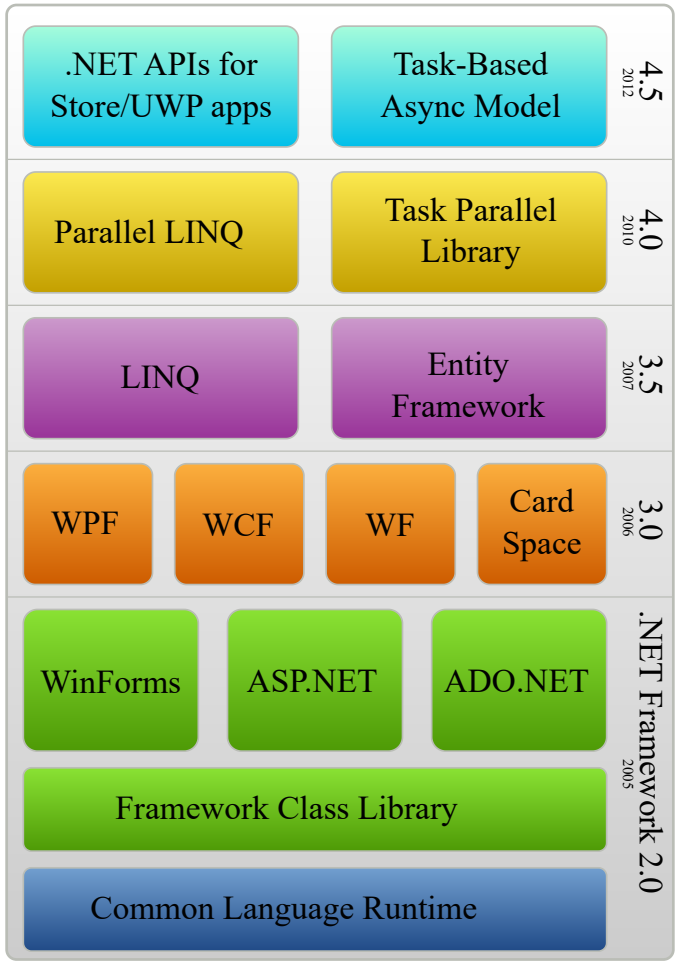
- **.NET Framework** : Initialement conçu pour les applications Windows.
- **.NET Core** : Introduit en 2016 pour supporter des applications multiplateforme.
- **.NET 5 et versions ultérieures** : Unification des deux plateformes précédentes, avec des améliorations continues et un support élargi.

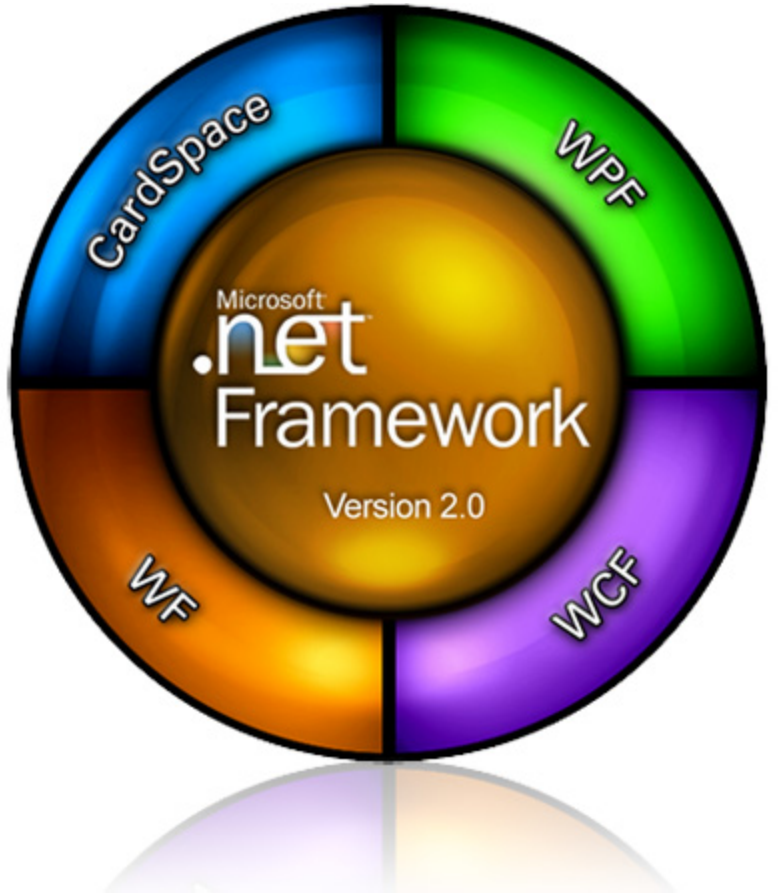
# Évolution de .NET



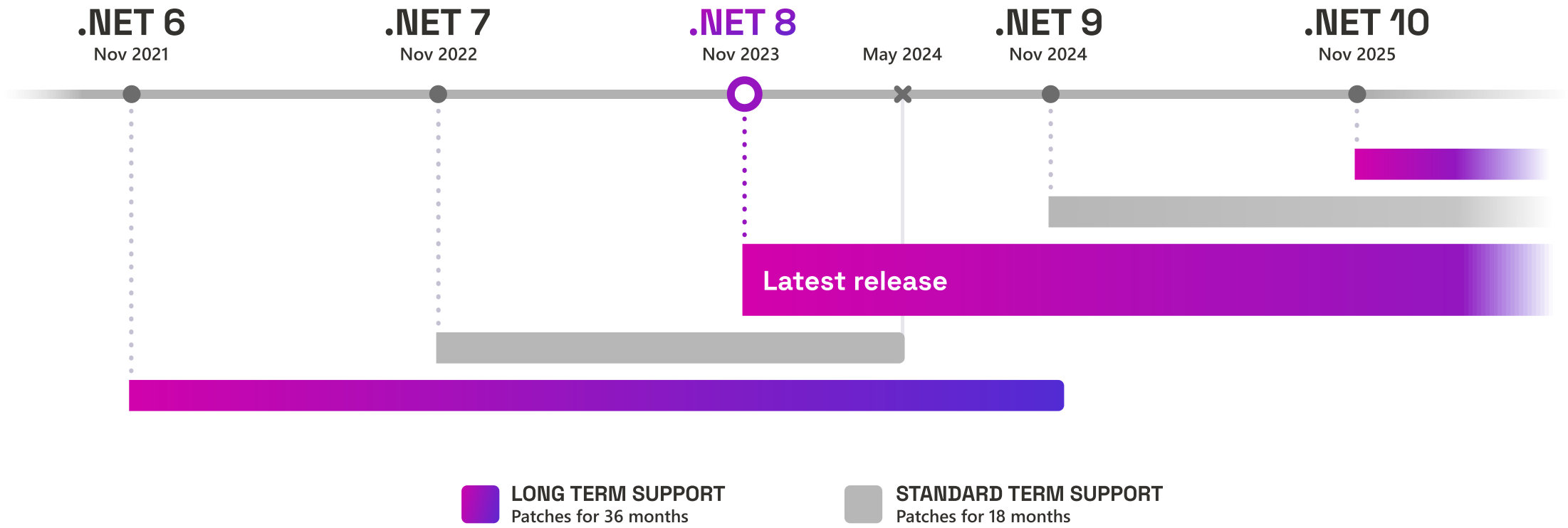

[Timeline complète]

# .NET Framework



# Versions LTS et STS




[Support Policy]

# Pourquoi choisir .NET ?

- **Performances élevées** : Optimisations constantes dans chaque nouvelle version.
- **Large écosystème** : Bibliothèques et outils pour presque tous les besoins de développement.
- **Interopérabilité** : Capacité à intégrer et à interagir avec d'autres technologies.
- **Sécurité** : Gestion avancée de la sécurité des applications, notamment avec la gestion des exceptions et le Garbage Collector.

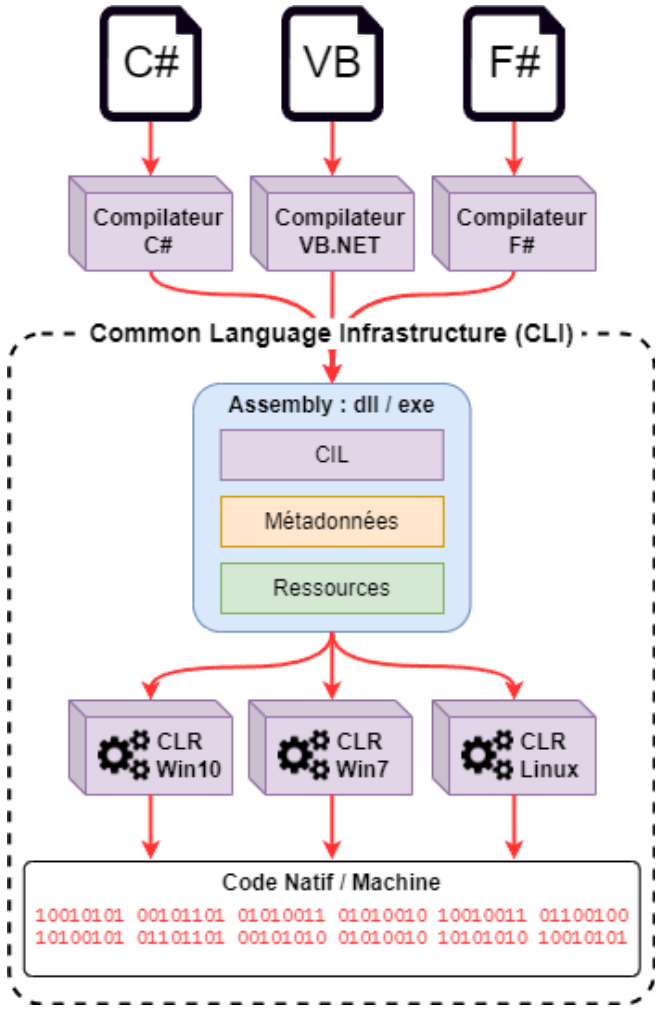
UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

# Common Language Infrastructure (CLI)

# Common Language Infrastructure

- Le CLI est un **cadre de spécifications standard** pour les **langages de programmation** et les **environnements d'exécution**.
- C’est une norme définie par l'**ECMA** (European Computer Manufacturers Association) et l'**ISO** (International Organization for Standardization).
- Il définit la **structure** et le **fonctionnement** des **langages de programmation**, des **compilateurs** et des **environnements d'exécution** qui sont compatibles entre eux.
- Les **spécifications .NET** sont des **implémentation du CLI**

# Aperçu du CLI



UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

# Common Language Specification (CLS)

# Common Language Specification (CLS)

- La **Common Language Specification (CLS)** est un **ensemble de règles** que **tout langage de programmation doit suivre** pour être **compatible avec .NET**.
- La CLS assure que les langages peuvent **interagir** de manière fluide, facilitant l'**interopérabilité**.
- Elle standardise des aspects comme les **types de données**, les **classes**, les **délégués**, et la gestion des **événements**.

UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

# Common Type System (CTS)

# Le Common Type System (CTS)

- Le **Common Type System (CTS)** définit les **types de données** que peuvent **utiliser tous les langages de .NET**.
- Le CTS assure que les **types** sont **uniformes** à travers différents langages, permettant une **interopérabilité complète**.
- Les **types de base** inclus dans le CTS sont des types comme `Boolean`, `Byte`, `Char`, etc.

UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

# Base Class Library (BCL)

# Base Class Library (BCL)

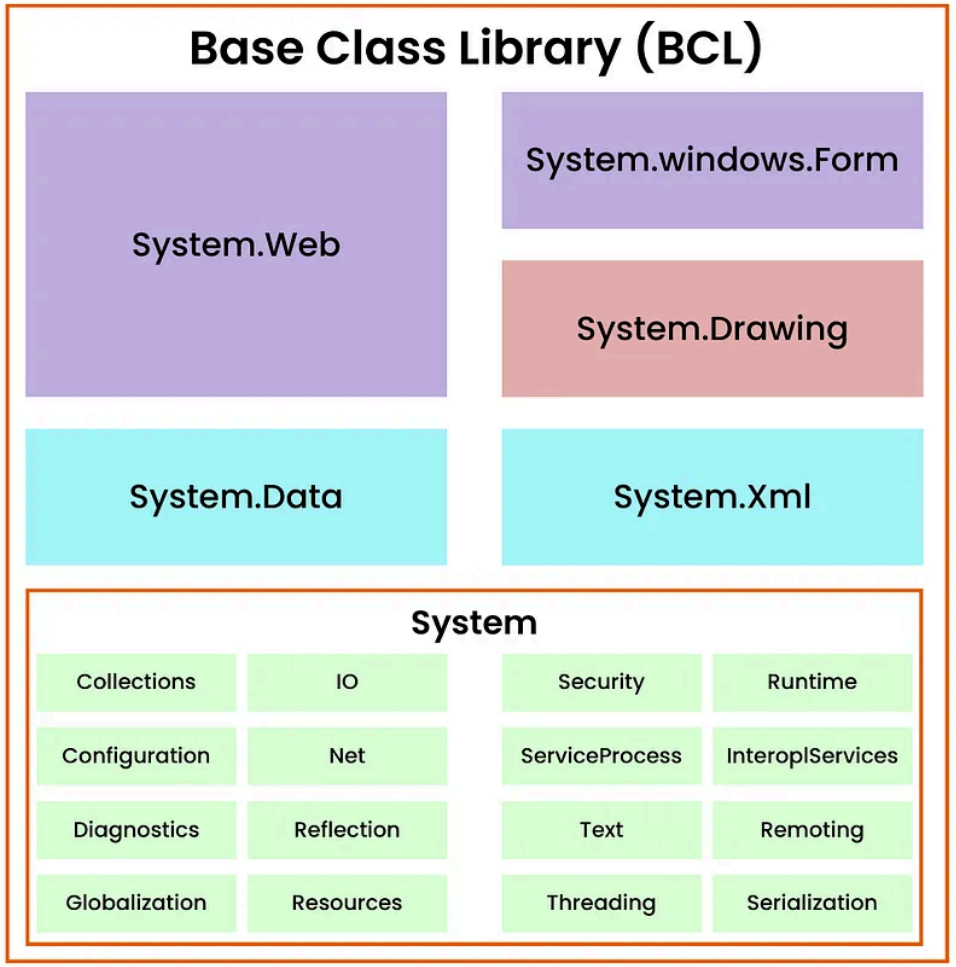
- La **Base Class Library (BCL)** est une **collection de types** réutilisables qui s'intègrent parfaitement avec le **Common Language Runtime** (CLR).
- La BCL est **orientée objet** et fournit des **types de base** pour les applications, comme les collections, la manipulation de fichiers, et bien plus encore.
- Elle assure une **programmation multi-langages** et **unifie les développements**.

# La BCL une API de programmation multi-langages

- La BCL supporte actuellement **27 langages**, les plus utilisés étant **C#**, **VB.NET**, et **C++.Net**.
- Chaque langage **conforme au CLS** peut **utiliser les classes et méthodes de la BCL**.

Liste complète des langages

# Couches de la BCL



# Les classes de la BCL

- Les classes de la BCL sont organisées en **espaces de noms** (**Name Spaces**) hiérarchisés.
- Exemple :
  - La classe `DataSet` dans `System.Data` se déclare/récupère de cette manière `System.Data.DataSet`.
  - La classe `Console` dans `System` se déclare/récupère comme `System.Console`.

UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

# Common Intermediate Language (CIL)

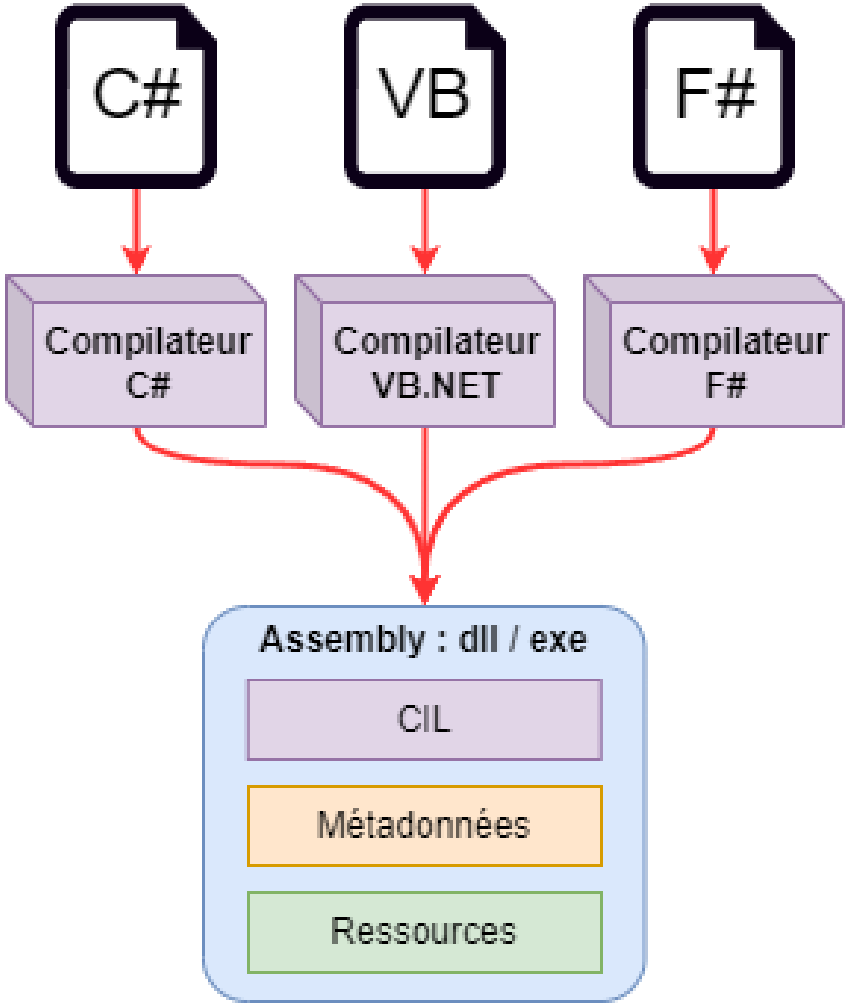
# Le Common Intermediate Language (CIL)

- Le **Common Intermediate Language (CIL)**, anciennement appelé MSIL, est le **code intermédiaire** vers lequel le **code source** .NET est **compilé**.
- Le CIL n'est **pas du code machine** et nécessite **une étape d'interprétation ou de compilation supplémentaire** pour être **exécuté** sur un **système d'exploitation**.
- Il est contenu dans un fichier **Assembly** (**.dll**/**.exe**)

# Le rôle du CIL

- La **compilation JIT (Just-In-Time)** convertit le CIL en code machine **au moment de l'exécution**.
- La **compilation AOT (Ahead-Of-Time)** peut également être utilisée pour **optimiser les performances**.

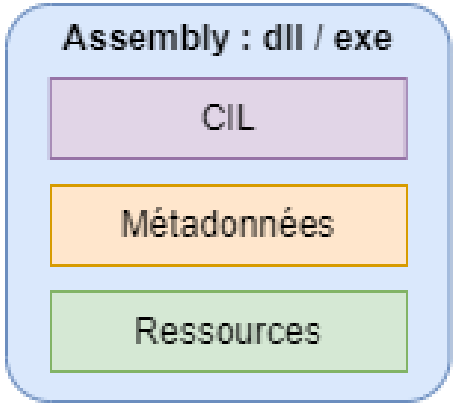
# Processus de compilation vers le CIL



UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

# Assemblage (Assembly)

# Assemblage (Assembly)




Un **Assembly** est le **fichier exécutable** (**.exe** ou **.dll**) **produit** par la **compilation** d'un code .NET.
Il contient :

- Le **code CIL**, il peut comporter **plusieurs classes** mais **un seul point d'entrée** (`Main`).
- Les **métadonnées**, qui incluent des **descriptions de types** et l'**Assembly manifest**.
- Les **ressources** (icônes, bitmaps, etc.)

# Assembly Manifest

| Information | Description | Détails |
|---|---|---|
| **Assembly name** | Nom de l'assembly. | Texte spécifiant le nom. |
| **Version number** | Numéro de version de l'assembly. | `X.X.X-xxx` : Majeur, mineur, révision, build. Utilisé pour appliquer la politique de version. |
| **Culture** | Culture ou langue supportée. | Désigne un assembly satellite avec des informations spécifiques à une culture ou langue. |
| **Strong name information** | Clé publique de l'éditeur. | Présente si l'assembly a un nom fort. |
| **List of all files** | Liste des fichiers dans l'assembly. | Hash et nom de chaque fichier. Tous les fichiers doivent être dans le même répertoire que le manifest. |
| **Type reference information** | Informations sur les références de types. | Utilisées pour mapper les références de type au fichier contenant leur déclaration et implémentation. |
| **Referenced assemblies** | Assemblies référencés statiquement. | Nom, métadonnées (version, culture, OS, etc.), clé publique si l'assembly a un nom fort. |

UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

# L’environnement CLR

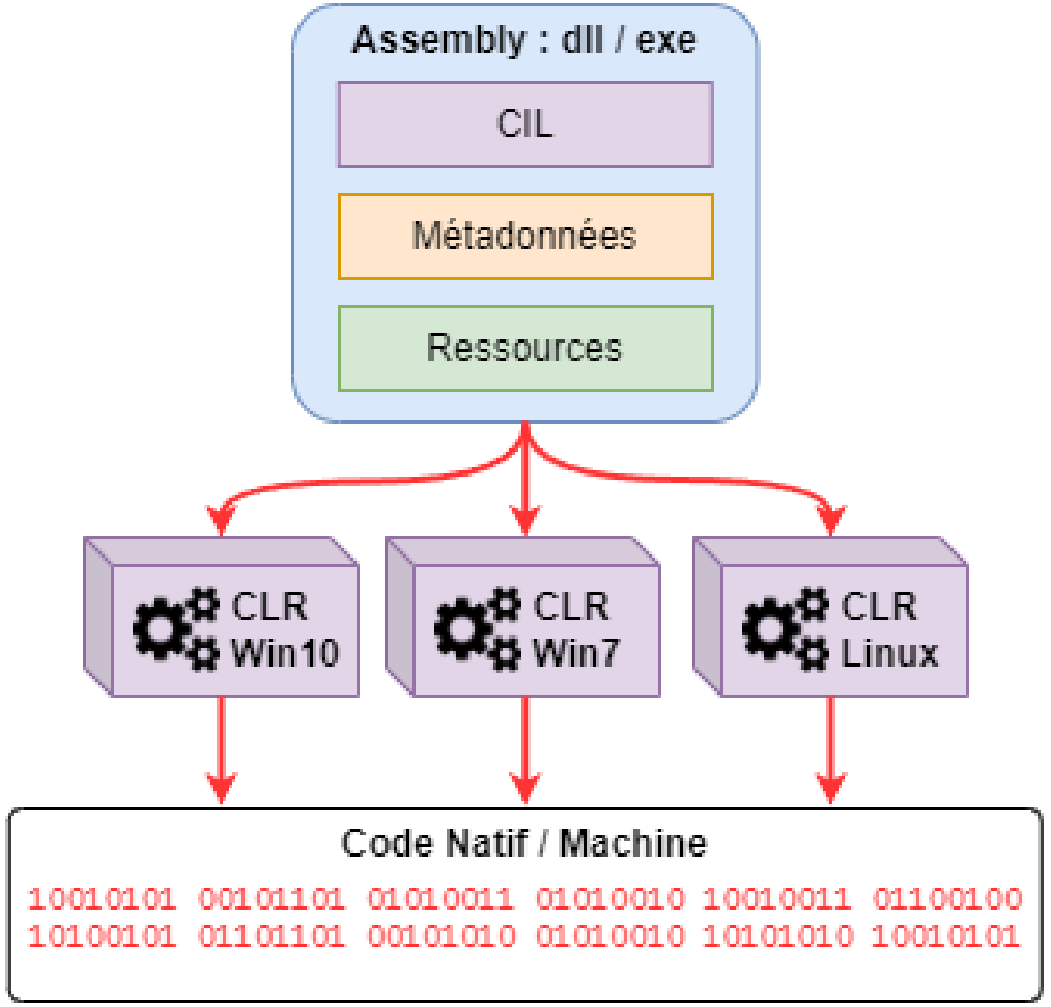
# Common Language Runtime (CLR)

- Le **Common Language Runtime (CLR)** est l'**environnement d'exécution** des programmes .NET.
- Il gère l'**exécution des programmes**, la **mémoire**, les **exceptions**, les **threads**, et bien plus encore.
- Le CLR **interprète les fichiers exécutables compilés en CIL** et **permet leur exécution** sur **différents systèmes d'exploitation**.

# Services fournis par le CLR

- **Gestion de la mémoire** : Utilisation du Garbage Collector.
- **Sécurité des types** : Vérification de la sécurité des types de données.
- **Gestion des exceptions** : Gestion des erreurs à l'exécution.
- **Interopérabilité** : Capacité d'utiliser des composants COM et des DLLs non managés.

# Diagramme du CLR



# Différences entre Code managé et Code natif

- **Code managé** : Programme exécuté sous la supervision du CLR.
- **Code non managé** : Programme exécuté directement par le système d'exploitation sans intervention du CLR.

UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

# Les Spécifications .NET

# Spécifications .NET

- **.NET Standard** : Spécification des API .NET communes à toutes les plateformes .NET.
- **.NET Framework** : Ancienne, qui a évolué vers .NET 5+.
- **.NET Core** : Ancienne, qui a évolué vers .NET 5+.
- **Mono** : Pour les applications mobiles et les jeux.
- **.NET 5+** : Unification de .NET Framework et .NET Core, avec des versions modernes.

UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

# .NET MAUI

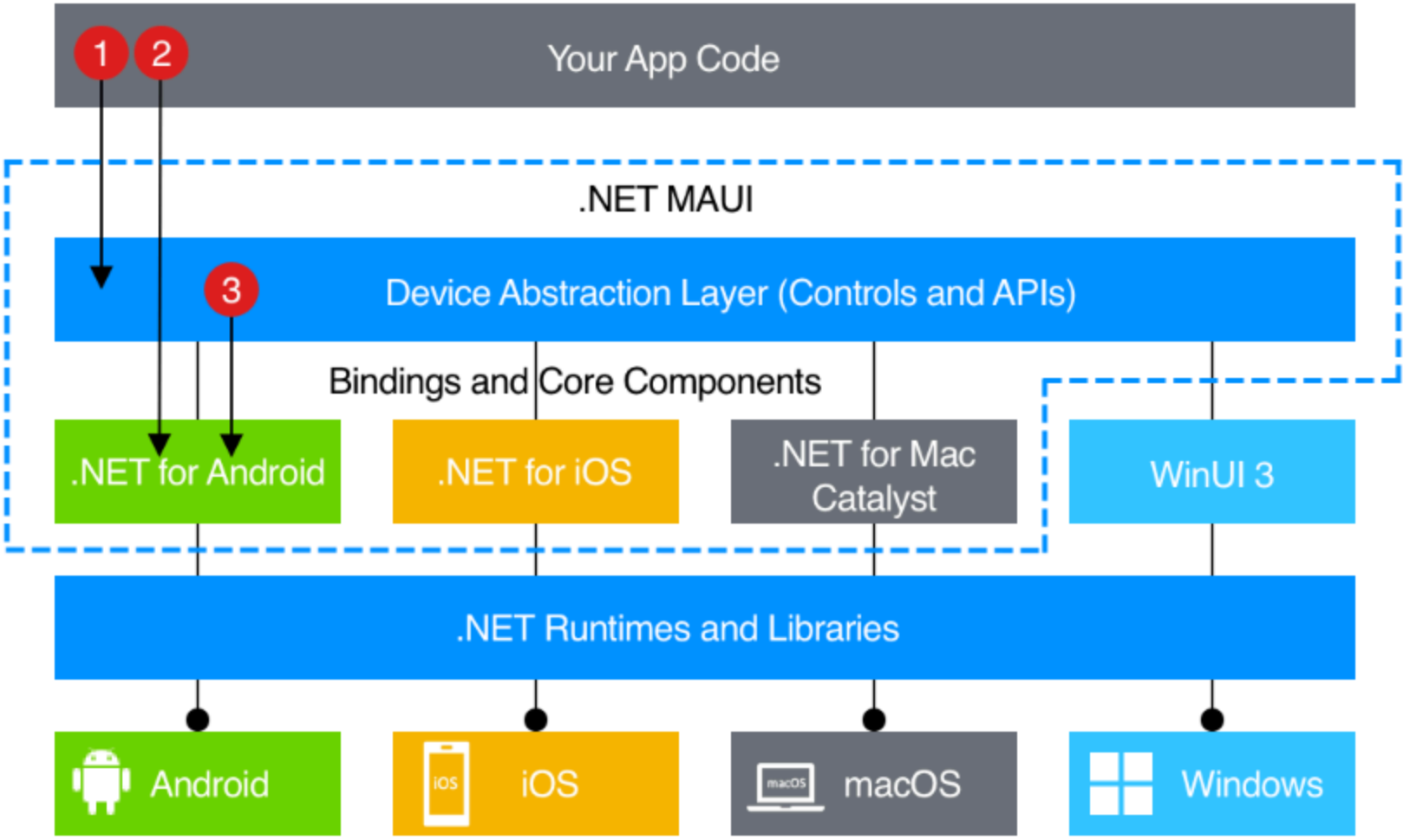
# Qu'est-ce que .NET MAUI ?

- **.NET MAUI** (**Multi-platform App UI**) est une **plateforme** pour **créer des applications natives multiplateformes** avec **une seule base de code**.
- .NET MAUI permet de développer des applications pour **Android**, **iOS**, **macOS** et **Windows** en utilisant **C# et XAML** ou **Blazor Hybrid**.

# Avantages de .NET MAUI

- **Code partagé** : Réduire le temps de développement et de maintenance en partageant une base de code unique.
- **Performance native** : Accéder aux API natives de chaque plateforme pour des performances optimales.
- **Productivité accrue** : Utiliser des outils et bibliothèques modernes comme .NET 6+, Visual Studio et les bibliothèques de composants.

# Architecture de .NET MAUI



UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

# Comparaison avec Java

# .NET, une Réponse à Java

| Caractéristique | .NET/C# | Java |
|---|---|---|
| Langages | C#, VB.NET, C++,... | Java, Kotlin, Groovy, ... |
| Environnement d'exécution | Common Language Runtime (CLR) | Java Virtual Machine (JVM) |
| Language intermédiaire | Common Intermediate Language | Bytecode |
| Spécification du langage et Système de types | Common Language Specification Common Type System | Non applicable |
| Plateforme de développement | .NET Framework, .NET Core, .NET 5+ | Java Development Kit (JDK) |
| Gestion de la mémoire | Garbage Collection (GC) | Garbage Collection (GC) |
| Outils de développement intégrés | Visual Studio, Visual Studio Code, Rider | IntelliJ IDEA, Eclipse, NetBeans |
| Framework web | ASP.NET | Spring |
| Date de première version | 2000 (C# 1.0) | 1995 (JDK 1.0) |
| Concepteur | Microsoft | Sun Microsystems => Oracle |
| Versions actuelles (2024) | .NET 8 / C# 12 | JDK 21 / Java 20 |

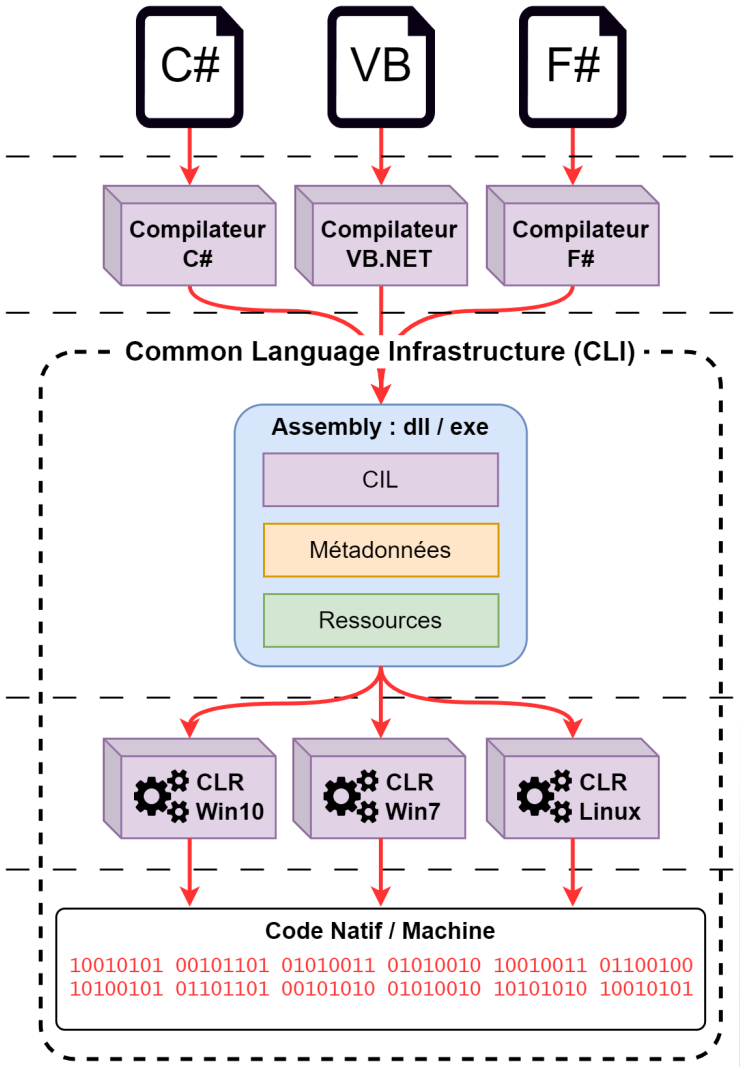
UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

# Conclusion

# Conclusion

- **.NET** est une plateforme de développement moderne, performante et multiplateforme, adaptée à une variété d'applications.
- Les versions récentes, telles que **.NET 8** et **.NET 9**, continuent d'améliorer les capacités de la plateforme, en offrant des performances accrues et des fonctionnalités avancées.
- En combinant des éléments de programmation orientée objet, une gestion efficace de la mémoire et une interopérabilité entre les langages, .NET reste un choix de premier ordre pour les développeurs.

## Fonctionnement




## Explications

- **Respectent** la **CLS** (Common Language Specification)
- **Respectent** le **CTS** (Common Type System)
- **Utilisent** les Types de la **BCL** (Base Class Library)
- **Nécessité** d'un **fichier projet** (ex: **.csproj**)

- Application qui **transforme le code source en CLI**, plus proche du langage natif
- Processus de **Compilation** et **Assemblage**
- **1** par **couple langage-version** de .NET (ex: C#11-.NET6)

**CLI** :
- **Cadre de spécifications standard** pour les langages
- Définie par l'**ECMA** et l'**ISO**

**Assembly** (**fichier**, **portable**) contenant :
- **CIL** (Common Intermediate Language)
  **Code source compilé**
  Un seul point d'entrée (`Main`)
- **Métadonnées**
  Descriptions des Types
  **Assembly Manifest** (Nom, Version, Culture, ...)
- **Ressources**
  images, audio, json, xml, html, css, ...

**CLR** (Common Language Runtime)
- **Environnement d'exécution**
- **1** par **couple Système-version** de .NET (ex: Win10-.NET6)
- Fournit des **services** de **sécurité** et de **contrôle** de l'exécution : **Garbage Collector**, interopérabilité, erreurs, threads, ...

**Code Natif:**
- Transformé **au moment** ou **juste avant** de l'exécution par le compilateur **JIT** (Just-In-Time Compiler) ou le compilateur **AOT** (Ahead-Of-Time Compiler)

# Liens

- Documentation dot.net
- Glossaire dotnet

UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

# Merci pour votre attention

## Des questions ?