L’architecture .NET

1. Présentation générale

.NET est un framework de développement créé par Microsoft.

Il permet de créer des applications console, desktop (Windows Forms, WPF), web (ASP.NET Core), mobiles (Xamarin, MAUI) et même cloud.

Il est composé de plusieurs éléments :

a) Le CLR (Common Language Runtime)

C’est la machine virtuelle de .NET.

Rôle : exécuter le code, gérer la mémoire, le ramasse-miettes (*Garbage Collector*), la sécurité, les exceptions.

b) Le CTS (Common Type System)

Définit comment les types de données (int, string, classes, etc.) sont représentés et utilisés.

Garantit la compatibilité entre différents langages (C#, VB.NET, F#...).

c) Le CLS (Common Language Specification)

Définit un ensemble minimal de règles que tous les langages .NET doivent respecter pour être interopérables.

d) La BCL / FCL (Base Class Library / Framework Class Library)

Ensemble de bibliothèques standards (collections, IO, réseau, XML, LINQ, etc.).

e) Outils associés

Visual Studio (IDE complet).

NuGet (gestionnaire de packages).

MSBuild (système de compilation).

Bonnes pratiques de développement en C#

1. **Principes généraux (SOLID)**

S : Single Responsibility (chaque classe a une seule responsabilité).

O : Open/Closed (ouvert à l'extension, fermé à la modification).

L : Liskov Substitution (une classe dérivée doit pouvoir remplacer sa base).

I : Interface Segregation (ne pas forcer une classe à implémenter des méthodes inutiles).

D : Dependency Inversion (dépendre d'abstractions, pas de classes concrètes).

2. **Gestion des exceptions**

Toujours utiliser try/catch/finally.

Ne pas attraper une exception si on ne peut pas la traiter.

Éviter les exceptions silencieuses.

3. **Lisibilité et maintenabilité**

Code clair, commenté uniquement si nécessaire.

Méthodes courtes (une seule responsabilité).

Utiliser le typage fort (int, string) plutôt que var sauf si évident.

4. **Tests unitaires**

Utiliser des frameworks comme MSTest, xUnit, ou NUnit.

Tester les cas normaux, limites et erreurs.

Conventions de nommage en C# (Microsoft)

Namespaces, Classes, Interfaces, Méthodes, Propriétés : PascalCase

```
namespace MonApplication.Formes;
```

```
class Cercle { }
interface IForme { }
public double CalculerAire() { }
```

Champs privés, variables locales, paramètres : camelCase

```
private int rayon;
double calculInterne;
void DessinerForme(int tailleMax) { }
```

Constantes et readonly : PascalCase

```
public const double Pi = 3.14;
```

Abréviations :

Éviter les sigles en majuscules : XmlReader (et pas XMLReader).

Préfixes :

Interfaces commencent par I (IComparable).

Champs privés souvent préfixés par _ (_rayon).

Résumé utilisable dans ton README.md :

Présentation claire de l'architecture .NET (CLR, CTS, CLS, BCL).

Bonnes pratiques = SOLID, exceptions, lisibilité, tests unitaires.

Conventions de nommage = PascalCase / camelCase avec règles Microsoft.