ÉVOLUTION DE L'ÉCOSYSTÈME .NET

DE .NET FRAMEWORK 4.8 À .NET 8

C# • ASP.NET API • TRAJECTOIRE LTS

OBJECTIFS

- Comprendre l’évolution de .NET
- Identifier les impacts développeur
- Se projeter dans une trajectoire maîtrisée

.NET : ÉVOLUTION DE LA PLATEFORME

.NET FRAMEWORK 4.8 AUJOURD'HUI

- Stable
- Supporté
- Fiable
- **Figé**

SUPPORT ≠ ÉVOLUTION

- Correctifs : oui
- Sécurité : oui
- Nouvelles fonctionnalités : non

ANALOGIE TERRAIN (BANQUE)

- Plateformes anciennes mais supportées
- Environnements critiques
- Fonctionnel volontairement maîtrisé

POINT CLÉ

LA PLATEFORME TECHNIQUE BORNE LE FONCTIONNEL

FAQ – PLATEFORME

- Peut-on rester en .NET 4.8 ?
- Quel est le risque réel ?
- Quand ça devient bloquant ?

C# : ÉVOLUTION DU LANGAGE

POURQUOI LE LANGAGE A ÉVOLUÉ

- Lisibilité
- Sécurité
- Expressivité

AVANT – C# 7.3

```
if (status == 200)
{
    return "OK";
}
else if (status == 404)
{
    return "Not Found";
}
else
{
    return "Error";
}
```

APRÈS – C# MODERNE

```
return status switch
{
    200 => "OK",
    404 => "Not Found",
    _   => "Error"
};
```

AVANT – DTO CLASSIQUE

```
public class UserDto
{
    public int Id { get; set; }
    public string Name { get; set; }

    public UserDto(int id, string name)
    {
        Id = id;
        Name = name;
    }
}
```

APRÈS – RECORD

```
public record UserDto(int Id, string Name);
```

AVANT – NULL IMPLICITE

```
string name = GetName();
Console.WriteLine(name.Length);
```

APRÈS – NULLABLE EXPLICITE

```
string? name = GetName();

if (name != null)
{
    Console.WriteLine(name.Length);
}
```

FAQ – C#

- Est-ce obligatoire ?
- Progressif ?
- Compatible ?

ASP.NET WEB API : ÉVOLUTION

ASP.NET WEB API (.NET 4.8)

- System.Web
- Global.asax
- Web.config
- Couplage IIS

ASP.NET WEB API : COMPARAISON

Web API 2 (.NET 4.8)	ASP.NET Core Web API
System.Web	Pipeline middleware
Global.asax	Program.cs
Web.config	appsettings.json
IIS obligatoire	Auto-hébergé possible
DI externe	DI native

AVANT – WEB API 2

```
[ApiController]
[Route("api/users")]
public class UsersController : ApiController
{
    [HttpGet("{id}")]
    public IHttpActionResult Get(int id)
    {
        var user = _service.Get(id);
        return Ok(user);
    }
}
```

APRÈS – ASP.NET CORE

```
 1 [ApiController]
 2 [Route("api/users")]
 3 public class UsersController : ControllerBase
 4 {
 5     [HttpGet("{id}")]
 6     public IActionResult Get(int id)
 7     {
 8         var user = _service.Get(id);
 9         return Ok(user);
10     }
11 }
```

APRÈS – ASP.NET CORE

```
 1 [ApiController]
 2 [Route("api/users")]
 3 public class UsersController : ControllerBase
 4 {
 5     [HttpGet("{id}")]
 6     public IActionResult Get(int id)
 7     {
 8         var user = _service.Get(id);
 9         return Ok(user);
10     }
11 }
```

APRÈS – ASP.NET CORE

```
 1 [ApiController]
 2 [Route("api/users")]
 3 public class UsersController : ControllerBase
 4 {
 5     [HttpGet("{id}")]
 6     public IActionResult Get(int id)
 7     {
 8         var user = _service.Get(id);
 9         return Ok(user);
10     }
11 }
```

CE QUI NE CHANGE PAS

- Controllers
- Routing
- Métier
- JSON

FAQ – WEB API

- Réécriture complète ?
- Contrôleurs différents ?
- Performances ?

CADENCE .NET & TRAJECTOIRE LTS

COMMENT ÉVOLUE .NET

- 1 version / an
- LTS tous les 2 ans
- STS rarement en production

SITUATION AU 24/01/2026

- .NET 8 : LTS (fin 11/2026)
- .NET 9 : STS
- .NET 10 : LTS (depuis 11/2025)

AVANT – VRAI CHANGEMENT

- .NET Framework → .NET moderne
- Changement de plateforme

APRÈS – MONTÉE LTS

- .NET 8 → .NET 10
- Même plateforme

FAQ – LTS

- Pourquoi pas attendre .NET 10 ?
- .NET 8 trop court ?
- Nouvelle migration ?

CONCLUSION

- .NET 8 est un point d’entrée
- Le vrai saut est déjà fait
- La trajectoire est maîtrisée

QUESTIONS