Guide de déploiement — Application .NET (module Forms Stimulsoft) sur Windows Server + IIS

*Précision importante : ce guide couvre le déploiement de **votre application .NET** qui intègre le module Forms de Stimulsoft (bibliothèque/NuGet, ex. Stimulsoft.Forms.Web ou équivalent) — pas l'installation du produit packagé « Stimulsoft Server ». Il s'agit donc d'un déploiement .NET classique, avec quelques points d'attention spécifiques à Stimulsoft (licence, taille des requêtes) détaillés à l'étape 6.*

*Hypothèse retenue : application **ASP.NET Core** (.NET 6/7/8/9), le cas le plus courant pour un projet récent (les exemples officiels Stimulsoft pour Forms sont fournis en ASP.NET Core). Si votre application est en réalité en **.NET Framework classique** (4.x), prévenez-moi — la démarche est plus simple encore (pas de module à installer, IIS gère nativement le pipeline .NET Framework).*

Comment vérifier rapidement lequel s'applique à vous : ouvrez le fichier `.csproj` du projet. S'il contient une ligne `<TargetFramework>net8.0</TargetFramework>` (ou net6.0, net7.0, net9.0…), c'est de l'ASP.NET Core → ce guide s'applique. S'il contient plutôt `<TargetFrameworkVersion>v4.8</TargetFrameworkVersion>` (ancien format de fichier projet, souvent plus long et verbeux), c'est du .NET Framework classique → dites-le-moi, je vous ferai la version adaptée.

Bonne nouvelle par rapport à Node.js ou Angular : IIS et .NET sont conçus par le même éditeur (Microsoft) pour fonctionner ensemble nativement. Il n'y a **pas besoin de NSSM, pas de reverse proxy à configurer manuellement, pas de port applicatif à gérer** — IIS héberge directement le process .NET via un module dédié (ASP.NET Core Module), et redémarre l'application automatiquement si elle plante. C'est le déploiement le plus simple des trois guides que je vous ai faits.

> Le sommaire ci-dessous est généré automatiquement par DokuWiki à partir des titres de section.

---

1. Objectif et architecture

Ce document décrit comment déployer une application ASP.NET Core sur un Windows Server existant, hébergée par IIS.

Le modèle d'hébergement par défaut est dit **« in-process »** : l'application .NET Core tourne directement à l'intérieur du process de travail IIS (`w3wp.exe`), via un module natif appelé **ASP.NET Core Module (ANCM)**. Contrairement à Node.js, il n'y a donc pas de process Node séparé à démarrer/surveiller (pas de NSSM), ni de reverse proxy à configurer à la main (pas de règle URL Rewrite vers un port local) — IIS gère tout nativement, y compris le démarrage automatique au boot et le redémarrage en cas de plantage.

Schéma : Internet → IIS (port 443, certificat SSL) → module ASP.NET Core (ANCM) → application .NET hébergée directement dans le worker process IIS.

2. Prérequis

- Accès administrateur au Windows Server (2016, 2019, 2022 ou équivalent).
- Le rôle « Serveur Web (IIS) » disponible via Gestionnaire de serveur.
- Le SDK .NET installé sur le poste/pipeline de build (pas nécessairement sur le serveur) pour publier l'application.
- Le code de l'application prêt, avec un `.csproj` valide.
- Un nom de domaine ou sous-domaine interne pointant vers l'IP du serveur, si un accès via nom d'hôte est souhaité.
- Un certificat SSL (interne ou public) si l'application doit être servie en HTTPS — fortement recommandé dès lors que le questionnaire collecte des données.

3. Étape 1 — Installer le rôle IIS

Si IIS n'est pas encore installé sur le serveur :

```
# PowerShell, en tant qu'administrateur
Install-WindowsFeature -Name Web-Server -IncludeManagementTools
```

Ou via l'interface graphique : Gestionnaire de serveur → Ajouter des rôles et fonctionnalités → Serveur Web (IIS) → cocher au minimum : Contenu statique, Document par défaut, Journalisation HTTP, Filtrage des requêtes, Console de gestion IIS.

4. Étape 2 — Installer le .NET Hosting Bundle

Étape **indispensable et spécifique à ASP.NET Core** : sans elle, IIS ne sait pas exécuter une application .NET Core (erreur 500.19 ou 502.5 sinon).

- Télécharger le « ASP.NET Core Runtime – Windows Hosting Bundle » correspondant à la version cible du projet (ex. .NET 8) depuis dotnet.microsoft.com/download/dotnet — bien prendre le **Hosting Bundle**, pas seulement le runtime standard : lui seul installe le module IIS ANCM.
- Lancer l'installeur (.exe) sur le serveur.
- Redémarrer IIS pour que le nouveau module soit pris en compte :

```
net stop was /y
net start w3svc
```

- Vérifier que le module est bien enregistré : dans IIS Manager, sélectionner le nœud du serveur → « Modules » → chercher `AspNetCoreModuleV2` dans la liste.

5. Étape 3 — Publier l'application

Sur le poste de build (ou directement sur le serveur si vous choisissez de builder sur place) :

```
dotnet publish -c Release -o ./publish
```

Le dossier `publish/` généré contient les DLL de l'application, ses dépendances, et surtout un fichier **`web.config` déjà généré automatiquement** par la commande publish — il contient la configuration nécessaire pour qu'IIS sache démarrer l'application via ANCM (chemin vers l'exécutable/DLL, modèle d'hébergement in-process, etc.). Il n'y a normalement rien à modifier dedans pour un déploiement standard.

6. Étape 4 — Copier les fichiers vers le serveur

Copier l'intégralité du dossier `publish/` vers un dossier dédié sur le serveur, par exemple `D:\Sites\questionnaire\`.

```
# Exemple avec robocopy, en local sur le serveur ou via un partage réseau
robocopy publish D:\Sites\questionnaire /MIR
```

7. Étape 5 — Créer le site dans IIS Manager

1. Ouvrir IIS Manager → clic droit sur « Sites » → Ajouter un site web.
2. Nom du site : ex. QuestionnaireApp.
3. Chemin d'accès physique : D:\Sites\questionnaire\ (le dossier contenant le web.config généré par dotnet publish).
4. Liaison (Binding) : type http, port 80 (et https / port 443 une fois le certificat installé, voir étape suivante), nom d'hôte = le domaine/sous-domaine prévu.
5. Pool d'applications : créer ou assigner un pool dédié, en mode **« No Managed Code »** — contrairement à une application .NET Framework classique, ASP.NET Core ne passe pas par le pipeline géré historique d'IIS, c'est le module ANCM qui gère tout.
6. Démarrer le site.

> **Identité du pool d'application :** vérifier que le compte utilisé par le pool (par défaut `ApplicationPoolIdentity`) a les droits de lecture sur le dossier `D:\Sites\questionnaire\`, et d'écriture sur le sous-dossier de logs si la journalisation stdout est activée (étape 10).

8. Étape 6 — Points spécifiques au module Forms Stimulsoft

6.1 Déployer la clé de licence

Stimulsoft charge sa licence soit par code (clé en dur dans le `Program.cs`/contrôleur), soit par fichier (`license.key`). Si votre code utilise la seconde méthode — typiquement :

```
var path = Path.Combine(hostEnvironment.ContentRootPath,
"Content\\license.key");
Stimulsoft.Base.StiLicense.LoadFromFile(path);
```

— assurez-vous que ce fichier `license.key` est bien copié dans le dossier `publish/` lors du

`dotnet publish` (propriété « Copy to Output Directory » sur le fichier dans le projet), puis vérifiez sa présence une fois copié sur le serveur, au même chemin relatif attendu par le code (ex. `D:\Sites\questionnaire\Content\license.key`). Un oubli de ce fichier au déploiement est une cause fréquente d'erreur de licence en production alors que tout fonctionne en local.

6.2 Taille maximale des requêtes

Les formulaires Stimulsoft peuvent transporter des données volumineuses (pièces jointes, images intégrées, export de rapports). Si vous rencontrez une erreur HTTP 404.13 ou 413 lors de la soumission d'un formulaire un peu chargé, augmentez la limite dans le web.config :

```
<system.webServer>
  <security>
    <requestFiltering>
      <requestLimits maxAllowedContentLength="52428800" />
      <!-- 52428800 = 50 Mo, à ajuster selon vos besoins -->
    </requestFiltering>
  </security>
</system.webServer>
```

Si l'application est en ASP.NET Core, vérifier également, côté code, la configuration Kestrel/formOptions correspondante (`MultipartBodyLengthLimit`, `MaxRequestBodySize`) si elle est explicitement définie dans `Program.cs`.

9. Étape 7 — Configurer HTTPS

Fortement recommandé dès que le questionnaire transmet des données, même peu sensibles.

- Obtenir un certificat SSL : certificat interne (PKI de l'entreprise) si le site n'est accessible qu'en interne, ou certificat public (ex. Let's Encrypt, via l'outil win-acme, ou un certificat acheté) si le site est exposé sur Internet.
- Dans IIS Manager, sélectionner le site → « Certificats de serveur » → importer le certificat.
- Ajouter une liaison https sur le port 443 pour le site, en sélectionnant ce certificat.
- Forcer la redirection HTTP → HTTPS : le plus simple ici est d'activer `app.UseHttpsRedirection()` directement dans le code (`Program.cs`), déjà présent par défaut dans la plupart des templates ASP.NET Core. Alternative sans toucher au code : utiliser le module URL Rewrite d'IIS avec une règle de redirection identique à celle des guides précédents.

10. Étape 8 — Pare-feu Windows

Autoriser uniquement les ports nécessaires en entrée : 80 et 443. Il n'y a ici aucun port applicatif interne à ouvrir (le module ANCM communique avec l'application via un canal interne, pas un port TCP exposé).

```
New-NetFirewallRule -DisplayName "IIS HTTP" -Direction Inbound -Protocol TCP
```

```
-LocalPort 80 -Action Allow
New-NetFirewallRule -DisplayName "IIS HTTPS" -Direction Inbound -Protocol
TCP -LocalPort 443 -Action Allow
```

11. Étape 9 — Tester le déploiement

1. Depuis un poste client : accéder à `http://nom-du-site` puis `https://nom-du-site` dans un navigateur : l'application doit répondre.
2. Vérifier que le questionnaire fonctionne de bout en bout (soumission, enregistrement des réponses).
3. Provoquer volontairement un arrêt du pool d'application (Arrêter puis Démarrer dans IIS Manager) et vérifier que le site répond de nouveau normalement après redémarrage — IIS/ANCM relance automatiquement l'application à la requête suivante.
4. Redémarrer le serveur et vérifier que le site est de nouveau accessible sans intervention manuelle (IIS démarre automatiquement au boot).

12. Étape 10 — Activer les logs (diagnostic)

Par défaut, les logs détaillés de l'application (stdout) ne sont pas activés — utile de les activer temporairement en cas de problème. Dans le `web.config` généré par `dotnet publish`, repérer la balise `aspNetCore` et ajuster :

```
<aspNetCore processPath="dotnet"
            arguments=".\QuestionnaireApp.dll"
            stdoutLogEnabled="true"
            stdoutLogFile=".\logs\stdout"
            hostingModel="InProcess" />
```

Créer le sous-dossier `logs\` à côté du web.config (IIS ne le crée pas automatiquement) et s'assurer que le pool d'application a le droit d'y écrire. Une fois le diagnostic terminé, repasser `stdoutLogEnabled` à `false` pour éviter une accumulation de fichiers.

Les logs IIS classiques restent aussi disponibles dans `%SystemDrive%\inetpub\logs\LogFiles\`, et les erreurs de démarrage du module ANCM apparaissent dans l'Observateur d'événements Windows (Journaux Windows → Application).

13. Étape 11 — Sécuriser l'application

- Activer HSTS (`app.UseHsts()`, déjà présent par défaut dans la plupart des templates hors environnement de développement).
- Valider et assainir les entrées du questionnaire côté serveur (pas seulement côté client).
- Ne jamais stocker de secrets (chaînes de connexion, clés API) en clair dans `appsettings.json` déployé — utiliser des variables d'environnement, le gestionnaire de secrets IIS, ou un coffre-fort (ex. Azure Key Vault si applicable).
- Restreindre les droits NTFS du dossier `D:\Sites\questionnaire\` pour que seuls les comptes de déploiement/administration puissent y écrire.

- Mettre en place une limite de débit si nécessaire (middleware `Microsoft.AspNetCore.RateLimiting`, inclus nativement depuis .NET 7).

14. Étape 12 — Procédure de mise à jour de l'application

1. Sur le poste de build : récupérer la nouvelle version du code, puis relancer `dotnet publish -c Release -o ./publish`.
2. Avant de copier les nouveaux fichiers, déposer un fichier `app_offline.htm` à la racine de `D:\Sites\questionnaire\` : IIS/ANCM détecte sa présence et arrête proprement l'application, libérant les DLL verrouillées (sans ça, la copie peut échouer si les fichiers sont en cours d'utilisation).
3. Copier les nouveaux fichiers (`robocopy publish D:\Sites\questionnaire /MIR`, en excluant `app_offline.htm` s'il est encore présent côté serveur).
4. Supprimer `app_offline.htm` : l'application redémarre automatiquement à la requête suivante.
5. Vérifier rapidement (étape 9) que le site répond correctement.

> **Bonnes pratiques :** pour limiter les interruptions, prévoir les mises à jour en dehors des heures d'utilisation du questionnaire, et garder une copie de la version précédente du dossier `publish/` pour pouvoir revenir en arrière rapidement en cas de problème.

15. Checklist finale

Élément	Statut
Rôle IIS installé	☐
.NET Hosting Bundle installé, module AspNetCoreModuleV2 vérifié	☐
Application publiée (dotnet publish -c Release)	☐
Fichiers copiés sur le serveur (dossier publish/, avec web.config)	☐
Site créé dans IIS, pool en « No Managed Code »	☐
Fichier license.key présent sur le serveur, au bon chemin	☐
Limite de taille des requêtes (maxAllowedContentLength) ajustée si besoin	☐
Binding http configuré	☐
Certificat SSL installé, binding https configuré	☐
Redirection HTTP → HTTPS active	☐
Pare-feu : ports 80/443 ouverts	☐
Test de bout en bout du questionnaire réussi	☐
Redémarrage serveur testé (reprise automatique)	☐
Droits NTFS restreints sur le dossier du site	☐
Secrets non stockés en clair (appsettings)	☐

16. Annexe — Dépannage rapide

Symptôme	Piste de résolution
Erreur HTTP 500.19	web.config invalide, ou module ANCM absent — vérifier que le Hosting Bundle est bien installé (pas seulement le runtime standard).

Symptôme	Piste de résolution
Erreur HTTP 500.30 (in-process start failure)	L'application plante au démarrage — activer stdoutLogEnabled (étape 9) et consulter l'Observateur d'événements Windows pour l'exception exacte.
Erreur HTTP 500.31 / 502.5	Le module ANCM ne trouve pas l'application ou une version du runtime incompatible — vérifier la version du Hosting Bundle installée vs le TargetFramework du projet.
Le site ne répond pas après une mise à jour	Vérifier qu'`app_offline.htm` a bien été retiré après la copie des nouveaux fichiers.
Erreur d'accès refusé lors du démarrage	Droits NTFS insuffisants pour le compte du pool d'application (`ApplicationPoolIdentity`) sur le dossier du site ou le sous-dossier logs.
Le site répond en HTTP mais pas en HTTPS	Vérifier le binding 443 et que le certificat sélectionné est valide et non expiré.
Erreur de licence Stimulsoft en production (fonctionne en local)	Le fichier license.key n'a probablement pas été copié dans le dossier publié — vérifier sa présence sur le serveur au chemin exact attendu par le code (étape 6.1).
Erreur 404.13 ou 413 lors de la soumission d'un formulaire volumineux	Limite de taille de requête IIS trop basse — augmenter maxAllowedContentLength dans le web.config (étape 6.2).

Document préparé le 18 août 2026.