Guide de déploiement — API .NET (Stimulsoft.PDF.Forms + EF Core SQLite) sur Windows Server + IIS

Projet réel : ASP.NET Core Web API, .NET 8, packages Stimulsoft.PDF.Forms (génération de formulaires PDF côté serveur, en paire avec le package Angular `stimulsoft-forms` côté client) et Microsoft.EntityFrameworkCore.Sqlite (base de données SQLite locale).

*Topologie retenue : cette API est déployée sur un **site/domaine séparé** de l'application Angular (ex. `api.mondomaine.local` vs `questionnaire.mondomaine.local`). Cela implique de configurer explicitement **CORS** pour que le frontend Angular soit autorisé à appeler l'API — voir étape 5.*

---

1. Objectif et architecture

Ce guide décrit le déploiement d'une API ASP.NET Core (.NET 8) sur un Windows Server via IIS, en tant que backend séparé pour l'application Angular (déjà couverte dans un guide dédié). L'API génère des formulaires PDF via Stimulsoft.PDF.Forms et persiste ses données dans une base SQLite locale au serveur (fichier unique, pas de serveur de base de données séparé).

Comme pour tout ASP.NET Core, IIS héberge l'application **in-process** via le module ASP.NET Core (ANCM) : pas de process séparé à surveiller, pas de reverse proxy manuel, IIS démarre/redémarre l'API automatiquement.

Schéma : Angular (autre site/domaine) → appels HTTPS avec en-têtes CORS → IIS (port 443, certificat SSL) → module ANCM → API .NET (Stimulsoft.PDF.Forms + EF Core) → fichier SQLite local sur le serveur.

Point d'attention spécifique à ce projet, à garder en tête tout du long : le fichier de base SQLite ne doit **jamais** se trouver dans le dossier qui sera écrasé à chaque déploiement (sous peine de perdre les données à la prochaine mise à jour) — voir étape 4.

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 8 installé sur le poste/pipeline de build.
- Un nom de domaine ou sous-domaine dédié à l'API (distinct de celui de l'Angular), ex. `api.mondomaine.local`.
- Un certificat SSL pour ce domaine — indispensable ici en plus d'être une bonne pratique : la plupart des navigateurs bloquent les appels CORS depuis une origine HTTPS (l'Angular) vers une API en simple HTTP.
- Le fichier de licence Stimulsoft (``license.key`` ou clé en dur selon votre implémentation).

3. Étape 1 — Installer le rôle IIS

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

4. Étape 2 — Installer le .NET Hosting Bundle

Indispensable pour qu'IIS sache exécuter une application .NET 8 (sans elle, erreur 500.19 ou 502.5).

- Télécharger le « ASP.NET Core Runtime 8.0 – Windows Hosting Bundle » depuis dotnet.microsoft.com/download/dotnet/8.0 (bien prendre le Hosting Bundle, pas le runtime seul).
- Lancer l'installeur sur le serveur.
- Redémarrer IIS :

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

- Vérifier dans IIS Manager → nœud serveur → « Modules » → présence de `AspNetCoreModuleV2`.

5. Étape 3 — Configurer CORS dans le code

Comme l'API et l'Angular sont sur deux origines différentes, le navigateur bloquera par défaut les appels de l'Angular vers l'API tant que CORS n'est pas explicitement autorisé côté serveur. Dans `Program.cs` :

```
var builder = WebApplication.CreateBuilder(args);

builder.Services.AddCors(options =>
{
    options.AddPolicy("AllowAngularApp", policy =>
    {
        policy.WithOrigins("https://questionnaire.mondomaine.local") //
origine exacte de l'Angular
              .AllowAnyHeader()
              .AllowAnyMethod();
        // .AllowCredentials() si l'API utilise des cookies
d'authentification
    });
});

// ... reste de la configuration ...

var app = builder.Build();
```

```
app.UseCors("AllowAngularApp");

// ... app.MapControllers(); etc.
```

> **Remplacer l'URL par la vraie origine de production** de l'Angular (protocole + domaine + port exact). Une politique CORS trop permissive (``AllowAnyOrigin``) est déconseillée dès lors que l'API traite des données de questionnaire.

6. Étape 4 — Préparer un emplacement persistant pour la base SQLite

C'est l'étape la plus importante à ne pas manquer sur ce projet. Si la chaîne de connexion SQLite par défaut (souvent `Data Source=app.db` dans `appsettings.json`) pointe vers un chemin relatif, le fichier se crée dans le dossier de l'application — c'est-à-dire exactement le dossier qui sera écrasé/synchronisé à chaque redéploiement. Résultat : toutes les réponses au questionnaire disparaissent au prochain déploiement.

- Créer un dossier dédié, en dehors de tout dossier de déploiement, ex. `D:\Data\questionnaire-api\`.
- Dans `appsettings.Production.json` (créé s'il n'existe pas encore, à côté d'`appsettings.json` dans le projet), définir un chemin absolu :

```
{
  "ConnectionStrings": {
    "DefaultConnection": "Data Source=D:\\Data\\questionnaire-api\\app.db"
  }
}
```

- S'assurer que le code lit bien cette clé de configuration (``builder.Configuration.GetConnectionString("DefaultConnection")``) plutôt qu'une valeur en dur.
- Donner au compte du pool d'application IIS (par défaut `IIS AppPool\<NomDuPool>`) les droits de **lecture et écriture** sur `D:\Data\questionnaire-api\` (clic droit sur le dossier → Propriétés → Sécurité → Ajouter → saisir `IIS AppPool\NomDuPool` → Modifier).
- Si des migrations EF Core doivent s'appliquer au démarrage (``dbContext.Database.Migrate()``), vérifier qu'elles s'exécutent bien contre ce chemin en production, et prévoir une sauvegarde du fichier `app.db` avant toute mise à jour du schéma.

> **Sauvegardes :** un fichier SQLite se sauvegarde simplement en copiant le fichier `.db` (idéalement pool arrêté, ou via l'outil `backup` de SQLite pour une copie à chaud cohérente). Mettre en place une tâche planifiée de copie régulière de `D:\Data\questionnaire-api\app.db` vers un emplacement de sauvegarde.

7. Étape 5 — Publier l'application

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

Le dossier `publish/` contient les DLL, les dépendances (dont les assemblies Stimulsoft.PDF.Forms), et un `web.config` généré automatiquement configurant le module ANCM.

> Vérifier que `appsettings.Production.json` (étape 4) et, le cas échéant, `license.key` (étape 8) sont bien présents dans ce dossier publié — les fichiers de configuration/licence ne sont inclus que s'ils sont correctement référencés dans le `.csproj` (propriété « Copy to Output Directory »).

8. Étape 6 — Copier les fichiers vers le serveur

Copier le dossier `publish/` vers un dossier dédié, par exemple `D:\Sites\questionnaire-api\` — bien distinct du dossier de données créé à l'étape 4.

```
robocopy publish D:\Sites\questionnaire-api /MIR
```

> **Rappel :** `/MIR` supprime côté serveur tout ce qui n'est pas dans le dossier source. Comme la base SQLite est désormais hors de ce dossier (étape 4), elle ne sera jamais affectée par cette commande — c'est précisément l'objectif.

9. Étape 7 — Créer le site dans IIS Manager

1. Ouvrir IIS Manager → clic droit sur « Sites » → Ajouter un site web.
2. Nom du site : ex. QuestionnaireAPI.
3. Chemin d'accès physique : D:\Sites\questionnaire-api\.
4. Liaison (Binding) : type https de préférence dès le départ (voir étape 10 pour le certificat), nom d'hôte = `api.mondomaine.local`.
5. Pool d'applications : dédié, en mode « No Managed Code ».
6. Forcer explicitement l'environnement de production, pour être sûr qu'`appsettings.Production.json` est bien utilisé et que les pages d'erreurs détaillées de développement sont désactivées — ajouter dans le web.config (dans la balise `<aspNetCore>`) :

```
<aspNetCore processPath="dotnet" arguments=".\QuestionnaireApi.dll"
hostingModel="InProcess">
  <environmentVariables>
    <environmentVariable name="ASPNETCORE_ENVIRONMENT" value="Production" />
  </environmentVariables>
</aspNetCore>
```

1. Démarrer le site.

10. Étape 8 — Licence Stimulsoft et taille des requêtes

10.1 Déployer la clé de licence

Si le code charge la licence par fichier (``Stimulsoft.Base.StiLicense.LoadFromFile(path)``), vérifier que ce fichier `license.key` est bien copié dans `publish/` puis présent sur le serveur au chemin

attendu. Si elle est chargée par code (clé en dur dans `Program.cs`), rien à déployer séparément, mais vérifier qu'elle n'est pas accidentellement liée à un environnement de développement uniquement (ex. clé d'essai).

10.2 Taille maximale des requêtes

La génération/soumission de formulaires PDF peut impliquer des payloads plus volumineux que la moyenne (documents, images intégrées). Si erreur 404.13 ou 413 :

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

11. Étape 9 — Configurer HTTPS

- Obtenir un certificat SSL pour `api.mondomaine.local` (interne ou public selon l'exposition).
- IIS Manager → site → « Certificats de serveur » → importer le certificat, puis ajouter une liaison https sur le port 443.
- Vérifier que ``app.UseHttpsRedirection()`` est bien présent dans `Program.cs` pour rediriger automatiquement le trafic HTTP restant vers HTTPS.

12. Étape 10 — Pare-feu Windows

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

13. Étape 11 — Tester le déploiement

1. Depuis un poste : https://api.mondomaine.local/swagger (si Swagger est activé pour un test ponctuel) ou un endpoint connu de l'API doit répondre.
2. Depuis l'application Angular réelle (pas juste Postman) : vérifier qu'un appel vers l'API ne remonte pas d'erreur CORS dans la console navigateur (message typique : « has been blocked by CORS policy »). Si c'est le cas, revérifier l'origine exacte configurée à l'étape 5 (protocole, casse du domaine, port).
3. Générer un formulaire PDF de test de bout en bout, en conditions réelles depuis l'Angular.
4. Vérifier que la base SQLite se remplit bien au bon endroit (`D:\Data\questionnaire-api\app.db` grossit après une soumission).
5. Redémarrer le serveur et vérifier la reprise automatique du site.

14. Étape 12 — Logs et diagnostic

Activer temporairement les logs stdout dans le web.config en cas de problème :

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

Créer le sous-dossier `logs\` avec droits d'écriture pour le pool d'application. Consulter aussi l'Observateur d'événements Windows (Journaux Windows → Application) pour les erreurs de démarrage du module ANCM. Repasser `stdoutLogEnabled` à `false` une fois le diagnostic terminé.

15. Étape 13 — Sécuriser l'API

- Confirmer que Swagger/OpenAPI (s'il est présent dans le projet) est bien désactivé ou protégé en production — ne pas l'exposer publiquement sans authentification.
- Restreindre la politique CORS à l'origine exacte de l'Angular (étape 5), jamais ``AllowAnyOrigin`` en production.
- Restreindre les droits NTFS de `D:\Sites\questionnaire-api\` et `D:\Data\questionnaire-api\` aux seuls comptes nécessaires (déploiement, pool d'application).
- Ne jamais committer `appsettings.Production.json` ou `license.key` dans un dépôt Git public — les gérer comme des secrets de déploiement.
- Valider côté serveur toutes les entrées reçues de l'Angular, même si elles sont déjà validées côté client.

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

1. Sur le poste de build : `dotnet publish -c Release -o ./publish`.
2. Optionnel mais recommandé avant une mise à jour de schéma : copier `D:\Data\questionnaire-api\app.db` vers un emplacement de sauvegarde.
3. Déposer un `app_offline.htm` à la racine de `D:\Sites\questionnaire-api\` pour libérer proprement les fichiers verrouillés.
4. `robocopy publish D:\Sites\questionnaire-api /MIR` (la base de données, hors de ce dossier, n'est pas affectée).
5. Supprimer `app_offline.htm`.
6. Vérifier (étape 11) que l'API répond correctement et que les données existantes sont toujours présentes.

17. Checklist finale

Élément	Statut
Rôle IIS installé	☐
.NET 8 Hosting Bundle installé, AspNetCoreModuleV2 vérifié	☐

Élément	Statut
Politique CORS configurée avec l'origine exacte de l'Angular	☐
Dossier de données SQLite créé hors du dossier de déploiement	☐
Chaîne de connexion pointant vers ce dossier (appsettings.Production.json)	☐
Droits NTFS du pool d'application sur le dossier de données	☐
Application publiée (dotnet publish -c Release)	☐
Fichiers copiés sur le serveur (web.config inclus)	☐
license.key présent sur le serveur si utilisé	☐
Limite de taille des requêtes ajustée si besoin	☐
Site créé dans IIS, pool en « No Managed Code », ASPNETCORE_ENVIRONMENT=Production	☐
Certificat SSL installé, binding https configuré	☐
Pare-feu : ports 80/443 ouverts	☐
Appel depuis l'Angular réel sans erreur CORS	☐
Génération de PDF testée de bout en bout	☐
Base SQLite vérifiée au bon emplacement, hors dossier de déploiement	☐
Sauvegarde de la base SQLite planifiée	☐
Swagger désactivé/protégé en production	☐

18. Annexe — Dépannage rapide

Symptôme	Piste de résolution
Erreur CORS dans la console navigateur	Vérifier l'origine exacte déclarée dans WithOrigins (protocole, domaine, port), et que app.UseCors(…) est bien appelé avant app.MapControllers().
Les données disparaissent après un déploiement	La base SQLite était encore dans le dossier de déploiement écrasé par /MIR — la déplacer vers un dossier persistant (étape 4) et restaurer depuis une sauvegarde si disponible.
Erreur d'accès refusé sur le fichier .db	Droits NTFS insuffisants pour le compte IIS AppPool\<NomDuPool> sur le dossier de données.
Erreur HTTP 500.19	web.config invalide ou Hosting Bundle absent.
Erreur HTTP 500.30 (in-process start failure)	Activer stdoutLogEnabled et consulter l'Observateur d'événements Windows pour l'exception exacte (souvent une erreur de connexion à la base ou de licence Stimulsoft au démarrage).
Erreur de licence Stimulsoft en production	license.key absent du dossier publié, ou chemin de chargement incorrect (voir étape 8.1).
Erreur 404.13 / 413 sur soumission de formulaire	Augmenter maxAllowedContentLength (étape 8.2).

Document préparé le 18 août 2026.

From:
https://wiki-logeas.fr/certif/ - **dokuwiki-certif**

Permanent link:
https://wiki-logeas.fr/certif/doku.php?id=certif:procedure:miseenplaceserveur:serveurdotnet

Last update: **2026/08/18 14:16**