Guide de déploiement — Application Angular (SPA) sur Windows Server + IIS

Une SPA Angular, une fois « buildée », n'est qu'un ensemble de fichiers statiques (HTML, CSS, JS, assets). Contrairement à une application Node.js, **aucun process ne doit tourner en continu sur le serveur** : IIS sert directement ces fichiers, comme n'importe quel site statique. Node.js et Angular CLI ne sont nécessaires que pour la phase de build (sur un poste de développement ou un pipeline CI), pas sur le serveur de production — sauf si vous choisissez de builder directement dessus.

---

1. Objectif et architecture

Ce document décrit comment déployer une application Angular (SPA) sur un Windows Server existant, servie par IIS. L'architecture est nettement plus simple que pour une application Node.js server-side : IIS sert des fichiers statiques, avec une seule règle spécifique à gérer — rediriger toutes les routes internes de l'application vers `index.html` pour que le routing Angular (côté client) fonctionne correctement, y compris lors d'un rafraîchissement de page (F5) sur une route profonde.

Schéma : Internet → IIS (port 443, certificat SSL) → fichiers statiques du dossier `dist/` (+ règle de réécriture vers index.html pour les routes Angular).

2. Prérequis

- Accès administrateur au Windows Server (2016, 2019, 2022 ou équivalent) pour la configuration IIS.
- Un poste (ou pipeline CI) avec Node.js (version LTS) et Angular CLI installés, pour générer le build de production — ce poste peut être différent du serveur.
- Le code source de l'application Angular, avec un `angular.json` et `package.json` valides.
- 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, Compression du contenu statique, Journalisation HTTP, Filtrage des requêtes, Console de gestion IIS.

4. Étape 2 — Installer le module URL Rewrite

Indispensable pour que le routing interne d'Angular fonctionne (sans lui, un F5 sur une route comme `/questionnaire/3` renvoie une erreur 404, car ce chemin n'existe pas physiquement sur le serveur). Télécharger et installer « URL Rewrite Module 2.1 » depuis le site officiel IIS (iis.net/downloads/microsoft/url-rewrite).

Un redémarrage d'IIS est recommandé après installation : `iisreset` en ligne de commande administrateur.

5. Étape 3 — Builder l'application Angular

Sur le poste de build (pas nécessairement le serveur) :

```
# Installer les dépendances
npm ci

# Générer le build de production
ng build --configuration production
```

Le résultat se trouve dans `dist/<nom-du-projet>/` (ou `dist/<nom-du-projet>/browser/` selon la version d'Angular). Ce dossier contient uniquement des fichiers statiques (index.html, fichiers .js/.css avec hash dans leur nom, assets) : c'est tout ce dont IIS a besoin.

> **Vérifier la configuration de build :** si l'application n'est pas servie à la racine du domaine (ex. https://serveur/questionnaire/), penser à définir le `base href` correspondant, soit dans `index.html`, soit via `ng build –base-href /questionnaire/`.

6. Étape 4 — Copier les fichiers vers le serveur

Copier l'intégralité du contenu de `dist/<nom-du-projet>/` 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 dist\questionnaire-app D:\Sites\questionnaire /MIR
```

L'option `/MIR` miroir le contenu (supprime côté serveur les anciens fichiers qui n'existent plus dans le nouveau build) — utile pour éviter d'accumuler d'anciennes versions des fichiers JS/CSS hashés.

7. Étape 5 — Configurer web.config (routing SPA, MIME types, compression)

Créer (ou compléter) un fichier `web.config` à la racine de `D:\Sites\questionnaire\` :

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <system.webServer>

    <!-- Redirige toutes les routes internes Angular vers index.html -->
    <rewrite>
      <rules>
        <rule name="Angular Routes" stopProcessing="true">
          <match url=".*" />
          <conditions logicalGrouping="MatchAll">
            <add input="{REQUEST_FILENAME}" matchType="IsFile" negate="true"
/>
            <add input="{REQUEST_FILENAME}" matchType="IsDirectory"
negate="true" />
          </conditions>
          <action type="Rewrite" url="/index.html" />
        </rule>
      </rules>
    </rewrite>

    <!-- Types MIME parfois manquants par défaut sur IIS -->
    <staticContent>
      <remove fileExtension=".json" />
      <mimeMap fileExtension=".json" mimeType="application/json" />
      <remove fileExtension=".woff" />
      <mimeMap fileExtension=".woff" mimeType="font/woff" />
      <remove fileExtension=".woff2" />
      <mimeMap fileExtension=".woff2" mimeType="font/woff2" />
      <clientCache cacheControlMode="UseMaxAge"
cacheControlMaxAge="7.00:00:00" />
    </staticContent>

    <!-- Compression pour réduire le poids des fichiers JS/CSS -->
    <httpCompression>
      <dynamicTypes>
        <add mimeType="*/*" enabled="false" />
      </dynamicTypes>
      <staticTypes>
        <add mimeType="application/javascript" enabled="true" />
        <add mimeType="text/css" enabled="true" />
        <add mimeType="text/html" enabled="true" />
      </staticTypes>
    </httpCompression>

  </system.webServer>
</configuration>
```

> **Cache des fichiers hashés :** Angular ajoute un hash au nom des fichiers JS/CSS à chaque build (ex. `main.a1b2c3.js`), donc un cache long (7 jours ci-dessus) sur ces fichiers ne pose pas de

> problème — un nouveau build génère de nouveaux noms de fichiers. En revanche, `index.html` ne doit jamais être mis en cache longtemps côté client, sous peine de servir une ancienne version de l'app après une mise à jour ; ajouter si besoin une règle de cache spécifique et courte pour ce fichier.

8. Étape 6 — Créer le site dans IIS Manager

1. Ouvrir IIS Manager → clic droit sur « Sites » → Ajouter un site web.
2. Nom du site : ex. QuestionnaireAngularApp.
3. Chemin d'accès physique : D:\Sites\questionnaire\ (le dossier contenant index.html et le web.config).
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 : un pool standard suffit (le mode « Managed Code » n'a pas d'impact ici puisqu'il n'y a pas de code exécuté côté serveur).
6. Démarrer le site.

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.
- Ajouter une règle de redirection HTTP → HTTPS dans le web.config (placée avant la règle « Angular Routes ») :

```
<rule name="ForceHTTPS" stopProcessing="true">
  <match url="(.*)" />
  <conditions>
    <add input="{HTTPS}" pattern="off" />
  </conditions>
  <action type="Redirect"
          url="https://{HTTP_HOST}/{R:1}"
          redirectType="Permanent" />
</rule>
```

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 (pas de Node.js en écoute), puisque IIS sert directement les fichiers statiques.

```
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. Accéder à `http://nom-du-site` puis `https://nom-du-site` dans un navigateur : la page d'accueil de l'application doit s'afficher.
2. Naviguer vers une route interne de l'app (ex. `/questionnaire/3`), puis rafraîchir la page (F5) : elle doit continuer à s'afficher correctement (et non renvoyer une erreur 404) — c'est le test clé qui valide la règle de réécriture.
3. Ouvrir les outils de développement du navigateur (onglet Réseau) et vérifier qu'il n'y a pas d'erreur 404 sur les fichiers JS/CSS/fonts, et que les fichiers statiques sont bien servis avec compression (en-tête `Content-Encoding: gzip` ou `br`).
4. Vérifier que le questionnaire fonctionne de bout en bout (soumission, appels vers l'API si l'application Angular en consomme une).

12. Étape 10 — Sécuriser l'application

- Ajouter des en-têtes de sécurité HTTP via le web.config (ou un module dédié), notamment `X-Content-Type-Options: nosniff`, `X-Frame-Options: SAMEORIGIN`, et une politique `Content-Security-Policy` adaptée à l'application.
- Si l'application Angular appelle une API distincte (backend), s'assurer que cette API est elle-même correctement sécurisée (HTTPS, authentification, CORS restreint au domaine du site) — ce guide ne couvre que la partie frontend statique.
- Ne jamais inclure de secrets (clés API, identifiants) dans le code Angular : tout ce qui est buildé côté client est visible par n'importe quel visiteur.
- Restreindre les droits NTFS du dossier `D:\Sites\questionnaire\` pour que seuls les comptes de déploiement/administration puissent y écrire.

13. Étape 11 — Procédure de mise à jour de l'application

1. Sur le poste de build : récupérer la nouvelle version du code, puis relancer `npm ci` et `ng build –configuration production`.
2. Copier le nouveau contenu de `dist/<nom-du-projet>/` vers `D:\Sites\questionnaire\` (avec `robocopy /MIR` pour nettoyer les anciens fichiers hashés).
3. Aucun redémarrage de service n'est nécessaire (il n'y a pas de process applicatif) — IIS sert immédiatement les nouveaux fichiers.
4. Vérifier rapidement (étape 9) que le site répond correctement après la mise à jour.

> **Bonnes pratiques :** conserver une copie de la version précédente du dossier `dist/` quelque part (ex. archivée par date) pour pouvoir revenir en arrière rapidement en cas de problème après une mise à jour.

14. Checklist finale

Élément	Statut
Rôle IIS installé (avec compression du contenu statique)	☐
Module URL Rewrite installé	☐
Build de production généré (ng build –configuration production)	☐
Fichiers copiés sur le serveur (dossier dist/)	☐
web.config en place (règle SPA, MIME types, compression)	☐
Site créé dans IIS, binding http configuré	☐
Certificat SSL installé, binding https configuré	☐
Redirection HTTP → HTTPS active	☐
Pare-feu : ports 80/443 ouverts	☐
Test de rafraîchissement (F5) sur une route interne réussi	☐
Aucune erreur 404 sur les fichiers statiques (JS/CSS/fonts)	☐
Test de bout en bout du questionnaire réussi	☐
En-têtes de sécurité HTTP en place	☐

15. Annexe — Dépannage rapide

Symptôme	Piste de résolution
Erreur 404 en rafraîchissant une route interne (ex. /questionnaire/3)	Règle « Angular Routes » absente ou mal placée dans le web.config, ou module URL Rewrite non installé.
Page blanche à l'ouverture du site	Vérifier le `base href` dans index.html (doit correspondre au chemin réel du site) et la console navigateur pour une erreur JS/404 sur un asset.
Fichiers .woff/.json en erreur 404 ou mal interprétés	Vérifier la section staticContent du web.config (types MIME manquants par défaut sur certaines configurations IIS).
L'app affiche encore l'ancienne version après une mise à jour	Cache navigateur ou cache trop long sur index.html — vérifier la politique de cache et forcer un rafraîchissement (Ctrl+F5) pour confirmer.
Erreur 500.19 sur le site IIS	web.config invalide (erreur de syntaxe XML) ou module URL Rewrite non installé.
Le site répond en HTTP mais pas en HTTPS	Vérifier le binding 443 et que le certificat sélectionné est valide et non expiré.

Document préparé le 18 août 2026.