Phase Bonus — Publipostage DOCX/PDF

🎯Objectif

Permettre la génération automatique d'une **fiche Pokémon imprimable** (format DOCX ou PDF) à partir de l'identifiant du Pokémon. Le but est de démontrer la **modularité du backend .NET** et sa **réutilisabilité inter-systèmes** (Angular, MuleSoft, Salesforce).

🧩Options techniques envisagées

Option A – HTML → PDF (Razor + wkhtmltopdf)

- Génération via RazorLight / Razor Pages.
- Conversion PDF par wkhtmltopdf ou PuppeteerSharp.
- ✅Avantages : rendu visuel fidèle (HTML/CSS), rapide, portable.
- ⚠️Limites : difficilement réutilisable dans d'autres contextes (Salesforce, Mule), nécessite Chromium ou dépendances lourdes.

Option B – DOCX → PDF (TemplateEngine.Docx) *(retenue)*

- Génération à partir d'un **template DOCX** avec content controls (SDT).
- Utilisation de **TemplateEngine.Docx** pour injecter les données.
- Conversion optionnelle en PDF via **LibreOffice headless**.
- ✅Avantages : format bureautique standard, réutilisable depuis Salesforce ou MuleSoft, démontre la modularité du backend.
- ⚠️Limites : rendu moins flexible qu'en HTML, dépendance à LibreOffice pour le PDF.

Raisons du choix de l'option B

Modularité et découplage

- L'API de génération DOCX est conçue comme un **service transversal** indépendant de l'UI ou de MuleSoft.
- Elle pourra être **appelée directement par Salesforce** dans une phase ultérieure, prouvant la réutilisabilité du code et le respect du principe **"une seule source de vérité"**.

Réutilisation dans Salesforce

- Une fois les Pokémon intégrés à Salesforce (via MuleSoft), la même API pourra être invoquée depuis Apex :
- Entrée : `pokemonId`
- Sortie : un fichier DOCX ou PDF rattaché à un objet `Pokemon__c` ou `ContentVersion`.
- Cela démontrera :
- la capacité à réaliser un **callout Apex sécurisé** vers une API externe,
- la **réutilisabilité d'un même service .NET** dans plusieurs contextes techniques,
- et la **cohérence d'intégration interstack** (Angular / MuleSoft / .NET / Salesforce).

Capitalisation des compétences acquises

- Ce choix illustre une **mise à profit des compétences techniques acquises sur des projets antérieurs** en environnement Salesforce :
- automatisation de la génération de documents à partir de données métier,
- intégration d'applications hétérogènes via des API sécurisées,
- et conception de services modulaires réutilisables.
- Ici, le service est pensé dès le départ pour être **transverse**, **interopérable** et **démonstratif** d'une architecture moderne et maintenable.

Implémentation prévue

- **Endpoints** :
- `POST /api/print/generate` (DOCX/PDF)
- `GET /api/print/templates` (liste des modèles disponibles)
- `POST /api/print/templates` (upload interne)
- **Sécurité** : même logique que les routes internes `/internal/*` avec header `X-Internal-Token`.
- **Stockage** : `/app/templates/{templateKey}/v{n}/template.docx` + metadata JSON.
- **Libs** : `TemplateEngine.Docx`, `LibreOffice headless` (option PDF).
- **Angular** : bouton "Exporter DOCX/PDF" sur la fiche Pokémon (sélection du template + langue).
