**Projet Python PDF + Egnyte (tri par date) - PDF Name AutoGenerator v1.0**

Créer un outil capable de générer automatiquement des noms de fichiers PDF à partir de leur contenu. L’objectif est de mettre en place un modèle de nommage personnalisable, capable d’extraire les informations clés de chaque PDF.

Le candidat idéal aura de l’expérience en extraction de données, en conception de modèles dynamiques, ainsi qu’en développement logiciel.  
En complément de ce projet, nous souhaitons également créer une fonction permettant de **trier automatiquement les PDF dans des dossiers spécifiques** en fonction du nom généré.

Tout ce système devra être **déployé dans notre solution de gestion cloud de fichiers :** [**Egnyte**](https://www.egnyte.com).( <https://www.egnyte.com/>)

**⚙️ Python ou PHP pour ce projet ?**

✅ **Python** est **le meilleur choix** pour ce projet, pour plusieurs raisons techniques :

| **Critère** | **Python** | **PHP** |
| --- | --- | --- |
| **Extraction de contenu PDF** | 🟢 Excellente (avec pdfminer, PyMuPDF, pdfplumber, OCRmyPDF…) | 🔴 Limité, moins de bibliothèques stables |
| **Manipulation de fichiers système/cloud** | 🟢 Très souple, excellent pour automatisation (avec os, shutil, API SDK) | 🟠 Moins adapté pour automatisation côté serveur |
| **API Egnyte** | 🟢 Intégration facile avec requests, gestion OAuth, etc. | 🟠 Possible, mais plus complexe à maintenir |
| **Déploiement en tâche planifiée / script autonome** | 🟢 Idéal (cron jobs, scripts cloud, Docker) | 🔴 PHP est conçu pour le web, pas pour les scripts autonomes |
| **Maintenance & évolutivité** | 🟢 Python est devenu un standard pour les outils de productivité automatisés |  |

✅ Prototype de base prêt !

Voici ce que fait le script :

1. 🔍 **Lit chaque PDF** dans le dossier source (/mnt/data/pdfs\_source)
2. ✏️ **Extrait les deux premiers mots** de la première page pour générer un nom (ex: Invoice\_123.pdf)
3. 📂 **Classe automatiquement** chaque PDF renommé dans un sous-dossier basé sur la première lettre de son nouveau nom (ex: I/Invoice\_123.pdf)
4. ✅ **Copie** le PDF renommé dans le dossier organisé (/mnt/data/pdfs\_sorted)

**🧱 Étape 1 – Extraction + renommage + tri local (date)**

* 🔍 Lecture du PDF avec **PyMuPDF**
* 🧠 Extraction de texte pour générer un nom intelligent (ex. : Facture\_123.pdf)
* 🗂️ Classement automatique selon la **date de création du PDF**, formatée ainsi :  
  **PDF-YYYY\_MM\_DD/**
* ✅ Si le dossier existe déjà → **ne pas le recréer**, juste y copier le fichier

📂 Dossier source : /mnt/data/pdfs\_source  
📂 Dossier organisé : /mnt/data/pdfs\_sorted/PDF-2025\_05\_08/Facture\_123.pdf

**🔐 Étape 2 – Connexion API Egnyte (OAuth2)**

* Création d'une application via https://developers.egnyte.com
* Obtention du :
  + client\_id
  + client\_secret
  + domain (ex. tonentreprise.egnyte.com)
* Authentification via Authorization Code Flow + refresh\_token

⚠️ On stockera les tokens dans un fichier .egnyte\_token.json localement

**☁️ Étape 3 – Synchronisation vers Egnyte**

* 📁 Pour chaque PDF trié localement :
  + Vérifier l'existence du dossier Shared/PDF-YYYY\_MM\_DD/ dans Egnyte
  + Créer le dossier s’il n’existe pas (POST /mkdir)
  + Uploader le fichier (POST /fs-content/...)
  + 📄 Vérifier si le fichier existe déjà → écraser ou ignorer selon paramètre

**🧠 Étape 4 – Templates de nommage dynamiques**

* Fichier config.json ou settings.yaml :

json

CopierModifier

{

"naming\_template": "first\_word\_second\_word",

"fallback\_name": "unnamed\_pdf",

"sort\_by": "creation\_date", // peut devenir "text\_date", "metadata\_date", etc.

"replace\_existing": true

}

* Tu pourras modifier les règles de renommage sans toucher au code
* Possibilité de future extension avec regex / IA sémantique

**🧭 Étape 5 – Interface Web (optionnelle mais recommandée)**

Avec Flask, on aura :

* 📤 Uploader un PDF → traitement + push Egnyte
* 📜 Voir les logs des fichiers traités
* ⚙️ Modifier les templates de nommage à la volée
* 🔐 Auth Egnyte simple avec refresh token en cache