***📘 Documentation – Configuration de Microsoft Graph API pour manipuler des listes SharePoint via Node.js & Axios***

1. **Introduction**

Cette documentation décrit les étapes nécessaires pour configurer et utiliser Microsoft Graph API afin de manipuler des listes SharePoint dans un projet Node.js à l’aide d’Axios.  
Elle met en évidence la différence entre l’API REST classique de SharePoint et l’API Graph, ainsi que son rôle clé dans l’intégration avec les services Microsoft 365 et la Power Platform.

1. **Différence entre l’API REST classique et Microsoft Graph**

**API REST SharePoint classique**

* 1. Permet de manipuler uniquement les données propres à SharePoint (listes, sites, fichiers).
  2. Chaque service Microsoft a sa propre API, ce qui implique une gestion multiple des points d’accès.
  3. Authentification souvent spécifique à SharePoint.

**Microsoft Graph API**

* 1. Fournit un **point d’accès unique** pour tous les services Microsoft 365 : SharePoint, OneDrive, Teams, Outlook, Entra ID, etc.
  2. Simplifie la gestion des permissions grâce à un modèle unifié.
  3. Compatible avec la **Power Platform** (Power Automate, Power Apps, Power BI) pour automatiser et enrichir les processus.
  4. Recommandé par Microsoft pour toute nouvelle intégration car il centralise la sécurité et l’administration.

**Conclusion** : utiliser **Graph API** au lieu de l’API REST classique garantit plus de flexibilité, une meilleure sécurité et une intégration native avec tout l’écosystème Microsoft.

1. **Étapes de configuration de Microsoft Graph API**

***Étape 1 – Accès au portail d’administration***

* Connectez-vous au **Microsoft 365 Admin Center**.
* Ce portail permet de gérer l’organisation, les utilisateurs et les autorisations globales.

***Étape 2 – Accéder à Microsoft Entra ID (Azure Active Directory)***

* Dans le menu de gauche, ouvrez **Microsoft Entra ID** (anciennement Azure AD).
* C’est ici que l’on gère les identités, applications et autorisations pour Graph API.

***Étape 3 – Inscription de l’application***

* Cliquez sur **Inscriptions d’applications** → **Nouvelle inscription**.
* Donnez un nom à l’application (ex. *NodeJS-SharePoint-Graph*).
* Enregistrez et conservez dans un fichier séparé :
  + **ID d’application (Client ID)**
  + **ID de l’annuaire (Tenant ID)**

**Étape 4 – Générer un secret client**

* Allez dans **Certificats & secrets**.
* Créez un **nouveau secret client**.
* Conservez sa **valeur** dans un fichier sécurisé (ce secret ne sera plus visible après fermeture).

**Étape 5 – Attribuer les autorisations nécessaires**

* Demandez au **Global Administrator** de l’entreprise d’accorder les droits.
* Ouvrez l’application enregistrée → **Autorisations d’API**.
* Cliquez sur **Ajouter une autorisation** → **Microsoft Graph** → **Autorisations d’application**.
* Sélectionnez :
  + Sites.ReadWrite.All → **Read and write items in all site collections**
* Cliquez sur **Accorder les autorisations d’administration**.

**Étape 6 – Intégration dans Node.js avec Axios**

Une fois les credentials obtenus (Client ID, Tenant ID, Client Secret), vous pouvez générer un **token d’accès** via @azure/identity, puis appeler Graph API avec Axios.

Exemple minimal :

Une image contenant texte, capture d’écran, document, Police

Le contenu généré par l’IA peut être incorrect.

1. **Conclusion**

Avec Microsoft Graph, vous disposez :

* d’une **authentification centralisée** via Microsoft Entra ID,
* d’une **gestion unifiée des permissions**,
* d’un **point d’accès unique** pour manipuler SharePoint et d’autres services Microsoft 365.

Cela en fait l’approche recommandée pour tout projet moderne intégrant des **listes SharePoint** avec **Node.js** et la **Power Platform**.