ChatGPT

Comparaison des APIs Google Books et OpenLibrary pour l'Auto-complétion d'un Formulaire de Livre

Champs de données disponibles dans chaque API

Les deux APIs fournissent de nombreuses métadonnées exploitables pour pré-remplir une fiche livre. Le tableau ci-dessous compare les principaux champs du modèle **Book** (titre, auteurs, éditeur, date de publication, nombre de pages, description, couverture, genres, etc.) avec leur nom dans la réponse JSON de chaque API :

Champ (donnée livre)	**Google Books API** (champ JSON dans `volumeInfo`)	**OpenLibrary API** (champ JSON)
Titre	`title` [1] (Titre du livre)	`title` [2] (Titre de l'édition ou du travail)¹
Sous-titre	`subtitle` [1] (Sous-titre du livre, si disponible)	*Peut exister comme* `subtitle` *si renseigné pour l'édition*¹
Auteur(s)	`authors` (liste de noms d'auteurs) [1]	`authors` (liste d'auteurs avec nom) [3] (chaque entrée contient un champ `name`)
Éditeur (Maison d'édition)	`publisher` [4]	`publishers` (liste d'éditeurs, chaque entrée a un `name`) [5]
Date de publication	`publishedDate` [4] (chaîne, ex. "2021-07-01" ou juste année)	`publish_date` [6] (chaîne, ex. "2009")
Nombre de pages	`pageCount` [7] (nombre entier)	`number_of_pages` [8] (nombre entier)
Description (Résumé)	`description` [9] (texte descriptif du livre)	`description` (texte descriptif, souvent présent via l'endpoint *details*) [10] ²
Couverture (Image)	`imageLinks` (objets URL d'images : `thumbnail`, `small`, `medium`, etc.) [11]	`cover` (URLs pour images en petite, moyenne, grande taille) [12]
Genres / Sujets	`categories` (liste de catégories/ genres) [13]	`subjects` (liste de sujets/genres) [14] ³
Identifiants (ISBN, etc.)	`industryIdentifiers` (liste d'identifiants : ISBN_10, ISBN_13, etc.) [15]	`identifiers` (objets d'identifiants : `isbn_10`, `isbn_13`, OCLC, etc.) [16]

Champ (donnée livre)	**Google Books API** (champ JSON dans `volumeInfo`)	**OpenLibrary API** (champ JSON)
Langue	`language` (code langue, ex. "en") [17]	`languages` (code langue sous forme de clé, ex. `/languages/eng`) [18]

¹ **Titre/Sous-titre :** Google Books fournit le titre (`title`) et éventuellement un sous-titre séparé (`subtitle`) [1]. OpenLibrary stocke aussi le sous-titre si disponible, soit dans un champ `subtitle` de l'édition, soit concaténé au titre selon le cas (la documentation indique que le titre et le sous-titre sont inclus dans les données renvoyées) [2].

² **Description :** L'API Google renvoie souvent un résumé du livre dans `description` [9]. Du côté d'OpenLibrary, le résumé/description peut être disponible via l'API d'édition détaillée. Par exemple, en utilisant `jscmd=details`, on obtient un champ `description` si la description du livre a été renseignée dans OpenLibrary [10]. (Le mode `jscmd=data` de l'API OpenLibrary de base ne retourne pas le résumé dans tous les cas.)

³ **Genres/Sujets :** Google Books fournit des catégories thématiques dans `categories` (par ex. "Fiction / Fantasy") [13]. OpenLibrary renvoie une liste de sujets sous `subjects` [14] (par ex. des thèmes ou genres associés). Ces champs peuvent servir à pré-remplir un champ "Genre(s)" ou "Catégorie" dans le formulaire, éventuellement en multi-sélection.

Proposition de logique d'intégration des deux sources

Pour maximiser le pré-remplissage des champs tout en évitant des données erronées ou en double, on propose la stratégie d'intégration suivante :

1. **Interroger Google Books en premier (source principale)** – Lors de la saisie (par exemple après avoir entré un ISBN ou un titre), l'application effectue une requête vers l'API Google Books. L'API permet de rechercher un volume par identifiant (ISBN) ou par mots-clés (titre/auteur) et peut retourner une liste de volumes correspondants [19].
2. **Si un résultat unique pertinent est trouvé** (par exemple, via ISBN, un livre précis) : utiliser les données de ce volume pour pré-remplir le formulaire. Chaque champ du modèle Book est rempli avec la valeur correspondante de `volumeInfo` (titre, auteurs, etc. comme indiqué dans le tableau ci-dessus).

3. **Si plusieurs résultats sont retournés** (cas d'une recherche par titre ambigu ou partiel) : soit sélectionner automatiquement le plus pertinent (ex. le premier résultat par pertinence), soit présenter à l'utilisateur une liste de suggestions de titres pour qu'il choisisse le bon livre. Il est recommandé de **laisser l'utilisateur confirmer** le choix lorsqu'il y a ambiguïté, afin d'éviter de pré-remplir avec le mauvais livre.

4. **Fallback sur OpenLibrary si Google n'a rien retourné** – Si la requête Google Books n'aboutit à aucun résultat (par exemple pour un ouvrage plus rare ou un ISBN que Google ne connaît pas), on interroge alors l'API OpenLibrary en second. On utilise le même identifiant ou critères (ISBN, titre, etc.) pour chercher dans OpenLibrary.

5. **Si OpenLibrary renvoie des données** : on pré-remplit le formulaire avec ces informations. Les champs disponibles (titre, auteurs, éditeur, etc.) sont remplis d'après la réponse JSON

OpenLibrary (voir tableau ci-dessus pour la correspondance des champs). Par exemple, `title`, `authors[*].name`, `publish_date`, `number_of_pages`, `cover.small/medium/large` pour l'image de couverture, etc. OpenLibrary a souvent des informations pour des livres plus anciens ou des éditions que Google pourrait ne pas lister, ce qui complète bien le service.

6. **Si OpenLibrary ne retourne rien non plus** : l'auto-complétion échoue et l'utilisateur devra renseigner manuellement les détails du livre. Dans ce cas, on peut afficher un message du type « Aucune information trouvée automatiquement » pour que l'utilisateur sache qu'il doit tout saisir.

7. **Fusionner les résultats des deux si nécessaire** – Dans la plupart des cas, on utilisera **soit** Google Books **soit** OpenLibrary selon le scénario ci-dessus. Cependant, il peut arriver que **les deux sources aient des informations complémentaires** pour un même livre (si on décide d'interroger OpenLibrary en complément même quand Google a répondu). Par exemple, Google Books pourrait fournir un résumé alors qu'OpenLibrary fournit une couverture de meilleure qualité, ou vice versa.

8. On peut **prioriser les champs de Google Books** par défaut (considérée comme source principale plus à jour sur les livres grand public) et utiliser OpenLibrary pour combler les champs manquants. Par exemple, si Google ne retourne pas la description du livre mais qu'OpenLibrary en a une, on pourrait la récupérer pour pré-remplir le champ résumé. De même, si la couverture n'est pas disponible via Google (ou de résolution insuffisante), on peut utiliser l'URL de couverture d'OpenLibrary [12].
9. En cas de **conflit ou de divergence** sur une même information (par ex. un nombre de pages différent entre les deux APIs), il est préférable de faire confiance à la source principale (Google) ou d'indiquer à l'utilisateur qu'une vérification est nécessaire. On pourrait éventuellement montrer les deux valeurs en suggestion afin que l'utilisateur tranche si c'est pertinent, mais cela complexifie l'interface.

En résumé, la logique serait : **Google Books d'abord**, puis **OpenLibrary en secours** si Google ne trouve rien. OpenLibrary peut aussi servir à enrichir les données si certaines informations font défaut côté Google.

Indications pour l'implémentation côté front-end

Du point de vue de l'interface utilisateur, voici quelques conseils pour intégrer ces données de façon fluide et conviviale :

- **Pré-remplissage éditable** – Tous les champs récupérés devraient être affichés dans le formulaire de saisie du livre avec leurs valeurs pré-remplies, mais en restant modifiables. L'utilisateur doit pouvoir corriger ou ajuster chaque champ. Par exemple, le titre et le nom de l'auteur peuvent être auto-complétés mais l'utilisateur pourra les éditer s'il le souhaite (p.ex. pour ajouter un sous-titre manquant, corriger une coquille, etc.). On veille à ce que le champ description soit en texte libre modifiable, que la liste des auteurs soit éditable (voire permettre d'en ajouter/supprimer), etc.

- **Indication de source/suggestion** – Il peut être utile d'indiquer visuellement que les champs ont été remplis automatiquement (par exemple en les surlignant ou en ajoutant un libellé "Suggestion issue de Google Books"). Ainsi, l'utilisateur comprend qu'il s'agit d'une donnée importée qu'il peut modifier. Une fois le champ édité par l'utilisateur, on peut retirer cette indication.

- **Gestion des résultats multiples** – Si la recherche initiale renvoie plusieurs livres possibles (notamment lors d'une recherche par titre ou auteur), il est recommandé d'**afficher une liste de suggestions** à l'utilisateur plutôt que de remplir directement les champs avec un résultat incertain. Par exemple, afficher une liste déroulante ou une modale avec les titres/auteurs correspondants parmi lesquels choisir. Une fois le livre sélectionné par l'utilisateur, remplir les champs avec les données de ce livre précis. Cela évite les erreurs de correspondance. *(Exemple : l'utilisateur tape "Harry Potter" — plusieurs éditions ou tomes ressortent ; l'interface propose "Harry Potter à l'école des sorciers – 1998 – 309 pages" et d'autres tomes en suggestions, plutôt que de choisir arbitrairement le premier.)*

- **Mise à jour de la couverture** – Si une image de couverture est récupérée (URL fournie par l'API), on peut l'afficher en aperçu dans le formulaire (par ex. une vignette). Ce champ étant aussi modifiable, l'utilisateur pourrait avoir la possibilité de le changer : soit en fournissant une autre URL d'image, soit en téléversant une image personnalisée si l'API n'en a pas fourni ou si l'utilisateur préfère une autre illustration.

- **Champs multi-valeurs** – Pour les champs comme les auteurs ou les genres (sujets), qui peuvent contenir plusieurs valeurs, l'interface doit gérer la liste. Par exemple, si le JSON Google renvoie deux auteurs, pré-remplir deux champs auteur ou une liste à puces. Assurez-vous que l'utilisateur puisse facilement ajouter ou supprimer des auteurs. Idem pour les genres/sujets : s'il y en a plusieurs (ex. « Fantasy », « Adventure »), on peut préremplir un champ multi-sélection ou des tags que l'utilisateur peut éditer.

- **Absence de données** – Prévoyez le cas où certains champs ne sont pas fournis par l'API. Par exemple, certains livres n'ont pas de résumé dans Google Books, ou pas de nombre de pages. Le formulaire doit alors laisser le champ vide (ou avec une indication comme "Non renseigné") afin que l'utilisateur puisse le remplir manuellement le cas échéant. Ne pas remplir un champ non retourné par l'API évite d'insérer de l'information incorrecte (« N/A » pourrait être confondu avec une donnée réelle).

En appliquant ces principes, l'auto-complétion améliorera grandement la vitesse et la facilité de saisie, tout en laissant le contrôle à l'utilisateur final. Celui-ci voit les données importées (depuis Google Books en priorité, OpenLibrary en secours) et peut les ajuster avant d'enregistrer la fiche du livre. Cette approche hybride garantit un **pré-remplissage maximal** tout en maintenant une **validation humaine** des informations pour une base de données livres de haute qualité.

Sources : Documentation officielle de l'API Google Books (métadonnées de Volume) [1] [20], et documentation de l'API OpenLibrary (Books API, champs JSON disponibles) [21] [22], consultées pour la liste des champs et exemples de réponses JSON.

---

[1] [4] [7] [9] [11] [13] [15] [17] [19] [20] Volume | Google Books APIs | Google for Developers
https://developers.google.com/books/docs/v1/reference/volumes

[2] [3] [5] [6] [8] [10] [12] [14] [16] [18] [21] [22] Developer Center / APIs / Books API | Open Library
https://openlibrary.org/dev/docs/api/books