

PROJET

APPLICATION DE RECHERCHE DE LIVRES  
EN UTILISANT L'API DE GOOGLE BOOKS  
AVEC POSTMAN

## DESCRIPTION DU PROJET

Ce projet vise à créer une application de recherche de livres en utilisant l'API de Google Books. Cette API permet de rechercher des livres en utilisant des mots-clés et de récupérer les détails d'un livre à partir de son identifiant. Cette API est utilisée par plusieurs sites web commerciaux tels que fnac, darty et amazon.



# MISSION

Le maire d'Arcachon souhaite donner de la visibilité sur les livres disponibles dans la bibliothèque municipale en utilisant une API. Pour s'assurer que l'API fonctionne correctement, il fait appel à nos services pour mettre en place une stratégie d'automatisation des tests de Web Service en utilisant l'application Postman.

Notre travail est très important car des modifications seront apportées dans l'avenir.

# MISSION

Pour éviter d'éventuelles régressions, il est important de s'assurer que l'API fonctionne correctement avant et après chaque modification. C'est là que l'automatisation des tests de Web Service entre en jeu.

Nous allons utiliser Postman pour automatiser les tests de l'API de la bibliothèque municipale d'Arcachon.

# MISSION

**Nous allons créer des scripts pour tester les différentes fonctionnalités de l'API, telles que la recherche de livres et la récupération des détails d'un livre. Nous allons également nous assurer que les résultats de la recherche de livres sont corrects et que les informations de détails des livres sont correctes.**

# MISSION

**En utilisant cette stratégie d'automatisation des tests, nous pouvons garantir que l'API de la bibliothèque municipale d'Arcachon fonctionne correctement et que toutes les modifications apportées à l'API ne causeront pas de régressions.**

**En conclusion, ce projet est essentiel pour garantir que les utilisateurs de la bibliothèque municipale d'Arcachon ont accès à des informations précises sur les livres disponibles. Grâce à l'automatisation des tests avec Postman, nous pouvons garantir que l'API fonctionne correctement à tout moment, même après des modifications futures.**

## CONTEXTE

Nous utiliserons principalement deux routes de l'API de Google Books : "LIST Books" et "GET Book":

- La route "LIST Books" sera utilisée pour rechercher des livres en utilisant des mots-clés,
- la route "GET Book" sera utilisée pour récupérer les détails d'un livre à partir de son identifiant.

Nous allons utiliser l'application Postman pour tester ces deux routes de l'API de Google Books. Nous allons également utiliser le paramètre "maxResults" pour limiter le nombre de résultats retournés lors de la recherche de livres.

## CONTEXTE

En utilisant cette application, les utilisateurs pourront rechercher des livres en utilisant des mots-clés et afficher les détails d'un livre particulier en utilisant son identifiant. Les utilisateurs peuvent également filtrer les résultats de recherche en utilisant les paramètres disponibles dans la documentation de l'API.

URL de documentation de l'API : <https://developers.google.com/books/docs/overview>

# OBJECTIF

L'objectif de ce projet est de créer une collection Postman pour effectuer des tests automatisés de non-régression sur l'API. À la fin du projet, nous aurons des tests pour chaque route de l'API, qui vérifieront le bon fonctionnement de la recherche de livres, le nombre de résultats retournés, le code de réponse, etc.

Nous allons également mettre en place des vérifications (assertions) pour vérifier la cohérence des réponses des deux routes. Nous pourrons choisir les points de comparaison, tels que les informations de l'auteur, le titre du livre, la date de publication, etc.

# CONCEPTION

Avant de se lancer dans la partie technique de la création d'un projet Postman, il est important de prendre le temps d'écrire des scénarios de test en langage naturel ou en Gherkin. Cela permettra de mieux comprendre les fonctionnalités de l'API et d'avoir une vue d'ensemble sur les différentes requêtes.



## LE PROJET CONSISTERA EN PLUSIEURS ÉTAPES :

1) Écriture des scénarios de test en langage naturel ou en Gherkin :

cette étape permettra de décrire les différents cas d'utilisation de

l'API, les entrées attendues et les résultats attendus. Il est important

de bien réfléchir à tous les scénarios possibles pour une utilisation

optimale de l'API.

## LE PROJET CONSISTERA EN PLUSIEURS ÉTAPES :

2) Inclusion des scénarios de test dans la documentation de chaque requête : cette étape permettra de documenter les tests et d'assurer une meilleure compréhension de l'API. Les scénarios seront inclus dans la documentation de chaque requête, ce qui permettra aux autres membres de l'équipe de mieux comprendre le fonctionnement de l'API.

## LE PROJET CONSISTERA EN PLUSIEURS ÉTAPES :

3) **Création des dossiers Postman** : cette étape consiste à créer deux dossiers( LIST BOOKS & GET BOOK) pour les scénario de test:  
Chaque dossier doit inclure les requêtes nécessaires pour réaliser le scénario, ainsi que les assertions pour vérifier que les résultats correspondent à ce qui est attendu.

## LE PROJET CONSISTERA EN PLUSIEURS ÉTAPES :

**4. Exécution des tests : il est temps de les exécuter pour vérifier le bon fonctionnement de l'API. Les résultats des tests seront documentés et examinés pour détecter d'éventuelles régressions.**

## LE PROJET CONSISTERA EN PLUSIEURS ÉTAPES :

5. Maintenance des tests : enfin, il est important de maintenir les tests tout au long du cycle de vie de l'API. Cela implique de mettre à jour les tests en fonction des modifications apportées à l'API et de s'assurer que les tests continuent de fonctionner correctement.

## LE PROJET CONSISTERA EN PLUSIEURS ÉTAPES :

En conclusion, le projet consiste à concevoir un projet Postman avec des scénarios de test en langage naturel ou en Gherkin. Les scénarios seront inclus dans la documentation de chaque requête, les collections Postman seront créées pour chaque scénario de test, et les tests seront exécutés et maintenus tout au long du cycle de vie de l'API.

# DÉCOUVERTE

Pour ce projet de découverte sur Postman, nous allons commencer par tester les différentes requêtes pour les routes LIST Books et GET Book en utilisant des données en dur.



# DÉCOUVERTE

Après créer une collection sur Postman et y ajouter deux dossiers : un pour la route LIST Books et l'autre pour la route GET Book. Nous allons ensuite utiliser des données en dur pour tester chacune de ces requêtes, en nous assurant que les résultats retournés sont cohérents et conformes aux spécifications de l'API.

# DÉCOUVERTE

Pour ce faire, nous allons utiliser la fonctionnalité "Pre-request Script" de Postman pour définir les données de test et les préparer avant l'exécution de chaque requête. Nous allons également utiliser des assertions pour vérifier que les résultats obtenus sont corrects et répondent aux exigences de l'API.

# DÉCOUVERTE

Une fois que nous aurons réussi à tester les deux routes, nous pourrons alors passer à la phase suivante du projet, qui consistera à créer des tests automatisés pour vérifier l'API.

## CONSTRUCTION

Pour la construction de ce projet sur Postman, nous allons utiliser les scripts de tests pour implémenter les tests que nous avons précédemment définis.

Nous allons commencer par ajouter ces tests à notre collection sur Postman, en utilisant les assertions et les variables globales pour effectuer des vérifications sur les résultats de chaque requête.



# CONSTRUCTION

Une fois que les tests sont ajoutés à notre collection, nous allons les exécuter manuellement en utilisant la fonctionnalité Collection Runner de Postman. Cela nous permettra de vérifier que chaque test est exécuté correctement et que les résultats obtenus sont conformes à ce que nous attendons.

# CONSTRUCTION

Cependant, l'objectif ultime est d'exécuter ces tests sur une chaîne d'Intégration Continue (CI). Pour commencer à atteindre cet objectif, nous allons exécuter ces tests avec Newman, l'interface en ligne de commande (CLI) de Postman. Nous allons créer un script qui utilise Newman pour exécuter notre collection de tests.

## REPORTING

Pour répondre à l'exigence du maire, nous allons mettre en place un rapport personnalisé et détaillé sur les résultats des tests. Pour cela, nous allons utiliser les fonctionnalités de reporting de Newman. Nous allons générer des rapports de différents types pour avoir une vue constante sur les résultats des tests.



# REPORTING

Le rapport doit inclure les informations suivantes :

- Le nom de la collection testée
- Le nombre de requêtes exécutées
- Le nombre de requêtes réussies et échouées
- Les détails des requêtes échouées, y compris les codes de réponse et les messages d'erreur
- Le temps total d'exécution des tests

# REPORTING

Nous allons utiliser Newman pour exécuter les tests automatiques et générer le rapport personnalisé. Le rapport sera stocké dans un fichier HTML pour une visualisation facile.

# BONUS

-GET /volumes?q={searchterms} : Rechercher dans l'API avec des mots clés.

Prend le paramètre : maxResults pour limiter le nombre de résultats.

Dans le reste du projet, on utilisera la nomination LIST Books pour cette  
**\*\*route\*\*.**

-GET /volumes/{volumeId} : Récupère un livre à partir de son Identifiant.

Dans le reste du projet, on utilisera la nomination GETB ook pour cette  
**\*\*route\*\*.**

# BONUS

⚠ Pour utiliser GET Book, il faut récupérer l'ID présent dans la réponse de LIST

Books ⚠

Les paramètres permettant de filtrer les différents résultats de recherche

(nombre maximale de résultats, formats de téléchargement disponible ..etc)

sont détaillées sur la documentation de l'API.

GOOD LUCK

