🧪 Série d'exercices AJAX (avec Bootstrap, PHP backend, MySQL, sans rechargement)

Chaque exercice est accompagné de **questions** pour guider les stagiaires et les aider à comprendre la logique. Je fournis aussi le **message affiché côté frontend** que tu pourras personnaliser.

🎯 Objectif général

Manipuler une API backend PHP en utilisant uniquement **GET/POST**, avec retour de **données JSON**, en interagissant via :

- `XMLHttpRequest` (version 1)
- `jQuery AJAX` (version 2)

🛠 Backend `api.php` (structure)

Voici les **actions GET/POST disponibles** :

Action	Méthode	Paramètres	Résultat attendu
`get_all`	GET	—	Liste complète des produits
`get`	GET	`id`	Détails d'un produit
`add`	POST	name, price, category...	Ajout d'un produit
`update`	POST	id + autres champs	Mise à jour du produit
`delete`	POST	`id`	Suppression
`filter`	POST	name, price_min/max...	Filtrage multi-champs

📜 Liste des exercices

✅ Exercice 1 – Affichage de la liste complète

Objectif : Charger tous les produits au démarrage de la page via un appel asynchrone.

- **URL appelée** : `api.php?action=get_all`
- **Méthode** : GET
- **Frontend** :
  - Afficher chaque produit dans un tableau Bootstrap.
  - Ajouter un bouton "Modifier" et "Supprimer" sur chaque ligne.

🗨 Message affiché (success) :

```
"Liste des produits chargée avec succès."
```

❓ **Questions pédagogiques** :

- Quelle méthode utiliser pour récupérer les données sans recharger la page ?
- Comment afficher dynamiquement chaque ligne du tableau en JS ?

🧍 Exercice 2 – Récupération d'un produit

Objectif : Cliquer sur "Modifier" remplit automatiquement le formulaire avec les infos du produit.

- **URL appelée** : `api.php?action=get_one&id=XX`
- **Méthode** : GET
- **Frontend** :
  - Remplir dynamiquement le formulaire (à droite) avec les données.

🗨️ **Message affiché** :

```
"Produit ID XX chargé pour modification."
```

❓ **Questions pédagogiques** :

- Que doit contenir la requête GET pour cibler un produit ?
- Comment insérer dynamiquement les valeurs dans le formulaire ?

➕ Exercice 3 – Ajout d'un produit

Objectif : Soumettre un formulaire pour ajouter un produit via JSON ou FormData (au choix).

- **URL appelée** : `api.php?action=add`
- **Méthode** : POST
- **Données envoyées** : name, price, category, description

🗨️ **Message affiché** :

```
"Produit ajouté avec succès."
```

❓ **Questions pédagogiques** :

- Quelle est la différence entre FormData et JSON.stringify ?
- Comment ajouter la nouvelle ligne au tableau sans recharger la page ?

🖊️ Exercice 4 – Modification d'un produit

Objectif : Soumettre le formulaire déjà rempli pour mettre à jour un produit.

- **URL appelée** : `api.php?action=update`
- **Méthode** : POST
- **Données envoyées** : id, name, price, etc.

🗨️ Message affiché :

```
"Produit modifié avec succès."
```

❓ Questions pédagogiques :

- Pourquoi on utilise POST ici au lieu de PUT ?
- Comment remplacer la ligne dans le tableau sans recharger ?

❌ Exercice 5 – Suppression d'un produit

Objectif : Cliquer sur "Supprimer", afficher un `confirm()`, puis supprimer via POST.

- **URL appelée** : `api.php?action=delete`
- **Méthode** : POST
- **Paramètre** : `id`

🗨️ Message affiché :

```
"Produit supprimé avec succès."
```

❓ Questions pédagogiques :

- Que retourne `confirm()` en JS ?
- Comment supprimer dynamiquement la ligne du tableau ?

🔍 Exercice 6 (Bonus) – Filtrage dynamique

Objectif : Filtrer par nom, prix min, prix max, catégorie sans recharger la page.

- **URL appelée** : `api.php?action=filter`
- **Méthode** : POST
- **Paramètres** : name, min_price, max_price...

🗨️ Message affiché :

```
"Filtrage appliqué."
```

❓ **Questions pédagogiques** :

- Pourquoi POST même pour un filtre ?
- Comment reconstruire le tableau dynamiquement après chaque filtre ?

🧩 Interface HTML + Bootstrap

Je te fournirai :

- Un modèle de **page HTML Bootstrap divisée en deux colonnes**
- Un `table` Bootstrap
- Un `form` pour ajout/modification
- Un `form` pour le filtre

✅ Résumé des livrables

Je peux maintenant te préparer :

1. 🎨 Le **fichier HTML** avec Bootstrap et les ID bien nommés pour JS
2. 🐘 Le fichier `api.php` avec toutes les actions simulées
3. 🧠 Les **fonctions JS** version 1 (XMLHttpRequest)
4. ⚡ Les **fonctions JS** version 2 (jQuery AJAX)

Souhaites-tu que je commence par :

- Le **HTML Bootstrap prêt à l'emploi** ?
- Ou d'abord le **backend `api.php`** pour tester avec Postman ?