Jour 3 : Frontend - Affichage des produits

Objectif : Personnaliser le frontend pour afficher dynamiquement les produits importés dans Magento.

3.1. Personnalisation du thème pour afficher les produits Amazon

Nous allons :

1. Créer un design spécifique pour la **page d'accueil** ou une autre page de votre choix.
2. Ajouter dynamiquement les produits à travers des **fichiers XML** et un fichier **.phtml**.
3. Afficher les produits dans un **bloc ou widget** au sein du thème.

Étapes détaillées :

A. Structure du thème personnalisé

Avant de commencer, assurez-vous d'avoir un **thème Magento actif** sur lequel appliquer les personnalisations.

Emplacement des fichiers de votre thème :

```
app/design/frontend/{Vendor}/{Theme}/
```

Si vous utilisez le thème de base **Luma**, vous pouvez le copier et créer un **nouveau thème enfant** pour éviter de modifier directement le thème principal.

B. Modifier le layout XML pour ajouter un bloc de produits

Magento utilise des fichiers **XML de mise en page (layout XML)** pour structurer le contenu des pages. Le but est d'ajouter un bloc personnalisé pour afficher les produits liés à Amazon à l'aide de nos données.

1. Chemin du layout XML pour la page d'accueil

Créez ou modifiez le fichier suivant dans votre thème personnalisé :

```
app/design/frontend/{Vendor}/{Theme}/Magento_Theme/layout/cms_index_index.xml
```

2. Contenu du fichier `cms_index_index.xml`

Ce fichier XML permet de définir des blocs sur la page d'accueil. Ajoutez un bloc pour les produits importés :

```
<?xml version="1.0"?>
<page xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="urn:magento:framework:View/Layout/etc/
page_configuration.xsd">
   <body>
      <!-- Ajout d'un container pour afficher les produits Amazon -->
      <referenceContainer name="content">
         <block class="MyNamespace\AmazonIntegration\Block\ProductList"
name="amazon.product.list"
```

```
template="MyNamespace_AmazonIntegration::products.phtml" />
        </referenceContainer>
    </body>
</page>
```

Explications :

- `referenceContainer name="content"` : Ajoute le bloc dans la section principale de la page.
- `block class` : Référence à une classe PHP (Block) qui fournira les données des produits.
- `template` : Fichier `.phtml` que nous allons créer pour la mise en page.

C. La classe PHP du bloc

Le **bloc** est une classe qui permet de passer des données du backend (Magento) au fichier `.phtml`.

1. Créez la classe `ProductList` dans votre module :
   Chemin :
   `app/code/MyNamespace/AmazonIntegration/Block/ProductList.php`

2. Contenu de la classe `ProductList` :

```
<?php

namespace MyNamespace\AmazonIntegration\Block;

use Magento\Catalog\Api\ProductRepositoryInterface;
use Magento\Framework\View\Element\Template;

class ProductList extends Template
{
    private $productRepository;

    public function __construct(
        Template\Context $context,
        ProductRepositoryInterface $productRepository,
        array $data = []
    ) {
        parent::__construct($context, $data);
        $this->productRepository = $productRepository;
    }

    /**
     * Récupère une liste de produits Amazon
     *
     * @return array
     */
    public function getAmazonProducts()
    {
        $result = [];

        // Exemple : Récupération des produits liés à Amazon (marqués avec un SKU
spécifique comme "AMAZON-")
        $searchCriteria = $this->_objectManager->create(\Magento\Framework\Api\
SearchCriteriaBuilder::class)
            ->addFilter('sku', 'AMAZON-%', 'like')->create();
```

```
        $products = $this->productRepository->getList($searchCriteria);

        foreach ($products->getItems() as $product) {
            $result[] = [
                'name' => $product->getName(),
                'price' => $product->getPrice(),
                'image' => $this->getImageUrl($product),
                'url' => $product->getProductUrl(),
            ];
        }

        return $result;
    }

    /**
     * Récupère l'URL de l'image du produit
     *
     * @param \Magento\Catalog\Api\Data\ProductInterface $product
     * @return string|null
     */
    private function getImageUrl($product)
    {
        $imageHelper = $this->_objectManager->create(\Magento\Catalog\Helper\
Image::class);

        return $imageHelper->init($product, 'product_base_image')->getUrl();
    }
}
```

Explications :

- La méthode `getAmazonProducts` filtre les produits par SKU (les produits Amazon ayant un préfixe comme `AMAZON-`).
- Elle renvoie un tableau contenant les données nécessaires pour l'affichage : nom, prix, image, et URL.

D. Fichier PHTML pour afficher la liste des produits

Créez le fichier `products.phtml`, relié au template spécifié dans `cms_index_index.xml`.

Chemin du fichier :

```
app/design/frontend/{Vendor}/{Theme}/MyNamespace_AmazonIntegration/templates/
products.phtml
```

Contenu de `products.phtml` :

Voici un exemple de code HTML pour afficher les produits dans une grille dynamique :

```
<?php
/** @var \MyNamespace\AmazonIntegration\Block\ProductList $block */
$products = $block->getAmazonProducts();
?>

<div class="amazon-products">
```

```
    <h2>Produits Amazon</h2>
    <?php if (!empty($products)): ?>
        <div class="products-grid">
            <?php foreach ($products as $product): ?>
                <div class="product-item">
                    <a href="<?= $product['url'] ?>">
                        <img src="<?= $product['image'] ?>" alt="<?= $product['name'] ?>" />
                        <h3><?= $product['name'] ?></h3>
                        <p>Prix : <?= $product['price'] ?> €</p>
                    </a>
                </div>
            <?php endforeach; ?>
        </div>
    <?php else: ?>
        <p>Aucun produit Amazon trouvé.</p>
    <?php endif; ?>
</div>
```

CSS suggéré pour styliser la grille de produits :

Ajoutez le fichier CSS suivant à votre thème :
Chemin :
`app/design/frontend/{Vendor}/{Theme}/web/css/source/_module.less`

Contenu CSS :

```
.amazon-products {
    .products-grid {
        display: flex;
        flex-wrap: wrap;
        gap: 20px;
    }
    .product-item {
        border: 1px solid #ddd;
        padding: 10px;
        max-width: 200px;
        text-align: center;
    }
    .product-item img {
        max-width: 100%;
        height: auto;
    }
    .product-item h3 {
        font-size: 1.1rem;
        margin: 10px 0;
    }
    .product-item p {
        color: #555;
    }
}
```

Compilez les styles avec :

```
php bin/magento setup:static-content:deploy
```

E. Ajouter les produits Amazon dans des blocs ou widgets

1. **Option avec Widgets Magento** :
   Vous pouvez également associer votre bloc avec les produits Amazon en créant un **widget Magento**. Cela permet à l'administrateur de positionner ce bloc directement via l'interface backend.

2. **Blocs supplémentaires** :
   Si les produits Amazon doivent être affichés sur plusieurs pages, pensez à inclure le bloc dans des fichiers XML de mise en page personnalisés pour d'autres pages comme la page catégorie ou produit.

Test et résultat

Après avoir ajouté les modifications :

1. Videz les caches :

```
php bin/magento cache:flush
```

2 Rechargez la page d'accueil.

Vous devriez voir un **bloc de produits importés d'Amazon** affiché dynamiquement avec le design défini dans le fichier `.phtml`.

Conclusion

Dans cette partie, nous avons appris à :

1. Ajouter un **bloc personnalisé** sur la page d'accueil via un fichier XML.
2. Récupérer dynamiquement les produits Amazon via une classe PHP, en utilisant les **Service Contracts**.
3. Afficher ces produits dans une mise en page stylisée et responsive à l'aide d'un fichier `.phtml` et un peu de CSS.

3.2. Personnalisation d’une page produit dédiée pour les produits Amazon

Objectifs :

1. Créer une vue personnalisée pour les produits Amazon.
2. Ajouter des informations spécifiques (telles que la disponibilité, les notes) provenant de l’API Amazon.
3. Modifier les templates `.phtml` et utiliser des helpers Magento pour afficher les données sur les pages des produits.

Étapes détaillées :

A. Création d'une vue produit pour les produits Amazon

Magento vous permet de cibler les types de produits spécifiques ou des critères particuliers (comme le SKU ou des attributs personnalisés) pour personnaliser les pages produit.

1. Identifier les produits Amazon

Pour différencier les produits Amazon, on peut utiliser :

- Un **préfixe spécifique dans le SKU**.
- Un **attribut personnalisé** (par ex. : `is_amazon_product`) défini dans Magento pour indiquer que le produit provient de l’API Amazon.

Si cet attribut n'existe pas encore, vous pouvez le créer dans l’administration de Magento sous **Stores > Attributes > Product**, ou via un script PHP.

---

2. Modifier le fichier XML de mise en page des pages produit

Pour afficher les informations spécifiques aux produits Amazon, ajoutez un bloc dédié dans la page produit, uniquement pour les produits identifiés comme Amazon.

Chemin du fichier de layout XML pour la page produit :

```
app/design/frontend/{Vendor}/{Theme}/Magento_Catalog/layout/
catalog_product_view.xml
```

Contenu du fichier `catalog_product_view.xml` :

```
<?xml version="1.0"?>
<page xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="urn:magento:framework:View/Layout/etc/
page_configuration.xsd">
    <body>
        <!-- Ajouter un bloc dans la page produit -->
        <referenceBlock name="product.info.main">
            <!-- Si le produit est identifié comme Amazon, afficher un template de données
spécifiques -->
```

```
        <block class="MyNamespace\AmazonIntegration\Block\ProductView"
name="amazon.product.info" before="-"
template="MyNamespace_AmazonIntegration::product/amazon_info.phtml">
            <!-- Transmet un paramètre supplémentaire au block -->
            <arguments>
                <argument name="is_amazon_product"
xsi:type="boolean">true</argument>
            </arguments>
        </block>
    </referenceBlock>
  </body>
</page>
```

Explications :

- On insère un bloc personnalisé dans la section principale du contenu produit (`product.info.main`).
- **Condition spécifique :** On peut contrôler l'affichage dans le fichier PHTML (`amazon_info.phtml`) selon un attribut du produit, comme `is_amazon_product`.

B. Ajouter une classe PHP pour les données spécifiques

Créez une nouvelle classe `ProductView` pour gérer les données spécifiques des produits Amazon.

Chemin de la classe PHP :

```
app/code/MyNamespace/AmazonIntegration/Block/ProductView.php
```

Contenu de `ProductView.php` :

```php
<?php

namespace MyNamespace\AmazonIntegration\Block;

use Magento\Catalog\Api\ProductRepositoryInterface;
use Magento\Framework\Registry;
use Magento\Framework\View\Element\Template;

class ProductView extends Template
{
    private $productRepository;
    private $registry;

    public function __construct(
        Template\Context $context,
        ProductRepositoryInterface $productRepository,
        Registry $registry,
        array $data = []
    ) {
        parent::__construct($context, $data);
        $this->productRepository = $productRepository;
        $this->registry = $registry;
    }
```

```
    /**
     * Récupère le produit courant
     *
     * @return \Magento\Catalog\Api\Data\ProductInterface|null
     */
    public function getCurrentProduct()
    {
        return $this->registry->registry('current_product');
    }

    /**
     * Vérifie si le produit est un produit Amazon
     *
     * @return bool
     */
    public function isAmazonProduct()
    {
        $product = $this->getCurrentProduct();
        return $product && $product->getCustomAttribute('is_amazon_product') &&
$product->getCustomAttribute('is_amazon_product')->getValue();
    }

    /**
     * Récupère les informations spécifiques à Amazon
     *
     * @return array|null
     */
    public function getAmazonSpecificData()
    {
        if (!$this->isAmazonProduct()) {
            return null;
        }

        $product = $this->getCurrentProduct();

        return [
            'availability' => $product->getCustomAttribute('amazon_availability')-
>getValue() ?? 'Non spécifié',
            'rating' => $product->getCustomAttribute('amazon_rating')->getValue() ??
'Aucune note',
            'source' => $product->getCustomAttribute('amazon_source')->getValue() ??
'Amazon',
        ];
    }
}
```

Explications :

- `getCurrentProduct()` : Récupère le produit actuel sur la vue produit.
- `isAmazonProduct()` : Vérifie si le produit est un produit Amazon via l'attribut personnalisé `is_amazon_product`.
- `getAmazonSpecificData()` : Récupère des données spécifiques comme la disponibilité ou les notes.

C. Créer le fichier PHTML pour afficher les informations spécifiques

Créez un fichier PHTML dédié pour afficher les données spécifiques.

Chemin du fichier :

```
app/design/frontend/{Vendor}/{Theme}/MyNamespace_AmazonIntegration/templates/
product/amazon_info.phtml
```

Contenu du fichier PHTML :

```
<?php
/** @var \MyNamespace\AmazonIntegration\Block\ProductView $block */
$product = $block->getCurrentProduct();
$data = $block->getAmazonSpecificData();
?>

<?php if ($block->isAmazonProduct() && $data): ?>
<div class="amazon-product-info">
    <h3>Informations spécifiques Amazon</h3>
    <ul>
        <li><strong>Disponibilité :</strong> <?= $data['availability'] ?></li>
        <li><strong>Notes :</strong> <?= $data['rating'] ?></li>
        <li><strong>Source :</strong> <?= $data['source'] ?></li>
    </ul>
</div>
<?php endif; ?>
```

D. Modifier les styles CSS (optionnel)

Pour styliser le bloc où les informations spécifiques apparaissent :

Chemin du fichier LESS :

```
app/design/frontend/{Vendor}/{Theme}/web/css/source/_module.less

.amazon-product-info {
    background-color: #f9f9f9;
    padding: 15px;
    border: 1px solid #ddd;
    margin-top: 20px;

    h3 {
        font-size: 1.2rem;
        color: #555;
    }

    ul {
        list-style: none;
        padding: 0;

        li {
            margin: 10px 0;
            strong {
                color: #333;
            }
        }
    }
}
```

Compilez les styles avec :

```
php bin/magento setup:static-content:deploy
```

E. Vider le cache et vérifier

1. **Videz le cache :**

```
php bin/magento cache:flush
```

2. Rechargez la page produit dans le navigateur.

Le bloc affichant les informations spécifiques liées aux produits Amazon doit maintenant apparaître uniquement pour les produits identifiés.

Résultat attendu

1. Une section supplémentaire intitulée **Informations spécifiques Amazon** apparaît sur les pages des produits Amazon.
2. Cette section affiche des détails tels que :
    - Disponibilité, notes, et source de récupération.
3. Les produits non-Amazon n'afficheront pas ce bloc, respectant la condition définie.

Résumé

Dans cette étape, nous avons vu comment :

1. Personnaliser l'affichage des **pages produit** pour afficher des données spécifiques aux produits Amazon.
2. Modifier le layout XML pour ajouter un bloc dynamique conditionnel.
3. Créer une classe PHP pour connecter les données backend aux templates frontend.
4. Ajouter un fichier `.phtml` pour afficher les informations spécifiques, avec un design simple et stylisé.

3.3. Mise en place des filtres et recherches

Objectifs :

1. Utiliser les **facettes** natives de Magento pour permettre la recherche et le filtrage des produits importés d'Amazon.
2. Configurer les couches de navigation pour la recherche et les catégories afin d'ajouter des filtres basés sur vos besoins (prix, disponibilité, notes Amazon, etc.).

Étapes détaillées :

A. Utilisation des facettes natives Magento pour rechercher les produits importés

Magento propose un système de recherche avancée avec des filtres (facettes) qui peuvent être configurés pour explorer et rechercher les produits dans votre catalogue.

1. Identifier et configurer vos attributs pour les facettes

Pour inclure les produits Amazon dans la recherche et permettre leur filtrage :

1. Créez un **attribut personnalisé** (par exemple, `is_amazon_product`) s'il n'existe pas encore.
2. Ajoutez cet attribut aux **attributs de recherche**.

Étape pour ajouter un attribut dans la recherche :

- Allez dans **Stores > Attributes > Product** (dans l'Admin).
- Sélectionnez l'attribut (par exemple : `is_amazon_product`) ou créez-en un si nécessaire.
- Configurez les propriétés suivantes :
  - **Use in Search** : Oui
  - **Use in Layered Navigation** : Oui
  - **Use for Filterable (with results)** : Oui
  - **Comparable on Frontend** : Oui (si nécessaire)

Exemple d'attribut utile :

- `is_amazon_product` : Booléen (Oui/Non).
- `amazon_availability`: Liste déroulante (Disponible, Hors stock, etc.).
- `amazon_rating`: Décimal (avec des notes comme `4.5, 5.0, etc.` pour filtres par évaluation).

Une fois les attributs configurés, passez à la prochaine étape.

---

2. Inclure les produits Amazon dans les résultats de recherche

Magento utilise des indexes pour optimiser les résultats de recherche. Assurez-vous que les produits importés possèdent les attributs nécessaires.

1. Vérifiez que les produits Amazon ont un ou plusieurs attributs distinctifs (par exemple, un attribut `is_amazon_product`).
2. Mettez à jour l'indexation :

```
php bin/magento indexer:reindex
```

B. Configurer les couches de navigation pour les pages catégories et recherche

1. Activer Layered Navigation pour les catégories

Si vous souhaitez appliquer des filtres aux produits dans une page catégorie spécifique (et pas uniquement dans la recherche) :

1. Assurez-vous que vos **catégories sont ancrées** (anchored).
   - Allez dans **Catalog > Categories**.
   - Sélectionnez une catégorie parent ou enfant où les produits Amazon seront visibles.
   - Dans les paramètres, réglez **Is Anchor** sur **Oui**.

Cela permet d'activer les couches de navigation (layered navigation) pour cette catégorie.

---

2. Configurer les filtres de navigation pour les produits Amazon

Avec Magento, vous pouvez configurer quels attributs seront disponibles dans les couches de navigation (layered navigation).

Lister les attributs dans la navigation filtrée :

Dans les propriétés de chaque attribut (voir **Stores > Attributes > Product**) :

- **"Use in Layered Navigation"** : activez pour permettre leur utilisation dans les filtres.
- **"Use for Sorting in Product Listing"** : activez si vous souhaitez permettre un tri spécifique.

Exemple de filtres utiles :

1. **Disponibilité Amazon (amazon_availability)** : "Disponible" ou "Hors stock".
2. **Notes (amazon_rating)** : pour filtrer les produits selon leur évaluation (vous pouvez créer des plages fixes comme `4-5 étoiles`, `3-4 étoiles`, etc.).
3. **Type de produit (is_amazon_product)** : pour séparer les produits Amazon des autres.

Regénérer ces filtres :

```
php bin/magento cache:flush
```

C. Ajouter les facettes personnalisées sur la vue

Magento ajoute automatiquement les attributs activés comme filtres dans la vue catégories ou dans les résultats de recherche. Cela peut être utilisé pour spécifiquement permettre de filtrer les produits Amazon.

Si nécessaire, vous pouvez personnaliser les aspects de ces facettes via le fichier XML ou en modifiant les templates.

Modifier le layout XML des catégories et résultats de recherche :

Chemin des layouts :

- **Catégorie :**

```
app/design/frontend/{Vendor}/{Theme}/Magento_Catalog/layout/
catalog_category_view.xml
```

Recherche :

```
app/design/frontend/{Vendor}/{Theme}/Magento_Search/layout/
catalogsearch_result_index.xml
```

Exemple pour filtrer uniquement les produits Amazon :

```
<?xml version="1.0"?>
<page xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="urn:magento:framework:View/Layout/etc/
page_configuration.xsd">
   <body>
      <!-- Personnalisation des filtres pour la recherche -->
      <referenceBlock name="catalog.leftnav">
         <block class="Magento\LayeredNavigation\Block\Navigation"
name="amazon.filters" template="Magento_LayeredNavigation::filters.phtml">
            <arguments>
               <argument name="is_amazon_product" xsi:type="string">true</argument>
            </arguments>
         </block>
      </referenceBlock>
   </body>
</page>
```

Ce fragment replace le catalog.leftnav (menus de navigation filtrée) avec un but précis : l'ajout ou le ciblage des produits Amazon.

D. Mise en page frontend : Personnaliser les facettes

Vous pouvez personnaliser le template pour les facettes si vous souhaitez modifier leur rendu visuel ou leur ordre.

1. **Chemin du fichier de facettes :**

```
vendor/magento/module-layered-navigation/view/frontend/templates/layer/view.phtml
```

Copiez ce fichier dans votre thème :

```
app/design/frontend/{Vendor}/{Theme}/Magento_LayeredNavigation/templates/layer/
view.phtml
```

Personnalisez la disposition ou l'ordre des facettes affichées. Par exemple, affichez d'abord les produits Amazon uniquement :

```
<?php foreach ($block->getFilters() as $filter): ?>
   <?php if ($filter->getName() == 'is_amazon_product'): ?>
      <!-- Afficher le filtre Amazon en priorité -->
      <div class="filter-item">
         <?= $filter->getHtml() ?>
      </div>
```

```
    <?php endif; ?>
<?php endforeach; ?>
```

E. Tester les filtres et résultats

1. **Videz les caches et recompilez** :

```
php bin/magento cache:flush
php bin/magento setup:upgrade
php bin/magento indexer:reindex
```

Accédez à :

- Une catégorie configurée pour afficher les produits Amazon avec filtres.
- La page de recherche et testez les facettes disponibles.

Résultat attendu :

1. Les **produits Amazon importés** sont visibles dans les résultats de recherche et/ou dans les pages des catégories configurées.
2. Les utilisateurs peuvent filtrer facilement les produits Amazon en utilisant des **facettes adaptées** (disponibilité, notes, type de produit, etc.).
3. Les couches de navigation pour les catégories utilisent des **filtres spécifiques** définis pour vos produits importés.

Résumé

Dans cette étape, nous avons :

1. Activé les **facettes natives Magento** pour rechercher et filtrer les produits Amazon.
2. Configuré les filtres pour les couches de navigation des catégories.
3. Ajouté des personnalisations possibles via les layouts XML et les templates `.phtml`.

Demandes pertinentes à une IA pour la 3e journée

1. Approfondissement des tâches cron et automatisations

- *"Comment planifier une tâche cron Magento qui synchronise à des intervalles différents, par exemple toutes les heures pour les stocks mais quotidiennement pour d'autres attributs produits ?"*
- *"Comment gérer les priorités entre plusieurs tâches cron pour éviter des conflits ou surcharges (verrous ou files d'attente) ?"*
- *"Peux-tu me montrer un exemple de gestion des incidents dans mes tâches cron (comme des alertes en cas d'échec ou de délai d'exécution prolongé) ?"*
- *"Comment optimiser une importation incrémentale : extraire uniquement les produits modifiés ou ajoutés sur Amazon pour réduire le temps de traitement ?"*

---

2. Gestion avancée des produits et catégories

- *"Peux-tu me guider pour convertir un produit configurable Amazon (couleur et taille différentes) en produits configurables dans Magento ?"*
- *"Comment créer une relation entre produits attributaires (par exemple, styles, tailles) lors de l'import dans Magento ?"*
- *"Comment migrer des attributs spécifiques des produits Amazon (par exemple : `brand`, `delivery_time`, etc.) en attributs Magento personnalisés, et les mapper dynamiquement ?"*
- *"Peux-tu générer un script d'ajout massif d'attributs ou options manquantes pour permettre aux produits importés de correspondre à la structure Magento ?"*
- *"Comment gérer les redondances dans les catégories ou créer une hiérarchie propre ? Peux-tu automatiser le nettoyage des doublons importés depuis Amazon ?"*

---

3. Optimisation et gestion des données massives

- *"Quel est le meilleur moyen de gérer des fichiers volumineux de données Amazon, comme le format CSV ou JSON ?"*
- *"Peux-tu m'aider à configurer une pagination efficace pour les appels API Amazon et la synchronisation des données volumineuses ?"*
- *"Comment indexer correctement les données après une importation massive pour améliorer la navigation dans Magento ?"*
- *"Comment planifier des sauvegardes ou un rollback avant d'importer une quantité importante de données ?"*

---

4. Customisation et personnalisation Frontend

- *"Peux-tu me guider pour afficher dynamiquement des informations spécifiques récupérées d'Amazon, comme le statut du produit ou des badges ?"*
- *"Comment puis-je ajouter une section personnalisée sur une fiche produit Magento pour afficher des attributs exclusifs Amazon (par ex. : certifications, origine) ?"*

- *"Comment modifier les templates frontend pour afficher des produits configurables avec des données provenant exclusivement du flux Amazon ?"*
- *"Peux-tu générer un exemple de bloc/mise à jour via KnockoutJS pour afficher une liste des produits récemment importés ?"*

---

5. Gestion des erreurs et robustesse du code

- *"Peux-tu me guider pour mettre en œuvre une stratégie de retry automatique lors d'échecs d'appel API ?"*
- *"Montre-moi un exemple de code qui valide systématiquement la réponse de l'API avant de l'injecter dans Magento pour éviter des données corrompues."*
- *"Comment puis-je configurer des notifications ou alertes spécifiques en cas d'échec répété des synchronisations ou de détection d'incohérences dans les données?"*
- *"Peux-tu proposer une méthode pour enregistrer les erreurs d'importation dans une table dédiée pour les traiter ultérieurement ?"*

---

6. Améliorations des performances globales

- *"Comment puis-je configurer la mise en cache des résultats de l'API Amazon pour réduire les appels ?"*
- *"Peux-tu me montrer un exemple de script réparti (multi-threading ou via tâches en parallèle) pour accélérer les importations massives ?"*
- *"Comment réduire les temps de traitement dans Magento lors de l'enregistrement des produits ou des catégories en gros volumes ?"*
- *"Quels indexes spécifiques dois-je surveiller ou reconstruire pour garantir la fluidité après synchronisations récurrentes ?"*

---

7. Gestion des relations tierces

- *"Comment puis-je automatiser la mise à jour des stock/products dans Magento en cas de modification des données externes Amazon (webhooks, polling, etc.) ?"*
- *"Peux-tu m'aider à écrire une classe pour convertir des données tierces récupérées ailleurs (par ex. API logistique) dans un format compatible avec les produits Magento Amazon ?"*
- *"Comment gérer des intégrations multicanaux en veillant à ce que l'Amazon feed fonctionne en parallèle avec d'autres API (eBay, Shopify) ?"*

---

8. Extension UX pour administrateurs

- *"Comment ajouter de nouvelles options dans l'écran d'administration pour permettre une synchronisation manuelle ou semi-automatisée des catégories Amazon ?"*
- *"Peux-tu m'aider à configurer une grille personnalisée pour surveiller directement les résultats d'appels API ou modifications des produits Amazon depuis l'Admin Panel?"*
- *"Comment intégrer un tableau de bord personnalisé qui affiche des KPI (produits importés, synchronisations réussies, erreurs, etc.) pour l'administrateur Magento ?"*

- *"Peux-tu me guider pour ajouter une section qui permet aux administrateurs de filtrer les produits récemment synchronisés sur base de leurs attributs Amazon ?"*

---

9. Mises en place d’exports Amazon → Magento

- *"Peux-tu me guider pour configurer un export de produits Magento vers un listing au format Amazon depuis le backend ? Les données doivent correspondre au standard JSON Amazon."*
- *"Comment développer une logique pour exporter les transactions ou commandes Magento au format Amazon et monitorer leur transmission ?"*
- *"Montre-moi un exemple d’adaptation des nomenclatures Magento pour respecter le mapping des catégories Amazon."*

---

10. Tests approfondis et validation

- *"Peux-tu générer un test unit de validation d’importation qui compare chaque champ Amazon avec ses champs correspondants dans Magento ?"*
- *"Comment écrire un test d'intégration pour simuler un processus complet d’import et détecter des erreurs potentielles dans la chaîne ?"*
- *"Montre-moi une structure pour un test de validation des synchronisations cron Magento avec un fichier de réponse API fictif."*

Pourquoi ces demandes sont pertinentes ?

1. **Concentration sur la robustesse du projet :** À ce stade de la formation, l’accent sera mis sur les performances, les tests approfondis et l'amélioration du workflow (via cron, tâches parallèles, validations des données, etc.).
2. **Autonomie dans les tâches complexes :** Ces demandes aident les apprenants à automatiser et optimiser des processus récurrents et souvent fastidieux, tout en maintenant la qualité.
3. **Élargissement du périmètre fonctionnel :** L’apprentissage progresse en intégrant d'autres usages comme la création de produits configurables, l’export de données ou encore les KPI pour les administrateurs.
4. **Renforcement des best practices liées aux performances et à la sécurité :** L'emphase est mise sur la réutilisabilité, l’efficacité des scripts, et la gestion proactive des erreurs.