Cahier de Tests - Base de données MongoDB FakeStore

Table des matières

1. Introduction
2. Environnement de test
3. Tests sur la collection Products
4. Tests sur la collection Users
5. Tests sur la collection Carts
6. Tests sur la collection Categories

Introduction

Ce document présente les scénarios de tests pour valider les opérations CRUD (Create, Read, Update, Delete) sur la base de données MongoDB FakeStore hébergée sur MongoDB Atlas. Chaque opération comprend un scénario passant et deux scénarios non passants.

Environnement de test

- **Base de données** : MongoDB Atlas
- **Collections** : products, users, carts, categories
- **Framework de test** : Robot Framework
- **Bibliothèque MongoDB** : robotframework-mongodb-library

Tests sur la collection Products

1. CREATE - Création d'un produit

TC_PROD_CREATE_01 : Création réussie d'un produit (Passant)

Objectif : Vérifier qu'un produit peut être créé avec toutes les informations requises
Prérequis : Connexion à la base de données établie
Données de test :

```
{
  "title": "Test Product",
  "price": 29.99,
  "description": "This is a test product",
  "category": "electronics",
  "image": "https://test.com/image.jpg",
  "rating": {
    "rate": 4.5,
    "count": 10
  }
}
```

Étapes :

1. Se connecter à la base de données
2. Insérer le document dans la collection products
3. Vérifier que l'insertion retourne un ObjectId
4. Vérifier que le produit existe dans la base

Résultat attendu : Le produit est créé avec succès et un ObjectId est généré

TC_PROD_CREATE_02 : Création d'un produit sans champ obligatoire (Non passant)

Objectif : Vérifier que la création échoue sans le champ title
Prérequis : Connexion à la base de données établie
Données de test :

```
{
  "price": 29.99,
  "description": "Product without title",
  "category": "electronics"
}
```

Étapes :

1. Se connecter à la base de données
2. Tenter d'insérer le document sans le champ title
3. Vérifier que l'insertion échoue

Résultat attendu : L'insertion échoue avec une erreur de validation

TC_PROD_CREATE_03 : Création d'un produit avec un prix invalide (Non passant)

Objectif : Vérifier que la création échoue avec un prix négatif
Prérequis : Connexion à la base de données établie
Données de test :

```
{
  "title": "Invalid Price Product",
  "price": -50.00,
  "description": "Product with negative price",
  "category": "electronics"
}
```

Étapes :

1. Se connecter à la base de données
2. Tenter d'insérer le document avec un prix négatif
3. Vérifier que l'insertion échoue

Résultat attendu : L'insertion échoue avec une erreur de validation

2. READ - Lecture de produits

TC_PROD_READ_01 : Lecture réussie d'un produit par ID (Passant)

Objectif : Vérifier qu'un produit peut être lu par son ID
Prérequis : Un produit existe dans la base avec un ID connu
Étapes :

1. Se connecter à la base de données
2. Rechercher le produit par son ObjectId
3. Vérifier que le produit retourné contient tous les champs attendus

Résultat attendu : Le produit est retourné avec toutes ses informations

TC_PROD_READ_02 : Lecture d'un produit avec un ID inexistant (Non passant)

Objectif : Vérifier le comportement avec un ID inexistant
Prérequis : Connexion à la base de données établie
Étapes :

1. Se connecter à la base de données
2. Rechercher un produit avec un ObjectId invalide
3. Vérifier qu'aucun résultat n'est retourné

Résultat attendu : Aucun document n'est trouvé

TC_PROD_READ_03 : Lecture avec un format d'ID invalide (Non passant)

Objectif : Vérifier le comportement avec un format d'ID invalide
Prérequis : Connexion à la base de données établie
Étapes :

1. Se connecter à la base de données
2. Rechercher un produit avec un ID au format incorrect ("abc123")
3. Vérifier qu'une erreur est retournée

Résultat attendu : Une erreur de format est retournée

3. UPDATE - Mise à jour de produits

TC_PROD_UPDATE_01 : Mise à jour réussie du prix d'un produit (Passant)

Objectif : Vérifier qu'un produit peut être mis à jour
Prérequis : Un produit existe dans la base
Données de test :

```
{
  "price": 39.99,
  "rating.count": 150
}
```

Étapes :

1. Se connecter à la base de données
2. Mettre à jour le prix et le nombre d'évaluations
3. Vérifier que la mise à jour retourne le nombre de documents modifiés
4. Vérifier que les nouvelles valeurs sont enregistrées

Résultat attendu : Le produit est mis à jour avec succès

TC_PROD_UPDATE_02 : Mise à jour d'un produit inexistant (Non passant)

Objectif : Vérifier le comportement lors de la mise à jour d'un ID inexistant
Prérequis : Connexion à la base de données établie
Étapes :

1. Se connecter à la base de données
2. Tenter de mettre à jour un produit avec un ObjectId inexistant
3. Vérifier qu'aucun document n'est modifié

Résultat attendu : Aucun document n'est modifié (matchedCount = 0)

TC_PROD_UPDATE_03 : Mise à jour avec des données invalides (Non passant)

Objectif : Vérifier le rejet de données invalides
Prérequis : Un produit existe dans la base
Données de test :

```
{
  "price": "not a number",
  "rating.rate": 6.0
}
```

Étapes :

1. Se connecter à la base de données
2. Tenter de mettre à jour avec des données invalides
3. Vérifier que la mise à jour échoue

Résultat attendu : La mise à jour échoue avec une erreur de validation

4. DELETE - Suppression de produits

TC_PROD_DELETE_01 : Suppression réussie d'un produit (Passant)

Objectif : Vérifier qu'un produit peut être supprimé
Prérequis : Un produit existe dans la base
Étapes :

1. Se connecter à la base de données
2. Supprimer le produit par son ObjectId
3. Vérifier que la suppression retourne deletedCount = 1
4. Vérifier que le produit n'existe plus

Résultat attendu : Le produit est supprimé avec succès

TC_PROD_DELETE_02 : Suppression d'un produit inexistant (Non passant)

Objectif : Vérifier le comportement lors de la suppression d'un ID inexistant
Prérequis : Connexion à la base de données établie
Étapes :

1. Se connecter à la base de données
2. Tenter de supprimer un produit avec un ObjectId inexistant
3. Vérifier que deletedCount = 0

Résultat attendu : Aucun document n'est supprimé

TC_PROD_DELETE_03 : Suppression avec un ID invalide (Non passant)

Objectif : Vérifier le comportement avec un format d'ID invalide
Prérequis : Connexion à la base de données établie
Étapes :

1. Se connecter à la base de données
2. Tenter de supprimer avec un ID au format incorrect
3. Vérifier qu'une erreur est retournée

Résultat attendu : Une erreur de format est retournée

Tests sur la collection Users

1. CREATE - Création d'un utilisateur

TC_USER_CREATE_01 : Création réussie d'un utilisateur (Passant)

Objectif : Vérifier qu'un utilisateur peut être créé avec toutes les informations
Prérequis : Connexion à la base de données établie
Données de test :

```
{
  "email": "test@example.com",
  "username": "testuser",
  "password": "hashedPassword123",
  "name": {
    "firstname": "Test",
    "lastname": "User"
  },
  "address": {
    "city": "Test City",
    "street": "123 Test Street",
    "zipcode": "12345",
    "geolocation": {
      "lat": "40.7128",
      "long": "-74.0060"
    }
  },
  "phone": "1-555-555-5555"
}
```

Étapes :

1. Se connecter à la base de données
2. Insérer le document utilisateur
3. Vérifier que l'insertion retourne un ObjectId
4. Vérifier que l'utilisateur existe dans la base

Résultat attendu : L'utilisateur est créé avec succès

TC_USER_CREATE_02 : Création d'un utilisateur avec email dupliqué (Non passant)

Objectif : Vérifier que la création échoue avec un email existant
Prérequis : Un utilisateur existe déjà avec l'email "existing@example.com"
Données de test :

```
{
  "email": "existing@example.com",
  "username": "newuser",
  "password": "password123"
}
```

Étapes :

1. Se connecter à la base de données
2. Tenter d'insérer un utilisateur avec un email existant
3. Vérifier que l'insertion échoue

Résultat attendu : L'insertion échoue avec une erreur de duplication

TC_USER_CREATE_03 : Création d'un utilisateur sans email (Non passant)

Objectif : Vérifier que la création échoue sans email
Prérequis : Connexion à la base de données établie
Données de test :

```
{
  "username": "userwithoutemail",
  "password": "password123",
  "name": {
    "firstname": "No",
    "lastname": "Email"
  }
}
```

Étapes :

1. Se connecter à la base de données
2. Tenter d'insérer un utilisateur sans email
3. Vérifier que l'insertion échoue

Résultat attendu : L'insertion échoue avec une erreur de validation

2. READ - Lecture d'utilisateurs

TC_USER_READ_01 : Lecture réussie d'un utilisateur par username (Passant)

Objectif : Vérifier qu'un utilisateur peut être trouvé par son username
Prérequis : Un utilisateur existe avec le username "johnd"
Étapes :

1. Se connecter à la base de données
2. Rechercher l'utilisateur par username
3. Vérifier que l'utilisateur retourné contient tous les champs

Résultat attendu : L'utilisateur est retourné avec toutes ses informations

TC_USER_READ_02 : Lecture avec un username inexistant (Non passant)

Objectif : Vérifier le comportement avec un username inexistant
Prérequis : Connexion à la base de données établie
Étapes :

1. Se connecter à la base de données
2. Rechercher un utilisateur avec username "nonexistent"
3. Vérifier qu'aucun résultat n'est retourné

Résultat attendu : Aucun document n'est trouvé

TC_USER_READ_03 : Lecture avec filtre invalide (Non passant)

Objectif : Vérifier le comportement avec un filtre mal formé
Prérequis : Connexion à la base de données établie
Étapes :

1. Se connecter à la base de données
2. Rechercher avec un filtre invalide
3. Vérifier qu'une erreur est retournée

Résultat attendu : Une erreur de syntaxe est retournée

3. UPDATE - Mise à jour d'utilisateurs

TC_USER_UPDATE_01 : Mise à jour réussie de l'adresse (Passant)

Objectif : Vérifier qu'une adresse utilisateur peut être mise à jour
Prérequis : Un utilisateur existe dans la base
Données de test :

```
{
  "address.city": "New City",
  "address.street": "456 New Street",
  "address.zipcode": "54321"
}
```

Étapes :

1. Se connecter à la base de données
2. Mettre à jour l'adresse de l'utilisateur
3. Vérifier que la mise à jour retourne matchedCount = 1
4. Vérifier que les nouvelles valeurs sont enregistrées

Résultat attendu : L'adresse est mise à jour avec succès

TC_USER_UPDATE_02 : Mise à jour d'un email avec valeur dupliquée (Non passant)

Objectif : Vérifier le rejet d'un email déjà utilisé
Prérequis : Deux utilisateurs existent dans la base
Étapes :

1. Se connecter à la base de données
2. Tenter de mettre à jour l'email avec une valeur existante
3. Vérifier que la mise à jour échoue

Résultat attendu : La mise à jour échoue avec une erreur de duplication

TC_USER_UPDATE_03 : Mise à jour avec des coordonnées invalides (Non passant)

Objectif : Vérifier le rejet de coordonnées GPS invalides
Prérequis : Un utilisateur existe dans la base
Données de test :

```
{
  "address.geolocation.lat": "invalid",
  "address.geolocation.long": "999.999"
}
```

Étapes :

1. Se connecter à la base de données
2. Tenter de mettre à jour avec des coordonnées invalides
3. Vérifier que la mise à jour échoue

Résultat attendu : La mise à jour échoue avec une erreur de validation

4. DELETE - Suppression d'utilisateurs

TC_USER_DELETE_01 : Suppression réussie d'un utilisateur (Passant)

Objectif : Vérifier qu'un utilisateur peut être supprimé
Prérequis : Un utilisateur existe dans la base
Étapes :

1. Se connecter à la base de données
2. Supprimer l'utilisateur par son ObjectId
3. Vérifier que deletedCount = 1
4. Vérifier que l'utilisateur n'existe plus

Résultat attendu : L'utilisateur est supprimé avec succès

TC_USER_DELETE_02 : Suppression d'un utilisateur avec commandes actives (Non passant)

Objectif : Vérifier la protection des utilisateurs avec commandes
Prérequis : Un utilisateur existe avec des commandes dans la collection carts
Étapes :

1. Se connecter à la base de données
2. Tenter de supprimer l'utilisateur
3. Vérifier que la suppression est bloquée

Résultat attendu : La suppression échoue avec une erreur de contrainte

TC_USER_DELETE_03 : Suppression avec critère invalide (Non passant)

Objectif : Vérifier le comportement avec un critère invalide
Prérequis : Connexion à la base de données établie
Étapes :

1. Se connecter à la base de données
2. Tenter de supprimer avec un critère mal formé
3. Vérifier qu'une erreur est retournée

Résultat attendu : Une erreur de syntaxe est retournée

Tests sur la collection Carts

1. CREATE - Création d'un panier

TC_CART_CREATE_01 : Création réussie d'un panier (Passant)

Objectif : Vérifier qu'un panier peut être créé

Prérequis : Un utilisateur et des produits existent dans la base
Données de test :

```
{
  "userId": "ObjectId_utilisateur_existant",
  "date": "2024-01-15T10:00:00.000Z",
  "products": [
    {
      "productId": "ObjectId_produit_1",
      "quantity": 2
    },
    {
      "productId": "ObjectId_produit_2",
      "quantity": 1
    }
  ]
}
```

Étapes :

1. Se connecter à la base de données
2. Insérer le document panier
3. Vérifier que l'insertion retourne un ObjectId
4. Vérifier que le panier existe dans la base

Résultat attendu : Le panier est créé avec succès

TC_CART_CREATE_02 : Création d'un panier sans userId (Non passant)

Objectif : Vérifier que la création échoue sans userId
Prérequis : Connexion à la base de données établie
Données de test :

```
{
  "date": "2024-01-15T10:00:00.000Z",
  "products": [
    {
      "productId": "ObjectId_produit",
      "quantity": 1
    }
  ]
}
```

Étapes :

1. Se connecter à la base de données
2. Tenter d'insérer un panier sans userId
3. Vérifier que l'insertion échoue

Résultat attendu : L'insertion échoue avec une erreur de validation

TC_CART_CREATE_03 : Création d'un panier avec quantité négative (Non passant)

Objectif : Vérifier le rejet de quantités négatives
Prérequis : Connexion à la base de données établie
Données de test :

```
{
  "userId": "ObjectId_utilisateur",
  "date": "2024-01-15T10:00:00.000Z",
  "products": [
    {
      "productId": "ObjectId_produit",
      "quantity": -5
    }
  ]
}
```

Étapes :

1. Se connecter à la base de données
2. Tenter d'insérer un panier avec quantité négative

3. Vérifier que l'insertion échoue

Résultat attendu : L'insertion échoue avec une erreur de validation

2. READ - Lecture de paniers

TC_CART_READ_01 : Lecture réussie des paniers d'un utilisateur (Passant)

Objectif : Vérifier la récupération des paniers par userId
Prérequis : Des paniers existent pour un utilisateur
Étapes :

1. Se connecter à la base de données
2. Rechercher tous les paniers d'un userId
3. Vérifier que les paniers sont retournés
4. Vérifier la structure des documents

Résultat attendu : Les paniers de l'utilisateur sont retournés

TC_CART_READ_02 : Lecture de paniers pour utilisateur inexistant (Non passant)

Objectif : Vérifier le comportement avec un userId inexistant
Prérequis : Connexion à la base de données établie
Étapes :

1. Se connecter à la base de données
2. Rechercher les paniers d'un userId inexistant
3. Vérifier qu'aucun résultat n'est retourné

Résultat attendu : Aucun document n'est trouvé

TC_CART_READ_03 : Lecture avec projection invalide (Non passant)

Objectif : Vérifier le comportement avec une projection mal formée
Prérequis : Connexion à la base de données établie
Étapes :

1. Se connecter à la base de données
2. Tenter une lecture avec projection invalide
3. Vérifier qu'une erreur est retournée

Résultat attendu : Une erreur de syntaxe est retournée

3. UPDATE - Mise à jour de paniers

TC_CART_UPDATE_01 : Ajout réussi d'un produit au panier (Passant)

Objectif : Vérifier l'ajout d'un produit à un panier existant
Prérequis : Un panier existe dans la base
Données de test :

```
{
  "$push": {
    "products": {
      "productId": "ObjectId_nouveau_produit",
      "quantity": 3
    }
  }
}
```

Étapes :

1. Se connecter à la base de données
2. Ajouter le nouveau produit au panier
3. Vérifier que la mise à jour retourne modifiedCount = 1
4. Vérifier que le produit est ajouté

Résultat attendu : Le produit est ajouté au panier

TC_CART_UPDATE_02 : Mise à jour d'un panier inexistant (Non passant)

Objectif : Vérifier le comportement avec un panier inexistant
Prérequis : Connexion à la base de données établie
Étapes :

1. Se connecter à la base de données
2. Tenter de mettre à jour un panier inexistant
3. Vérifier que matchedCount = 0

Résultat attendu : Aucun document n'est modifié

TC_CART_UPDATE_03 : Mise à jour avec productId invalide (Non passant)

Objectif : Vérifier le rejet d'un productId invalide
Prérequis : Un panier existe dans la base
Données de test :

```
{
  "$push": {
    "products": {
      "productId": "invalid_id_format",
      "quantity": 1
    }
  }
}
```

Étapes :

1. Se connecter à la base de données
2. Tenter d'ajouter un produit avec ID invalide
3. Vérifier que la mise à jour échoue

Résultat attendu : La mise à jour échoue avec une erreur

4. DELETE - Suppression de paniers

TC_CART_DELETE_01 : Suppression réussie d'un panier (Passant)

Objectif : Vérifier qu'un panier peut être supprimé
Prérequis : Un panier existe dans la base
Étapes :

1. Se connecter à la base de données
2. Supprimer le panier par son ObjectId
3. Vérifier que deletedCount = 1
4. Vérifier que le panier n'existe plus

Résultat attendu : Le panier est supprimé avec succès

TC_CART_DELETE_02 : Suppression multiple de paniers anciens (Non passant)

Objectif : Vérifier la tentative de suppression sans critère
Prérequis : Plusieurs paniers existent dans la base
Étapes :

1. Se connecter à la base de données
2. Tenter de supprimer sans filtre (deleteMany({}))
3. Vérifier que l'opération est bloquée

Résultat attendu : L'opération échoue pour sécurité

TC_CART_DELETE_03 : Suppression avec filtre de date invalide (Non passant)

Objectif : Vérifier le comportement avec un format de date invalide
Prérequis : Connexion à la base de données établie
Étapes :

1. Se connecter à la base de données
2. Tenter de supprimer avec date au format incorrect
3. Vérifier qu'une erreur est retournée

Résultat attendu : Une erreur de format est retournée

Tests sur la collection Categories

1. CREATE - Création d'une catégorie

TC_CAT_CREATE_01 : Création réussie d'une catégorie (Passant)

Objectif : Vérifier qu'une catégorie peut être créée
Prérequis : Connexion à la base de données établie
Données de test :

```
{
  "name": "home & garden",
  "description": "Articles pour la maison et le jardin",
  "image": "https://example.com/category/home-garden.jpg"
}
```

Étapes :

1. Se connecter à la base de données
2. Insérer la nouvelle catégorie
3. Vérifier que l'insertion retourne un ObjectId
4. Vérifier que la catégorie existe

Résultat attendu : La catégorie est créée avec succès

TC_CAT_CREATE_02 : Création d'une catégorie avec nom dupliqué (Non passant)

Objectif : Vérifier le rejet d'un nom de catégorie existant
Prérequis : Une catégorie "electronics" existe déjà
Données de test :

```
{
  "name": "electronics",
  "description": "Nouvelle description",
  "image": "https://example.com/new-image.jpg"
}
```

Étapes :

1. Se connecter à la base de données
2. Tenter d'insérer une catégorie avec nom existant
3. Vérifier que l'insertion échoue

Résultat attendu : L'insertion échoue avec erreur de duplication

TC_CAT_CREATE_03 : Création d'une catégorie sans nom (Non passant)

Objectif : Vérifier que le nom est obligatoire
Prérequis : Connexion à la base de données établie
Données de test :

```
{
  "description": "Catégorie sans nom",
  "image": "https://example.com/image.jpg"
}
```

Étapes :

1. Se connecter à la base de données
2. Tenter d'insérer sans le champ name
3. Vérifier que l'insertion échoue

Résultat attendu : L'insertion échoue avec erreur de validation

2. READ - Lecture de catégories

TC_CAT_READ_01 : Lecture réussie de toutes les catégories (Passant)

Objectif : Vérifier la récupération de toutes les catégories
Prérequis : Plusieurs catégories existent dans la base
Étapes :

1. Se connecter à la base de données
2. Récupérer toutes les catégories
3. Vérifier que les catégories sont retournées
4. Vérifier la structure des documents

Résultat attendu : Toutes les catégories sont retournées

TC_CAT_READ_02 : Lecture d'une catégorie par nom inexistant (Non passant)

Objectif : Vérifier le comportement avec un nom inexistant
Prérequis : Connexion à la base de données établie
Étapes :

1. Se connecter à la base de données

2. Rechercher une catégorie "nonexistent"
3. Vérifier qu'aucun résultat n'est retourné

Résultat attendu : Aucun document n'est trouvé

TC_CAT_READ_03 : Lecture avec regex invalide (Non passant)

Objectif : Vérifier le comportement avec une regex mal formée
Prérequis : Connexion à la base de données établie
Étapes :

1. Se connecter à la base de données
2. Rechercher avec une regex invalide
3. Vérifier qu'une erreur est retournée

Résultat attendu : Une erreur de syntaxe est retournée

3. UPDATE - Mise à jour de catégories

TC_CAT_UPDATE_01 : Mise à jour réussie de la description (Passant)

Objectif : Vérifier la mise à jour d'une catégorie
Prérequis : Une catégorie existe dans la base
Données de test :

```
{
  "description": "Description mise à jour",
  "image": "https://example.com/updated-image.jpg"
}
```

Étapes :

1. Se connecter à la base de données
2. Mettre à jour la description et l'image
3. Vérifier que modifiedCount = 1
4. Vérifier les nouvelles valeurs

Résultat attendu : La catégorie est mise à jour

TC_CAT_UPDATE_02 : Tentative de modification du nom (Non passant)

Objectif : Vérifier que le nom ne peut pas être modifié
Prérequis : Une catégorie existe dans la base
Données de test :

```
{
  "name": "nouveau_nom",
  "description": "Description mise à jour"
}
```

Étapes :

1. Se connecter à la base de données
2. Tenter de modifier le nom de la catégorie
3. Vérifier que le nom n'a pas changé

Résultat attendu : Le nom de la catégorie reste inchangé

TC_CAT_UPDATE_03 : Mise à jour d'une catégorie inexistante (Non passant)

Objectif : Vérifier le comportement avec une catégorie inexistante
Prérequis : Connexion à la base de données établie
Étapes :

1. Se connecter à la base de données
2. Tenter de mettre à jour une catégorie avec un ObjectId inexistant
3. Vérifier que matchedCount = 0

Résultat attendu : Aucun document n'est modifié

4. DELETE - Suppression de catégories

TC_CAT_DELETE_01 : Suppression réussie d'une catégorie (Passant)

Objectif : Vérifier qu'une catégorie peut être supprimée
Prérequis : Une catégorie existe dans la base
Étapes :

1. Se connecter à la base de données
2. Supprimer la catégorie par son ObjectId
3. Vérifier que deletedCount = 1
4. Vérifier que la catégorie n'existe plus

Résultat attendu : La catégorie est supprimée avec succès

TC_CAT_DELETE_02 : Suppression d'une catégorie avec produits associés (Non passant)

Objectif : Vérifier la protection des catégories avec produits
Prérequis : Une catégorie existe avec des produits associés
Étapes :

1. Se connecter à la base de données
2. Vérifier qu'il existe des produits dans cette catégorie
3. Tenter de supprimer la catégorie
4. Vérifier le comportement de suppression

Résultat attendu : La suppression est gérée selon les contraintes d'intégrité

TC_CAT_DELETE_03 : Suppression d'une catégorie inexistante (Non passant)

Objectif : Vérifier le comportement avec une catégorie inexistante
Prérequis : Connexion à la base de données établie
Étapes :

1. Se connecter à la base de données
2. Tenter de supprimer une catégorie avec un ObjectId inexistant
3. Vérifier que deletedCount = 0

Résultat attendu : Aucun document n'est supprimé

Résumé des tests

Statistiques par collection

Collection	CREATE	READ	UPDATE	DELETE	Total
Products	3	3	3	3	12
Users	3	3	3	3	12
Carts	3	3	3	3	12
Categories	3	3	3	3	12

Total des scénarios

- **Scénarios passants** : 16 (4 par collection)
- **Scénarios non passants** : 32 (8 par collection)
- **Total général** : 48 scénarios de test

Couverture des tests

Les tests couvrent :

- Validation des données d'entrée
- Gestion des erreurs et exceptions
- Contraintes d'intégrité référentielle
- Formats de données invalides
- Opérations sur des ressources inexistantes
- Duplication de données uniques
- Validation des types de données

Recommandations

1. Exécuter les tests dans un environnement isolé
2. Nettoyer les données de test après chaque exécution
3. Vérifier les logs en cas d'échec
4. Maintenir à jour les données de test avec l'évolution du schéma

---

Annexes

A. Configuration MongoDB Atlas

- Créer un cluster MongoDB Atlas gratuit
- Configurer les règles d'accès réseau (IP Whitelist)
- Créer un utilisateur avec les permissions appropriées
- Obtenir la chaîne de connexion (Connection String)

B. Installation de Robot Framework

```
pip install robotframework
pip install robotframework-mongodbLibrary
```

C. Structure des documents MongoDB

Product

```
{
  "_id": ObjectId(),
  "title": String,
  "price": Number,
  "description": String,
  "category": String,
  "image": String,
  "rating": {
    "rate": Number,
    "count": Number
  }
}
```

User

```
{
  "_id": ObjectId(),
  "email": String,
  "username": String,
  "password": String,
  "name": {
    "firstname": String,
    "lastname": String
  },
  "address": {
    "city": String,
    "street": String,
    "zipcode": String,
    "geolocation": {
      "lat": String,
      "long": String
    }
  },
  "phone": String
}
```

Cart

```
{
  "_id": ObjectId(),
  "userId": ObjectId(),
  "date": Date,
  "products": [
    {
      "productId": ObjectId(),
      "quantity": Number
    }
  ]
}
```

Category

```
{
  "_id": ObjectId(),
  "name": String,
  "description": String,
  "image": String
}
```

Document créé le : 15 Janvier 2024
Version : 1.0
Auteur : Équipe QA
Révision : Mme SAMB