Contrat d'API Complet - Marketplace Microservices

Contexte et Conventions Générales

Architecture : API Gateway unique + microservices indépendants (users:3001, catalog:3002, cart:3004, orders:3003, payments:3005)
Format : JSON bodies uniquement
Authentification : JWT via `Authorization: Bearer <token>`, Gateway ajoute `X-User-Id`, `X-User-Role`
Rôles : `client`, `vendeur`, `admin`
Pagination : `?page=0&limit=20&sort=price:desc`
Erreurs : `{ "error": "message", "code": "ERR_INVALID_INPUT" }`
Base URL externe : `http://localhost:3000/api`

---

1. API Externe (Front-end → API Gateway)

1.1 Authentification

POST /auth/register
Description : Création de compte utilisateur
Body : `{ "username": string (required, unique, 3-30 chars), "password": string (required, min 8 chars), "email": string (required, valid email), "role": string (optional, default "client") }`
Response : `201` - `{ "userId": "uuid", "username": "string", "email": "string", "role": "string" }`
Erreurs : `400` (validation), `409` (username/email exists)
Autorisation : Aucune (public)

POST /auth/login
Description : Authentification et génération JWT
Body : `{ "username": string (required), "password": string (required) }`
Response : `200` - `{ "token": "string (JWT)", "userId": "uuid", "role": "string", "expiresIn": 86400 }`
Erreurs : `401` (invalid credentials)
Autorisation : Aucune (public)

GET /auth/me
Description : Profil utilisateur connecté
Body : none
Response : `200` - `{ "userId": "uuid", "username": "string", "email": "string", "role": "string", "createdAt": "ISO date" }`
Erreurs : `401` (no token)
Autorisation : Token requis

POST /auth/logout
Description : Invalidation session (blacklist token)
Body : none
Response : `204` No Content
Erreurs : `401` (no token)
Autorisation : Token requis

1.2 Utilisateurs (Routé → users-service:3001)

GET /users/:userId
Description : Consulter profil utilisateur
Params : `userId` (uuid, required)
Body : none
Response : `200` - `{ "userId": "uuid", "username": "string", "email": "string", "role": "string", "createdAt": "ISO date", "stats": { "ordersCount": number, "productsCount": number } }`
Erreurs : `403` (access denied), `404` (not found)
Autorisation : Token requis (client: seulement son compte, admin/vendeur: tous)

PUT /users/:userId
Description : Modifier profil utilisateur
Params : `userId` (uuid, required)
Body : `{ "username": string (optional), "email": string (optional), "password": string (optional, min 8) }`
Response : `200` - `{ "userId": "uuid", "username": "string", "email": "string", "role": "string" }`
Erreurs : `400` (validation), `403` (not owner or admin), `409` (conflict)
Autorisation : Token requis (propriétaire ou admin)

PUT /users/:userId/role
Description : Changer rôle utilisateur
Params : `userId` (uuid, required)
Body : `{ "role": string (required, "client"|"vendeur"|"admin") }`
Response : `200` - `{ "userId": "uuid", "role": "string" }`
Erreurs : `400` (invalid role), `403` (not admin), `404` (not found)
Autorisation : Token requis, rôle admin

DELETE /users/:userId
Description : Désactivation compte (soft delete)
Params : `userId` (uuid, required)
Body : none
Response : `204` No Content
Erreurs : `403` (not owner or admin), `404` (not found)
Autorisation : Token requis (propriétaire ou admin)

1.3 Catalogue Produits (Routé → catalog-service:3002)

POST /products
Description : Créer nouveau produit

Body : `{ "name": string (required, 1-200 chars), "description": string (optional, max 2000), "price": number (required, >0), "currency": string (optional, default "EUR"), "stock": number (required, >=0), "categoryId": "uuid" (required), "images": array<string> (optional), "attributes": object (optional) }`
Response : `201` - `{ "productId": "uuid", "name": "string", "description": "string", "price": number, "currency": "string", "stock": number, "categoryId": "uuid", "sellerId": "uuid", "createdAt": "ISO date", "images": array, "attributes": object }`
Erreurs : `400` (validation), `403` (not vendeur/admin), `404` (category not found)
Autorisation : Token requis, rôle vendeur ou admin

GET /products
Description : Recherche produits paginée
Query : `q=string` (search), `categoryId=uuid`, `minPrice=number`, `maxPrice=number`, `sellerId=uuid`, `available=true`, `page=0`, `limit=20`, `sort=price:asc|desc|newest`
Body : none
Response : `200` - `{ "products": array[{ "productId": "uuid", "name": "string", "price": number, "currency": "string", "stock": number, "images": array, "seller": { "userId": "uuid", "username": "string" } }], "total": number, "page": number, "limit": number }`
Erreurs : `400` (invalid filters)
Autorisation : Aucune (public)

GET /products/:productId
Description : Détails produit complet
Params : `productId` (uuid, required)
Body : none
Response : `200` - `{ "productId": "uuid", "name": "string", "description": "string", "price": number, "currency": "string", "stock": number, "categoryId": "uuid", "sellerId": "uuid", "seller": { "username": "string" }, "images": array, "attributes": object, "createdAt": "ISO date", "avgRating": number, "reviewsCount": number }`
Erreurs : `404` (not found)
Autorisation : Aucune (public)

PUT /products/:productId
Description : Modifier produit (propriétaire uniquement)
Params : `productId` (uuid, required)
Body : `{ "name": string (optional), "description": string (optional), "price": number (optional), "currency": string (optional), "stock": number (optional), "categoryId": "uuid" (optional), "images": array (optional), "attributes": object (optional) }`
Response : `200` - Produit complet mis à jour
Erreurs : `400` (validation), `403` (not owner or admin), `404` (not found)
Autorisation : Token requis, vendeur propriétaire ou admin

DELETE /products/:productId
Description : Désactivation produit (soft delete)
Params : `productId` (uuid, required)
Body : none
Response : `204` No Content

Erreurs : `403` (not owner or admin), `404` (not found)
Autorisation : Token requis, vendeur propriétaire ou admin

POST /products/:productId/images
Description : Ajouter images produit
Params : `productId` (uuid, required)
Body : `{ "images": array<string> (required, URLs) }`
Response : `200` - `{ "productId": "uuid", "images": array }`
Erreurs : `400` (validation), `403`, `404`
Autorisation : Token requis, vendeur propriétaire ou admin

1.4 Catégories Produits

GET /categories
Description : Liste hiérarchique catégories
Query : `parentId=uuid`, `page=0`, `limit=50`
Body : none
Response : `200` - `{ "categories": array[{ "categoryId": "uuid", "name": "string", "parentId": "uuid|null", "childrenCount": number }], "total": number }`
Erreurs : `400` (invalid filters)
Autorisation : Aucune (public)

GET /categories/:categoryId
Description : Détails catégorie + produits
Params : `categoryId` (uuid, required)
Query : `page=0`, `limit=20`
Body : none
Response : `200` - `{ "categoryId": "uuid", "name": "string", "parentId": "uuid|null", "description": "string", "products": array, "totalProducts": number }`
Erreurs : `404` (not found)
Autorisation : Aucune (public)

1.5 Panier (Routé → cart-service:3004)

GET /cart
Description : Panier utilisateur courant
Body : none
Response : `200` - `{ "items": array[{ "productId": "uuid", "name": "string", "price": number, "currency": "string", "quantity": number, "total": number }], "totalItems": number, "totalAmount": number, "currency": "string" }`
Erreurs : `401` (no token)
Autorisation : Token requis

POST /cart/items
Description : Ajouter/modifier article panier
Body : `{ "productId": "uuid" (required), "quantity": number (required, 1-99) }`
Response : `200` - Panier complet mis à jour

Erreurs : `400` (invalid quantity), `401`, `404` (product not found/available)
Autorisation : Token requis

PUT /cart/items/:productId
Description : Modifier quantité article
Params : `productId` (uuid, required)
Body : `{ "quantity": number (required, 0-99, 0=suppression) }`
Response : `200` - Panier complet
Erreurs : `400`, `401`, `404`
Autorisation : Token requis

DELETE /cart/items/:productId
Description : Supprimer article panier
Params : `productId` (uuid, required)
Body : none
Response : `204` No Content
Erreurs : `401`, `404` (not in cart)
Autorisation : Token requis

DELETE /cart
Description : Vider panier complet
Body : none
Response : `204` No Content
Erreurs : `401`
Autorisation : Token requis

1.6 Commandes (Routé → orders-service:3003)

POST /orders
Description : Créer commande depuis panier
Body : `{ "shippingAddressId": "uuid" (required), "billingAddressId": "uuid" (required), "paymentMethod": string (required, "card|paypal|bank"), "couponCode": string (optional) }`
Response : `201` - `{ "orderId": "uuid", "status": "pending", "items": array, "shippingAddress": object, "billingAddress": object, "total": number, "currency": "string", "createdAt": "ISO date" }`
Erreurs : `400` (cart empty/invalid addresses), `401`, `402` (payment required)
Autorisation : Token requis

GET /orders
Description : Liste commandes utilisateur
Query : `status=pending|confirmed|shipped|canceled`, `page=0`, `limit=20`
Body : none
Response : `200` - `{ "orders": array[{ "orderId": "uuid", "status": "string", "total": number, "currency": "string", "createdAt": "ISO date" }], "total": number }`
Erreurs : `401`
Autorisation : Token requis (client: ses commandes, admin: toutes)

GET /orders/:orderId
Description : Détail commande complète
Params : `orderId` (uuid, required)
Body : none
Response : `200` - `{ "orderId": "uuid", "status": "string", "userId": "uuid", "items": array[{ "productId": "uuid", "name": "string", "quantity": number, "price": number }], "shippingAddress": object, "billingAddress": object, "total": number, "paymentStatus": "string", "trackingNumber": "string|null" }`
Erreurs : `403` (not owner/admin), `404`
Autorisation : Token requis

PUT /orders/:orderId/cancel
Description : Annulation commande (avant paiement)
Params : `orderId` (uuid, required)
Body : `{ "reason": string (optional) }`
Response : `200` - `{ "orderId": "uuid", "status": "canceled" }`
Erreurs : `400` (cannot cancel current status), `403` (not owner/admin), `404`
Autorisation : Token requis (propriétaire ou admin)

PUT /orders/:orderId/status
Description : Changer statut (admin uniquement)
Params : `orderId` (uuid, required)
Body : `{ "status": string (required, "confirmed|shipped|delivered") }`
Response : `200` - Commande mise à jour
Erreurs : `400` (invalid transition), `403` (not admin), `404`
Autorisation : Token requis, rôle admin

1.7 Paiements (Routé → payments-service:3005)

POST /payments
Description : Initialiser paiement commande
Body : `{ "orderId": "uuid" (required), "provider": string (required, "stripe|paypal"), "paymentMethod": string (optional) }`
Response : `201` - `{ "paymentId": "uuid", "orderId": "uuid", "status": "pending", "provider": "string", "amount": number, "currency": "string", "redirectUrl": "string" }`
Erreurs : `400` (invalid order), `402` (payment required), `404`
Autorisation : Token requis

GET /payments/:paymentId
Description : Statut paiement
Params : `paymentId` (uuid, required)
Body : none
Response : `200` - `{ "paymentId": "uuid", "orderId": "uuid", "status": "pending|success|failed", "provider": "string", "amount": number, "currency": "string", "transactionId": "string|null" }`
Erreurs : `403` (not owner/admin), `404`
Autorisation : Token requis

POST /payments/:paymentId/confirm
Description : Confirmer paiement externe (webhook/callback)
Params : `paymentId` (uuid, required)
Body : `{ "transactionId": "string" (required), "status": "success|failed" (required) }`
Response : `200` - Paiement mis à jour
Erreurs : `400` (invalid data), `403`, `404`
Autorisation : Token interne uniquement (webhook)

1.8 Adresses

POST /addresses
Description : Ajouter adresse utilisateur
Body : `{ "type": string (required, "billing|shipping"), "line1": string (required), "line2": string (optional), "city": string (required), "postalCode": string (required), "country": string (required, ISO), "isDefault": boolean (optional) }`
Response : `201` - `{ "addressId": "uuid", "type": "string", "line1": "string", ... }`
Erreurs : `400` (validation), `401`
Autorisation : Token requis

GET /addresses
Description : Liste adresses utilisateur
Query : `type=billing|shipping`
Body : none
Response : `200` - `{ "addresses": array[complete address objects] }`
Erreurs : `401`
Autorisation : Token requis

---

2. API Interne (Gateway → Microservices)

Headers internes : `X-User-Id`, `X-User-Role`, `X-Request-Id`

2.1 Users Service (Port 3001)

POST /internal/users | GET /internal/users/:userId | PUT /internal/users/:userId | POST /internal/users/authenticate

2.2 Catalog Service (Port 3002)

POST /internal/products | GET /internal/products | GET /internal/products/:productId | PUT /internal/products/:productId

2.3 Cart Service (Port 3004)

GET /internal/cart | POST /internal/cart/items | PUT /internal/cart/items/:productId | DELETE /internal/cart/items/:productId

2.4 Orders Service (Port 3003)

POST /internal/orders | GET /internal/orders | GET /internal/orders/:orderId | PUT /internal/orders/:orderId/status

2.5 Payments Service (Port 3005)

POST /internal/payments | GET /internal/payments/:paymentId | POST /internal/payments/:paymentId/confirm

---

3. Flux Métier Exemple : Achat Complet

```
text
```
1. POST /auth/login → JWT token
2. GET /products → Recherche produits
3. POST /cart/items → Ajout panier
4. POST /addresses → Ajout adresse livraison
5. POST /orders → Création commande (réserve stock)
6. POST /payments → Initie paiement Stripe
7. [Redirect] → Stripe checkout
8. POST /payments/:id/confirm → Webhook finalise
9. PUT /orders/:id/status → "confirmed"

Conforme contraintes projet : microservices indépendants, persistance SQL/NoSQL, communication réseau.projet-annuel.pdf

TABLEAU SQL vs NoSQL par Section du Contrat d'API

Section	Titre	Type Base	Service	Justification
1.1	Authentification	🗄 SQL/PostgreSQL	users:3001	Utilisateurs + rôles ACID projet-annuel.pdf
1.2	Utilisateurs	🗄 SQL/ PostgreSQL	users:3001	Comptes, transactions utilisateurs
1.3	Catalogue Produits	🗃NoSQL/MongoDB	catalog:3002	Recherche full-text, scalabilité
1.4	Catégories Produits	🗃 NoSQLMongoDB	catalog:3002	Hiérarchie flexible, indexation
1.5	Panier	🗃NoSQL/MongoDB	cart:3004	Éphémère TTL 30j haute volumétrie
1.6	Commandes	🗄 SQL/PostgreSQL	orders:3003	Transactions ACID critiques

1.7	Paiements	🗄 SQL/PostgreSQL	payments:3005	Audit PCI-DSS, conformité légale
1.8	Adresses	🗄 SQL/PostgreSQL	addresses:3006	Relations users/commandes

---

CONTRAT D'API MODIFIÉ avec annotations SQL/NoSQL

```
text
```
Contexte et Conventions Générales
Architecture : API Gateway + microservices indépendants ✓ [file:1]

1. API Externe (Front-end → API Gateway)
├── 1.1 Authentification [🗄 □ SQL PostgreSQL - users:3001]
├── 1.2 Utilisateurs [🗄 □ SQL PostgreSQL - users:3001]
├── 1.3 Catalogue Produits [🗃 NoSQL MongoDB - catalog:3002]
├── 1.4 Catégories Produits [🗃 NoSQL MongoDB - catalog:3002]
├── 1.5 Panier [🗃 NoSQL MongoDB - cart:3004]
├── 1.6 Commandes [🗄 □ SQL PostgreSQL - orders:3003]
├── 1.7 Paiements [🗄 □ SQL PostgreSQL - payments:3005]
├── 1.8 Adresses [🗄 □ SQL PostgreSQL - addresses:3006]
└── 2. API Interne [Hybride SQL/NoSQL]

✓ 4 services SQL + 2 services NoSQL = CONFORME PROJET [file:1]

SQL = données critiques transactionnelles (4/8 sections)
NoSQL = données scalables recherche (2/8 sections)projet-annuel.pdf

1. MAPPING BASES DE DONNÉES PAR SERVICE

Service	Port	Base	Justification
users	3001	PostgreSQL	ACID, rôles, transactions projet-annuel.pdf
catalog	3002	MongoDB	Recherche full-text, scalabilité projet-annuel.pdf
cart	3004	MongoDB	Éphémère TTL 30j, haute volumétrie projet-annuel.pdf
orders	3003	PostgreSQL	Transactions critiques ACID projet-annuel.pdf
payments	3005	PostgreSQL	Conformité PCI-DSS, audit projet-annuel.pdf
addresses	3006	PostgreSQL	Relations users/commandes projet-annuel.pdf

SQL = données critiques | NoSQL = catalogue scalable projet-annuel.pdf

---

---

2. Schémas SQL - PostgreSQL (Données critiques)

2.1 Utilisateurs & Rôles

```
sql
CREATE TABLE roles (id SERIAL PRIMARY KEY, code VARCHAR(20) UNIQUE NOT NULL);
CREATE TABLE users (
   id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
   username VARCHAR(30) UNIQUE NOT NULL,
   email VARCHAR(255) UNIQUE NOT NULL,
   password_hash VARCHAR(255) NOT NULL,
   role_id INT REFERENCES roles(id)
);
CREATE TABLE user_roles (user_id UUID REFERENCES users(id), role_id INT REFERENCES roles(id),
PRIMARY KEY (user_id, role_id));
```

2.2 Commandes

```
sql
CREATE TABLE orders (
   id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
   user_id UUID REFERENCES users(id),
   status VARCHAR(20) NOT NULL,
   total_amount DECIMAL(12,2) NOT NULL
);
CREATE TABLE order_items (
   order_id UUID REFERENCES orders(id) ON DELETE CASCADE,
   product_id UUID NOT NULL,
   quantity INT NOT NULL CHECK (quantity > 0)
);
```

2.3 Paiements

```
sql
CREATE TABLE payments (
   id UUID PRIMARY KEY,
   order_id UUID UNIQUE REFERENCES orders(id),
   status VARCHAR(20) NOT NULL,
   amount DECIMAL(12,2) NOT NULL
);
```

2.4 Adresses

```
sql
CREATE TABLE addresses (
   id UUID PRIMARY KEY,
   user_id UUID REFERENCES users(id) ON DELETE CASCADE,
   type VARCHAR(20) CHECK (type IN ('billing','shipping'))
);
```

---

3. Schémas NoSQL - MongoDB (Catalogue scalable)

3.1 Produits

```javascript
// Collection products
{
  productId: "uuid",
  sellerId: "uuid",      // ref SQL users
  name: "iPhone 15 Pro",
  price: 1299.99,
  categoryIds: ["uuid"],
  stock: 23,
  images: ["url1.jpg"],
  searchIndex: "iphone 15 pro max apple"  // full-text
}
```

3.2 Catégories

```javascript
// Collection categories
{
  categoryId: "uuid",
  name: "Smartphones",
  parentId: null,
  path: ["uuid_root", "uuid_electronique"]
}
```

3.3 Paniers

```javascript
// Collection carts (TTL 30j)
{
  userId: "uuid",     // ref SQL users
  items: [{productId: "uuid", quantity: 1}],
  totalAmount: 1299.99,
  updatedAt: ISODate()
}
```

---

4. API Interne (Gateway → Microservices)

Headers : `X-User-Id`, `X-User-Role`, `X-Request-Id`
/internal/ prefix pour tous les services

---

5. Flux Métier : Achat Complet

```text
1. POST /auth/login [SQL] → JWT
2. GET /products [NoSQL] → Recherche
3. POST /cart/items [NoSQL] → Panier
4. POST /addresses [SQL] → Adresse
5. POST /orders [SQL] → Commande ACID
```

6. POST /payments [SQL] → Paiement

✓ CONFORME PROJET : SQL + NoSQL, choix justifiés, frontières clairesmicroservices projet-annuel.pdf