TECHNOLOGIE DU COMMERCE ÉLECTRONIQUE

INF27523-MS

UNIVERSITÉ DU QUÉBEC À RIMOUSKI

DÉPARTEMENT DE MATHÉMATIQUES, D'INFORMATIQUE ET DE GÉNIE

TRAVAIL

Travail Pratique 2

ÉLÈVE :
Bah Oumar Diogo
Eli Daniel Senyo

PROFESSEUR :
Yacine Yaddadden

Date : 30 avril 2025

Table des matières

1 Vue d'Ensemble

L'application ECOM repose sur une architecture microservices sophistiquée qui compartimente les fonctionnalités en services autonomes et spécialisés. Cette approche architecturale permet une maintenance ciblée, une évolution indépendante des composants et une résilience globale du système. L'architecture se compose de sept services principaux interconnectés :

- Service d'Authentification
- Service Utilisateur
- Service Produits
- Service Panier
- Service Commandes
- Service Paiement
- Passerelle (Gateway)

Sur le plan technique, le système exploite ASP.NET Core 9.0 comme framework principal, associé à Entity Framework Core pour gérer la persistance des données dans SQL Server. Les communications entre services s'effectuent exclusivement via des API REST avec échange de données au format JSON. La sécurité est assurée par un système d'authentification JWT centralisé, garantissant que les accès aux ressources sont correctement autorisés et authentifiés.

2 Analyse Détaillée des Microservices

2.1 Service d'Authentification (ECOM_AuthentificationMicroservice)

Le service d'authentification constitue la pierre angulaire de la sécurité du système ECOM. Il prend en charge l'ensemble du processus d'authentification des utilisateurs en exposant des endpoints REST pour la connexion, l'enregistrement et la validation des sessions. Sa fonction principale est de générer et valider des tokens JWT sécurisés qui seront utilisés par les autres services pour authentifier les requêtes.

L'architecture interne de ce service s'articule autour d'un contrôleur `AuthController` qui expose trois endpoints essentiels :

- `/api/auth/login` - Authentification et génération de token
- `/api/auth/register` - Enregistrement de nouveaux utilisateurs
- `/api/auth/validate` - Vérification de validité des tokens

Le `TokenService` est responsable de la génération des JWT avec une sécurité renforcée, tandis que l'`AuthService` orchestre l'ensemble du processus d'authentification et communique avec le service utilisateur pour vérifier les identifiants.

La gestion des tokens JWT est particulièrement raffinée, avec l'inclusion de claims d'identité et de rôle qui distinguent les clients des vendeurs. Des claims spécifiques sont ajoutés pour les vendeurs, contenant les informations d'entreprise essentielles. Les to-

kens sont configurés avec une durée de validité de 7 jours et font l'objet d'une validation complète incluant la vérification de l'expiration, de la signature et du format.

2.2 Service Utilisateur (ECOM_UtilisateurMicroservice)

Le service utilisateur centralise la gestion des profils et des informations personnelles dans le système ECOM. Il permet une gestion complète du cycle de vie des utilisateurs, depuis la création jusqu'à la suppression, en passant par les mises à jour de profil. Ce service supporte deux catégories distinctes d'utilisateurs : les clients standards et les vendeurs qui disposent d'attributs supplémentaires liés à leur activité commerciale.
Sur le plan des données, ce service s'appuie sur une entité `User` complète qui stocke aussi bien les informations personnelles (nom, prénom, email) que professionnelles pour les vendeurs (nom d'entreprise, adresse, téléphone). L'énumération `UserType` permet de distinguer clairement les deux types d'utilisateurs. La validation des données est assurée par des attributs Data Annotations qui imposent des règles métier strictes, notamment l'obligation pour les vendeurs de fournir un nom d'entreprise.
L'API du service utilisateur expose une série d'endpoints RESTful pour manipuler les ressources utilisateur :

- `GET /api/users` - Liste de tous les utilisateurs
- `GET /api/users/{id}` - Détails d'un utilisateur spécifique
- `GET /api/users/by-email/{email}` - Recherche par email
- `POST /api/users` - Création d'utilisateur
- `PUT /api/users/{id}` - Mise à jour d'utilisateur
- `DELETE /api/users/{id}` - Suppression d'utilisateur
- `GET /api/users/clients` - Liste des clients uniquement
- `GET /api/users/vendeurs` - Liste des vendeurs uniquement

Cette API complète permet aux autres services, notamment l'authentification, d'accéder aux informations utilisateur nécessaires à leur fonctionnement.

2.3 Service Produits (ECOM_ProductsMicroservice)

Le service produits forme le cœur du catalogue de l'application e-commerce. Il gère l'intégralité du cycle de vie des produits et offre des fonctionnalités avancées de recherche et de filtrage. Ce service permet également de mettre en avant les nouveautés et d'organiser les produits par catégories pour faciliter la navigation. Un aspect crucial de ce service est la sécurisation des opérations, où seuls les vendeurs authentifiés peuvent modifier les produits, avec une restriction supplémentaire : un vendeur ne peut manipuler que ses propres produits.
Le modèle de données central, l'entité `Product`, est particulièrement riche et comprend tous les attributs nécessaires à la description d'un produit commercial : nom, descriptions (courte et détaillée), prix, catégorie, note, image et statut de nouveauté. Le lien avec le vendeur est assuré par le champ `SellerId`, permettant d'associer chaque produit

à son propriétaire légitime. Cette association est fondamentale pour les mécanismes de sécurité.
L'API du service produits expose des endpoints complets pour la gestion du catalogue :

- `GET /api/products` - Liste des produits
- `GET /api/products/{productId}` - Détails d'un produit spécifique
- `POST /api/products` - Ajout de produit (vendeurs uniquement)
- `PUT /api/products/{productId}` - Mise à jour de produit
- `DELETE /api/products/{productId}` - Suppression de produit
- `GET /api/products/category/{category}` - Filtrage par catégorie
- `GET /api/products/new-arrivals` - Nouveaux produits
- `GET /api/products/search` - Recherche textuelle

La sécurité du service est renforcée par une double vérification : d'abord l'authentification JWT pour confirmer que l'utilisateur est bien un vendeur, puis une vérification que le vendeur manipule uniquement ses propres produits, évitant ainsi toute modification non autorisée.

2.4 Service Panier (ECOM_PanierMicroservice)

Le service panier gère l'expérience d'achat temporaire des utilisateurs en permettant l'ajout, la modification et la suppression d'articles avant validation de la commande. Il assure la persistance des paniers entre les sessions, permettant aux utilisateurs de retrouver leurs sélections même après déconnexion. Un atout majeur de ce service est le calcul automatique et en temps réel des montants (sous-total, taxes, frais de livraison et total) à mesure que le contenu du panier évolue.
La structure de données s'articule autour de deux entités principales : `Cart`, qui représente le panier global lié à un utilisateur spécifique, et `CartItem`, qui contient les détails de chaque article ajouté (produit, quantité, prix). L'entité `Cart` intègre également un statut (`Active`, `CheckedOut` ou `Abandoned`) permettant de suivre l'état du panier.
Les calculs commerciaux automatisés comprennent :

- Sous-total = Somme des (prix × quantité) pour chaque article
- Frais d'expédition = 10,00€ (si sous-total ¿ 0)
- Taxe = 7% du sous-total
- Total = Sous-total + Frais d'expédition + Taxe

L'API du service expose cinq endpoints principaux :

- `GET /api/cart` - Consulter le panier actuel
- `POST /api/cart/add` - Ajouter un article
- `PUT /api/cart/update-quantity/{productId}` - Modifier une quantité
- `DELETE /api/cart/remove/{productId}` - Retirer un article spécifique
- `DELETE /api/cart/clear` - Vider entièrement le panier

Le processus d'ajout au panier suit une séquence bien définie : réception d'une demande d'ajout avec ID produit et quantité, vérification de l'existence du produit via le service produits, recherche du panier utilisateur (ou création si nécessaire), puis ajout ou mise

à jour de l'article dans le panier. Si l'article existe déjà, sa quantité est incrémentée ; sinon, un nouvel article est créé avec les informations récupérées du service produits.

2.5 Service Commandes (ECOM_CommandesMicroservice)

Le service commandes représente une étape cruciale dans le parcours d'achat, transformant un panier temporaire en engagement ferme d'achat. Il assure la création des commandes à partir des paniers utilisateurs, la gestion complète du cycle de vie des commandes à travers différents états, et le stockage permanent de l'historique des commandes par utilisateur. Le service gère également les informations de livraison nécessaires à l'acheminement des produits.
Le modèle de données de ce service s'appuie sur deux entités principales : `Order`, qui contient les informations générales de la commande (utilisateur, montants, adresse de livraison, statut), et `OrderItems`, qui préserve un instantané des produits au moment de la commande. Cette approche d'instantané est cruciale car elle garantit que les caractéristiques des produits (nom, prix, description) sont figées au moment de la commande, même si les produits sont modifiés ultérieurement dans le catalogue.
Le cycle de vie des commandes est géré via l'énumération `OrderStatus` qui définit cinq états possibles :

- `Pending` - Commande en attente de traitement
- `Processing` - Commande en cours de traitement
- `Shipped` - Commande expédiée
- `Delivered` - Commande livrée
- `Cancelled` - Commande annulée

L'API du service expose des endpoints complets pour la gestion des commandes :

- `POST /api/order` - Créer une nouvelle commande
- `GET /api/order/{orderId}` - Consulter une commande spécifique
- `GET /api/order/user` - Visualiser toutes les commandes de l'utilisateur
- `PUT /api/order/{orderId}/status` - Mettre à jour le statut
- `DELETE /api/order/{orderId}` - Supprimer une commande en attente

La logique métier du service commandes est particulièrement élaborée. Lors de la création d'une commande, le service suit plusieurs étapes clés : récupération du contenu du panier, création d'un instantané des articles, enregistrement des informations de livraison, calcul des montants, vidage du panier utilisateur, et attribution du statut initial `Pending`. Les transitions d'état suivent des règles strictes, où seules certaines transitions sont permises (par exemple, une commande livrée ne peut plus être annulée).

2.6 Service Paiement (ECOM_PayementMicroservice)

Le service paiement constitue la phase finale du processus d'achat, permettant aux utilisateurs de finaliser leurs commandes par un règlement sécurisé. Ce service s'appuie sur l'intégration de l'API Stripe, une solution de paiement en ligne reconnue, pour gérer les

transactions financières de manière sécurisée. Il assure la création et la gestion des intentions de paiement, la confirmation des paiements avec les données de carte bancaire, et le suivi précis des statuts de transaction.
La structure de données du service s'articule autour de quatre modèles principaux : `PaymentTransaction` qui trace les transactions de paiement, `CreditCardInfo` qui centralise temporairement les données de carte (numéro, date d'expiration, code CVC), `OrderResponse` qui représente les informations de commande nécessaires au paiement, et l'énumération `PaymentStatus` qui définit les états possibles d'une transaction (Pending, Succeeded, Failed, Cancelled).
L'API du service expose six endpoints stratégiques :

- `POST /api/payment/create-order-payment` - Initialiser un paiement
- `POST /api/payment/{paymentIntentId}/confirm` - Confirmer un paiement
- `GET /api/payment/{paymentIntentId}/status` - Vérifier l'état
- `GET /api/payment/{paymentIntentId}` - Obtenir les détails
- `PUT /api/payment/{paymentIntentId}` - Ajuster le montant
- `POST /api/payment/{paymentIntentId}/cancel` - Annuler un paiement

L'intégration avec Stripe est réalisée via le `StripeService` qui encapsule toutes les interactions avec l'API de paiement. Ce service crée des objets `PaymentIntent` dans Stripe avec le montant et la devise appropriés, confirme les paiements en associant une méthode de paiement valide, et gère l'ensemble du cycle de vie des transactions. Une validation préalable des cartes est effectuée pour vérifier que la date d'expiration n'est pas dépassée, évitant ainsi des tentatives de paiement vouées à l'échec. La gestion des erreurs Stripe est soigneusement implémentée pour fournir des messages d'erreur clairs et exploitables en cas de problème.

3 Architecture Technique

Le système ECOM s'appuie sur une pile technologique moderne et robuste, centrée sur l'écosystème Microsoft. Les principales technologies utilisées sont :

- ASP.NET Core 9.0 comme framework principal de développement
- Entity Framework Core 9.0.4 pour l'accès aux données
- SQL Server comme système de gestion de base de données
- JWT (JSON Web Tokens) pour l'authentification
- API Stripe pour le traitement des paiements

Les communications entre microservices suivent exclusivement le paradigme REST, avec des échanges de données au format JSON. Cette approche permet une intégration souple et une évolution indépendante des services. La propagation des tokens d'authentification entre services est réalisée via des en-têtes HTTP standards, assurant que le contexte de sécurité est maintenu tout au long des appels chaînés entre services. Bien que synchrone par nature, cette architecture de communication offre une bonne réactivité et une transparence dans les échanges.

La sécurité du système est particulièrement soignée, avec une authentification JWT centralisée qui génère des tokens signés contenant les informations d'identité et de rôle. Les autorisations sont appliquées à deux niveaux : d'abord via les rôles globaux (Client/Vendeur) qui déterminent les fonctionnalités accessibles, puis via un contrôle d'accès aux ressources qui vérifie que l'utilisateur ne manipule que ses propres données (ses commandes, son panier, ses produits pour un vendeur). La validation des données entrantes est systématiquement réalisée via des Data Annotations qui définissent des contraintes strictes sur les modèles.

4 Points Forts et Conclusion

L'architecture ECOM présente trois atouts majeurs qui en font une solution particulièrement adaptée aux exigences modernes du e-commerce :

- **Modularité** : Services autonomes avec responsabilités clairement définies, facilitant la maintenance ciblée et l'évolution indépendante
- **Sécurité** : Système d'authentification robuste avec JWT et contrôle d'accès granulaire aux ressources
- **Flexibilité** : Architecture évolutive permettant l'ajout de nouvelles fonctionnalités sans perturber l'existant

Ce projet d'architecture microservices s'est révélé particulièrement stimulant à réaliser. La décomposition d'un système e-commerce complet en services interconnectés a constitué un défi technique enrichissant, permettant d'explorer en profondeur les bonnes pratiques de conception distribuée. L'intégration de technologies modernes comme ASP.NET Core, Entity Framework et Stripe a offert l'opportunité de mettre en œuvre des solutions innovantes pour répondre aux exigences métier d'une plateforme e-commerce.
La mise en place des mécanismes de sécurité, la gestion des communications inter-services et l'orchestration des flux utilisateurs complets ont été des aspects particulièrement passionnants qui ont nécessité une réflexion approfondie et des solutions techniques élégantes. Ce projet a non seulement permis de développer une architecture robuste et évolutive, mais également d'acquérir une expérience précieuse dans la conception de systèmes distribués modernes.

5 Références

- DotNetMastery (2023). *.NET Core Microservices - Full Course*. [Vidéo en ligne]. YouTube. Disponible sur : `https://www.youtube.com/watch?v=Nw4AZs1kLAs&t=12176s&ab_channel=DotNetMastery`
- DotNetMastery (2023). *.NET Core Web API Authentication with JWT*. [Vidéo en ligne]. YouTube. Disponible sur : `https://www.youtube.com/watch?v=_uZYOgzYheU&t=3966s&ab_channel=DotNetMastery`

— Scalable Scripts (2022). *Microservices with .NET and Docker*. [Vidéo en ligne]. YouTube. Disponible sur : https://www.youtube.com/watch?v=WNKwdgr34fA&list=PLlameCF3cMEvk_3QFm_GgJ-H4akG0V2jM&ab_channel=ScalableScripts
— github