TP02 – Création d'un *back-end* avec une *architecture* en *microservices*.

Technologies du commerce électronique (INF27507)
Hiver 2023
yacine_yaddaden@uqar.ca

Table des matières

1 Objectifs du travail

L'objectif de ce deuxième travail pratique est de vous initier aux concepts d'API REST en utilisant la technologie **Microsoft ASP.NET Web API** qu'on a eu l'occasion de voir en cours. Il sera principalement question des concepts suivants :

- ✓ Utilisation de l'architecture en *microservices*,
- ✓ Création des **API REST** avec **ASP.NET**,
- ✓ Utilisation de l'authentification par *jeton*,
- ✓ Utilisation de l'outil **SWAGGER** pour la documentation du *back-end*,
- ✓ Exploitation de l'API pour le paiement fournie par **Stripe**,
- ✓ ...

L'atteinte de cet objectif se fera à travers la réalisation d'un *back-end* pour la plateforme de e-commerce ou bien une boutique en ligne réalisée dans le premier travail pratique. Ceci, en respectant l'architecture en *microservices*.

2 Technologies à utiliser

Dans le cadre de ce travail pratique, vous serez amenés à utiliser les technologies suivantes :

- ☞ **Microsoft Visual Studio** → https://visualstudio.microsoft.com/fr/
- ☞ **Langage de programmation** → C Sharp ou **C#**

Pour la partie *persistance* des données, il faut installer :

- ☞ **Microsoft SQL Server 2019 Express** → `SqlLocalDB.msi`.

Pour la partie **ORM**, il faut installer à travers le *gestionnaire de paquets* **NuGet** :

- ☞ **Microsoft.EntityFrameworkCore.SqlServer** → établir la communication avec Microsoft SQL Server.
- ☞ **Microsoft.EntityFrameworkCore.Tools** → bénéficier de l'utilitaire pour les lignes de commandes.
- ☞ **Microsoft.EntityFrameworkCore.Design** → conception automatique de la base de données des *classes* vers les *tables*•.

Afin de mettre en place l'authentification par *jeton* :

- ☞ **Microsoft.AspNetCore.Identity.EntityFrameworkCore** → permet la gestion des utilisateurs et des jetons.
- ☞ **Microsoft.AspNetCore.Authentication.JwtBearer** → permet la génération des jetons et leurs vérification.

Afin d'utiliser **SWAGGER** :

- ☞ **Swashbuckle.AspNetCore** → permet de pouvoir utiliser l'outil SWAGGER (*documentation et interface Web*).
- ☞ **Microsoft.AspNetCore.Mvc.NewtonsoftJson** → Utilitaire permettant d'avoir des données imbriquées dans les réponses au format JSON.

Pour la mise en place de la passerelle **Ocelot** avec intégration **SWAGGER** pour les services :

- ☞ **Ocelot** → permet de créer une *passerelle* (ou *gateway*) pour l'architecture en *microservices*.
- ☞ **MMLib.SwaggerForOcelot** → permet de pouvoir utiliser Ocelot avec SWAGGER pour avoir l'ensemble des services sur une seule interface.

Pour la passerelle de paiement **Stripe** :

- ☞ **Stripe.net** → permet d'avoir accès à l'API pour effectuer des paiement à travers la passerelle Stripe.

3 Description détaillée

Afin de réaliser le projet, il est nécessaire de suivre les étapes suivantes :

3.1 Mettre en place l'architecture en *microservices*

La première étape est de décomposer la plateforme de e-commerce en différents *services*, chacun doit être *indépendant* des autres, mais *interagissent* entre eux. Quelques exemples de services : *produits*, *utilisateurs*, *commandes*, *paniers*, etc. Chaque service sera un projet (*avec différents ports*) à part faisant parti de la solution.

3.2 Création d'une passerelle d'API en utilisant Ocelot

En plus des différents services, il est nécessaire de créer une *passerelle* qui s'occupe d'intercepter les requêtes et les transmettre au bon service. Pour y arriver, il faut utiliser **Ocelot**. Il faut s'assurer des bonnes configurations (le fichier `ocelot.json`). De plus chaque projet doit être de type **ASP.NET Web API**.

3.3 Ajouter l'authentification par jeton

Un des services implémentés doit avoir comme rôle l'authentification des utilisateurs en passant par un jeton (ou *token*). Il faut configurer **SWAGGER** afin de permettre d'effectuer l'authentification.

3.4 Installer et configurer SWAGGER

Afin de documenter chacun des services, il faut installer et configurer **SWAGGER** dans chacun des services. De plus, il faut avoir une interface unifiée pour **SWAGGER** au niveau de la *passerelle* en passant par l'outil **MMLib.SwaggerForOcelot**.

3.5 Procéder à l'intégration du paiement électronique

Pour toute la partie paiement, il est nécessaire d'avoir un autre service dédié. Ce dernier communiquera avec l'API fournie par **Stripe**. Il faut procéder comme on avait fait en cours et créer un compte afin d'obtenir les clés *privée* et *publique*. La différence est que l'appel à l'API ne se fera pas à travers un formulaire, mais une requête.

3.6 Persistance des données

En ce qui concerne la partie base de données, chaque projet aura sa propre base de données au niveau du même serveur de base de données. Il faut reprendre les mêmes *entités* utilisées dans le premier travail pratique. Évidemment, il faudra les adapter afin que ça puisse fonctionner correctement.

4 Modalité d'évaluation

Le travail pratique comptera pour 25% (*points*) de la note du cours. Ces points sont répartis de la manière suivante :

Partie	Points
Création du back-end	21.00
→ Architecture en *microservice* (*arborescence*)	2.00
→ Création et implémentation des différents services	6.00
→ Mise en place de la *passerelle* avec les configurations	2.00
→ Service d'authentification par *jeton*	3.50
→ Configuration de **SWAGGER** + Interface *unifiée*	2.00
→ Configuration du paiement électronique avec **Stripe**	2.50
→ Création des *entités* et configuration de la *persistance*	3.00
Le rapport (*incluant la qualité du français*)	2.00
Démonstration vidéo	2.00

5 Date de remise

→ Date limite pour la remise : **26 Avril à 23h00**
→ Les fichiers à remettre :
  ✓ Le code source dans un fichier compressé → `source_code`,
  ✓ Un rapport (en Word ou PDF) → `rapport.pdf` ou `rapport.docx`,

✓ Une courte vidéo de démonstration → `demo.mp4`.

IMPORTANT : *Si la vidéo est trop volumineuse, il est possible de la téléverser sur YouTube en mode non référencé et d'inclure le lien en commentaire de la remise.*

6 Assistance

Dans le cas où vous avez besoin d'assistance, vous pouvez prendre contact avec nos auxiliaires d'enseignement en informatique :

☞ **Jérémy Bourget** (Lévis) → Jeremy_Bourget@uqar.ca
☞ **François Gosselin** (Rimouski) → Francois_Gosselin@uqar.ca

Vous pouvez également me contacter si vous n'arrivez toujours pas à avoir l'information dont vous avez besoin.

7 Points importants

- **Note I :**
  → Le travail est à réaliser en équipe de deux,
  → Le professeur peut poser des questions liées au travail pratique,
  → Le non respect du contenu de l'énoncé peut occasionner une perte de points,
  → En cas de plagiat, le ou les étudiants seront sanctionnés (*politique de l'université*),
  → Le retard de remise peut occasionner une perte de points.
- **Note II :** *Dans le cas où il y a des aspects qui ne sont pas clairs, n'hésitez pas à m'en faire part afin que je puisse apporter des éclaircissements et éventuellement mettre à jour l'énoncé du travail pratique.*

8 Documentation

Le contenu du cours est suffisant pour effectuer le travail pratique. Cependant, je vous encourage à vous documenter sur internet pour avoir plus d'informations.