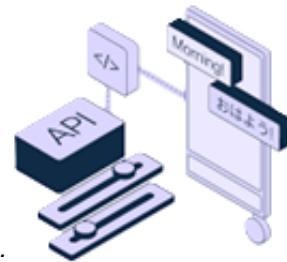


SOMMATIF 1

CONSOMMATION D'API WEB AVEC C# WPF

TRAVAIL A REALISER SEUL

 BALISE IAG :  LIMITÉ : Autorisé uniquement pour assister l'apprentissage.



Dans le cadre de ce travail, vous devez développer une application **WPF** qui consomme une API REST locale nommée « **StudentWebAPI** ». Cette API est connectée à une base de données MySQL et permet de gérer quatre entités liées à la gestion d'une école : étudiants, cours, départements, genres.

 Cependant, avant de pouvoir consommer les ressources de « **StudentWebAPI** », votre application devra obligatoirement passer par une étape d'authentification externe via une API publique.

OBJECTIFS PEDAGOGIQUES

- Comprendre et utiliser les opérations CRUD (Create, Read, Update, Delete) via HTTP
- Consommer une API REST depuis une application WPF
- Manipuler les verbes HTTP (GET, POST, PUT, PATCH, DELETE) de manière appropriée
- Gérer la sérialisation et désérialisation JSON
- Implémenter une architecture client en couches (séparation des responsabilités), similaire à l'exercice sur la gestion d'événements
- Gérer les erreurs réseau et les codes de statut HTTP

ÉTAPE 1 : MISE EN PLACE DE L'API FOURNIE

L'API « **StudentWebAPI** » fournie est une API REST qui gère un système de gestion académique. Elle expose plusieurs ressources organisées autour d'une base de données MySQL.

- 1- Ouvrez la solution dans Visual Studio
- 2- Si Visual Studio vous le demande, mettez à jour les **packages NuGet** de la solution fournie.
- 3- Créez une base de données ou utilisez-en une existante
- 4- Configurez les paramètres de connexion à la base de données dans le fichier « `appsettings.json` », dans la section « `ConnectionStrings` ».
- 5- Exécutez la commande : « `Update-Database` » pour effectuer la migration de la base de données.
⇒ Menu **Outils** → **Gestionnaire de package NuGet** → **Console du Gestionnaire de package**
Cela créera automatiquement les tables nécessaires dans votre base de données.
- 6- Générez le diagramme de la base de données à l'aide MySQL Workbench afin de mieux comprendre la structure de la base de données et les relations entre les tables.

7- Démarrez l'API en utilisant le bouton habituel d'exécution des applications dans **Visual Studio**.

- Une fenêtre de navigateur s'ouvre automatiquement.
- L'URL sera probablement : « <https://localhost:7080/swagger/index.html> »
- Vous verrez l'interface **Swagger** affichant l'ensemble des endpoints disponibles.

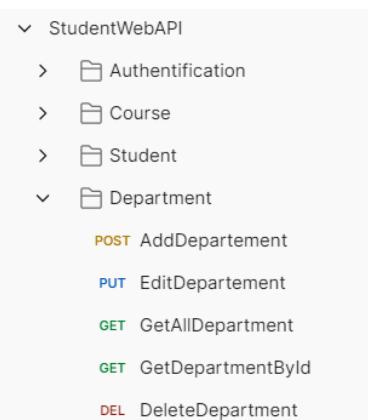
The screenshot shows the Swagger UI interface for the **StudentWebApi**. At the top, it says "1.0 OAS3". Below that, the URL "https://localhost:7080/swagger/v1/swagger.json" is shown. The main area is titled "Course" and contains a list of endpoints:

- GET /Course
- POST /Course
- GET /Course/{courseId}
- PUT /Course/{courseId}
- DELETE /Course/{courseId}
- POST /Course/registerStudentCourse/{courseId}/{studentId}
- DELETE /Course/deleteStudentCourse/{courseId}/{studentId}
- GET /Course/StudentsByCourse/{courseId}

ÉTAPE 2 : TEST DE L'API AVEC POSTMAN

- Testez les endpoints principaux de l'API à l'aide de Postman.
- Organisez les requêtes dans une **collection Postman**.
- Chaque catégorie de ressources de la collection (ex. étudiant, cours, département) doit être placée dans un dossier distinct.
 - Référez-vous à la documentation fournie dans Swagger.
 - Note : Swagger permet aussi de tester l'API, mais Postman offre des fonctionnalités plus complètes.

La structure à créer dans Postman :



- Ajoutez des données de test afin de faciliter la mise en place de l'application WPF qui devra consommer cette API. **Vous devez minimalement ajouter :**

3 départements

6 étudiants

5 cours

2 genres

ÉTAPE 3 : CONFIGURATION DE L'APPLICATION WPF

1. Créez un projet WPF nommé « StudentClient »
2. Installez les packages NuGet nécessaires
3. Configurez un fichier de configuration « appsettings.json » qui stockera les quatre valeurs suivantes :
 - L'URL de base de l'API publique d'authentification externe
 - L'URL de base de l'API locale « StudentWebAPI »
 - La clé d'API qui sera générée à l'étape suivante
 - Un token qui sera fourni par l'API publique d'authentification externe
4.  Structurez votre projet pour garantir la séparation des responsabilités

ÉTAPE 4 : AUTHENTIFICATION EXTERNE

Avant de pouvoir consommer les ressources de StudentWebAPI, votre application WPF doit d'abord s'authentifier auprès d'une API publique.

1. Créez un compte sur <https://app.reqres.in/api-keys>
2. Créez une clé API afin de pouvoir consommer le service d'authentification offert
3. Testez la ressource login à l'aide de Postman afin de vous assurer de son bon fonctionnement, en envoyant une requête **POST** vers l'API : <https://reqres.in/api/login>.

Header	Body
x-api-key : votre_clé_api	{ "email": "eve.holt@reqres.in", "password": "cityslicka" }

4. Sauvegardez cette requête dans la collection Postman dans un dossier nommé « *Authentification* ».
5. Dans votre projet WPF, implémentez un écran de connexion (login)
6. Pour vérifier l'email et le mot de passe saisis par l'utilisateur, envoyez une requête **POST** vers l'API : <https://reqres.in/api/login>, en utilisant la même structure json que celle employée lors du test avec Postman.
7. Gérez les réponses comme suit :

200 OK :	<ul style="list-style-type: none">• l'utilisateur est considéré comme authentifié• stockez le « token » reçu dans le fichier de configuration (même si StudentWebAPI n'en a pas besoin pour le moment)	l'accès à l'interface de l'application cliente WPF est autorisé
400 Bad Request :	afficher le message : « Identifiants invalides »	bloquer l'accès aux fonctionnalités de l'application WPF

REMISE DES TACHES DE CE LUNDI : AU PLUS TARD JEUDI 12 FÉVRIER 2026, FIN DE JOURNÉE

ÉTAPE 5 : CONSOMMATION DE « STUDENTWEBAPI »

Une fois l'utilisateur authentifié, l'accès à l'interface principale de l'application **WPF** est autorisé et l'application peut alors consommer l'API locale **StudentWebAPI**.

TÂCHES À RÉALISER – JEUDI 12 FÉVRIER 2026

50%

Les autres tâches de programmation à effectuer seront dévoilées jeudi, le 12 février 2026, et devront être réalisées en classe, sans possibilité d'accès aux outils d'IAG.

Lors de cette séance, vos écrans seront surveillés en temps réel. **Toute utilisation d'un outil non autorisé, quelle qu'en soit la forme, entraînera automatiquement l'attribution de la note zéro (0)**, sans possibilité de recours.

HEURE DE REMISE DES TACHES DU JEUDI 12 FÉVRIER 2026 : 12H00

TRAVAIL INDIVIDUEL

BALISE IAG :  **INTERDIT**

BON TRAVAIL !