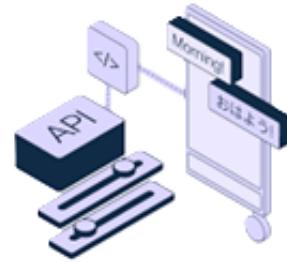


# SOMMATIF 1

CONSOMMATION D'API WEB AVEC C# WPF

12% DE LA NOTE FINALE

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.

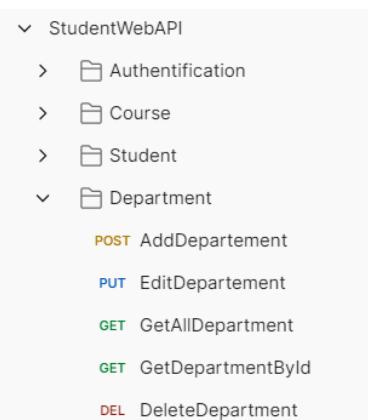
The screenshot shows the **StudentWebApi** documentation with the **Course** endpoint group expanded. The group contains the following endpoints:

- GET /Course** (blue button)
- POST /Course** (green button)
- GET /Course/{courseId}** (blue button)
- PUT /Course/{courseId}** (orange button)
- DELETE /Course/{courseId}** (red button)
- POST /Course/registerStudentCourse/{courseId}/{studentId}** (green button)
- DELETE /Course/deleteStudentCourse/{courseId}/{studentId}** (red button)
- GET /Course/StudentsByCourse/{courseId}** (blue button)

## 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
<b>x-api-key</b> : 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 :

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

**REMISE ÉTAPES 2,3 ET 4 : AU PLUS TARD LUNDI 16 FÉVRIER 2026 : 23H59**

## ÉTAPE 5 : CONSOMMATION DE « STUDENTWEBAPI »

### CONSIGNES STRICTES

**Balise IAG :  INTERDIT**

**TRAVAIL INDIVIDUEL**

Vos écrans sont surveillés en temps réel. **Toute utilisation d'un outil non autorisé, quelle qu'en soit la forme, entraînera automatiquement la note zéro (0)**, sans possibilité de recours.

**CETTE PARTIE COMpte SUR  50% DE LA NOTE DE CETTE EVALUATION FORMATIVE**

### MISE EN SITUATION

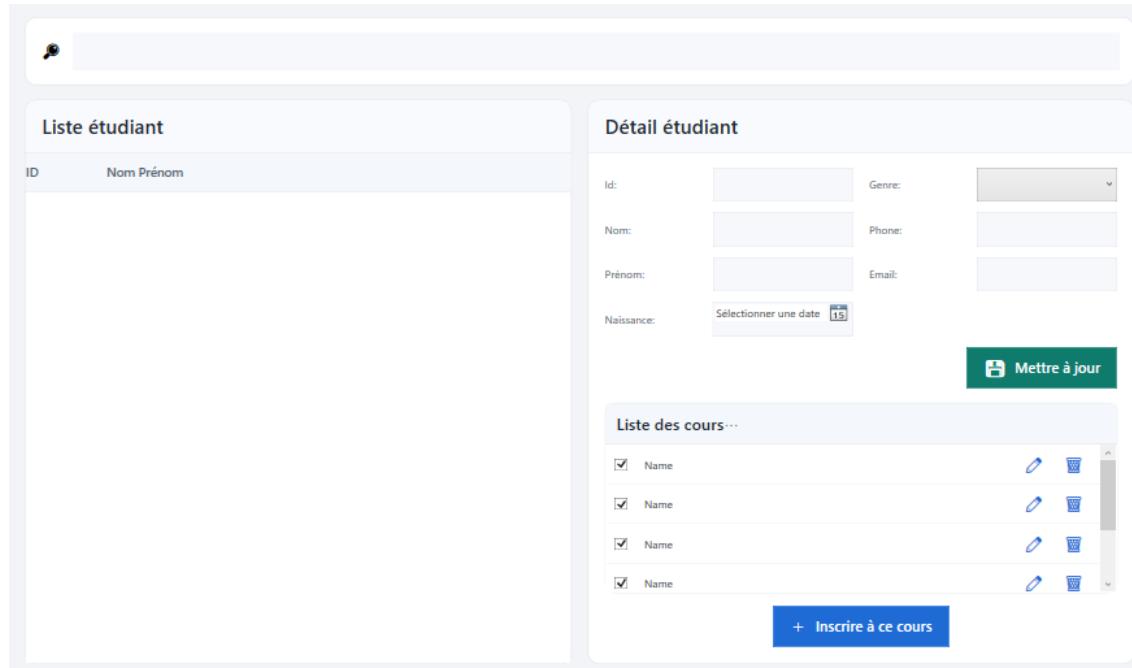
Une fois l'utilisateur authentifié, votre application WPF doit permettre de :

- consulter la liste complète des étudiants;
- consulter les informations personnelles détaillées d'un étudiant ;
- mettre à jour les informations d'un étudiant ;
- visualiser la liste des cours disponibles ;
- inscrire un étudiant à un nouveau cours.

L'interface graphique doit respecter fidèlement la maquette fournie :

- panneau gauche : liste des étudiants
- panneau droit : détails de l'étudiant sélectionné
- section inférieure droite : liste des cours disponibles

L'objectif est de reproduire un scénario réel où une application cliente (WPF) consomme une API REST afin de manipuler des données distantes.



The screenshot shows a WPF application window divided into two main sections. The left section is titled 'Liste étudiant' and contains a table with columns 'ID' and 'Nom Prénom'. The right section is titled 'Détail étudiant' and includes fields for 'Id', 'Nom', 'Prénom', 'Phone', 'Email', and 'Naissance' (with a date picker). Below these details is a list titled 'Liste des cours...' containing several items, each with a checkbox and a delete icon. At the bottom of this list is a blue button labeled '+ Inscire à ce cours'.

*Si cela peut vous faire gagner du temps, étant donné que le temps est limité, vous pouvez utiliser le fichier « .xaml » fourni.*

## FONCTIONNALITES OBLIGATOIRES A REMETTRE A MIDI

**REMISE : AUJOURD'HUI - JEUDI 12 FÉVRIER 2026 : 12H00**

### 1. Lister les étudiants

Au démarrage, l'application doit récupérer la liste des étudiants depuis l'API et l'afficher dans la grille « Liste étudiant » afin de permettre la sélection d'un étudiant.

### 2. Afficher les détails d'un étudiant

Lorsque l'utilisateur sélectionne un étudiant dans la liste, l'application doit afficher automatiquement ses informations détaillées dans le panneau de droite.

### 3. Mettre à jour les informations d'un étudiant

L'application doit permettre la mise à jour complète des informations de l'étudiant sélectionné (nom, prénom, genre, date de naissance, email, téléphone) à partir du formulaire affiché dans le panneau de droite.

Lorsque l'utilisateur clique sur le bouton « Mettre à jour », les nouvelles informations doivent être enregistrées via l'API.

## AUTRES FONCTIONNALITES OBLIGATOIRES POUVANT ETRE REMISES LUNDI 16 FÉVRIER 2026

**REMISE : AU PLUS TARD LUNDI 16 FÉVRIER 2026 : 23H59**

### 4. Afficher la liste des cours

L'application doit afficher dans la grille « Liste des cours » la liste complète des cours disponibles, indépendamment de l'étudiant sélectionné. Chaque ligne doit afficher au minimum le nom du cours et les icônes Modifier / Supprimer.

⚠ **Attention :** Il ne s'agit pas de la liste des cours associés à un étudiant, puisque l'API ne dispose pas de endpoint permettant de récupérer cette information.

### 5. Incrire un étudiant à un cours

L'application doit permettre d'inscrire l'étudiant sélectionné au cours coché ou sélectionné dans la grille « Liste cours » lorsque l'utilisateur clique sur le bouton « Incrire à ce cours ».

⚠ L'API ne dispose pas de endpoint permettant de vérifier si l'inscription a été effectuée ou non. La validation sera faite par l'enseignant directement dans la base de données.

## ATTENTES ET CRITÈRES D'ÉVALUATION

1. Vous devez démontrer une organisation claire et structurée de votre application, avec une bonne séparation des responsabilités.
2. Votre application doit gérer correctement les réponses de l'API et informer l'utilisateur en cas de problème.
3. Elle doit gérer les exceptions afin de rester stable en toutes circonstances.
4. Le code doit respecter les bonnes pratiques et être clair, cohérent et bien structuré.
5. L'application doit respecter la maquette et fonctionner correctement du début à la fin.

**BON TRAVAIL !**