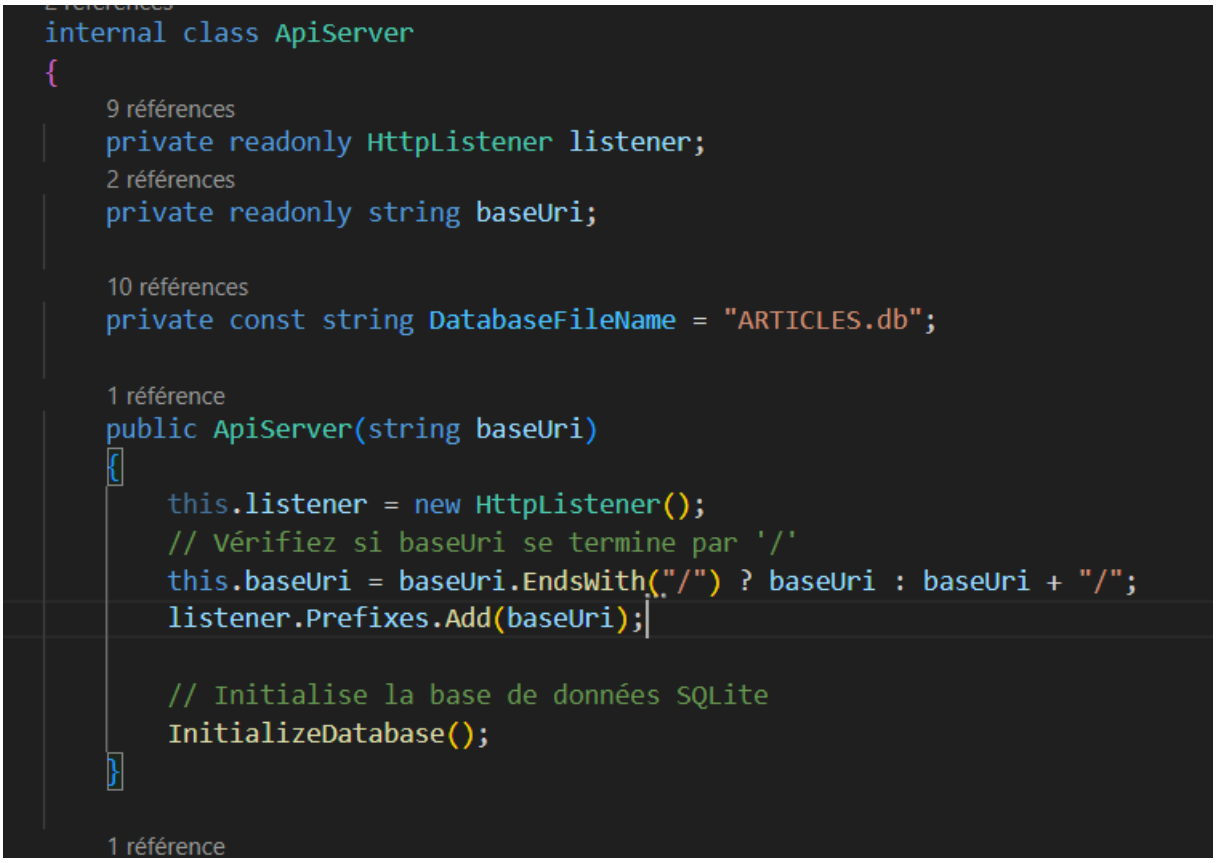
Documentation de l'API

Introduction

Bienvenue dans la documentation de l'API de gestion d'inventaire et de stock développée en C#. Cette API permet de gérer des produits et des utilisateurs au sein d'une base de données SQLite.

Étape 1 : Création de l'API C#

Nous avons entamé le projet en créant une API en langage C# pour la gestion des articles et des utilisateurs. Cette API offre des fonctionnalités telles que la création, la lecture, la mise à jour et la suppression d'articles, ainsi que la gestion des utilisateurs. Pour garantir la robustesse de notre solution, nous avons fait en sorte que l'API soit accessible via HTTP.



```
internal class ApiServer
{
    9 références
    private readonly HttpListener listener;
    2 références
    private readonly string baseUri;

    10 références
    private const string DatabaseFileName = "ARTICLES.db";

    1 référence
    public ApiServer(string baseUri)
    {
        this.listener = new HttpListener();
        // Vérifiez si baseUri se termine par '/'
        this.baseUri = baseUri.EndsWith("/") ? baseUri : baseUri + "/";
        listener.Prefixes.Add(baseUri);

        // Initialise la base de données SQLite
        InitializeDatabase();
    }

    1 référence
```

Étape 2 : Documentation Approfondie

La documentation de l'API revêt une importance cruciale. Nous avons généré un fichier au format PDF qui décrit de manière détaillée chaque point d'extrémité (endpoint) de l'API, les paramètres requis, les formats de données acceptés, ainsi que les retours possibles. Cela facilitera grandement la compréhension et l'utilisation de l'API par d'autres développeurs.

Étape 3 : Opérations CRUD

Nous avons mis en œuvre des opérations CRUD (Create, Read, Update, Delete) pour les articles et les utilisateurs. Chaque opération est associée à un point d'extrémité spécifique de l'API. Cela permet à l'utilisateur d'interagir avec la base de données, de manière à créer, lire, mettre à jour et supprimer des données avec une grande facilité.

```
private void AjouterProduit(string produitData)
```

```
private void SupprimerProduit(int produitId)
{
private void MettreAJourProduit(int produitId, string nouveauNom, decimal nouveauPrix, string nouvelleDate)
private void SupprimerUtilisateur(int utilisateurId)
{
private void SupprimerTousProduitsParId(int utilisateurId)
```

...

Étape 4 : Serveur Web

La configuration du serveur web était essentielle pour héberger notre API. Nous avons veillé à ce que le serveur soit correctement configuré pour répondre aux requêtes HTTP dirigées vers notre API. Cela garantit une accessibilité sans faille pour les utilisateurs finaux et les autres services qui peuvent consommer notre API.

```
public void Start()
{
    try
    {
        listener.Start();
        Console.WriteLine($"Serveur démarré sur {baseUri}");
        listener.BeginGetContext(ListenerCallback, listener);
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Erreur lors du démarrage du serveur : {ex.Message}");
    }
}

1 référence
public void Stop()
{
    try
    {
        listener.Stop();
        Console.WriteLine("Serveur arrêté");
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Erreur lors de l'arrêt du serveur : {ex.Message}");
    }
}
```

Étape 5 : Notions Avancées

Nous avons exploré des notions avancées telles que l'asynchronisme en C#. L'utilisation de mécanismes asynchrones améliore les performances de l'API en permettant le traitement simultané de plusieurs requêtes sans bloquer le fil d'exécution.

```
private void ListenerCallback(IAsyncResult result)
{
    try
    {
        HttpListenerContext context = listener.EndGetContext(result);
        listener.BeginGetContext(ListenerCallback, listener);
        HttpListenerRequest request = context.Request;
        HttpListenerResponse response = context.Response;

        Console.WriteLine($"Requête reçue : {request.HttpMethod} {request.Url}");

        if (request.HttpMethod == "GET" && request.Url.LocalPath == "/api/produits/")
        {
            string responseData = GetProduits();
            WriteResponse(response, responseData);
        }
        else if (request.HttpMethod == "POST" && request.Url.LocalPath == "/api/produits/ajouter/")
        {
            string requestData;
            using (StreamReader reader = new StreamReader(request.InputStream, request.ContentEncoding))
            {
                requestData = reader.ReadToEnd();
            }

            AjouterProduit(requestData);

            WriteResponse(response, "Produit ajouté avec succès.");
        }
```

Étape 6 : Interaction avec la Base de Données

L'intégration de la base de données MySQL dans notre application était une étape clé. Nous avons suivi les meilleures pratiques pour interagir avec la base de données de manière sécurisée et efficace, en utilisant des requêtes SQL appropriées pour les opérations CRUD.

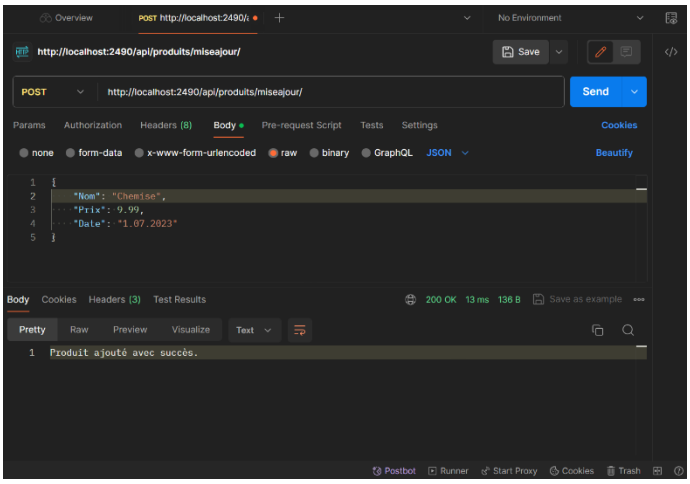
Ces étapes ont contribué à créer une API complète, performante et bien documentée pour la gestion d'inventaire.

```
string createTableQuery = "CREATE TABLE Produits (Id INTEGER PRIMARY KEY AUTOINCREMENT, Nom TEXT, Prix DECIMAL, Date TEXT)";
using (SQLiteCommand command = new SQLiteCommand(createTableQuery, connection))
```

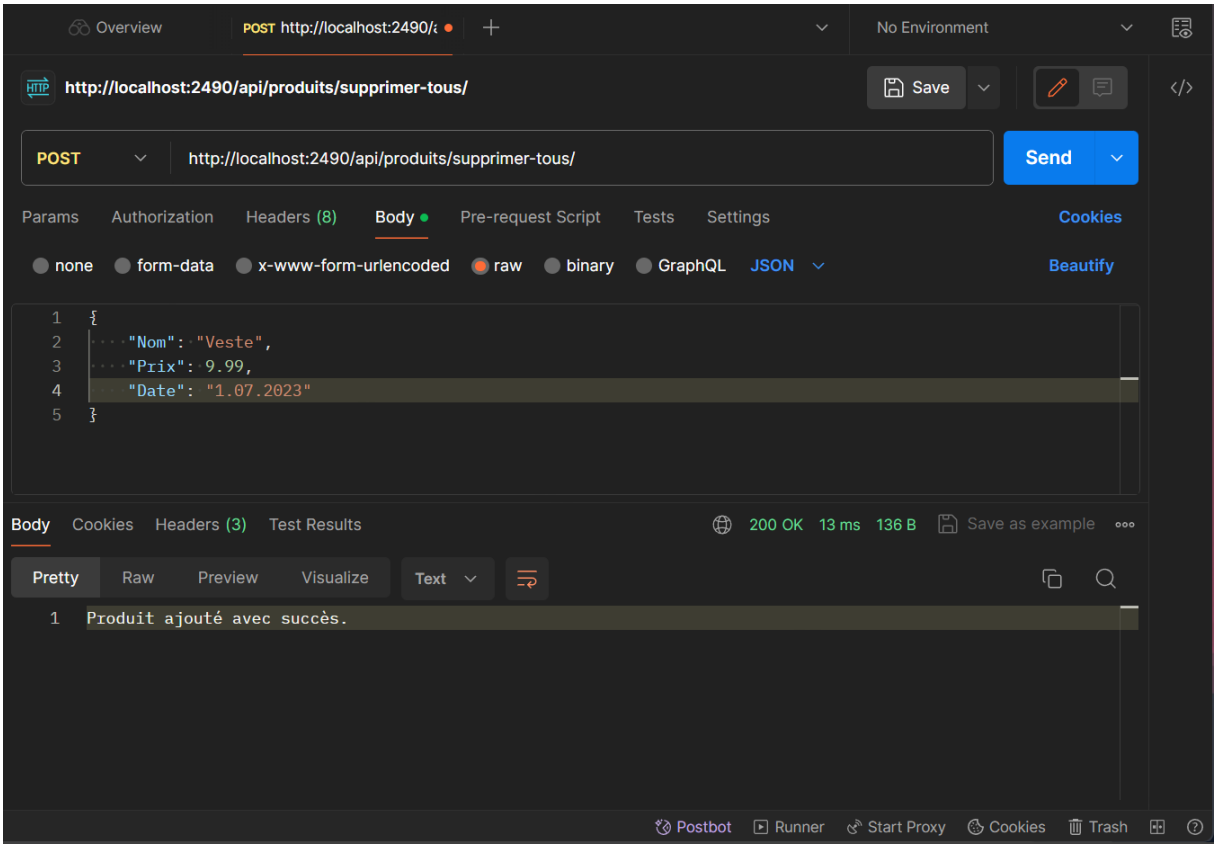
Etape 7 : Utilisation Postman

Afin d'utiliser le CRUD il faut changer les liens du localhost

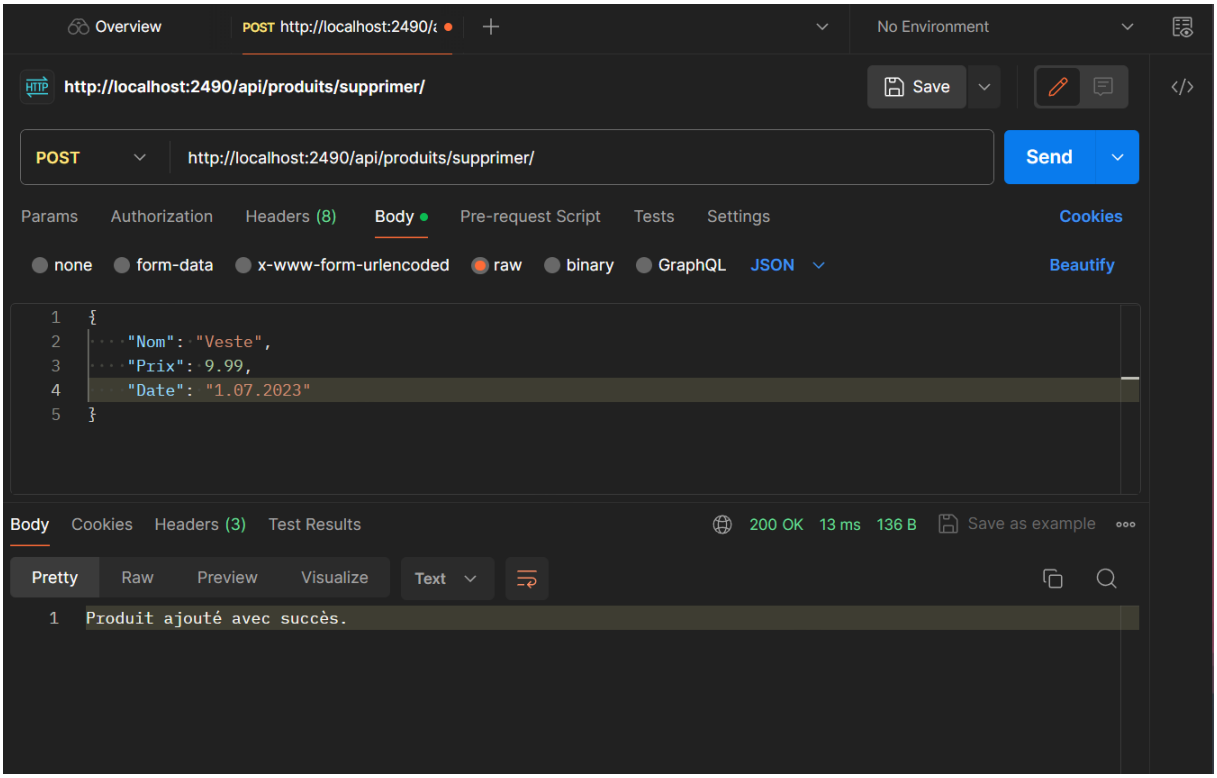
Comme ci-dessous avec « /miseajour/ » cela sert à modifier le contenu



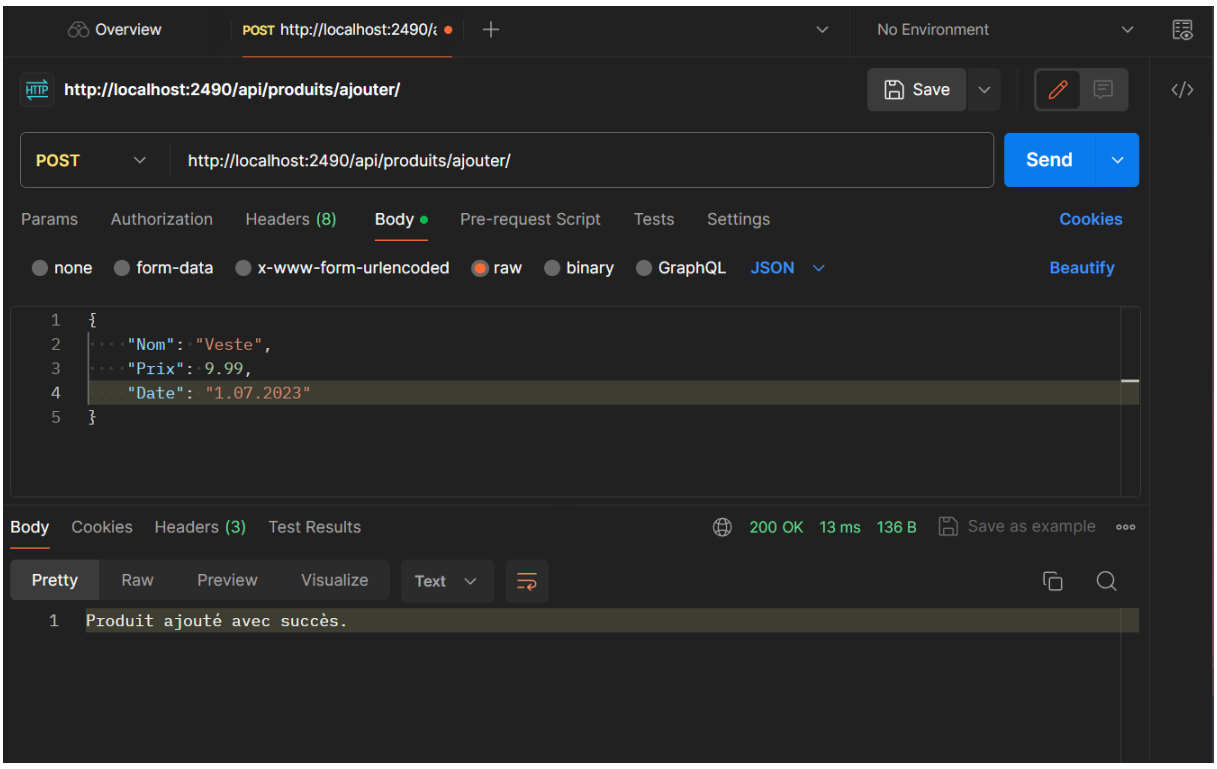
« /supprimer-tous/ » afin de tout supprimer a partir de l'ID



« /supprimer/ » afin de supprimer les données rentré



« /ajouter/ » afin de pouvoir ajouter les données entrées



Étape 8: Gestion de Version avec GitHub

Pour assurer une gestion efficace de notre code, nous avons utilisé GitHub. Nous avons mis en place des branches, des pull requests et des commentaires pour faciliter la collaboration au sein de l'équipe.

baayvin17/Projet_C_API (github.com)

Étape 9 : Gestion de Projet avec Trello

Trello a été notre outil central pour organiser les tâches de l'équipe. Nous avons créé des tableaux, des listes et des cartes pour suivre le progrès du projet. Chaque membre de l'équipe avait des rôles clairs et des tâches spécifiques.

https://trello.com/invite/b/J8FueJYq/ATTIdb3bc5ddfbce94f1edb2c3dc9a807c16648CDE4D/projet-cde-gestion-dinventaire

Baayvin SOUBRAMANIEN

Hocine CAUCHOIX

Terence AMBA