1. Préparation et Création du Projet

Installation de .NET Core SDK : La première étape était de s'assurer que le .NET Core SDK était installé sur votre machine, ce qui est essentiel pour développer des applications ASP.NET Core.

Création du Projet : Nous avons utilisé la commande dotnet new webapi -n quest\_web -o quest\_web\_dotnet pour créer un nouveau projet ASP.NET Core avec un modèle de Web API. Cette commande crée un projet nommé quest\_web dans un dossier quest\_web\_dotnet.

2. Configuration du Projet

Désactivation de HTTPS : Habituellement, un projet ASP.NET Core est configuré pour utiliser HTTPS. Nous avons modifié le fichier Properties/launchSettings.json pour désactiver HTTPS, en supprimant ou en commentant les références à HTTPS et au port SSL.

Changement du Port : Toujours dans launchSettings.json, nous avons changé le port d'écoute de l'application pour utiliser le port 8090 au lieu du port par défaut.

3. Création du Contrôleur

Ajout du DefaultController : Nous avons créé un fichier DefaultController.cs dans le dossier Controllers. Ce contrôleur est chargé de gérer les requêtes HTTP pour l'API.

Écriture des Méthodes du Contrôleur : Trois méthodes ont été implémentées dans DefaultController :

TestSuccess pour retourner un code HTTP 200 et le message "success".

TestNotFound pour retourner un code HTTP 404 et le message "not found".

TestError pour retourner un code HTTP 500 et le message "error".

Ces méthodes utilisent des annotations pour mapper les requêtes HTTP GET aux URL correspondantes.

4. Démarrage et Test de l'API

Démarrage de l'API : Avec le projet configuré, nous avons démarré l'API en utilisant la commande dotnet run dans le terminal.

Test avec Postman : Postman, un outil populaire pour tester les API, a été utilisé pour envoyer des requêtes GET aux différentes URL et vérifier les réponses.

Test de /Default/testSuccess : Cette requête devait retourner un statut HTTP 200 avec le message "success".

Test de /Default/testNotFound : Cette requête devait retourner un statut HTTP 404 avec le message "not found".

Test de /Default/testError : Cette requête devait retourner un statut HTTP 500 avec le message "error".

Points Clés à Comprendre

Rôle des Namespaces : Le namespace dans le fichier DefaultController.cs doit correspondre au namespace global de l'application pour que le contrôleur soit correctement reconnu.

Importance du Routage : Le routage dans ASP.NET Core détermine comment les requêtes HTTP sont mappées aux méthodes du contrôleur. Les annotations telles que [HttpGet("testSuccess")] sont cruciales pour définir ces routes.

Gestion des Réponses HTTP : Chaque méthode du contrôleur renvoie un type IActionResult, ce qui permet de retourner facilement différents types de réponses HTTP.