**Exercice :**

**Implémentation**

**d’une API en .Net Core**



**Sommaire**

[**I)** **Présentation  :** 3](#_Toc19022371)

[**II)** **Niveau 1 : Une question de questions** 4](#_Toc19022372)

[II.1) Objectif : 4](#_Toc19022373)

[II.2) Autre attente : 5](#_Toc19022374)

[II.3) Comment débuter ? 6](#_Toc19022375)

# **Présentation  :**

L’objectif de cet exercice va être d’implémenter une solution permettant de générer une API Rest Full en .Net Core, le langage utilisé étant C#.

Sachant que vous n’avez probablement jamais implémenté une API de votre vie, sachez qu’une API Rest est dite « RestFull » lorsque toutes les opérations du CRUD (Create, Read, Update, Delete) peuvent être exécuté via l’API. Ceci vous sera ré expliquer lors de la présentation du niveau 1 de l’Exercice.

Une API permet d’éxécuter un traitement sur une donnée que ce soit en lecture ou en écriture. Pour appeler les méthodes de l’Url, il suffira ainsi d’écrire une url sur le navigateur.

Par exemple dans le cadre de notre exercice, l’url suivante « http://localhost:55124/api/question/getAll » permettra de récupérer la liste des questions qui sera présente dans un document Json.

C’est ce que nous allons vous demander d’implémenter en vous guidant progressivement au cours de cet exercice.

Je tiens à vous rappeler que l’objectif final est d’allumer toutes les diodes en vert qui correspondent aux tests effectués pour l’exercice.

Alors commençons dés maintenant à créer notre API !

# **Niveau 1 : Une question de questions**

Pour commencer, sachez qu’il va falloir créer une API permettant de traiter une liste de questions. Les autres niveaux (2 et 3) devant vous permettre d’arriver rapidement à une architecture permettant de gérer plusieurs quizz avec des utilisateurs qui répondent à ces derniers.

Pour gérer les données, dans un premier temps, vous aurez à votre disposition un document Json, « Questions.json ». Le document sera dans sa version originale via un 2ème document « Questions\_initial.json », ce qui va permettre de récupérer la version initiale du document Json après avoir commencer à tester votre API. Elle sera aussi utile pour les tests, il est préférable de ne pas y toucher.

Ensuite, vous devez prendre connaissance des différentes requêtes possibles via une API Restful.

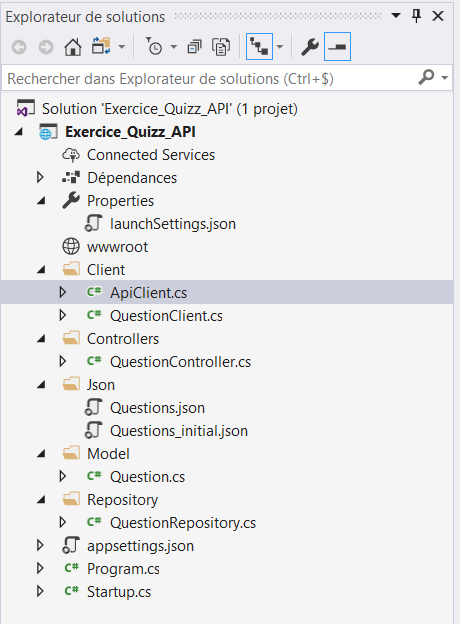
## II.1) Objectif :

En effet, l’objectif dans cette première partie de l’exercice est :

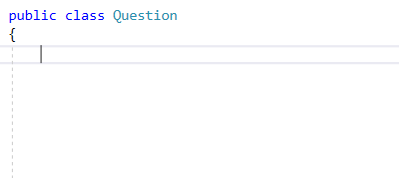
* De récupérer l’ensemble des questions contenus dans le document « Question.json » via une requête de type « GET ALL ». La liste des questions devra être accessible via l’url suivante : « http://localhost:55124/api/question/getAll ». Il s’agira du R pour « Read » du CRUD
* Lire une question précise de l’API via une requête de type « GET » avec le numéro d’identification de la question. Par exemple, si on veut récupérer l’intitulé de la question 3, elle devra être accessible en utilisant l’url suivante : « http://localhost:55124/api/question/get/3 ». Il s’agira là aussi du R du CRUD pour « Read » ou encore lecture.
* Ajouter une question dans le document « Question.json » via une requête de type POST. Par exemple si on veut ajouter une question, il faudra fournir l’intitulé de la question avec la réponse, au format Json. Pour cela, il faudra utiliser l’url suivante : « http://localhost:55124/api/question/addQuestion ». Il s’agira du C du CRUD pour « Create ».
* Modifier une question dans le document « Question.json » via les requêtes de type PUT avec le numéro d’identification de la question. Par exemple, si on veut modifier la question 3, elle devra être accessible en utilisant l’url suivante : « http://localhost:55124/api/question/updateQuestion/3 ». Il faudra, comme pour l’ajout d’une question, préciser l’intitulé de la question mais aussi la réponse de la question que l’on voudra modifier. Il s’agira du U du CRUD pour « Update » ou encore mise à jour.
* Supprimer une question dans le document « Question.json » via les requêtes de type DELETE avec là aussi le numéro d’identification de la question. Il faudra bien vérifier au préalable que la question n’a pas déjà été supprimé ou n’existe pas. Par exemple, si on veut supprimer la question 3, il faudra utiliser l’url suivante : « http://localhost:55124/api/question/deleteQuestion/3 ». Il s’agira du D du CRUD pour « Delete » ou encore Suppression.

## II.2) Autre attente :

Lorsque vous allez récupérer le document, la structure de la solution devrait ressembler à ceci :



Et quand la classe Question.cs est chargée, elle ressemble à ceci :



L’objectif est d’ajouter les paramètres de la classe « Question.cs » qui seront au nombre de 3, en lecture et en écriture, ce qui est important c’est leur nom et leur type :

* QuestionId : de type int,
* QuestionIntitule : de type string,
* Answer : de type string.

***(à ajouter) : Procédure pour lancer le projet de test et l’api simultanément (pas encore testé au 10/09/2019).***

Si vous vous sentez prêt, vous pouvez commencer l’exercice dés maintenant, vous possédez suffisament d’informations pour le faire.

Néanmoins si ce n’est pas le cas, pour vous aider dans cette tâche, voici plus de précisions sur les objectifs à atteindre.

## II.3) Comment débuter ?

1. Ouvrez le dossier « ExerciceAPIQuizzTest » dans Visual Studio.
2. Une fois ouvert, dans l’explorateur de Solution, vous verrez une solution « ProjetTest>>.
3. Déroulez le contenu de la solution et ouvrez les fichiers suivants : « QuestionController.cs », « Question.cs », « QuestionRepository.cs », « QuestionClient »
4. Dans un premier temps nous allons nous intéresser au contenu du fichier Question.cs, et établir un modèle objet pour ce dernier en se basant sur le fichier Json.



Il faut reprendre les noms des clés contenus dans document « Question.json » en faisant juste attention de remplacer chaque 1er lettre par une majuscule, ainsi id devient Id en paramètre.

Optionel :

Une fois cela fait, il faudra lui créer un constructeur afin de préciser les propriétés nécessaires à l’ajout d’une nouvelle question.

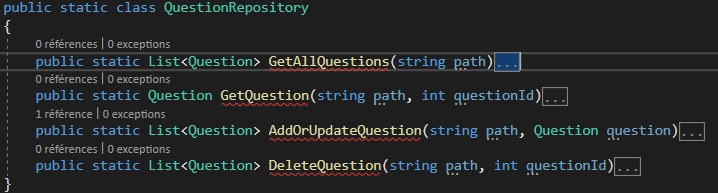
**Une fois cela fait, vous avez votre modèle objet prêt à l’emploi et vous êtes prêt à implémenter l’API Restful !**

1. Il faudra ensuite modifier la classe permettant d’accéder aux données, la classe « QuestionRepository.cs »

Pour ce faire vous devez donc compléter les méthodes des autres fichiers

Pour commencer il va falloir modifier le fichier « QuestionRepository.cs »

Dans le fichier vous voyez diverses méthodes déjà présentes mais vide :

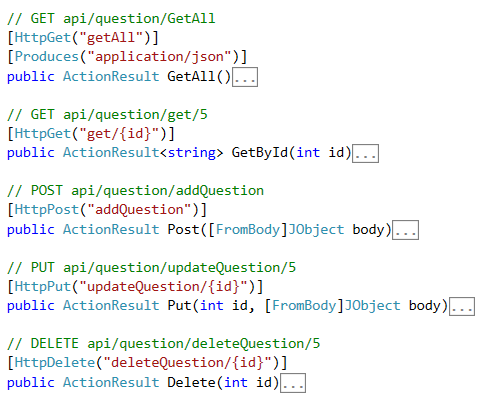


Ces méthodes ont pour rôle d’effectuer la requête du même nom lors de leur appel.

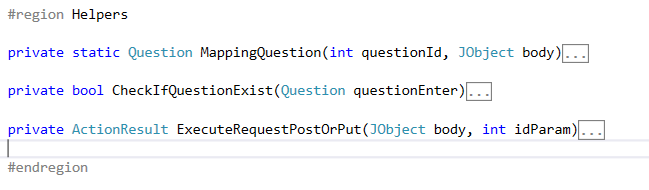
Vous devez donc implémenter les différents types de requête dans ce fichier dans la méthode du nom correspondant afin de finir la mise en place de l’API Restful.

1. Une fois toutes les méthodes complétées, vous devez alors mettre en place l’interaction des méthodes du repository avec celles de votre controller, afin de compléter l’implémentation de l’API Restful

Rendez vous donc dans le fichier « QuestionController.cs » et complétez les méthodes suivantes :



Pour information, une région s’intitulant « Helpers » existe dans le fichier « QuestionController.cs ». Les méthodes, à compléter sont ici à titre indicatif pour vous donner une idée du travail de factorisation à effectuer.



La factorisation consiste à appliquer un principe fondamentale de la programme, « DRY » pour « Don’t Repeat YourSelf », pas forcément facile à maîtriser et qui consiste à ne pas répéter son code ou du moins à le répéter le moins possible.

Enfin dernier conseil, bien lire le nom des méthodes, ceci permet de savoir quelle est l’objectif de la méthode, ce qu’elle est censé faire.

Nous vous avons fourni assez d’information pour commencer l’exercice de niveau 1, n’oubliez pas que vous pouvez me poser toutes les questions concernant cette exercice et qui vous passe par la tête, n’hésitez pas !