

**SESSION JANVIER 2020**

***Bachelier en Informatique de Gestion***

*Catégorie économique*

Nom : UE : 301

Prénom : AA : ECIG3B01IG301A

Groupe : Bloc : 3 N° d'ordre :

Nom de l’enseignant : PALERMO O.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Gestionnaire de questions à choix multiples

1. Objectif

L’objectif de cet examen sera de créer une application Web permettant à l’utilisateur de créer des QCM. Les utilisateurs pourront créer leur propre QCM et seront exposés à plusieurs choix pour répondre à une question.

Tout naturellement, il sera possible de permettre à l'utilisateur de tester ses connaissances grâce aux questions qui seront sauvegardées dans la base de données. Pour chaque question, seule une réponse est possible. La vérification de l'exactitude de sa réponse sera exécutée avec l'aide d'un bouton qui se trouvera en dessous des propositions de chaque question[[1]](#footnote-1). S'il trouve la bonne réponse, vous le lui signalerez. Dans le cas échéant, vous lui soulignerez son erreur tout en lui transmettant une explication.

Il sera possible de créer des questions et de les lier à des propositions. Lorsque l'utilisateur souhaite créer une question, il est obligé de fournir une liste initiale de propositions.

Votre logiciel ne permettra pas l'ajout de choix de réponses.

Pour rappel, une question à choix multiples (soit QCM) est composée d'une question et de plusieurs propositions de réponses.

La création de l'application se déroulera en deux phases : un serveur développé en C# et ASP.NET proposera un Web API pour exposer ou modifier les informations contenues de la base de données (en SQL Server). Quant à l’interface graphique client, celle-ci sera codée avec l’aide du framework Angular.

1. Consignes pour rendre les projets

Deux projets seront créés : l’un en C#, ASP.NET, SQL Server[[2]](#footnote-2) et l’autre en Angular. Ceux-ci devront respectivement se trouver dans deux dossiers compressés "examenASPNOMPrenom" et "examenAngularNOMPrenom". Dans le dossier "examenASPNOMPrenom", veuillez mettre un fichier JSON où résidera un exemple de donnée qui permettra d'exécuter une requête de type POST vers votre API.

Avant de mettre en ligne vos fichiers sur la plateforme ConnectED, vous prendrez soin de compresser vos dossiers et **d’exclure le dossier node\_modules** de votre projet Angular.

1. Caractéristiques de l’abstraction des données

Une question possède un identifiant et un libellé.

Un choix possède un identifiant et un contenu.

Une proposition possède un identifiant et fait le lien entre une question et un choix. De plus, elle permet également de savoir s'il s'agit de la réponse et donne aussi une explication (l'explication peut être facultative).

Exemple :

Quel est le résultat de cette opération mathématique : 1 + 1 **(correspond au libellé d'une question)**

* 1 **(le contenu "1" provient d'un choix et son association à cette question symbolise une proposition)**
* 2 => c'est la bonne réponse, car il s'agit d'une addition ! (**correspond à la bonne réponse et il y a une explication**. Toutefois cette phrase est cachée tant que l'utilisateur n'aura pas cliqué sur le bouton permettant le test de son choix).
* 3
* 4

1. Points techniques à réaliser

C# & ASP.NET

* Création d’un fichier SQL qui contiendra les instructions pour créer les tables et les remplir ;
* Conception des différents modèles ;
* Développement de DAO pour faciliter l’accès aux informations de la base de données ;
* Proposition d’un Web API qui exposera les points d'entrées suivants :
  + GET sur /api/qcm : permet de lister tous les QCM (les questions et leurs propositions);
  + POST sur /api/qcm : permet de créer un QCM (une question et ses propositions) ;
  + DELETE sur /api/questions/{id} : permet de supprimer une question.

Angular

* Création d’une interface graphique donnant la possibilité à l’utilisateur de tester ses connaissances. Ainsi, l'intégralité des questions et leurs propositions seront affichées.
* Pouvoir laisser l'utilisateur créer de nouvelles questions et les lier à des propositions ou autrement dit, il faut qu'il puisse créer de nouveaux QCM.
* Élaboration d’un « service » pour opérer les appels sur la base de données et des « components » pour ficeler le tout ;
* Utilisation d’une hiérarchie évolutive : « smart + dumb component » accompagnée d’un respect des bonnes pratiques de la gestion des différents dossiers ;
* Intégration de l’ergonomie : recevoir des feedbacks lors des suppressions, ajouts, etc. Le rafraîchissement automatique des données doit être exercé. À savoir : le rafraîchissement manuel est pénalisé !
* Il est à noter qu’il est nécessaire d'indiquer à l'utilisateur le nombre de questions dont il a réussi à trouver la réponse (même chose pour les erreurs) et le nombre de questions.

Quelques captures d’écrans

Ci-dessous se trouve un exemple de réponse lorsque le point d'entrée /api/qcm est contacté.

Une image contenant texte, carte

Description générée automatiquement

Ci-dessous se trouve le "body" d'une requête lorsque l'on souhaite créer une question à travers le point d'entrée /api/qcm :

Une image contenant capture d’écran

Description générée automatiquement

1. Par conséquent, s'il y a 10 questions, il y aura 10 boutons pour vérifier ses choix. [↑](#footnote-ref-1)
2. **Le fichier SQL se trouvera dans le dossier compressé "examenASPNOMPrenom".** [↑](#footnote-ref-2)