

**SESSION Juin 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.

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

# Liste de tâches contextualisée

1. Objectif

L’objectif de cet examen sera de développer une application Web permettant à l’utilisateur de créer des listes de tâches qui se trouvent dans un contexte.

L'utilisateur pourra créer un contexte. Un contexte peut ressembler à ces exemples : création d'un programme, organisation d'une fête, lancement d'une nouvelle année scolaire, etc. Donc, il doit être possible de tous les lister et d'en créer.

Lorsqu'un contexte sera créé, il sera possible d'y créer des listes. Voici quelques exemples de listes : à faire, en cours, terminé, suggestions, etc. Les manipulations possibles sont identiques à celles qui concernent le contexte : on peut les afficher et les créer.

Enfin, dans une liste, on peut y insérer de nouvelles tâches. Là encore, la seule limite est fixée par l'imagination. Il sera une nouvelle fois possible de les lister et d'en créer. À cela, on peut rajouter la suppression.

Ci-dessous se trouve une image qui expose un scénario fictif.

Une image contenant capture d’écran

Description générée automatiquement

En bleu est noté le contexte, en gris se trouvent les listes et enfin, les éléments en orange symbolisent les tâches.

La création de l'application se déroulera en deux phases : un serveur développé en C# et ASP.NET Core 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 Core, SQL Server[[1]](#footnote-1) et l’autre en Angular. Ceux-ci devront respectivement se trouver dans deux dossiers compressés "examenASPNOMPrenom" et "examenAngularNOMPrenom".

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

Un contexte possède un identifiant et un nom.

Une liste possède un identifiant, un nom et il est possible de savoir à quel contexte elle est rattachée.

Une tâche possède un identifiant, un nom et il est possible de savoir à quelle liste elle est rattachée.

1. Points techniques à réaliser

C# & ASP.NET Core

* Création d’un fichier SQL qui contiendra les instructions pour créer les tables et les remplir de données fictives ;
* Conception des différentes classes ;
* 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[[2]](#footnote-2) :
  + GET sur /api/contexts : permet de lister tous les contextes (pour chaque contexte, on renvoie juste son identifiant et son nom) ;
  + POST sur /api/contexts: permet de créer un contexte ;
  + GET sur /api/contexts/{id}/lists : permet de récupérer toutes les listes appartenant au contexte qui possède l'identifiant transmis dans l'url (pour chaque liste, on renvoie juste son identifiant et son nom) ;
  + POST sur /api/contexts/{id}/lists : permet de créer une liste ;
  + GET sur /api/contexts/{id}/lists/{idlist}/tasks : permet de récupérer toutes les tâches (pour chaque tâche, on renvoie juste son identifiant et son nom) ;
  + POST sur /api/contexts/{id}/lists/{idlist}/tasks : permet de créer une tâche ;
  + DELETE sur /api/tasks/{id} : permet de supprimer une tâche.

Angular

* Création d’une interface graphique donnant la possibilité à l’utilisateur d'opérer toutes les fonctionnalités citées plus haut.
* Élaboration d’un ou plusieurs « services » 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 !

1. **Le fichier SQL se trouvera dans le dossier compressé "examenASPNOMPrenom".** [↑](#footnote-ref-1)
2. Le non-respect de la structure des routes sera pénalisé. [↑](#footnote-ref-2)