Suite à votre réponse positive concernant la participation au test technique, voici l’énoncé de l’exercice à renvoyer dans les 72h après réception du mail :

- Objectif : voir la qualité de code fournit, le découpage utilisé entre les différentes classes et la logique employée. Vous serez non pas noté sur le résultat (afficher les valeurs) mais sur le moyen utilisé pour répondre aux questions posées.

- Délais : vous avez 72h maximum pour répondre à ce mail.

- Format autorisé de la réponse à cette évaluation : un lien de partage public (Github, Bitbucket, …) ou en pièces jointes (format .zip, .rar, …).

- Temps estimé pour développer la réponse : 1h

- Enoncé :

1. Faire une application Console en C# qui permet :

o D’afficher une collection de cercles, triangles ou carrés. Chaque cercle, triangle ou carré possède un nom, une superficie et un prix. Il peut par exemple y avoir 2 cercles et un 1 triangle. Voici la façon dont on veut les afficher : § [Cercle|Triangle|Carré] – [Nom] – [Superficie]cm² Exemple d’output console : - Cercle – Nom1 - 12 cm² - Cercle – Nom2 - 15 cm² - Triangle – Nom3 - 16 cm²

2. Après avoir affiché cette collection, on veut pouvoir ajouter un cercle ou un triangle ou un carré à cette collection a. Pour cela l’utilisateur devra taper par exemple : « Cercle MonNom 10 » : le séparateur étant l’espace, le premier segment désigne si c’est un Cercle, triangle ou carré, le suivant le nom et le dernier segment représente la superficie. Si on devait afficher le dernier élément rajouté on aurait ici : Cercle – MonNom – 10 cm² Pour gérer ce rajout, on voudrait passer par une classe « Service » qui aura pour but de parser la chaine saisie par l’utilisateur (dans notre exemple : « Cercle MonNom 10 » et de créer le nouvel élément (Cercle, Triangle ou Carré) à partir de cette chaine saisie puis de l’ajouter à la collection déjà existante.

3. On veut à la suite de cet ajout sauvegarder dans un fichier au format JSon la collection : a. Juste après l’ajout, sauvegarder la collection/liste de cercle, triangle ou carré dans un fichier collection.json au format JSON.

4. Après la sauvegarde dans le fichier, utiliser une classe « Pricer » qui permet d’estimer le coût d’un cercle, triangle ou carré à partir de sa superficie et afficher le coût de tous les éléments de la collection. Voici l’interface de la classe Pricer et son implémentation : public interface IPricer { int Price(int superficie) ; } public class Pricer : IPricer { public int Price(int superficie) { return superficie \* 2 ; } } Exemple d’utilisation : var pricer = new Pricer() ; var superficie = 5 ; var cout = pricer.Price(superficie) ; On veut afficher de cette façon : - Cercle – Nom1 - 12 cm² - Estimation cout : 24€ - Cercle – Nom2 - 15 cm² - Estimation cout : 30€ - Triangle – Nom3 - 16 cm² - Estimation cout : 32€

5. Après l’affichage de l’estimation du cout, refaire la même chose (c’est-à-dire afficher pour toute la collection l’estimation du cout) mais avec un autre Pricer : public class OtherPricer : IPricer { public int Price(int superficie) { return superficie \* 4 ; } } On veut afficher de cette façon : - Cercle – Nom1 - 12 cm² - Estimation cout : 48€ - Cercle – Nom2 - 15 cm² - Estimation cout : 60€ - Triangle – Nom3 - 16 cm² - Estimation cout : 64€