Examen Intra Formatif

# 420-5W5

## Consignes

* Vous avez 1h50 pour compléter l’examen.
* Toute documentation est permise.

À partir des fichiers de départ, suivre les instructions. Le style CSS ne sera pas évalué.

**Attention** : Utilisez **http** et non pas https (pour éviter d’avoir des problèmes de certificats). Vous pouvez lancer **angular** avec **ng serve** simplement (**PAS** de --ssl)

# Question 1 – Guards – 3 points

Il existe 4 pages en tout :

* **Login** (accessible à tous)
* **Home** (accessible à un utilisateur connecté)
* **Dog** (accessible à un utilisateur connecté)
* **Cat** (accessible à un utilisateur connecté qui préfère les chats)

Il faut utiliser **des** **guards** pour limiter l’accès aux pages.

Pour vérifier l’accès avec un utilisateur connecté, il faut simplement sélectionner un utilisateur et se connecter dans la page de login.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

C’est possible de se déconnecter en cliquant sur déconnecter.

Une image contenant texte, Police, capture d’écran, logo

Description générée automatiquement

1. Un utilisateur est redirigé vers la page **login** s’il n’est pas connecté et accède à une page autre que login.
2. Lors d’un clic sur le bouton « Acheter un animal ! » qui mène vers la page **cat**, si la propriété **prefercat** a la valeur **false**, il sera redirigé à la page **dog** l’aide d’un Guard. Sinon, il peut accéder à la page **cat**.

# Question 2 – SignalR – 10 points

À partir des fichiers de départs, compléter une application qui permet de :

* Choisir une sorte de pizza
  + Il faut joindre le groupe de notre pizza (Il y a une méthode GetGroupName)
  + Il faut mettre le client à jour avec le prix de la pizza choisi
  + Il faut mettre le client à jour avec le nombre de pizzas et la quantité d’argent de ce groupe
* Ajouter de l’argent pour cette sorte de pizza
  + L’argent que vous ajoutez s’additionne à l’argent pour ce groupe
  + On ajoute 2$ à chaque clique et on doit mettre les clients à jour
* Acheter une pizza
  + Il faut réduire l’argent du groupe de la valeur du prix de la pizza et augmenté le nombre de pizza de 1 et mettre les clients à jour
* Lors d’une connexion/déconnexion, il faut mettre le nombre d’utilisateurs à jour

**Conseil :** Essayer le client avant de modifier le code, il permet déjà de choisir une pizza et de changer son choix.

Votre solution **DOIT** utiliser 4 évènements pour communiquer entre le serveur et le client :

* **UpdateNbUsers** pour mettre à jour le nombre d’utilisateurs connectés
* **UpdateMoney** pour mettre à jour la quantité d’argent
* **UpdateNbPizzasAndMoney** pour mettre à jour le nombre de pizzas et la quantité d’argent
* **UpdatePizzaPrice** pour mettre à jour le prix de la pizza

**Note** : Le système pourrait fonctionner avec un seul event pour mettre à jour le nombre de pizzas et l’argent, mais on demande d’utiliser **UpdateMoney** lorsque l’on ajoute seulement de l’argent et **UpdateNbPizzasAndMoney** le reste du temps.

**Note :** Vous pouvez simplement faire un appel avec **selectedChoice (type number)** pour les méthodes qui reçoivent un **PizzaChoice** en paramètre sur le serveur. Donc, rien de compliqué !

## Hub

Il faut compléter le code de **PizzaHub.cs .**

Les méthodes sont déjà en place, mais il faut ajouter la logique.

Pour ajouter de l’argent et acheter des pizzas, utilisez simplement \_pizzaManager.

Si vous avez besoin d’un nom pour un groupe, utilisez également \_pizzaManager.

## Angular

Il faut compléter le code de **app.component.ts .**

Dans connect(), il faut ajouter la connexion et la gestion des événements.

Pour chacune des autres méthodes, il faut faire la bonne action sur le serveur.

# Question 3 – Reactive Forms – 7 pts

Compléter le formulaire pour qu’il utilise **ReactiveForms**. Les messages d’erreurs doivent s’afficher lors de la validation et ils doivent correctement indiquer les erreurs du formulaire.

* Nom
  + Requis
* Numéro de rue (roadnumber)
  + Requis
  + Entre 1000 et 9999
* Code postal
  + Facultatif (postalcode)
  + Doit correspondre à l’expression régulière "^[A-Z][0-9][A-Z][ ]?[0-9][A-Z][0-9]$"
    - Il existe déjà un validateur pour faire cette vérification.
* Commentaire
  + Facultatif
  + Minimum de 10 **mots** (pas 10 lettres)
    - La méthode string.split(" ") est suffisante pour compter le nombre de mots (même si ajouter un dernier espace compte pour un mot…)
* Formulaire
  + S’il y a un commentaire, il ne doit pas inclure le texte inscrit dans le champ Nom. Sinon, afficher une erreur qui dit : « Le commentaire ne doit pas mentionner le nom de l’utilisateur »

Le bouton doit être **désactivé** lorsque le formulaire n’est pas valide et il doit être **activé** lorsque le formulaire est **valide**.