- ❖ Le patrimoine choisit : Immatériel (musique,costumes,tradition,art..)
  - Chaque element est décrit par :
    Id :String(unique et generer automatiquement)
    Nom :string(pour designer le nom de la patrimoine)
    Photo :string
    4 autres champs OBLIGATOIRE de type booléen,date,string,number
    1 champ de type objet(liste des commantaire,lieux,.... ?chat)
    Autre champs OPTIONNEL si vous voulez(categorie,description,...)

- ❖ Partie back office (admin):
  - 1.1. Consulter, ajouter, modifier, supprimer un patrimoine (que vous avez défini)
  - 1.2. Rechercher un patrimoine
  - 1.3. S'authentifier à l'application
  - 1.4. Modifier son mot de passe
  - 1.5. Ajouter une fonctionnalité complémentaire de votre choix
- ❖ Partie front office(internautes qui pourraient consulter le site)SANS AUTH
  - 2.1. Consulter la liste des patrimoines. L'affichage doit être partiel en ne ciblant que quelque champs (par exemple : nom, photo et catégorie).
  - 2.2. Consulter le détail d'un patrimoine (tous les champs), après sélection dans la liste
  - 2.3. Rechercher un patrimoine selon 2 critères combinables à définir. L'utilisateur peut combiner ou utiliser séparément les 2 critères. Par exemple, il est possible de rechercher un patrimoine dont le nom commence par "a" et dont le prix d'entrée est inférieur à un certain montant, ou simplement de rechercher par nom.(integration d'un message d'erreur si l'utilisateur veut combiner plus que deux critères ?)
    2.4. Consulter une information externe (ex. : photos, météo actuelle, etc.) issue d'une API gratuite de votre choix, à l'emplacement que vous jugerez pertinent. !!!!!!choisir deux API pour deux information externe !!!!!!
  - 2.5. Consulter une page d'informations statiques présentant le site et ses (vous)
  - 2.6. Ajouter une fonctionnalité complémentaire de votre choix

- Exigences techniques : L'application doit définir :
  - Définir une interface typescript pour représenter le patrimoine et d'autres interfaces si nécessaire
  - Contenir un ou plusieurs services pour la gestion de la logique métier.
  - Afficher la liste des patrimoines dans un composant parent, chaque patrimoine étant représenté par un composant enfant.
  - Inclure un ou plusieurs pipes dont au moins un pipe personnalisé - Utiliser le binding de classe ([class])
  - Disposer d'un menu de navigation adapté à l'internaute et d'un menu distinct pour l'administrateur
  - Définir des routes différentes pour l'affichage de la liste et du détail d'un patrimoine (front office).
  - Inclure des routes imbriquées (routes enfants). - Utiliser des formulaires réactifs avec validations.
  - Utiliser Bootstrap, Angular Material.
- De même l'application doit :
  - Pouvoir accéder à un serveur JSON, les données étant sauvegardées dans un fichier json unique.
  - Être versionnée avec Git et déposée sur Github
  - Etre déployée sur Surge.sh ou autre plateforme équivalentz
- Utilisez les signals et les animations d'angular.