1 Contexte

L'entreprise *API fools*, nouvelle startup innovante sur le marché des APIs propose un grand nombre de services ouverts en ligne. Ces services permettent l'affichage de données thématiques. Les technologies et capacités de ces APIs commencent à être désuètes. *API fools* vous demande donc de régénérer leurs APIs avec une technologie plus à jour. Elle souhaite également pouvoir proposer des sites web présentant les informations de leurs apis sous forme visuelle.
Pour obtenir ce marché, il est nécessaire de produire rapidement une/des apis utilisant la technologie Java. La solution devra permettre de stocker et présenter en json + page web les données présentées par une des apis du système existant (`http://filrouge.uha4point0.fr`).
Tout le travail réalisé doit être présenté sur un repository git vous appartenant dans votre espace personnel (`https://git.uha4point0.fr/projects/new`). Cela permettra à l'entreprise de voir le déroulé du projet pour évaluer si un investissement dans cette technologie est envisageable. Pensez à envoyer le lien de votre repo par mail à l'entreprise. Le projet et le repository doivent être appelés nom.prenom-API.
Nota : dans chaque couple d'API se trouve une donnée sous forme de tableau. Celle ci ne doit pas être transcrite dans le travail réalisé.

2 Objectifs principaux

1. proposer un fichier README.MD qui présente comment lancer le projet, décrit très brièvement la structure du model (classes) en place, liste toutes les routes qui peuvent être utilisées une fois le projet lancé, décrit comment accéder à la console H2 et dit quelles étapes de l'examen ont été réalisées (pas de style nécéssaire dans le document).
2. proposer une hiérarchie de classes permettant de stocker les informations de l'api,
3. persister les objets des classes dans une base de données (h2 fortement conseillée),
4. intégrer une procédure ou un fichier SQL qui créé(e) des données au lancement dans l'application (reprendre les données de l'API, pas forcément toutes),
5. rendre disponible les objets par une API sous forme de JSON (le format des données peut être modifié légèrement par rapport à la solution existante, du moment que toute l'information est présente). Idéalement vous avez une route qui présente l'information "image" et une qui présente l'information "user"
6. ajouter à l'API des capacités d'édition des données (ajout, suppression, voir édition des objets).
7. proposer une collection Postman qui permet de manipuler l'API pour afficher/modifier les éléments contenus,

3 Déploiement du site web

1. rendre disponible les objets par des pages web qui présentent l'information clairement (l'éstétisme de la page n'est pas recherché. Il faut par contre que les textes affichés soient accessibles pour une personne lambda),
2. proposer un ou des formulaires permettant de modifier ou ajouter des données depuis les pages web,
3. proposer une route d'api permettant d'afficher l'ensemble des éléments appartenant à une catégorie (e.g. pour une API User + image, elle permet d'afficher toutes les images qui appartiennent à un user spécifique)

4 Cerise sur le gateau

1. proposer une page HTML qui donne le détail de chaque catégorie (e.g. pour une API User + image dans la page présentant les users, on aura un bouton détail a coté de chaque description de l'utilisateur. Le détail affiche le nom de l'utilisateur en grand ainsi que toutes ses images)
2. intégrer du style dans la / les pages réalisées
3. il y a un commit git par tache réalisée
4. une route d'api permet d'ajouter un nouvel element dont les données sont générées aléatoirement à la catégorie sélectionnée (utilisation de java faker - les liens des images peuvent rester vide)