**Questionnaire d’examen**

|  |  |
| --- | --- |
| Evaluation de Javascript : avancé (Vinci) | |
| Titulaire(s) : | Raphaël Baroni, Sébastien Strebelle |
| Année(s) d’études : | Bloc 2 |
| Durée : | 2h30 de 13h30 à 16h00  Pas de sortie ni soumission durant les 60 premières minutes |
| Modalités : | Accès à internet & aux supports de cours |

###### Consignes générales

Vous avez trouvé sur EvalMoodle un fichier **examen\_js.zip**.

Cette archive contient un boilerplate pour chaque question et les ressources à utiliser pour cet examen.

###### Développement de vos applications

Votre dossier d’examen doit se trouver localement sur votre machine : il n’est pas autorisé que celui-ci se trouve sur un disque réseau de Vinci ou sur le cloud (OneDrive, Gitlab, Github ou autres).

Renommez le dossier **examen\_js** en **NOM\_PRENOM**, comme par exemple **UCHIHA\_ITACHI**

Pour chaque question, installez d’abord les packages associés au boilerplate.

Vous pouvez modifier ou ajouter autant de fichiers que nécessaires pour chaque question. N’hésitez pas à installer de nouveaux packages si nécessaires ou à utiliser du code offert dans les support du cours de JS. Ne vous attardez pas sur l’esthétisme de vos pages, cela ne sera pas évalué.

###### Soumission de votre code

Vous devez **effacer** les répertoires **node\_modules** se trouvant dans vos sous-répertoires **./question1** & **./question2**. Si vous ne le faites pas, vous ne pourrez pas soumettre votre projet sur evalMoodle ! Vous devez aussi **effacer** le répertoire **question1-api** !

Créez **un fichier .zip** nommé **NOM\_PRENOM.zip** de votre répertoire **NOM\_PRENOM**.  
**Vérifiez bien votre .zip avant de le poster.**

**Remettez ce fichier .zip sur Evalmoodle dans le devoir Examen de Javascript.**

**Remarques**

**La collaboration entre les étudiants est interdite et sera donc lourdement sanctionnée si elle se produit. Un outil de détection de plagiat sera utilisé.**

La génération de code par des outils d’Intelligence Artificielle tels que Github Copilot et ChatGPT est autorisée.

###### Si une question d’examen ne s’exécute pas ou ne donne aucun résultat fonctionnel, vous aurez d’office moins de 50% des points pour cette question.

###### Objectif

Vous allez développer un frontend et une API en lien avec des grands penseurs.

# Question 1 : Création d’IHM pour SPA (13 points)

Votre application doit permettre d’afficher des données associées à de grands penseurs.

Une API backend a déjà été créée par une autre équipe. Elle vous est fournie dans le dossier **/question1-api** : entrez dans ce dossier et lancez le web service en utilisant les commandes suivantes :

$ *npm i*

$ *npm start*

Voici la documentation d’une opération proposée par le web service que vous allez devoir utiliser :

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Path** | **Action** | **Format** |
| **GET** | **/quotes** | Lire toutes les citations de grands penseurs | Returns : [{thinker:…, quote:.., image:…}, {thinker:…, quote:…, image:…}] |

Vous devez créer les deux pages qui suivent pour votre application web (la page d’accueil ne doit rien afficher, et aucune autre page ne doit être disponible).

## Carrousel de citations

Veuillez créer la **page d’affichage d’un carrousel de citations**, disponible avec l’URI **/quotecarousel** et avec un lien nommé « **Carrousel de citations** » dans la barre de navigation.

En utilisant la requête **GET** du web service, cette page doit afficher tous les 5000 millisecondes une citation.

Pour une citation, vous devez afficher :

* le penseur ou la penseuse (son nom & prénom);
* le texte de la citation ;
* l’image associée à la penseuse ou au penseur: Si le lien donné dans **image** ne correspond pas à un fichier image existant sur le cloud (🤝 tips : soit gérer l’événement "error" associé à une balise image, soit faire un fetch associé à l’URL de l’image), vous devez afficher la photo se trouvant dans **/question1/img/ tingey-injury-law-firm-unsplash-low-res.jpg**

Attention, lorsqu’on affiche la page d’un carrousel de citations, on ne doit jamais afficher deux fois la même citation ! Après avoir affiché toutes les citations une fois, vous devez afficher le message : « Rechargez la page si vous souhaitez réafficher le carrousel des citations ! ».

Voici un exemple illustratif de la manière dont votre application web pourrait être conçue :

|  |  |
| --- | --- |
| Figure 1- Affichage de la 1ère citation | Figure 2 - Affichage de la 2ème citation après 5000ms |
| Figure 1- Après l’affichage de la dernière citation |  |

## Mettre à jour la configuration du carrousel

Veuillez créer la **page de configuration du carrousel**, disponible avec l’URI **/manage-carousel** et avec un lien nommé « **Configuration du carrousel** » dans la barre de navigation.

Cette page doit offrir un formulaire permettant de paramétrer le nombre de millisecondes entre chaque affichage d’une citation. Par défaut, le nombre de millisecondes est de 5000.

Veuillez faire en sorte qu’une fois que vous avez soumis le formulaire, l’utilisateur soit **redirigé** vers la **page d’affichage d’un carrousel de citations** **(/quotecarousel**) et que la vitesse d’affichage des citations s’adapte à la valeur entrée dans le formulaire.

Voici un exemple illustratif de la manière dont votre application web pourrait être conçue :

|  |  |
| --- | --- |
| Figure 1- Affichage du formulaire | Figure 2 - Affichage du carrousel de citations après avoir soumis le formulaire |
| Figure 1- Après l’affichage de la dernière citation |  |

# Question 2 : Création de services web (7 points)

Vous allez développer un web service permettant de gérer des évaluations de citations.

Le boilerplate pour cette question se trouve dans le dossier **/question2**. Il correspond au boilerplate *jwt-api-boilerplate* vu au cours.

Veuillez offrir une opération permettant d’évaluer les citations qui sont données dans le fichier **question2/data/quotes.json**. Vous devez choisir un chemin pour atteindre cette opération, une méthode http et des paramètres qui respectent les conventions REST.

Cette opération doit être protégée par un token JWT.

Lors d’une requête demandant l’enregistrement de l’évaluation d’une citation, le client envoie :

* l’identifiant d’une citation,
* un score de 0 à 10,
* un token contenant son username.

Vous devez renvoyer un « status code » approprié :

* Si le score n’est pas compris entre 0 et 10 ;
* Si la citation demandée n’existe pas ;
* Si l’utilisateur a déjà fait une évaluation pour cette citation.

Si tout est OK, vous devez enregistrer de manière persistante l’évaluation au sein de **/question2/data/quotes.json** ou au sein d’un autre fichier de votre choix (vous êtes libre de décider !) le résultat de l’évaluation :

* en indiquant le username (qui provient du token) ;
* en reprenant l’identifiant de la citation ;
* en y ajoutant le score.

La réponse de l’opération d’enregistrement de l’évaluation doit renvoyer ces informations (les valeurs des propriétés sont juste un exemple, il est normal si vous avez d’autres valeurs) :  
{"idQuote":1, "username" : "ladyBird", "score":8}

Les seules autres opérations qui doivent être disponibles dans votre web service sont les opérations POST /auths/register et POST /auths/login.

Vous devez également ajouter dans le fichier **/question2/REST Client/tests.http** toutes les requêtes nécessaires pour tester l’API afin de :

1. Créer une évaluation en envoyant un **score correct** pour l’utilisatrice nommée **pinkLady** ;
2. Créer une évaluation en envoyant un **score correct** pour l’utilisateur nommé **greenMan** ;
3. Tenter de créer une évaluation en envoyant un **mauvais score**pour l’utilisatrice nommée **pinkLady ;**
4. Tenter de créer une évaluation pour une citation qui n’existe pas pour l’utilisatrice **pinkLady** ;
5. Tenter de créer une évaluation d’une citation déjà évaluée par l’utilisateur **greenMan**;
6. Tenter de créer une évaluation d’une citation pour un utilisateur anonyme (non authentifié).