**Questionnaire d'examen**

| Evaluation de JS / Node.js (Vinci) | |
|---|---|
| Titulaire(s) : | Raphaël Baroni, Sébastien Strebelle |
| Année(s) d'études : | Bloc 2 |
| Durée : | 3h de 13h30 à 16h30 ; 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 3 répertoires **node_modules** se trouvant dans vos sous-répertoires **./question1**, **./question2/spa** & **./question2/frontend**. Si vous ne le faites pas, vous ne pourrez pas soumettre votre projet sur evalMoodle !
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.**

**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.**

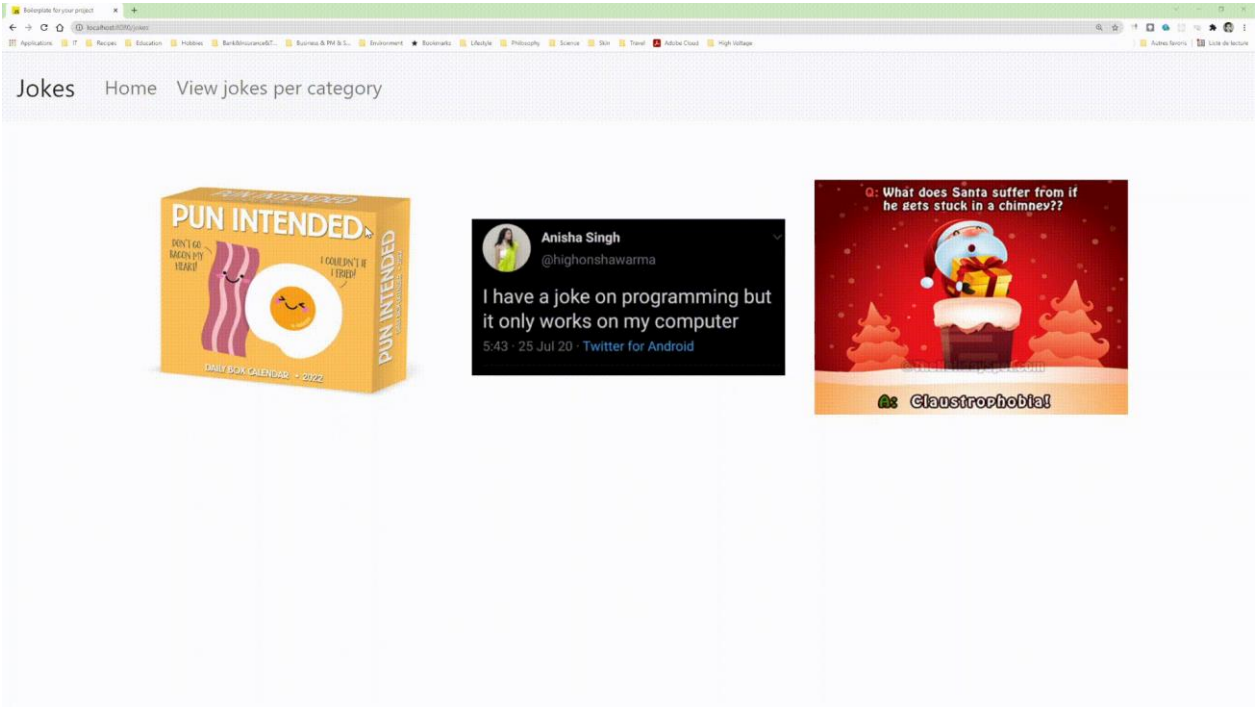
## Objectif

Nous allons développer deux frontends et une API permettant d'afficher et gérer des blagues.

# 1 Question 1 : frontend only (8 points)

Nous souhaitons affichez des blagues de manière interactive.

Voici un exemple de ce à quoi pourrait ressembler votre application pour la question1 :




Le boilerplate pour cette question se trouve dans **/question1**.

Vous trouverez trois images dans le répertoire **/question1/src/img** ainsi qu'un fichier contenant des blagues au format JSON dans le fichier **/question1/src/utils/jokes.js**

Veuillez créer une page affichant les 3 images offertes. Cet page doit être accessible lorsqu'on clique sur un bouton (ou un élément d'une barre de navigation) appelé « View jokes per category » et via cette URI « /jokes ».

Lors d'un clic sur une des trois images :

- Faite disparaître les 3 images ;
- Faite apparaître de manière aléatoire une des blagues associées à la catégorie sélectionnée ; chaque image correspond à une catégorie (« Programming », « Pun » ou « Christmas »), le nom de la catégorie se retrouvant dans le nom de l'image :
- Lors de l'affichage d'une blague, affichez d'abord la question associée à la blague ;
- Puis 5 secondes plus tard, la réponse associée à la blague ainsi qu'un bouton « Reselect a joke » ;

Lors du clic sur le bouton « Reselect a joke », affichez à nouveau les trois images permettant de faire apparaître une blague de manière aléatoire.

Contraintes d'implémentation :

- Veuillez utiliser le fichier **/question1/src/utils/jokes.js** qui reprend des blagues par catégorie. Vous devez mettre à jour ce fichier pour pouvoir l'utiliser dans votre code JS comme module.

# 2 Question 2 : SPA (12 points)

## 2.1 RESTful API

Le boilerplate pour l'API de cette question se trouve dans **/question2/api**. Nous avons généré pour vous une application Express et rajouté un fichier **utils/json.js**.

Au sein de **/question2/api**, vous devez créer une RESTful API qui permettra de gérer des blagues simples.
NB : Une blague simple est une simple histoire drôle, qui ne se présente pas sous forme d'une question et d'une réponse.

Veuillez créer ces trois opérations :

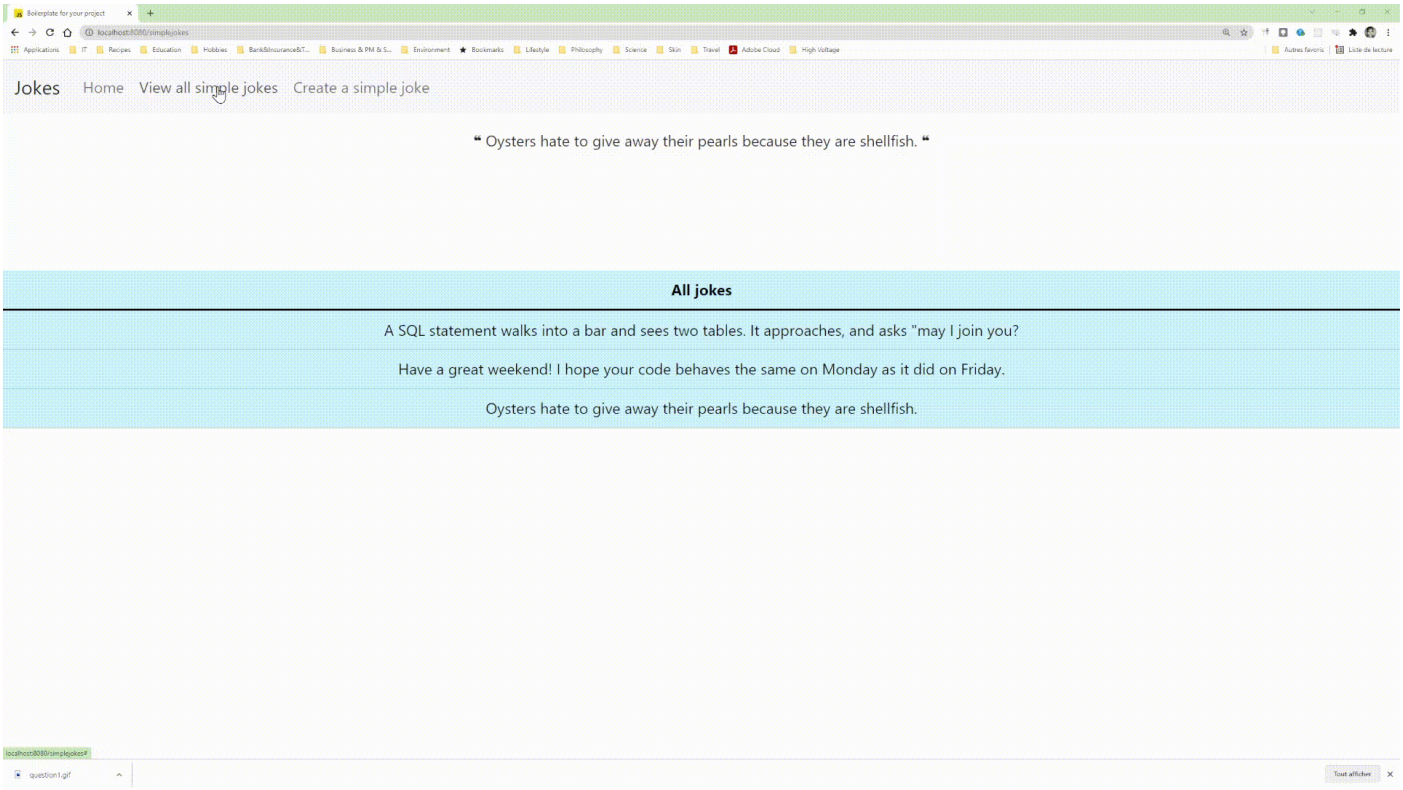
- Lecture de toutes les blagues simples ;
- Création d'une blague simple ; Vous devez générer l'identifiant d'une blague simple à l'aide du package **uuid** ;
- Lecture d'une blague simple de manière aléatoire ; l'API renvoie de manière aléatoire une des blagues.

Contraintes d'implémentation :

- Les demandes qui sont créées doivent persister et donc survivre au redémarrage de votre application. Vous pouvez faire persister les demandes côté serveur de la manière que vous voulez, du moment que cela soit en JSON. N'hésitez pas à utiliser **utils/json.js**.
- Mettez à jour le boilerplate se trouvant dans **/question2/api** afin que celui-ci ne contienne plus de routeurs & routes superflues.
- Aucune autorisation JWT est nécessaire.
- Quand aucune blague n'a été créé via l'API, le fichier des blagues (en JSON) ne peut pas exister.
- Veuillez faire en sorte que ces 3 blagues soient renvoyées par défaut si aucune blague n'a été créé via l'API :
    - A SQL statement walks into a bar and sees two tables. It approaches, and asks "may I join you?
    - Have a great weekend! I hope your code behaves the same on Monday as it did on Friday.
    - Oysters hate to give away their pearls because they are shellfish.

## 2.2 Frontend de la SPA

Nous souhaitons consommer l'API développée dans **/question2/api** pour afficher & créer des blagues simples. Voilà à quoi pourrait ressembler votre application Web pour le frontend :




Le boilerplate pour cette question se trouve dans **/question2/frontend**.

Veuillez créer une page :

- qui doit être accessible lorsqu'on clique sur un bouton (ou un élément d'une barre de navigation) appelé « View all simple jokes » et via cette URI « /simplejokes ».
- qui affiche dynamiquement toutes les blagues simples renvoyées par l'API développée dans **/question2/api**.

Veuillez créer une autre page :

- qui doit être accessible lorsqu'on clique sur un bouton (ou un élément d'une barre de navigation) appelé « Create simple joke » et via cette URI « /addsimplejoke ».
- qui affiche un formulaire de création d'une blague simple ;
- qui consomme l'API développée dans **/question2/api** pour la création d'une blague simple ;
- qui redirige vers la page affichant toutes les blagues simples une fois l'ajout réussi (URI « /simplejokes »).

Veuillez créer un composant JS :

- qui affiche tous les 5 secondes une blague simple renvoyée aléatoirement par l'API développée dans dans **/question2/api**.
- qui se trouve dans un module nommé **RandomJokes.js**
- qui est appelé dans le header de toutes les pages.