Test technique pour le poste de
Développeur Web Full Stack

Objectifs

Ce test technique est une des étapes du processus de votre recrutement pour vous évaluer sur les points suivants :

- La réalisation de l'application.
- Le respect des délais.
- La structure de votre code.
- L'optimisation de votre code.
- L'utilisation des bonnes pratiques.
- La méthode de travail.

Prérequis

Afin de pouvoir passer ce test, il est nécessaire d'avoir des bonnes connaissances dans les technologies suivantes :

- Node.js
- Express.js
- React et/ou Next.js
- MongoDB
- Git
- Compétences dans le déploiement.
- Pas de Nest.js.
- Pas de TypeScript.

Important

- Le produit final doit être ajouté aux deux référentiels Github (Backend et Frontend) fournis avec ce document.
- Votre méthode d'utilisation de Git (push, commit, etc.) sera prise en compte dans l'évaluation.
- Utilisez les logos fournis avec ce document dans le fichier **assets.zip**.

Description

Le test technique consiste à créer une simple application qui sera utilisée par deux types d'utilisateurs : les administrateurs et les clients. Les administrateurs peuvent se connecter à l'application et insérer de nouveaux utilisateurs et logos. Les clients peuvent à leur tour créer un compte, obtenir une clé pour utiliser l'API et consulter la liste des logos.

Backend

Initialisation

- Clonez le référentiel Github du Backend fourni avec ce document.
- Initialisez votre projet Node.js à l'aide de la version la plus récente du framework **Express.js**.
- Configurez votre base de données MongoDB à l'aide de la dernière version de **Mongoose**.

Modélisation

- Créez les modèles correspondant aux collections suivantes :

Users

```
{
  _id(ObjectId)₁: id de l'utilisateur.
  email(String)₁: email de l'utilisateur.
  password(String): mot de passe de l'utilisateur.
  isAdmin(Boolean): indique si l'utilisateur est un admin ou pas.

}
```

Logos

```
{
  _id(ObjectId)₁: id du logo.
  svg(String): code svg du logo.
  file(String): chemin de fichier source du logo.
}
```

1 : Valeur unique

Important : La vérification des contraintes doit être effectuée sur chaque attribut au niveau de la base de données, du backend et du frontend.

API REST

Afin de sécuriser l'API, créez les middlewares suivants :

- **checkToken :** vérifie si le JWT token de l'utilisateur qui envoie la requête est valide, sinon un message d'erreur sera envoyé à l'utilisateur.

Le token de l'utilisateur expirera après **une heure**.

- **isAdmin :** Vérifie si l'utilisateur qui envoie la requête est un administrateur, sinon un message d'erreur sera envoyé à l'utilisateur.

- **checkGeneralKey :** Vérifie si la clé API **"x-general-key"** envoyée par l'utilisateur est valide, sinon un message d'erreur sera envoyé à l'utilisateur.

La clé **"x-general-key"** est générée comme suit :

- Dans le côté Frontend, la date courante (en millisecondes) sera récupérée et cryptée à l'aide de la méthode **Advanced Encryption Standard (AES)**.

- La date courante cryptée sera envoyée en tant que "x-general-key" dans la requête.

- Sur le backend, au niveau du middleware, la clé "x-general-key" sera déchiffrée à l'aide de la méthode AES.

- Si la différence entre la valeur de la date déchiffrée et la date du moment de la réception de la requête au niveau du Backend est inférieure à **1 minute** alors la clé est valide et la requête sera acceptée, sinon un message d'erreur sera envoyé à l'utilisateur.

Exemple :

Date courante (Frontend) : 1659647531356 ms.

Cryptage de la date avec AES (Frontend) : UvRYeGGyjAfa7gYqe6ZNkA==

Envoi de la date cryptée dans **x-general-key** au Backend.

Décryptage de la date avec AES (Backend) : 1659647531356

Cas 1 : Date courante après décryptage (Backend) : 1659647555312 ms

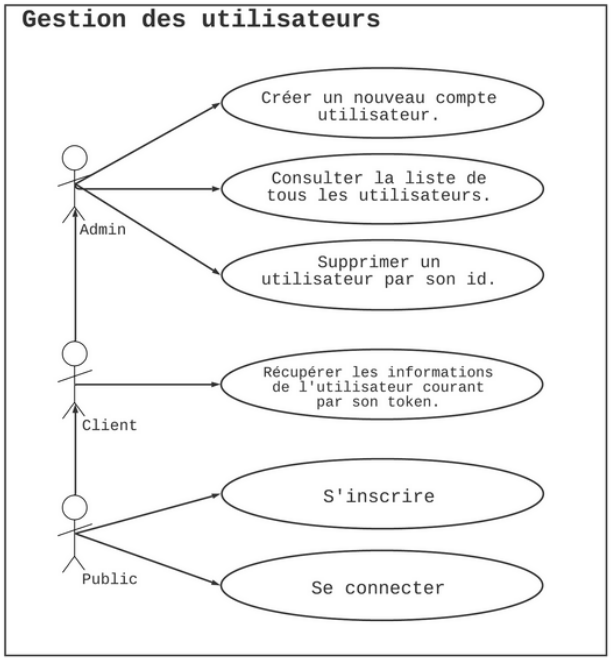
1659647555312 - 1659647531356 = 23956 ms = 0.4 min <= 1 (**x-general-key valide**).

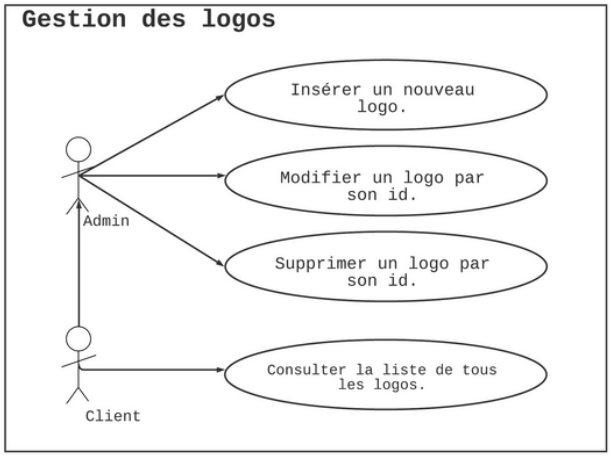
Cas 2 : Date courante après décryptage (Backend) : 1659648048906 ms

1659648048906 - 1659647531356 = 517550 ms = 8,6 min > 1 (**x-general-key non valide**).

Suggestion : Vous pouvez utiliser le package NPM **crypto-js** pour chiffrer/déchiffrer la valeur de la clé "x-general-key".

Créez les routes, les contrôleurs et les services qui permettent de répondre aux besoins présentés dans les diagrammes de cas d'utilisation suivants :





- Chaque cas d'utilisation dans les diagrammes de cas d'utilisation correspond à une route.

- Toutes les routes de l'administrateur et du client sont protégées par JWT Token.

- Toutes les routes sont protégées par la clé **"x-general-key"**.

Documentation

Utilisez **Postman** pour créer la documentation de l'API REST et ajoutez un lien vers la documentation dans le fichier Readme.me du répertoire Github.

Déploiement

Déployez le Backend et ajoutez l'URL dans le fichier Readme.me du répertoire Github.

Suggestions :

- Vous pouvez utiliser **MongoDB Atlas Shared** pour déployer la base de données.
- Vous pouvez utiliser **Heroku** pour le déploiement du Backend.

Frontend

Initialisation

- Clonez le référentiel Github du Frontend fourni avec ce document.
- Initialisez votre projet React ou Next.js.
- Configurez un cadre de style comme Tailwind ou Bootstrap (l'utilisation de CSS directement est un plus).

Implémentation du design

- Ouvrez le design **Figma** à partir de ce lien.
- Créez la page d'accueil.
- Créez les fenêtres de connexion et d'inscription.
- Stockez le token de l'utilisateur authentifié dans un cookie **"auth-token"** qui expire dans une heure. Si l'utilisateur quitte le navigateur sans se déconnecter et le rouvre avant l'expiration du cookie, il restera connecté.
- Créez la page qui permet de consulter la liste des logos (cette page s'affiche uniquement pour les utilisateurs authentifiés) :

  - Récupérez la liste des logos à partir du Backend.

  - Affichez la liste des logos dans le carrousel.

  - Affichez les couleurs et les polices utilisées pour chaque logo.

Optimisation pour les moteurs de recherche

- Configurez et ajoutez le fichier **sitemap.xml** à l'application.
- Configurez et ajoutez le fichier **robots.txt** à l'application.

Déploiement

Déployez le Frontend et ajoutez l'url dans le fichier Readme.me du répertoire Github.

Suggestions : Vous pouvez utiliser Netlify ou Vercel pour le déploiement du Frontend.

Performances

Utilisez l'outil **Google Lighthouse** pour évaluer les performances de l'application après la mise en production. L'objectif est d'obtenir des résultats supérieurs ou égaux aux résultats suivants (dans la version desktop et mobile) :