# 420-15D-FX TP3 (40%)

---

## Travail pratique #3 : API REST et Vues.js

Pondération : 40%
Travail à faire **en équipe de deux personnes**
Date de remise : **Jeudi le 20 mai avant minuit**

## Contexte

Ce travail pratique porte sur le framework **Vue.js** et sur la complétion d'une **API REST** de base. La documentation devra être réalisée avec Swagger.

**L'API** devra être hébergée sur un serveur distant.
La partie "front" en **Vue.js** fonctionne en local et n'a pas besoin d'être hébergée.

L'API devra respecter un jeu de tests automatisés et exhaustifs réalisé par les étudiants.

Les technologies suivantes sont utilisées : Vue.js, Node.js, Express.js, MongoDB, Mongoose et Swagger.

## Description

Vous avez été engagé par la compagnie *Deep Rock Galactic* pour créer une application leur permettant de gérer les missions de collecte de ressources interplanétaires (whaa félicitation).

*Deep rock Galactic* engage des 'mineurs' (avec des pioches, pas des fausses cartes) indépendants et leur propose des contrats de collecte partout à travers la galaxie.

Une fois un contrat accepté par un mineur, celui-ci voyage jusqu'à la planète concernée, y récolte les ressources demandées et empoche une prime une fois terminé.

# Réservation de contrat

Le mineur doit pouvoir rechercher et réserver un contrat à partir d'un formulaire.

Les critères de recherche sont :

- La date d'échéance
- La planète
- Le niveau de danger (!)
- La prime ($)

Une liste des contrats disponibles doit s'afficher et le mineur doit pouvoir sélectionner un contrat ou retourner à la recherche pour modifier les critères.

La liste doit comporter :

- Une image de la planète
- Le nom de la ressource à collecter
- La quantité (en tonne) de ressource à collecter
- La date d'échéance
- La prime ($)
- Le niveau de danger (!)

Si le mineur est connecté, un élément visuel doit être affiché afin qu'il puisse sélectionner un contrat.

Une fois le contrat sélectionné, un écran récapitulatif permet de vérifier toutes les informations du contrat.

Un bouton permet au mineur de réserver le contrat.

Un message de remerciement doit être affiché.

## Autres informations

Le mineur doit pouvoir **s'inscrire** sur le site en saisissant un courriel, un nom et un mot de passe.

Le mineur doit pouvoir **se connecter** sur le site en saisissant un courriel et un mot de passe.

Pensez à valider les informations des formulaires "côté front et côté back".
L'authentification doit être gérée grâce à un **jeton JWT**.
Le mineur connecté doit pouvoir accéder à une page de **statistiques** lui affichant l'historique des missions complétées ainsi que le total d'argent récolté dans sa carrière ($).

Un mineur qui n'est pas connecté peut effectuer des recherches sur le site mais il doit être connecté pour pouvoir effectuer une réservation de contrat.

Il faut prévoir un menu pour pouvoir accéder aux pages de connexion, d'inscription, de recherche et de statistiques.

La page de recherche peut être la page d'accueil.

Le nom des images est enregistré dans la base de données (1.jpg, 2.jpg...) Vous choisirez une dizaine d'images de planètes qui seront hébergées "côté front" et vous les nommerez correctement.

## Niveaux utilisateurs

Dans la table "mineurs", un niveau est requis.

Un mineur normal a un **niveau 1**. Il peut seulement s'inscrire, se connecter, faire une réservation et afficher ses statistiques.

Un mineur administrateur a un **niveau 2.** Il peut faire toutes les actions offertes par l'API: Créer / modifier / supprimer une planète, un contrat, un mineur.
Toutes ces actions pourront être effectuées dans Postman.

## API

L'api doit être réalisée avec Node.js.

Vous devez compléter le code qui est fourni.

Vous pouvez modifier le code existant sauf pour les routes.

Toutes les routes définies dans le dossier routes doivent être implémentées même si elles ne sont pas toutes utilisées dans l'application et dans Postman.

## Vue.js

La partie "front" doit être codée avec le framework Vue.js et doit permettre à un mineur de s’inscrire, de rechercher et réserver un contrat ainsi que de consulter ses statistiques.

Vous pouvez utiliser un framework css si vous le souhaitez.

## Tests

Vous devez réaliser un jeu de tests complet avec Postman pour :

- Création d'un compte utilisateur de niveau 1
- Connexion d'un utilisateur de niveau 1
- Recherche d'un contrat
- Réservation d'un contrat
- Création d'un compte utilisateur de niveau 2
- Connexion d'un utilisateur de niveau 2
- Ajout d'une planète
- Suppression d'une planète

La collection de test doit pouvoir être exécutée de manière séquentielle comme pour le TP2.

Les requêtes doivent être dynamiques et utiliser des variables d'environnement.

La base de données doit être dans le même état après l'exécution des tests. Par exemple, si vous créez un contrat, vous devez le supprimer.

Vous fournirez dans la remise une copie de la base de données qui est utilisée pour les tests.

## HATEOAS

Le principe HATEOAS doit être implémenté pour toutes les routes qui sont utilisées par l'application Vue.js. L'application n'aura donc pas connaissance des routes et devra utiliser les urls renvoyées par l'API.

La seule route qui est **connue** par l'application est celle qui permet de faire la recherche de contrat.

## Documentation

Vous devez réaliser la documentation de l'API avec swagger qui sera accessible avec la route GET /api-docs .

## Hébergement

L'API doit être hébergée sur un serveur distant (Heroku).

## Caractéristiques générales du code

Respectez les règles de base de la programmation vues en cours :

- Présentation du code
- Indentation,
- Saut de ligne
- Nommage des fonctions et variables de manière intelligible et raisonné.

- etc.

Une attention particulière sera portée à la qualité du code.

Le code doit être le plus simple possible.

# À remettre

Votre projet doit être remis sur Léa au format zip.

Il doit comporter dans des dossiers séparés :
- le code de l'API
- le code Vue.js
- un export des requêtes Postman au format JSON pour les tests
- un export des variables d'environnement de Postman au format JSON
- un fichier d'export de la base de données au format Mondodb

Dans le dossier zip, vous devez ajouter une page de présentation du travail en ajoutant des informations que vous jugerez utile de fournir.

L'API doit être hébergée sur un serveur distant et accessible pendant deux semaines après la date de la remise.

# Annexe

## Base de données

Structure de la base de données.
Vous pouvez la modifier si vous le souhaitez

**mineur**
- _id
- nom
- email
- motdepasse
- niveau (1 ou 2)

**reservation**
- _id
- mineurId
- contratId
- estTermine

**contrat**

- _id
- planeteId
- prime
- danger
- ressource
- quantiteRessource
- dateExpiration

**planete**

- _id
- nom
- image

## Maquette ? Impressionnez-moi !

L'image ci-dessous est un exemple de maquette pour illustrer le système de recherche, la liste des contrats et le modules pour confirmer réservation (il manque une navigation, la page de connexion/ d’inscription, de statistiques, etc.).

À vous de faire preuve d'originalité pour que ce soit plus beau ;) !

Si vous souhaitez diviser ces fonctionnalités en plusieurs pages, libre à vous, c'est votre front-end !

# DEEP ROCK GALACTIC

DANGER. DARKNESS. DWARVES

## FILTRES

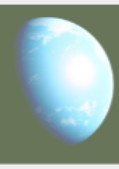
Planète Saturne

Prime Min 0 100 000 000

Date d'échéance Avant 2077-05-04

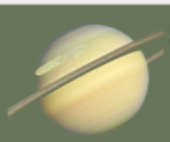
Niveau de danger ⚠⚠⚠⚠

## CONTRATS DISPONIBLES



**TITAN7**
500 000 $
SÉLECTIONNER
10t - Pu
2077-03-01
⚠⚠

**XANDAR**
100 000 $
SÉLECTIONNER
1t - Fe
2078-09-12
⚠



**SATURNE**
1 000 000 $
SÉLECTIONNER
100t - He
2076-02-11
⚠⚠⚠

## DÉTAILS D'UN CONTRATS

| | |
|---|---|
| PLANÈTE | Xandar |
| PRIME | 100 000$ |
| QUANTITÉ | 2 tonnes |
| RESSOURCES | Fer |
| AVANT LE | 2079-03-01 |
| DANGER | Calme (!) |

RÉSERVER