# Création d'une API REST à l'aide de NodeJS et PostgreSQL

## Infos concernant cette activité

- Cette activité est-elle notée ?
  - Oui, la note de cette activité fait ½ de la note de la ressource R307.
- Quand dois-je terminer cette activité?
  - La Semaine du 04/12 (A la fin de la séance suivante, vous devrez présenter le projet)
- Comment puis-je soumettre mon projet ?
  - Ceci est détaillé à la fin de ce document.
- Projet – à réaliser en binômes. (un seul monôme autorisé si nombre impair d'étudiants).

## Prérequis:

- Dans ce projet, vous allez utiliser un jeu de données JSON concernant les prix Nobel :
  - https://cours-info.iut-bm.univ-fcomte.fr/upload/supports/S3/web/cot%20serveur/data/prize.json.zip
  - Téléchargez le fichier en utilisant le lien ci-dessus, il a été mis à jour. Les lauréats 2023 ont été ajoutés.


```
{
  "prizes": [
    {
      "year": "2023",
      "category": "chemistry",
      "laureates": [
        {
          "id": "1029",
          "firstname": "Moungi",
          "surname": "Bawendi",
          "motivation": "\"for the discovery and synthesis of quantum dots\"",
          "share": "3"
        },
        {
          "id": "1030",
          "firstname": "Louis",
          "surname": "Brus",
          "motivation": "\"for the discovery and synthesis of quantum dots\"",
          "share": "3"
        },
        {
          "id": "1031",
          "firstname": "Aleksey",
          "surname": "Yekimov",
          "motivation": "\"for the discovery and synthesis of quantum dots\"",
          "share": "3"
        }
      ]
    },
```

Ce jeu de données représente une liste de prix Nobel attribués dans différentes catégories. Voici une explication détaillée de sa structure :

Racine du JSON : Le document commence par un objet racine qui contient une seule clé, "prizes", associée à une liste d'objets.

Objets "prize" : Chaque objet dans la liste "prizes" représente un prix Nobel pour une année et une catégorie spécifiques. Par exemple, il y a un objet pour le prix Nobel de chimie en 2023, un autre pour le prix Nobel d'économie en 2023, etc.

Détails de chaque prix : Chaque objet "prize" contient plusieurs informations :

- year: L'année d'attribution du prix (par exemple, "2023").
- category: La catégorie du prix Nobel (par exemple, "chemistry" pour la chimie).
- laureates: Une liste d'objets représentant les lauréats de ce prix.

Objets "laureate" : Chaque objet dans la liste "laureates" représente un lauréat du prix Nobel. Il contient les informations suivantes :

- id: Un identifiant unique pour le lauréat.
- firstname: Le prénom du lauréat.
- surname: Le nom de famille du lauréat.
- motivation: La motivation pour laquelle le prix a été attribué, souvent citée entre guillemets.
- share: La part du prix attribuée à ce lauréat, souvent divisée entre plusieurs lauréats.

Par exemple, pour le prix Nobel de chimie en 2023, il y a trois lauréats (Moungi Bawendi, Louis Brus et Aleksey Yekimov), chacun ayant une part égale (1/3) du prix, attribué pour "la découverte et la synthèse de points quantiques".

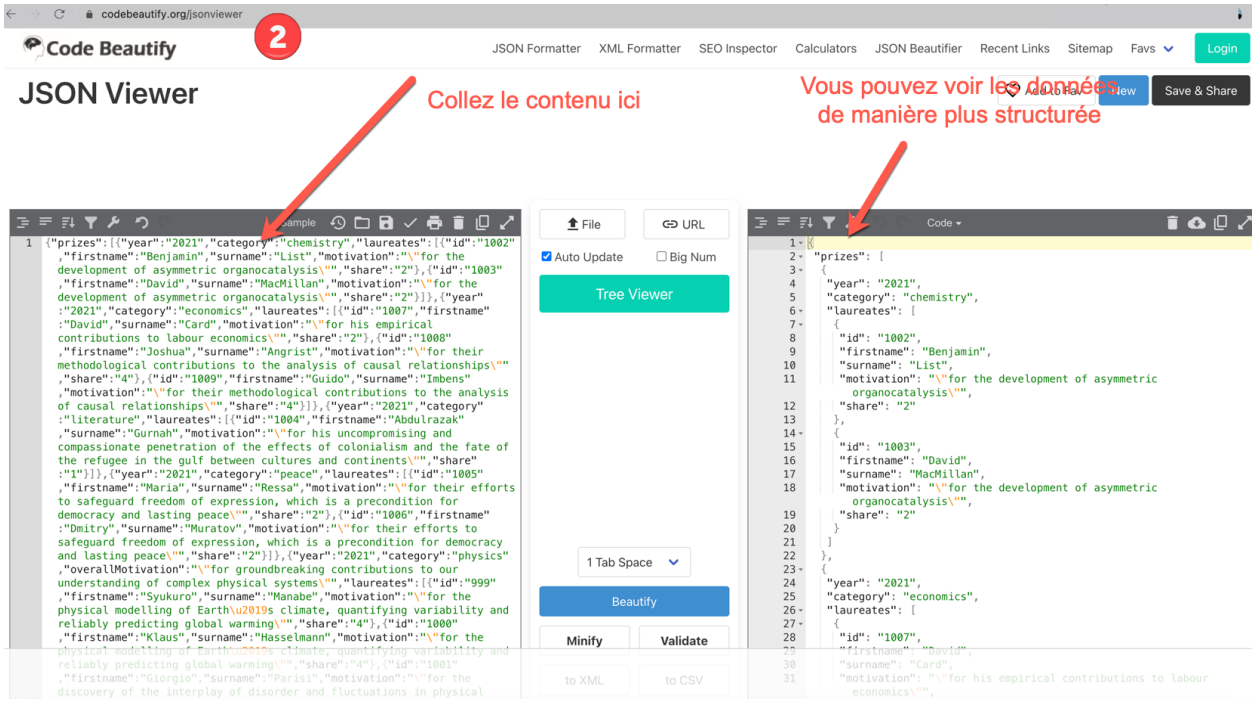
Dans ce projet, vous créerez une API dans laquelle les services (en utilisant le modèle Routers-Controllers-Services) liront les données et stockeront les données dans une base de données PostgreSQL au lieu d'un fichier JSON.

## Base de données initiale : **prize.json**

La base de données à utiliser dans ce projet est un fichier JSON qui contient des informations sur les prix Nobel et les lauréats du prix Nobel. Le nom du fichier est **prize.json**. Vous pouvez le télécharger sur cours-info:

- https://cours-info.iut-bm.univ-fcomte.fr/upload/supports/S3/web/cot%20serveur/data/prize.json.zip

Pour mieux comprendre ce que contient ce fichier, ouvrez-le dans un éditeur de texte, copiez le contenu, ouvrez un site Web Formateur JSON (JSON beautifier) tel que https://codebeautify.org/jsonviewer et collez les données JSON.


Ce fichier JSON contient un tableau "prizes" avec environ 670 objets. Chaque objet contient l'année, la catégorie du prix, et les gagnants stockés à l'intérieur d'un tableau "laureates".

# Tâches à faire :

## TÂCHE 1 : CRÉER UNE BASE DE DONNÉES POSTGRESQL A PARTIR DU FICHIER PRIZE.JSON

Lorsque vous regardez le fichier prize.json :

- Comment pouvez-vous représenter les données de manière structurée ?
- De combien de tables allons-nous avoir besoin pour ce fichier ?
- Quels sont ces tableaux et quelles colonnes contient chaque tableau ?

**Ne demandez pas à votre instructeur quelles tables vous devez créer, ni si le schéma de votre base de données est correct ou non, il s'agit d'une tâche notée et vous devez la faire vous-même. Il n'y a pas une seule bonne réponse à cette question. Deux groupes d'étudiants peuvent avoir des tableaux et des schémas différents, mais les deux peuvent être corrects.**

Dans la première tâche, vous allez créer un **parseur JSON-SQL à l'aide de NodeJS**.

**Dans ce contexte, un "parseur" (parser en anglais) est un programme ou un script écrit en NodeJS qui a pour fonction de lire et d'interpréter le fichier JSON prize.json. Il transforme les données structurées en JSON en un format compatible avec SQL pour qu'elles puissent être insérées dans une base de données PostgreSQL. En d'autres termes, le parseur s'occupe de la conversion des données depuis le format JSON, utilisé pour la représentation des données dans le fichier, vers des commandes SQL qui serviront à insérer ces données dans les tables de votre base de données, en veillant à respecter les associations et à éviter les doublons.**

Après avoir analysé les données JSON, créez les tables qui représentent le mieux ces données de manière structurée tout en tenant compte des colonnes de clé primaire et de clé étrangère.

Après avoir créé les tables, vous devez les remplir avec des données dans le fichier JSON. Créez un parseur NodeJS qui itère dans le tableau JSON et insérez les données dans les tables correspondantes tout en respectant les associations. Assurez-vous de ne pas insérer d'éléments en double dans la base de données.

- Commencez par créer un script SQL **nobelprize.sql** dans lequel vous créez toutes les tables.
- Créez un fichier **parser.js** dans lequel vous implémentez votre parseur. Notez que le fichier **parser.js** ne contient pas de serveur Web et que vous pouvez l'exécuter dans votre terminal: **node parser.js**

# TÂCHE 2 :REST API

- **Votre API doit offrir les fonctionnalités suivantes :**
    - **F1:** Lister tous les lauréats (id, prénom, nom).
    - **F2:** Étant donné un identifiant, affichez les informations du lauréat avec cet identifiant (prénom, nom, les prix remportés).
        - Ex: ID = 6
            - Marie Curie
                - 1911 chemistry in recognition of her services to the…
                - 1903 physics in recognition of the extraordinary services…
    - **F3:** Combien ont remporté plus d'un prix Nobel ?
        - (prénom, nom, nombre de prix gagnés)
        - Ex: Marie Curie, 2
    - **F4:** Lister toutes les catégories des prix nobel
        - Chemistry, economics, etc
    - **F5**: Déterminez quelle catégorie a produit le plus grand nombre de lauréats du prix Nobel.
    - **F6:** Pour chaque année, indiquez combien de lauréats avaient remporté un prix nobel.
        - Ex: 2021, 13
    - **F7:** Afficher toutes les années au cours desquelles aucun prix Nobel n'a été décerné.
    - **F8**: Afficher toutes les années de prix nobel triées par nombre de lauréats ascendant/descendant.
        - ?sort=asc_laureates ⇒ ascendant ⇒ commencer par les années avec le plus petit nombre de lauréats
        - ?sort=desc_laureates ⇒ descendant ⇒ commencer par les années avec le plus grand nombre de lauréats
        - Exclure les années où aucun prix Nobel n'a été décerné

    - **F9:** Supprimer un lauréat avec un identifiant donné.
    - **F10:** Mettre à jour la motivation d'un lauréat avec un identifiant donné dans une année donnée et une catégorie donnée.
    - **F11 :** Filtrer les lauréats par nom de famille similaire
      Votre API doit permettre aux utilisateurs de rechercher des lauréats en utilisant une partie de leur nom de famille. Ce filtre ne nécessite pas que le nom de famille soit exact, mais utilisera une correspondance partielle (comme l'opérateur SQL LIKE). Lorsqu'un utilisateur fait une requête avec une partie d'un nom de famille, l'API renverra une liste de tous les lauréats

ayant un nom de famille similaire, accompagnée de leurs informations pertinentes et du nombre total de prix Nobel qu'ils ont remportés.

        - Par exemple, si un utilisateur recherche le terme "Cur", l'API pourrait retourner des lauréats tels que Marie Curie et Pierre Curie, en indiquant les informations de chaque lauréat et le nombre total de prix qu'ils ont gagnés.
    - **F12:** Pagination des résultats de la liste des lauréats
        - Pour améliorer la performance et la facilité d'utilisation de votre API, implémentez un système de pagination pour la liste des lauréats. Cela signifie que les résultats seront divisés en "**pages**" de taille fixe, et l'utilisateur pourra demander une page spécifique. Vous devrez fournir des paramètres dans l'URL pour la page (par exemple, **page**) et le nombre de résultats par page (par exemple, **limit**).
        - Exemple d'utilisation : GET /<uri>?page=2&limit=10 renverrait la deuxième série de 10 lauréats.

Notes IMPORTANTES:

- Vous pouvez tester votre API en utilisant POSTMAN.
- Aucune vue n'est demandée dans ce projet. Uniquement une API qui renvoie JSON.
- **Vous n'êtes pas autorisé à utiliser un ORM dans cette activité.**

# Comment présenter ce projet ?

## Correction instantanée

Si vous avez terminé le projet avant la fin de la deuxième séance, vous pouvez appeler l'instructeur et il évaluera le projet en place.
Ce qui est recommandé :

- Présenter le script de création des tables et le parseur à la fin de la première séance.
- Présenter l'API à la fin de la deuxième séance.
    - Si vous n'avez pas réussi à créer le parseur et à remplir les tables à la fin de la première séance, appelez l'instructeur et montrez-le-lui dès que vous l'avez terminé lors de la deuxième séance.

L'instructeur vérifiera :

- Votre script sql pour la création de tables
    - Le nom des tables, les colonnes de chaque table et les associations doivent être clairs.

- Votre parseur JSON-SQL, vos tables postgreSQL doivent être remplies avec les données initiales.
    - **Faites attention aux doublons, si l'instructeur constate que vous avez des doublons, des points négatifs seront appliqués à cette tâche.**

- Routes d'API fonctionnelles.
    - L'instructeur peut demander à vérifier les requêtes/le code dans vos services/controlleurs NodeJS.
- L'instructeur peut vous demander d'expliquer la structure de votre projet et le rôle de chaque fichier/dossier.

**Pour clarifier les attentes en matière d'intégrité académique et d'originalité des travaux, il est important de souligner que l'instructeur n'a pas prescrit de noms spécifiques pour les routes (API), ni de schéma détaillé pour la base de données. Cela a été délibérément fait afin que chaque groupe puisse développer son propre schéma, choisir les noms de la base de données, des tables, des colonnes, ainsi que des routes, encourageant ainsi la créativité et l'unicité dans chaque projet.**
**En conséquence, l'instructeur s'attend à une diversité significative entre les travaux des différents groupes. Tout manquement à cette attente, comme la duplication de code, de schéma de base de données, ou le partage de solutions sur des plateformes telles que Discord, entraînera l'attribution d'une note de zéro pour la tâche concernée. Cette démarche vise à promouvoir une approche pédagogique axée sur l'innovation individuelle et le respect des principes académiques.**