

Faculté des Sciences

Master Intelligence Artificielle et Analyse des Données MIAAD

PROJET BIG DATA:

Intitulé :

APPLICATION NEO4J & JAVA

Spring Boot / Angular

Préparé par :

- ✓ KARMOUT Ismail
- ✓ El Goumri Youssef

TABLE DE MATIERE

TABLE DE MATIERE	2
I- Introduction	3
1- Contexte du projet	3
2- Objectif du rapport.....	3
II- Technologies Utilisées	4
1- Java	4
2- Spring Boot	4
3- Angular	4
4- Neo4j	4
III- Conception de la Base de Données Neo4j	5
1- Schéma de la base de données	5
2- Choix de Neo4j pour ce projet.....	8
IV- Implémentation Backend avec Spring Boot.....	8
1- Configuration de Spring Boot avec Neo4j	8
V - Implémentation Frontend avec Angular	12
VI- Conclusion.....	15

I- Introduction

1- Contexte du projet

Le projet émerge dans le contexte de la modernisation d'un petit systèmes de gestion académique, où la nécessité de stocker et de manipuler efficacement les données des étudiants, des modules et des départements devient cruciale. L'utilisation de technologies contemporaines telles que Java avec Spring Boot pour le backend, Angular pour le frontend, et l'intégration avec la base de données Neo4j, s'inscrit dans une démarche visant à améliorer la flexibilité et la performance de l'application. Ce projet vise à offrir une solution minimiser pour la gestion des étudiants, intégrant les modules et les départements de manière intuitive. L'adoption de Neo4j en tant que base de données graphique permet une représentation efficace des relations complexes au sein de l'écosystème académique, offrant ainsi une perspective novatrice pour la gestion de l'information.

2- Objectif du rapport

L'objectif principal de cette application est de fournir une petit plateforme de gestion académique, mettant l'accent sur la gestion des étudiants, des modules et des départements au sein d'une institution éducative. En intégrant les technologies Java (Spring Boot) pour le backend, Angular pour le frontend, et en exploitant la puissance de la base de données Neo4j, Cette application vise à simplifier les processus liés à la gestion académique, offrant aux utilisateurs une interface conviviale pour enregistrer, consulter et mettre à jour les informations relatives aux étudiants, aux modules et aux départements. À travers cette initiative, nous visons à optimiser l'efficacité opérationnelle des institutions éducatives en fournissant une solution technologique simple et efficace pour une gestion transparente et structurée des données académiques.

II- Technologies Utilisées

1- Java



Java est un langage de programmation polyvalent, orienté objet et basé sur la plateforme Java. Il est largement utilisé pour le développement d'applications backend en raison de sa portabilité, de sa fiabilité et de sa grande communauté de développeurs. Dans ce projet, Java est utilisé avec le framework Spring Boot pour la gestion du backend, permettant de créer des API REST robustes et modulaires.

2- Spring Boot



Spring Boot est un framework Java qui simplifie le développement d'applications basées sur Spring en offrant des fonctionnalités par défaut et une configuration automatique. Il facilite la création d'applications Java autonomes et prêtes à l'emploi. Dans notre projet, Spring Boot est utilisé pour la gestion du backend, fournissant une architecture solide pour la création d'API REST, la gestion des dépendances et la configuration simplifiée.

3- Angular



Angular est un framework JavaScript/TypeScript développé par Google, principalement utilisé pour la création d'applications web monopages (SPA). Il offre une structure modulaire, des fonctionnalités de liaison de données bidirectionnelles et facilite la création d'interfaces utilisateur interactives. Dans ce projet, Angular est utilisé pour le développement du frontend, fournissant une expérience utilisateur dynamique et réactive.

4- Neo4j



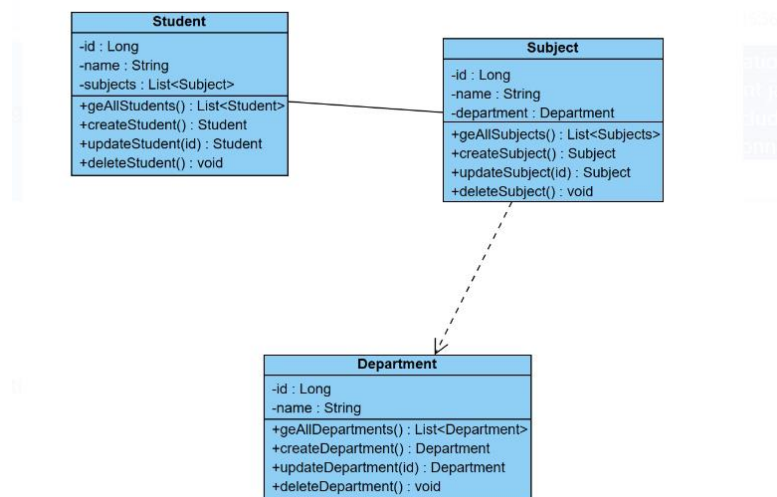
Neo4j est une base de données graphique qui utilise le modèle de graphe pour stocker et traiter les données. Contrairement aux bases de données relationnelles traditionnelles, Neo4j est optimisé pour représenter et interroger les relations complexes entre les entités. Dans notre application, Neo4j est choisi comme base de données pour stocker et récupérer les données liées aux étudiants, aux modules et aux départements de manière efficace, en exploitant la structure de graphe pour représenter les relations entre ces entités de manière naturelle.

En combinant ces technologies, nous créons une application robuste et moderne qui tire parti des avantages spécifiques de chacune pour répondre aux besoins de gestion académique de manière efficace et évolutive.

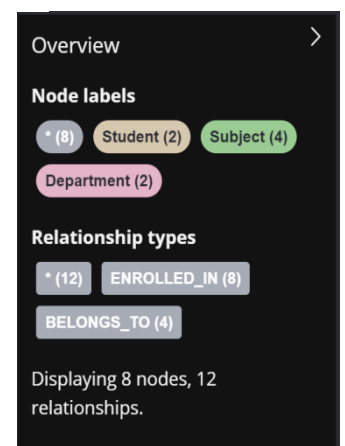
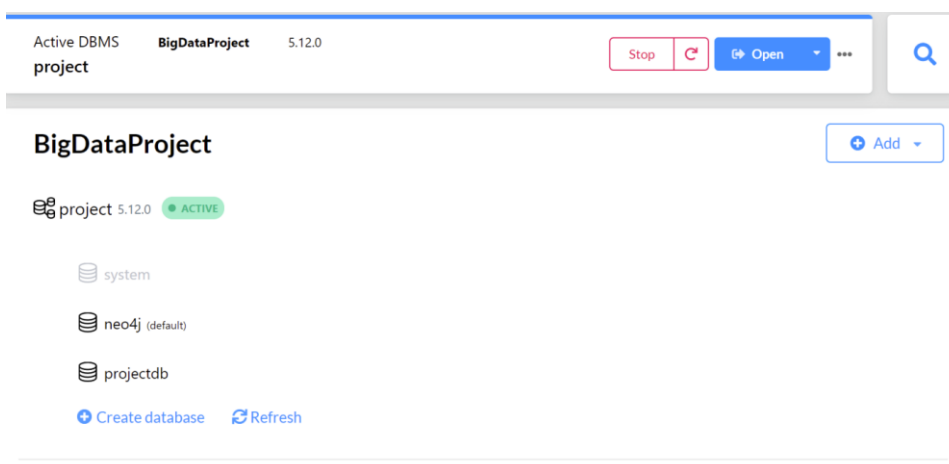
III- Conception de la Base de Données Neo4j

1- Schéma de la base de données

❖ Class Diagramme

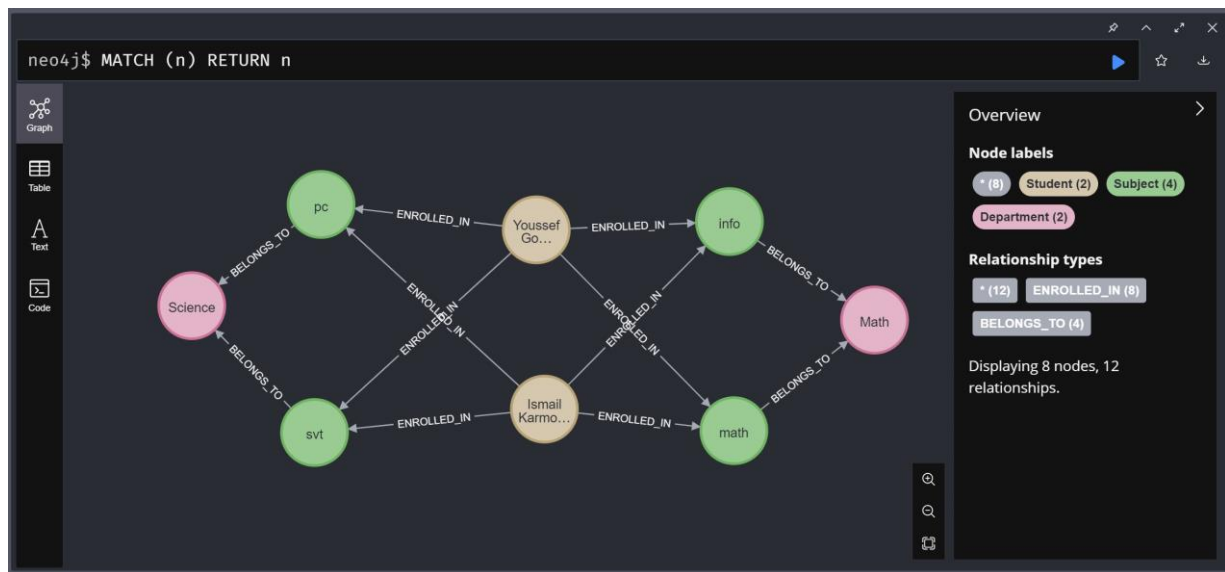


❖ Neo4j



➤ Exécution des requêtes Cypher pour récupérer et stocker les données

❖ Afficher tous les nœuds et les relations dans la base de données:



❖ Afficher le nombre des modules associées à l'étudiant Youssef El Goumri

The screenshot shows the Neo4j Desktop interface with a Cypher query and its result.

```
1 MATCH(student:Student)-[r:ENROLLED_IN]-(subject:Subject)
2 WHERE student.name="Youssef El Goumri"
3 RETURN count([student])
```

The result is displayed in a table:

	count(student)
1	4

Started streaming 1 records in less than 1 ms and completed in less than 1 ms.

❖ *Afficher le graphe des nœuds pour l'étudiant "Ismail Karmout"*

```
1 MATCH(student:Student)-[r:ENROLLED_IN]→(subject:Subject)
2 WHERE student.name="Ismail Karmout"
3 RETURN student, subject
```

Overview

Node labels

- * (5)
- Student (1)
- Subject (4)

Relationship types

- * (4)
- ENROLLED_IN (4)

Displaying 5 nodes, 0 relationships.

❖ *Afficher le nombre des nœuds Student dans la base de données*

```
1 MATCH(student:Student)
2 RETURN count(student) as StudentsNumbers
```

StudentsNumbers	
1	2

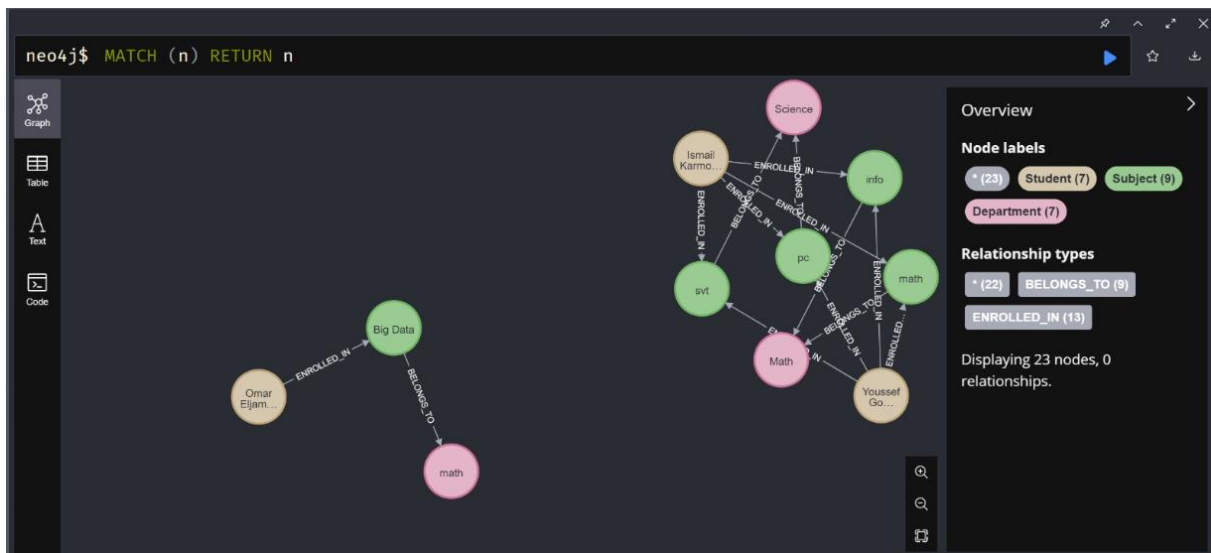
Started streaming 1 records after 7 ms and completed after 8 ms.

❖ *Ajouter un étudiant avec les modules et departement du module, en utilisant les requêtes Cypher de Neo4j*

```
1 CREATE (student:Student {id: 50, name: 'Omar Eljamghili'})-[r:ENROLLED_IN]→(subject:Subject {name: 'Big Data'})-[rb:BELONGS_TO]→(department:Department{name: 'math'})
2 RETURN student, subject, department
```

student	subject	department
{ "identity": 43, "labels": ["Student"], "properties": { "name": "Omar Eljamghili", "id": 50 }, "elementId": "43" }	{ "identity": 44, "labels": ["Subject"], "properties": { "name": "Big Data" }, "elementId": "44" }	{ "identity": 45, "labels": ["Department"], "properties": { "name": "math" }, "elementId": "45" }

Added 3 labels, created 3 nodes, set 4 properties, created 2 relationships, started streaming 1 records in less than 1 ms and completed after 5 ms.



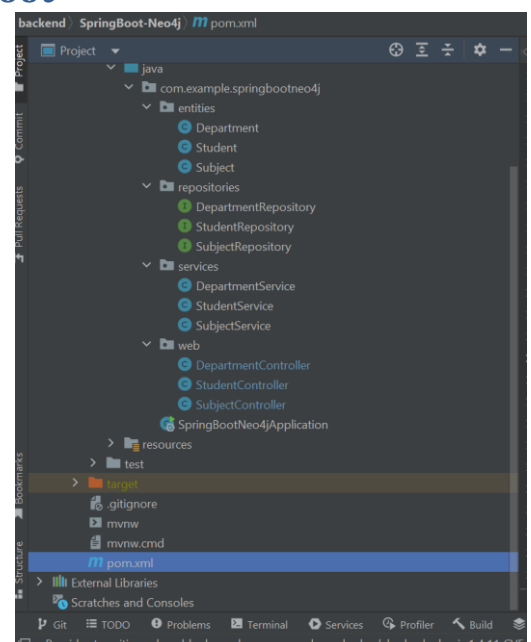
2- Choix de Neo4j pour ce projet

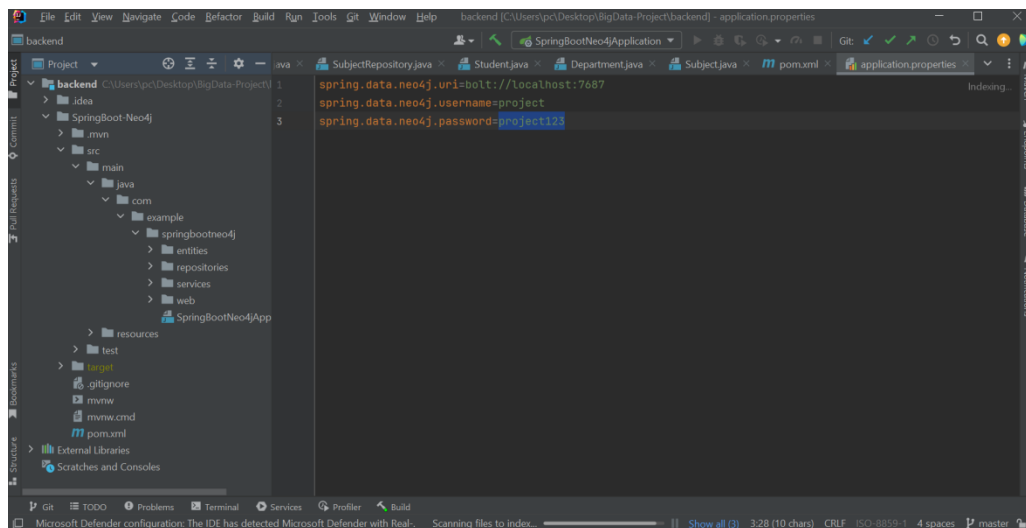
Le choix de Neo4j en tant que base de données pour ce projet s'appuie sur la nature intrinsèquement relationnelle des données académiques que nous gérons. Neo4j, en tant que base de données orientée graphe, excelle dans la représentation et la gestion de relations complexes entre les entités. Dans le contexte de la gestion des étudiants, des modules et des départements, les liens et les connexions entre ces entités sont souvent tout aussi importants que les entités elles-mêmes.

IV- Implémentation Backend avec Spring Boot

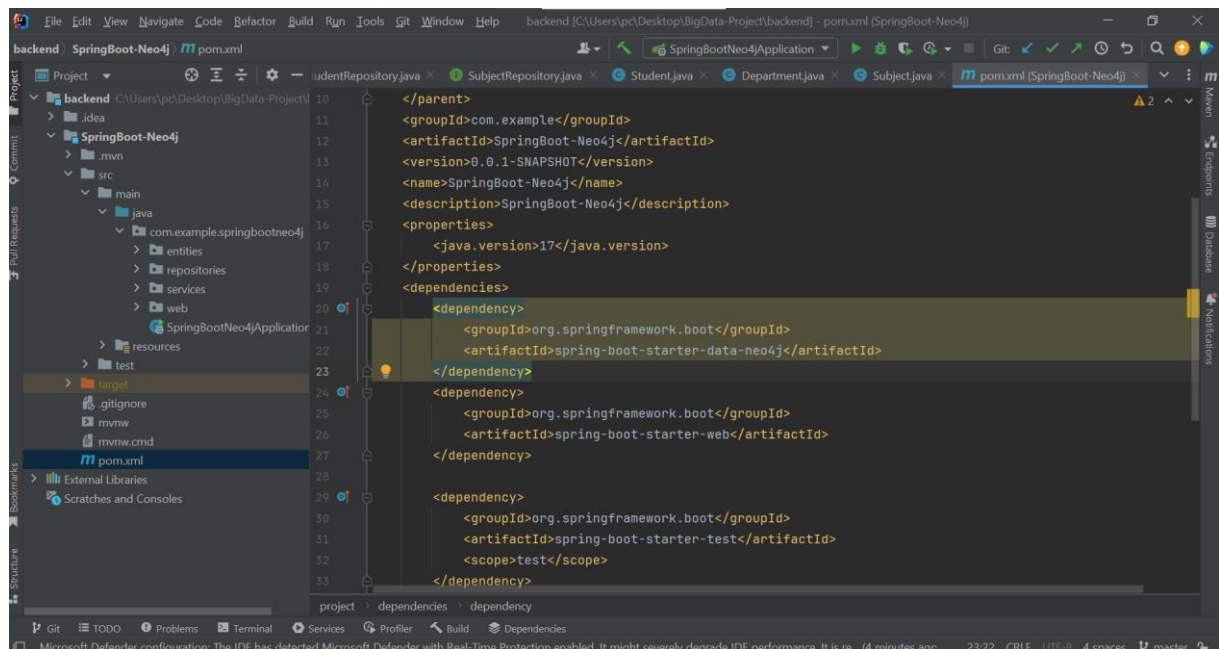
1- Configuration de Spring Boot avec Neo4j

- ❖ **Structure des Fichiers**
- ❖ **Fichier Application.properties**





❖ Fichier pom.xml dependencies

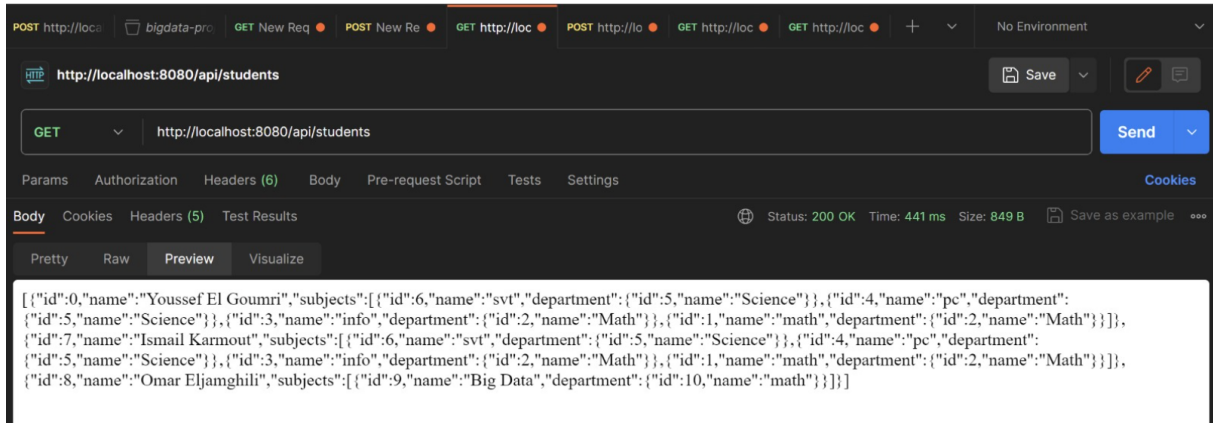


➤ Run the Application Backend

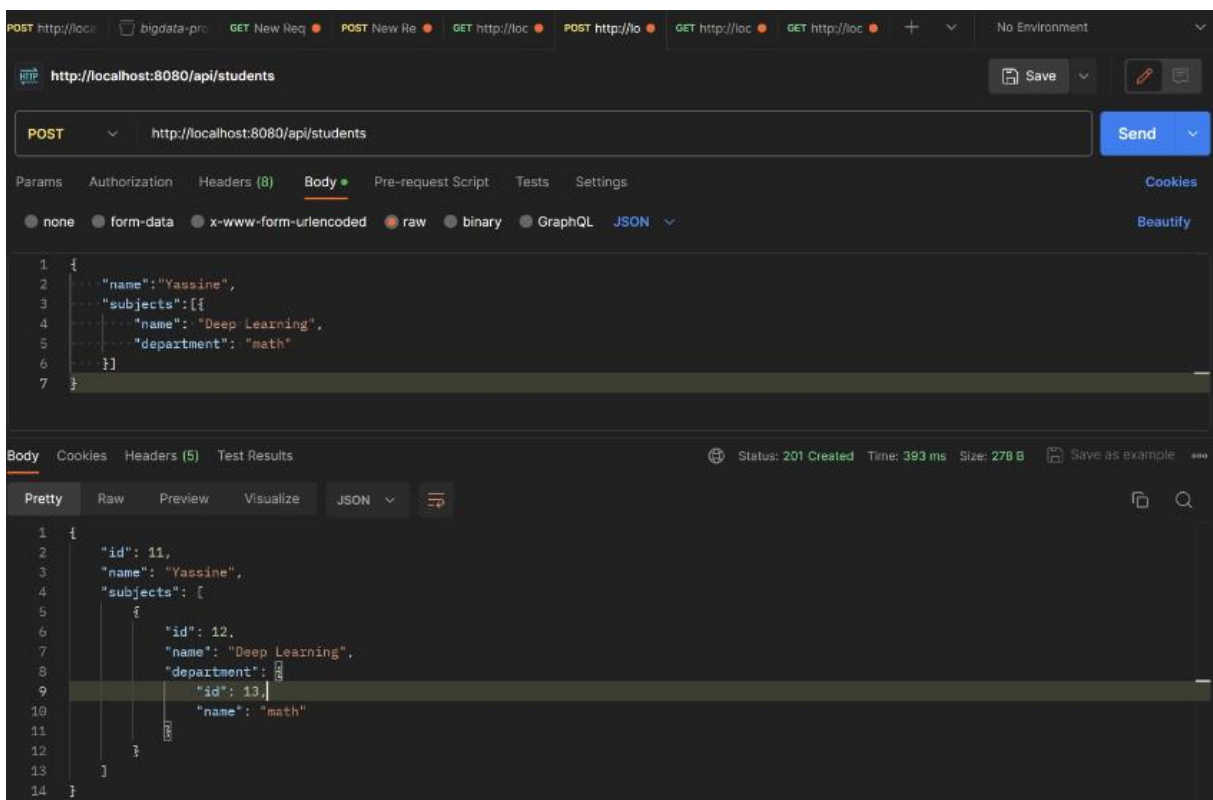


■ Tester les APIs de backend Spring boot avec Postman :

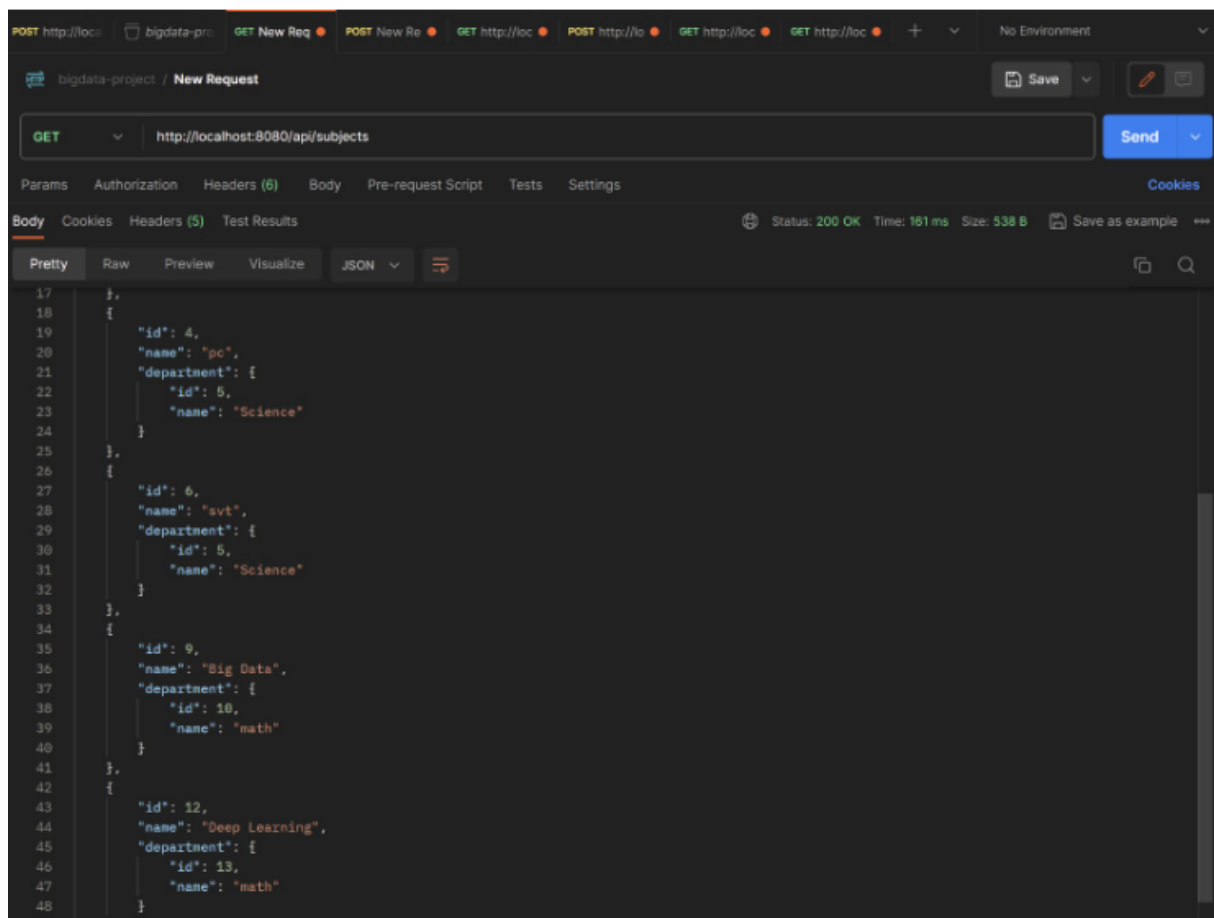
Afficher les étudiants



Ajouter un nouveau étudiant



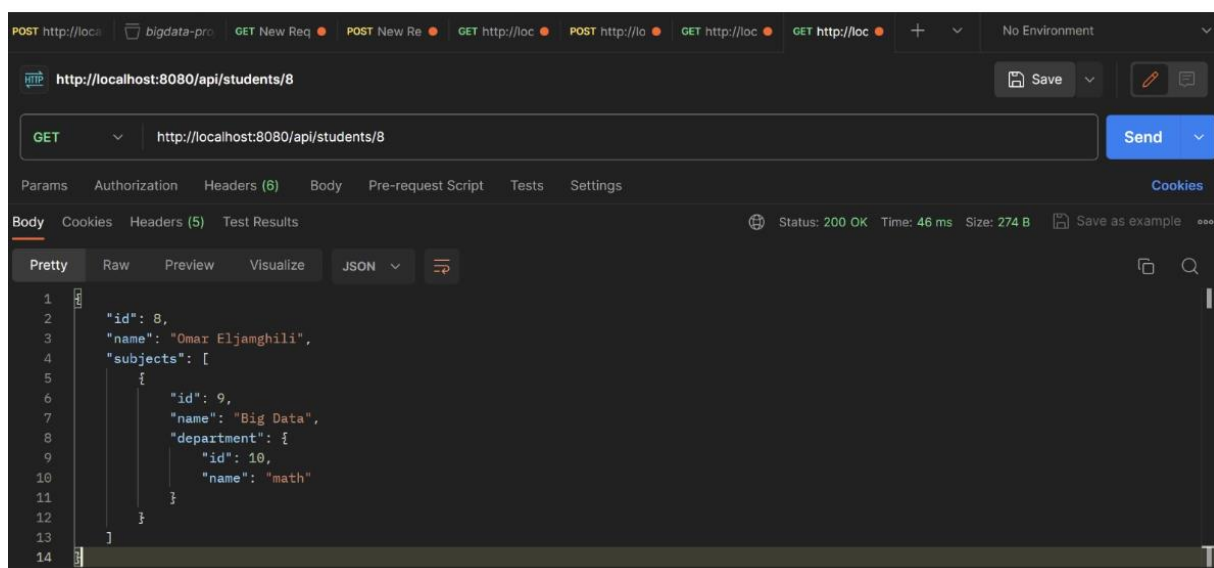
Afficher les modules avec leurs departement



The screenshot shows a Postman interface with a GET request to `http://localhost:8080/api/subjects`. The response is a JSON array of three subjects, each with an ID, name, and a department object containing an ID and name.

```
17  },
18  [
19    {
20      "id": 4,
21      "name": "pc",
22      "department": {
23        "id": 5,
24        "name": "Science"
25      }
26    },
27    {
28      "id": 6,
29      "name": "svt",
30      "department": {
31        "id": 5,
32        "name": "Science"
33      }
34    },
35    {
36      "id": 9,
37      "name": "Big Data",
38      "department": {
39        "id": 10,
40        "name": "math"
41      }
42    },
43    {
44      "id": 12,
45      "name": "Deep Learning",
46      "department": {
47        "id": 13,
48        "name": "math"
49      }
50    }
51  ]
52 }
```

Afficher l'étudiant que nous avons déjà ajouté à l'aide des requêtes Cypher

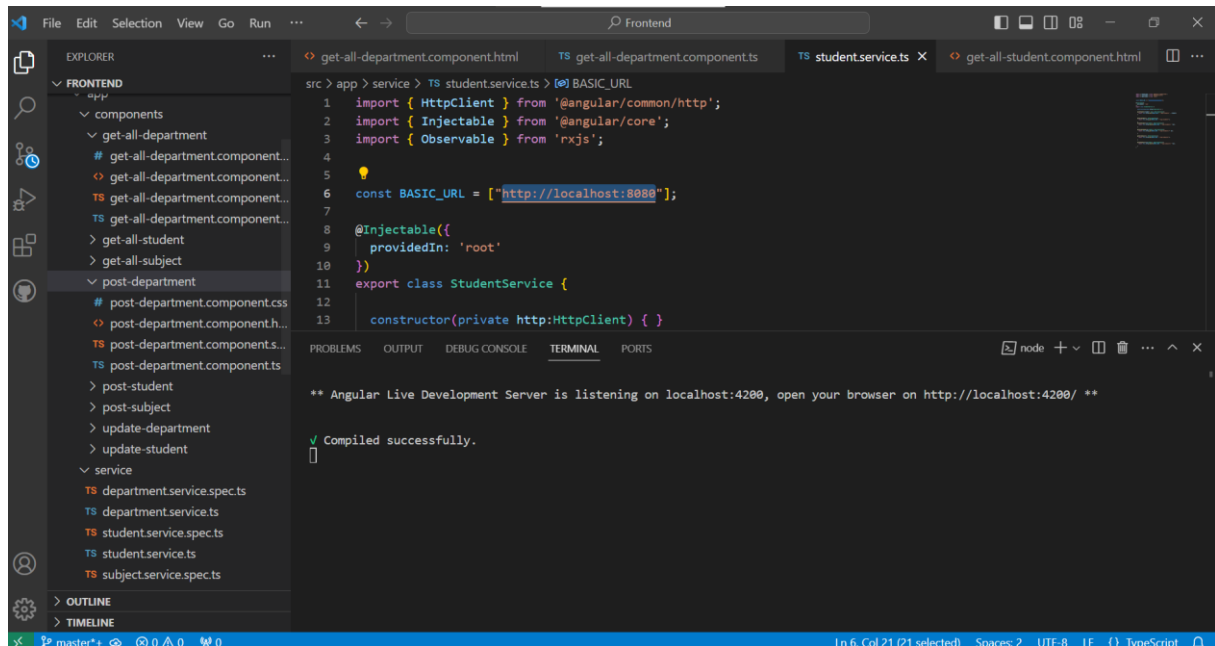


The screenshot shows a Postman interface with a GET request to `http://localhost:8080/api/students/8`. The response is a JSON object representing a student with an ID, name, and a list of subjects.

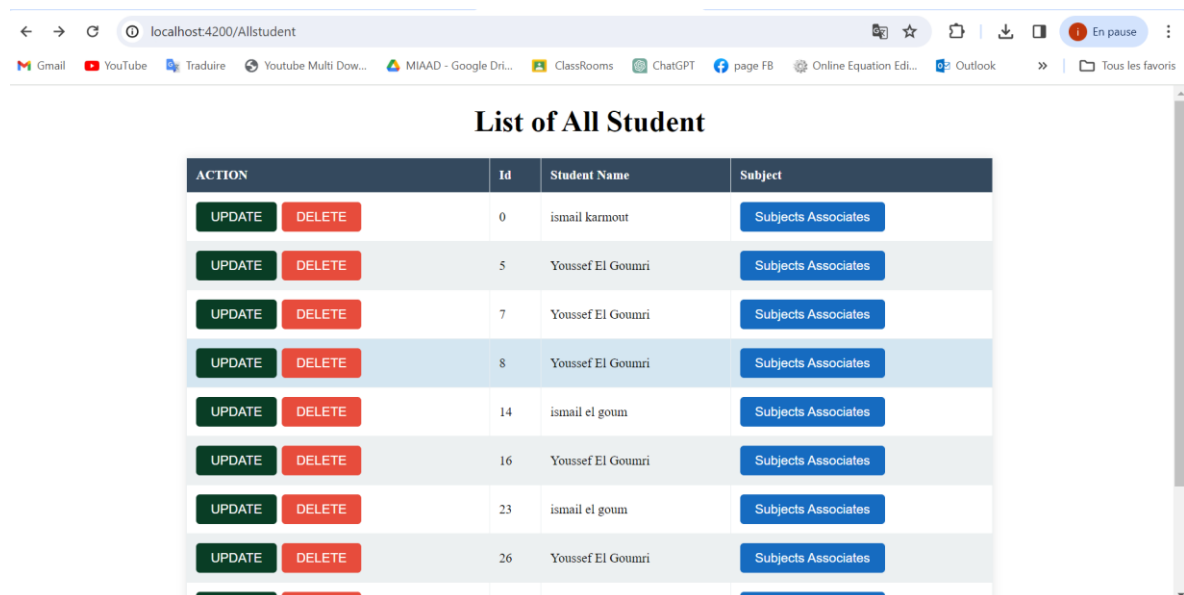
```
1  {
2    "id": 8,
3    "name": "Omar Eljamghili",
4    "subjects": [
5      {
6        "id": 9,
7        "name": "Big Data",
8        "department": {
9          "id": 10,
10         "name": "math"
11       }
12     }
13   ]
14 }
```

V - Implémentation Frontend avec Angular

❖ Structure des Fichiers



❖ List of All Student (En peut Ajouter / Modifier / et Supprimer Un Etudiant)



❖ Ajouter un Etudiant

Add Student

Student Name:

Omar El jamghili

Subject:

Big Data

Submit

Subject Associates pour chaque Etudiant :

List of Subject Associate to Student

ACTION	Id	Subject Name	Department
UPDATE DELETE	1	math	Department Associate
UPDATE DELETE	4	pc	Department Associate
UPDATE DELETE	10	info	Department Associate
UPDATE DELETE	13	svt	Department Associate

❖ **List of all Subject**

List of All Subject

ACTION	Id	Subject Name	Department
UPDATE DELETE	1	math	Department Associate
UPDATE DELETE	4	pc	Department Associate
UPDATE DELETE	10	info	Department Associate
UPDATE DELETE	13	svt	Department Associate
UPDATE DELETE	15	svt	Department Associate
UPDATE DELETE	17	math	Department Associate
UPDATE DELETE	19	info	Department Associate
UPDATE DELETE	20	pc	Department Associate

❖ **Département Associates pour chaque Subjectif :**

Add a New Department

List of Department Associate to Subject

ACTION	Id	Name Of Department
UPDATE DELETE	12	Informatique
UPDATE DELETE	18	Mathématique

List of All Departement :

List of All Department

ACTION	Id	Name Of Department
UPDATE DELETE	18	Mathématique
UPDATE DELETE	21	Science
UPDATE DELETE	28	Math
UPDATE DELETE	38	Science
UPDATE DELETE	40	informatique
UPDATE DELETE	41	economic
UPDATE DELETE	42	Math
UPDATE DELETE	46	Math
UPDATE DELETE	49	Science

VI- Conclusion

En conclusion, ce projet d'application de gestion académique utilisant Java (Spring Boot-Angular) avec Neo4j comme base de données représente une avancée significative dans l'optimisation des processus éducatifs. En intégrant des technologies modernes, nous avons créé une solution robuste et flexible pour la gestion des étudiants, modules et départements. L'utilisation de Neo4j a permis une représentation naturelle et efficace des relations complexes, tandis que Java avec Spring Boot et Angular ont assuré la création d'une application cohérente, réactive et facile à maintenir.