Rapport de développement de l’API REST pour la gestion des conférences scientifiques

Réaliser par :
ZANOUN Manel
LAIDI Aya
SADI Silya
AMIR Maria

Introduction:

Ce projet consiste a développer une API REST pour la gestion des conférences scientifiques. L'objectif était de créer une solution permettant de gérer les auteurs, les évaluateurs ,les éditeurs ,les conférences ,les soumissions et les évaluations. L'API doit aussi gérer les processus d'évaluation des articles soumis a une conférence et l'affectation des rôles aux utilisateurs.

Diagramme de classe:

- **Utilisateur** (classe mère abstraite):

  Attributs : id, nom, prenom, email et roles .

  Sous-classes : Auteur, editeur et evaluateur .

  Relation : Un Utilisateur peut avoir plusieurs rôles dans différentes conférences (éditeur, auteur, évaluateur).

- **Conférence :**

  Attributs: id ,titre ,dateDebut,dateFin,thematique,editeur ,soumission et l'état (ouverte, fermée, en évaluation)

  Relation : une conférence peut contenir plusieurs soumissions

- **Evaluation** :

  Attributs: id **,**evaluateur , soumission ,note , commentaires , etat

  Relation : chaque évaluation est liée a une soumission et un évaluateur.

○Soumission :

Attributs : id,titre,resume,documentPdf,auteur,conference,evaluateurs et évaluations

Relation :plusieurs soumission sont liées a une conférence

Relation entre les entités et implémentation dans l'API:

les relations entre les entités sont implémentées en utilisant JPA(Java Persistence API)
@OneToMany et **ManyToOne** pour les relations un a plusieurs.
@ManyTOMany pour les relation plusieurs a plusieurs avec une table de jointure .
@JoinColumn pour définir les clés étrangères .

Exemple :

```
@ManyToOne
@JoinColumn(name = "editeur_id", nullable = false)
@NotNull(message = "L'éditeur est obligatoire")
private Editeur editeur;

@OneToMany(mappedBy = "conference", cascade = CascadeType.ALL, orphanRemoval = true)
private List<Soumission> soumissions = new ArrayList<>();
```

La structure des endpoints et leur utilisation : voici les principaux endpoints de l'API et leur usage :

Gestion des Conférences :

GET/conferences : Récupérer toutes les conférences .
POST/conference : créer une nouvelle conférence .
PUT/conferences{id}/etat : Modifier l'état d'une conférence .

Gestion des soumissions:

POST/soumissions/{id}affecter-evaluateurs : affecter des évaluateur a une soumission .

Gestion des Evaluations :

GET/evaluations/evaluateur/{Id} : récupérer les évaluateurs d'une soumission

Utiles utilisés:

Spring Boot: framework principal pour la creation de l'API .
JPA(hibernate) : pour l'interaction avec la base de donnees .
H2 Database : Base de donnees en memoire pour les tests locaux .
Postman : permet de tester les endpoints de l'API et de valider qu' ils fonctionnent correctement .
Intellij IDEA : Environnement de développement .

Principales Difficultés Rencontrer et Résolutions:

Problème: Erreur de plugin Maven(spring-boot:run)
Solution : ajouter des nouvelles dépendances dans le fichier pom.xml
Problème: Configuration des relations entre les entités
Solution : Utilisation des annotations JPA telles que @ManyToMany et JoinTable pour résoudre des problèmes de mapping
Problème: un utilisateur peut avoir plusieurs rôle dans une conférence (éditeur ,evaluateur,auteur) dans une même conférence ce qui complique la gestion des permissions
Solution : Utilisation des entités intermédiaires pour lier un utilisateur a un rôle spécifique dans une conférence .

Problemes non resolus :

Certains aspect fonctionnels n'ont pas pu etre entierement implementes dans le cadre de projet :

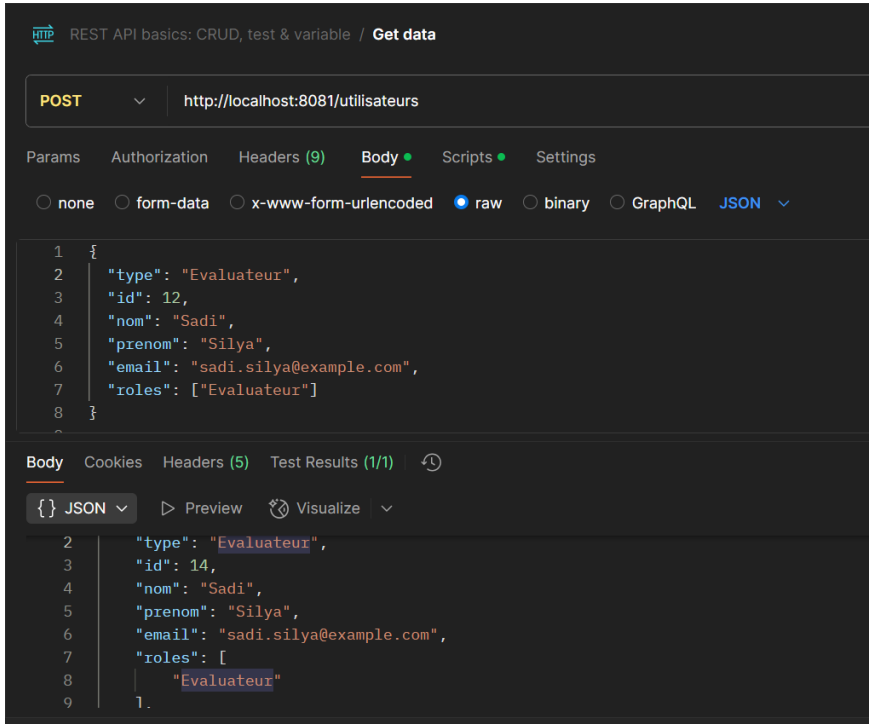
Restriction d'évaluation pour les auteurs :

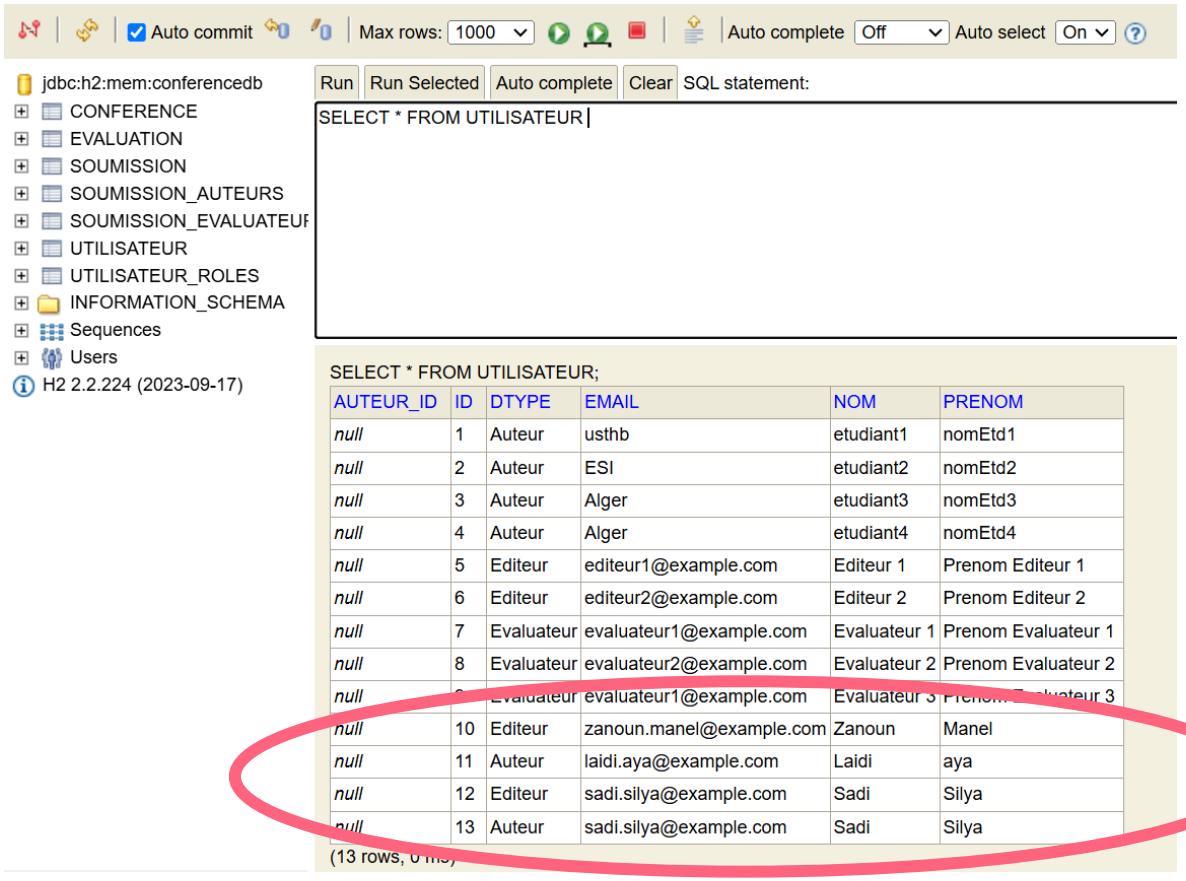
Aucun évaluateur ne peut évaluer une soumission dont il est l'un des auteurs Cette contrainte n'a pas été implémentée

Evaluateur : Évalue les soumissions qui lui sont assignées .

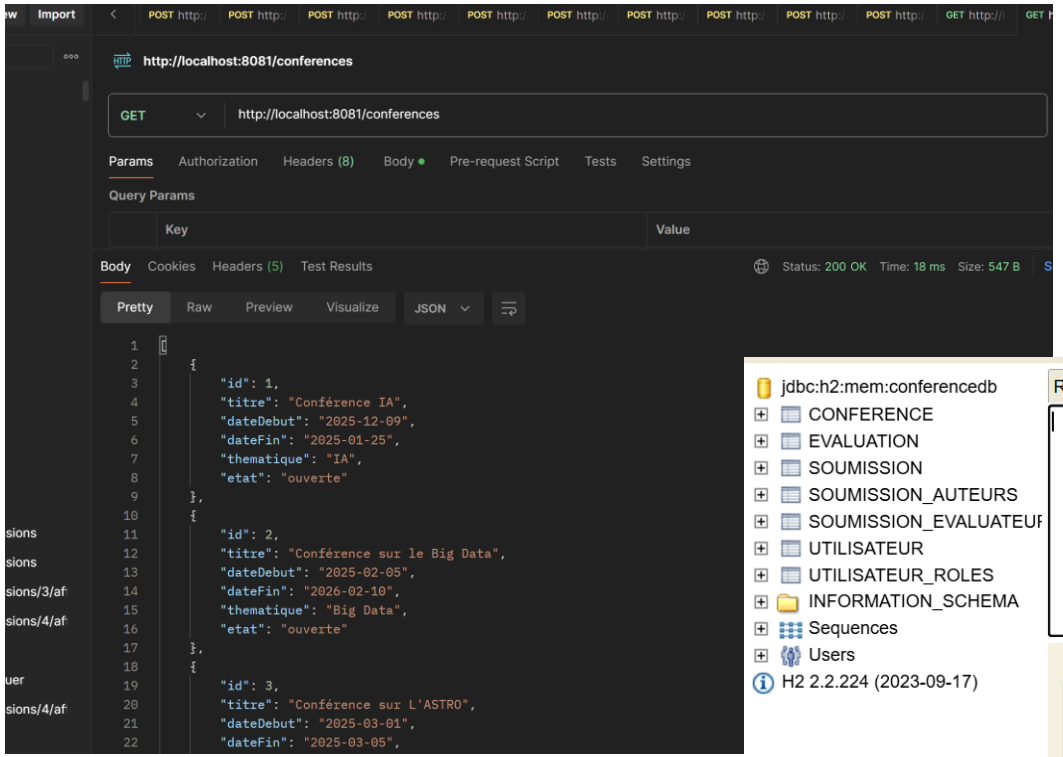
Tests de quelques endpoints avec Postman : Pour valider le bon fonctionnement des différents endpoints, Postman a été utilisé comme outil de test.

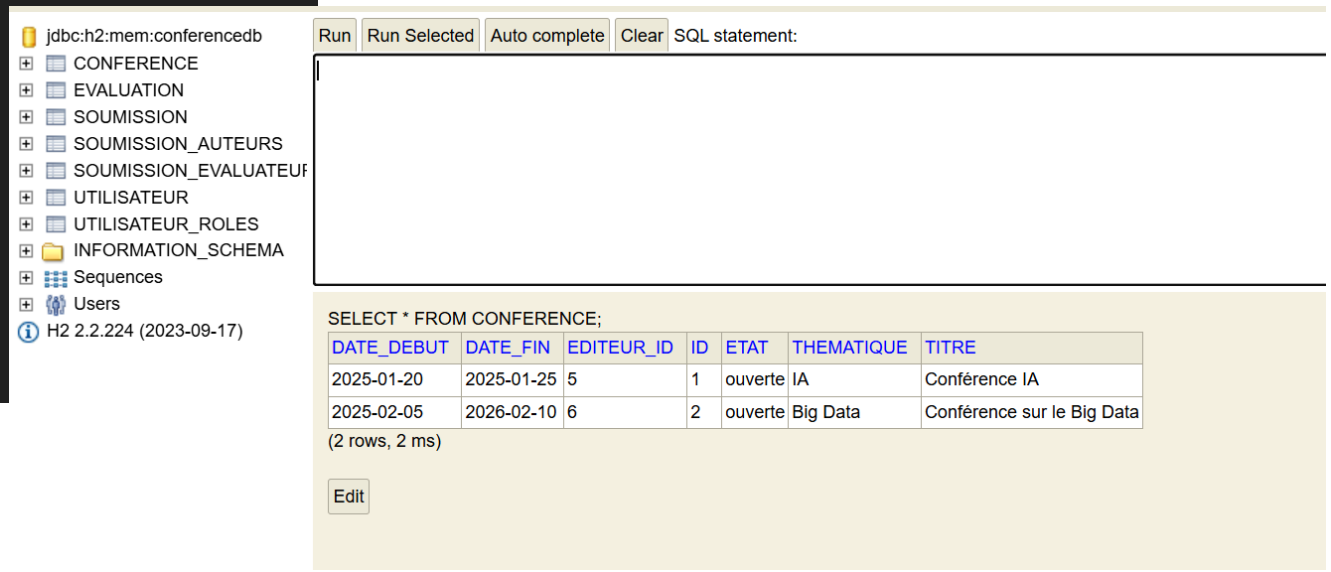
Création des utilisateurs :





Création des conférences:





Création d’une soumission :

Max rows: 1000 | Auto complete Off | Auto select On

Run | Run Selected | Auto complete | Clear | SQL statement:

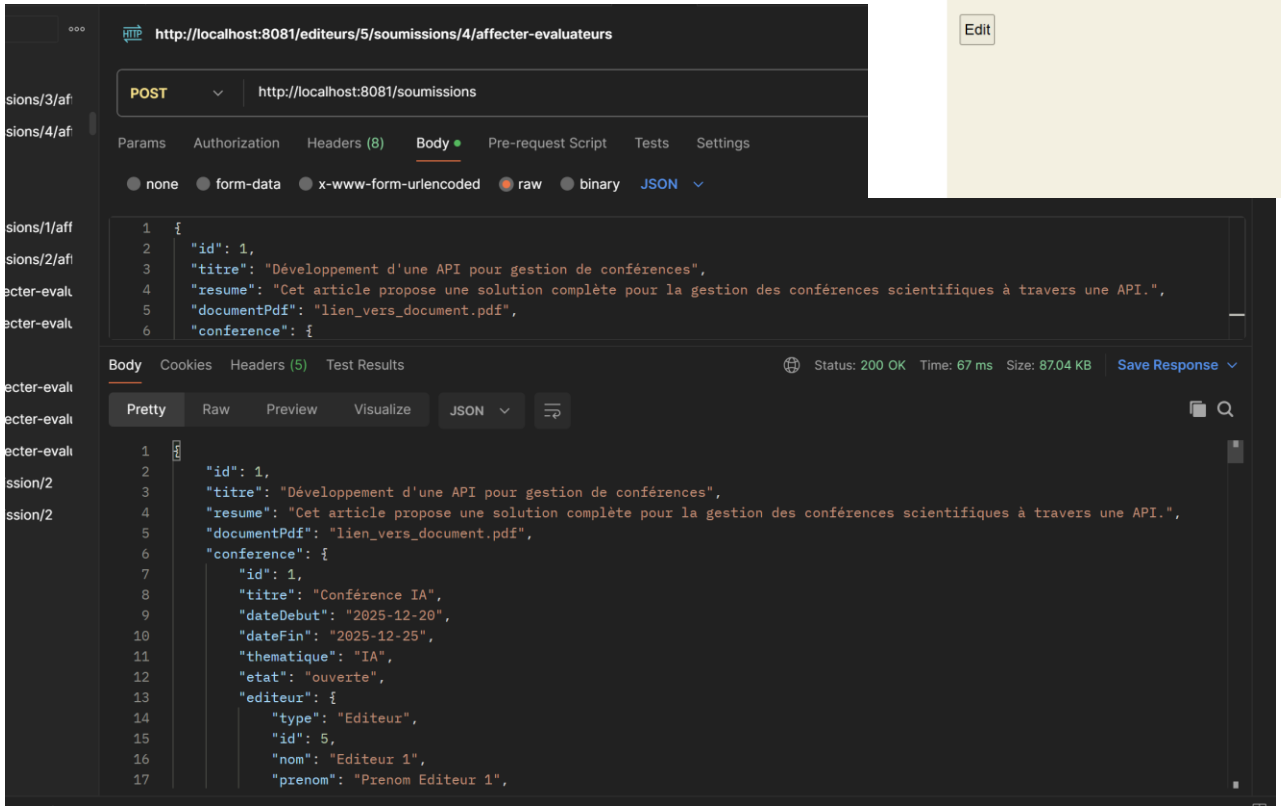
```
SELECT * FROM SOUMISSION
```

```
SELECT * FROM  SOUMISSION;
```

AUTEUR_ID	CONFERENCE_ID	ID	RESUME	DOCUMENT_PDF	TITRE
1	1	1	Cet article propose une solution complète pour la gestion des conférences scientifiques à travers une API.	lien_vers_document.pdf	Développement d'une API pour gestion de conférences
null	2	2	Résumé 2	document2.pdf	Titre 2
null	1	3	Résumé 1	document1.pdf	Titre 1

(3 rows, 2 ms)

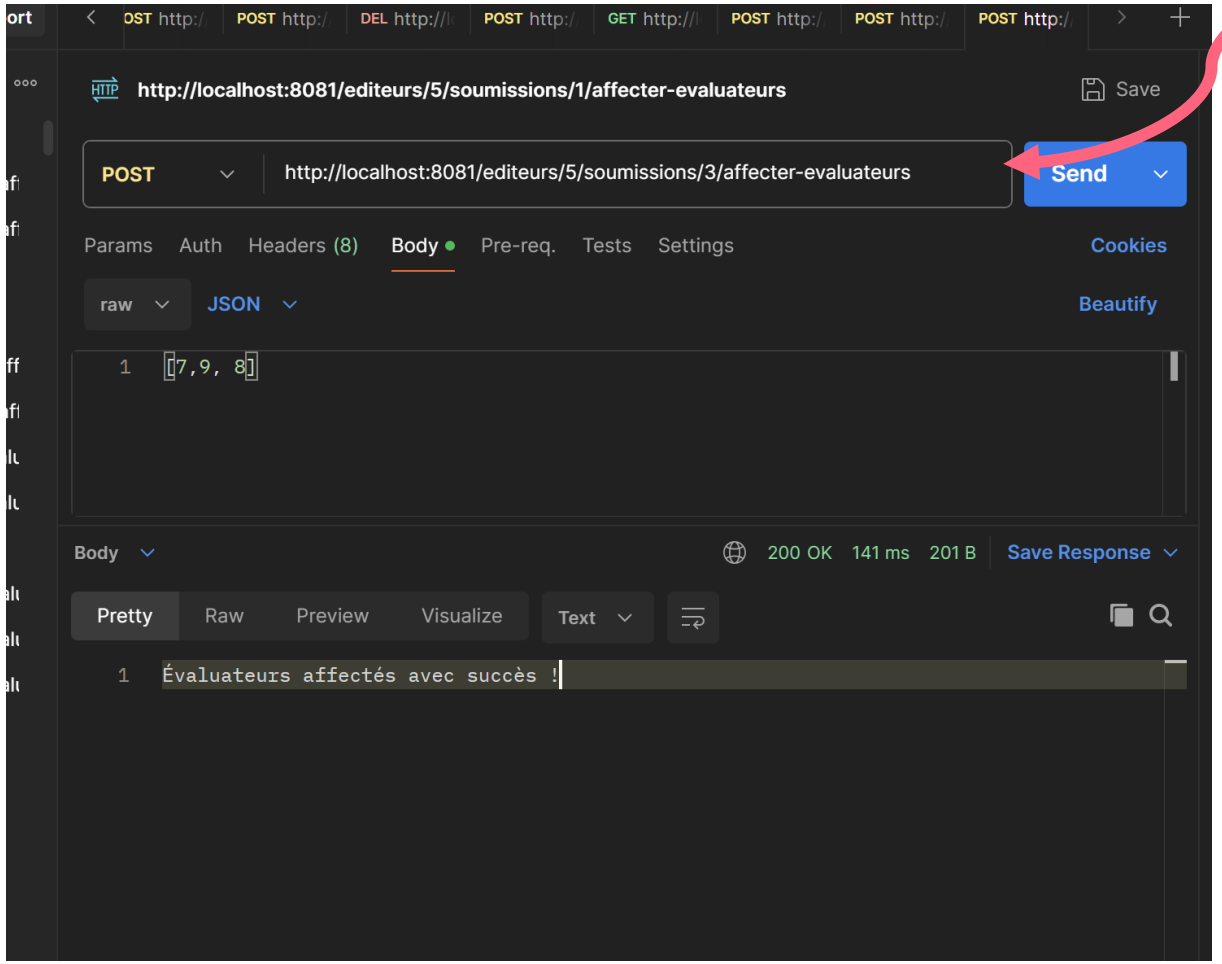
Edit

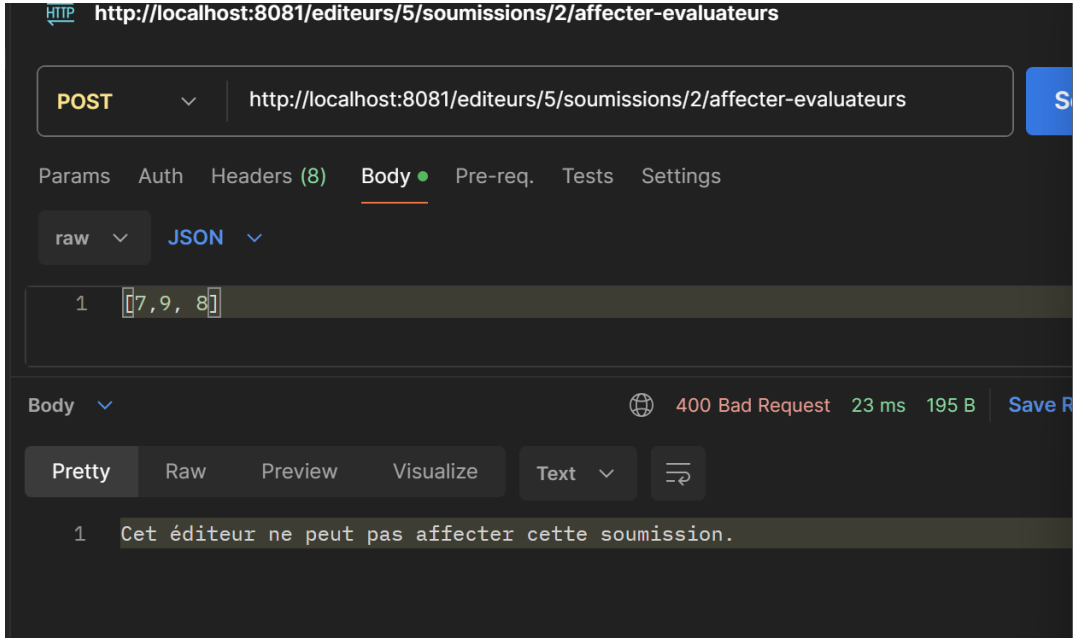



```
{
  "id": 1,
  "titre": "Développement d'une API pour gestion de conférences",
  "resume": "Cet article propose une solution complète pour la gestion des conférences scientifiques à travers une API.",
  "documentPdf": "lien_vers_document.pdf",
  "conference": {
    "id": 1,
    "titre": "Conférence IA",
    "dateDebut": "2025-12-20",
    "dateFin": "2025-12-25",
    "thematique": "IA",
    "etat": "ouverte",
    "editeur": {
      "type": "Editeur",
      "id": 5,
      "nom": "Editeur 1",
      "prenom": "Prenom Editeur 1",
```

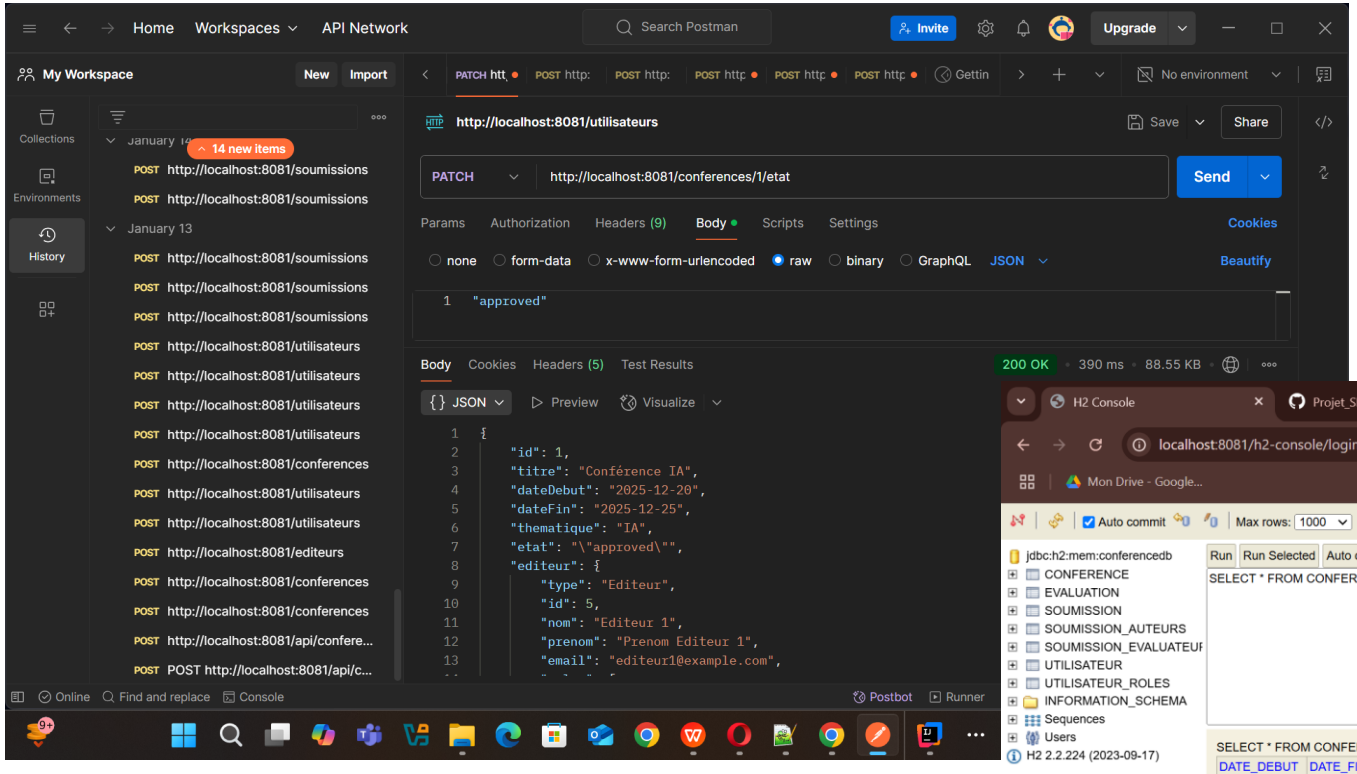
Affecter une soumission a un evaluateur par un editeur :

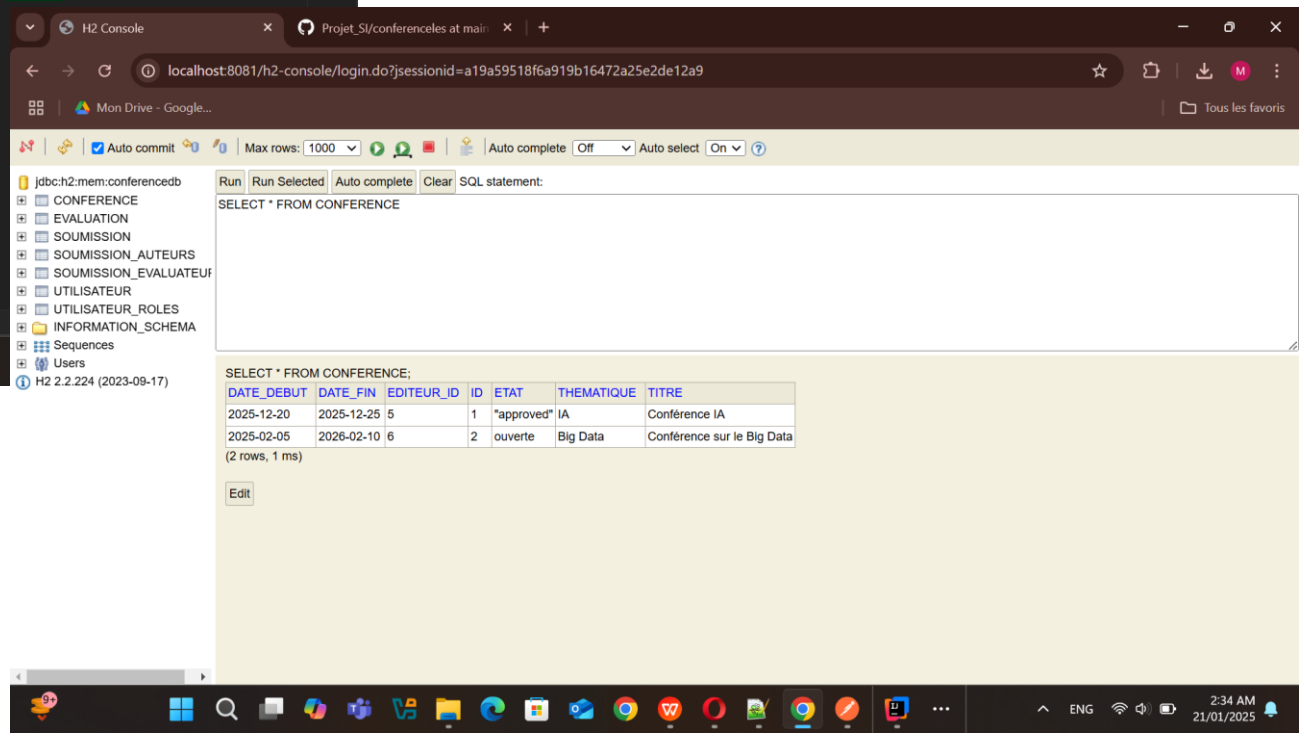
5 = Id éditeur ,
3=Id soumission ,
7,8 et 9= Id des évaluateurs au quel il a affecté la soumission par l'éditeur





Modification d'état d’une conference :





Conclusion:

Ce projet a abouti à une API REST fonctionnelle, conforme aux exigences initiales, permettant une gestion efficace des conférences scientifiques. Chaque fonctionnalité a été développée en respectant les normes de conception, comme le respect des relations entre entités et une structure claire des endpoints. L'utilisation d'outils modernes tels que Spring Boot, JPA et Postman a permis de garantir une implémentation robuste et des tests approfondis.