

Rapport de projet de micro service

Thème :

« « Conception et réalisation d'une
application de gestion d'un centre de santé

Présenté par :

Abdoulaye DIANKHA

Sous la supervision de :

Pr. Samba NDIAYE

PLAN :

Introduction

- I- Présentation de micro service
- II- Description du projet
 - 1. Technologies utilisées
 - 2. Description du projet
 - 3. Réalisation de l'enregistrement des informations des patients dans MySQL
 - 4. Réalisation de l'enregistrement des informations de paiements dans MySQL
 - 5. Réalisation de l'enregistrement des informations de consultations dans MySQL

Conclusion

Introduction

Depuis de nombreuses années, il existe des systèmes et nous les améliorons. Plusieurs technologies, modèles architecturaux et meilleures pratiques ont émergé au cours de ces années. Les micro services sont l'un de ces modèles architecturaux issus du monde de la conception axée sur le domaine, de la livraison continue, de l'automatisation des plates-formes et de l'infrastructure, des systèmes évolutifs, de la programmation polyglotte et de la persistance.

C'est dans ce cadre qu'il nous a été demandé de réaliser avec une architecture microservice un projet ayant un certain nombre de fonctionnalité.

Notre travail consiste ainsi à réaliser une application composée de 3 micro-service dépendamment entre eux. Certaines informations du micro-service 2 dépend du micro-service 1 et certaines informations du micro-services 3 dépend des micro-services 1 et 2 et chaque micro-services aura sa propre base de données.

I- Présentation de micro service

Les micro services sont des composants d'une application ou d'un écosystème plus large conçus pour fonctionner indépendamment - chacun étant responsable d'un domaine commercial ou technique spécifique

Une architecture de microservices adopte cette même approche et l'étend aux services faiblement couplés qui peuvent être développés, déployés et maintenus indépendamment. Chacun de ces services est responsable de tâches discrètes et peut communiquer avec d'autres services via de simples API pour résoudre un problème commercial complexe plus vaste

Une fois développés, ces services peuvent également être déployés indépendamment les uns des autres et il est donc facile d'identifier les services chauds et de les mettre à l'échelle indépendamment de l'ensemble de l'application. Les microservices offrent également une meilleure isolation des pannes grâce à laquelle, en cas d'erreur dans un service, l'ensemble de l'application ne cesse pas nécessairement de fonctionner. Lorsque l'erreur est corrigée, elle peut être déployée uniquement pour le service respectif au lieu de redéployer une application entière.

II- Description du projet

Il nous a été demandé de réaliser avec une architecture micro-service et en langage java, une application de gestion d'un centre de santé dans laquelle l'enregistrement de l'informations des patients, des paiements, des consultations se feront dans 3 bases de données MySQL, chaque fonctionnalité citée ci-dessus aura sa propre base de données.

1. Technologies utilisées

Pour chacun de ces trois projets, nous avons utilisé :

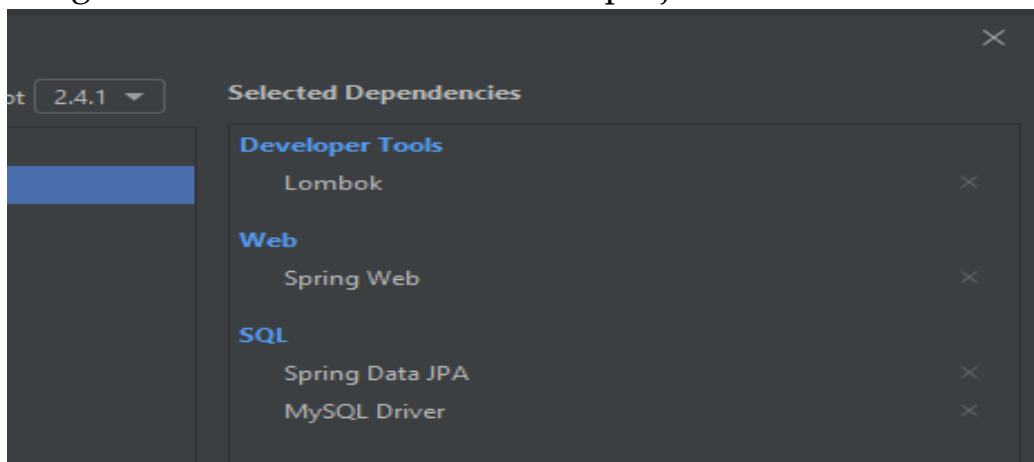
IntelliJ IDEA : qui est notre environnement de développement intégré pour la réalisation de ces trois micro services, qui est un environnement de programmation largement destiné à Java.

Spring boot : qui est le Framework utilisé lors de la création du projet, qui est open source et basé sur Java. Il contient de nombreuses librairies parmi lesquelles on a utilisé :

- **Lombok** : qui est une petite bibliothèque qui peut être utilisé pour réduire la quantité de code Java.
- **Spring web** : pour la création de service web basée sur des documents.
- **Spring JPA** : qui est la spécification SUN pour les objets persistants dans l'application.
- **Driver** du SGBD : pour la bonne liaison du micro service et du SGBD utilisé.

Postman : qui est un outil de test d'API évolutif qui s'intègre rapidement dans le pipeline CI/CD.

Image illustrative d'une création de projet micro service :



2. Description du projet

Chaque micro-service contient, entre autres des classes, un fichier `pom.xml`, un fichier de propriété.

- Le fichier **`pom.xml`** : qui contient des informations sur le projet, et les détails de configuration utilisés par Maven.
- Le fichier **`application.properties`** : utilisé pour configurer un certain nombre de propriétés dans un seul fichier pour exécuter l'application dans des environnements différents.
- Des classes : parmi lesquelles on a :
 - une classe **`controller`** : qui est une classe contrôleur contenant essentiellement l'annotation `@RestController` qui permet d'agir comme un stéréotype pour la classe annotée, indiquant son rôle.
 - une interface **`repository`** : qui est une classe d'abstraction du référentiel Spring Data qui a comme objectif de réduire la quantité de code standard requis pour implémenter.
 - une classe simple : pour la création d'une variable de type étudiant comportant les champs de représentation du type de l'objet en question.
 - une classe **`application`** : qui est la classe principale contenant essentiellement l'annotation `@SpringBootApplication` qui dispose une multitude de fonctionnalité parmi lesquelles la configuration de Spring basé sur Java, analyse des composant et en particulier pour activer la fonction de configuration automatique de Spring Boot.

3. Réalisation de l'enregistrement des informations des patients dans MySQL

Ce micro-service permet d'afficher l'ensemble les informations des patients dans MySQL et d'en enregistrer plusieurs. Il contient entre autres un contrôleur **PatientsRestController**, un repository **patientsRepositories**, une classe **patients** etc.

- La classe **patients** : contient les champs, *numtickets*, *nom*, *prenom*, *date*, *age*, *poids* et *température* ayant chacun un constructeur et un getter et setter pour chacun des champs.
- L'interface **patientsRepositories** : qui étend *JpaRepository<patients, Integer>* nous permettant de définir une interface de référentiel pour chaque entité de domaine de l'application.
- La classe **PatientsRestController** qui contient :
 - La fonction *listepatients ()* : permettant ainsi de renvoyer la liste de tous les patients grâce à l'annotation *@GetMapping* qui mappe les requêtes http GET sur des méthodes de gestionnaire spécifiques.
 - La fonction *save ()* : qui permet d'ajouter des patients dans MySQL grâce à l'annotation *@PostMapping* pour gérer le type de methode de requête POST etc.

a. DESCRIPTION DE LA BASE DE DONNEES PATIENTS

La base de données des patients est nommée bdtickets qui dispose d'une table patient contenant un certain nombre d'enregistrements.

```
MySQL 8.0 Command Line Client - Unicode
mysql> show databases;
+-----+
| Database |
+-----+
| bdconsultations |
| bdpayements |
| bdtickets |
| information_schema |
| mysql |
| performance_schema |
| sys |
+-----+
7 rows in set (0.60 sec)

mysql> use bdtickets
No connection. Trying to reconnect...
Enter password: *****
Connection id: 2041
Current database: *** NONE ***

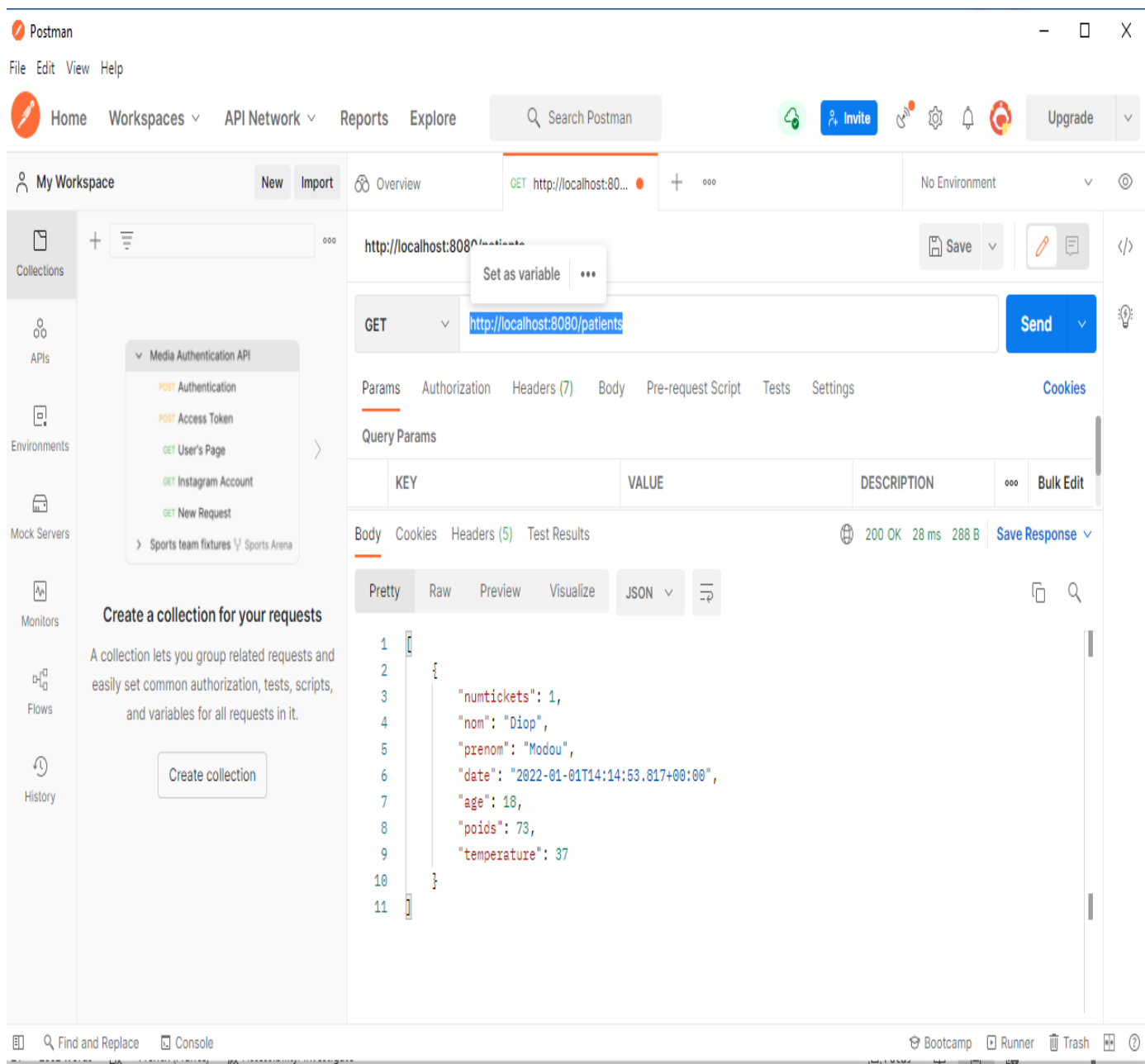
Database changed
mysql> select * from bdtickets.patients;
+-----+-----+-----+-----+-----+-----+-----+
| numtickets | age | date | nom | poids | prenom | temperature |
+-----+-----+-----+-----+-----+-----+-----+
| 1 | 18 | 2022-01-01 14:14:53.817000 | Diop | 73 | Modou | 37 |
+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.02 sec)

mysql>
```


b. FONCTIONNEMENT DE L’AFFICHAGE ET DE L’ENREGISTREMENT DES PATIENTS SUR POSTMAN

Ainsi la saisie de l’url « `http://localhost:8080/patients` » avec comme type de requête « GET » dans postman, renvoie les enregistrements de la table `bdtickets.patients` sous format **Json**.

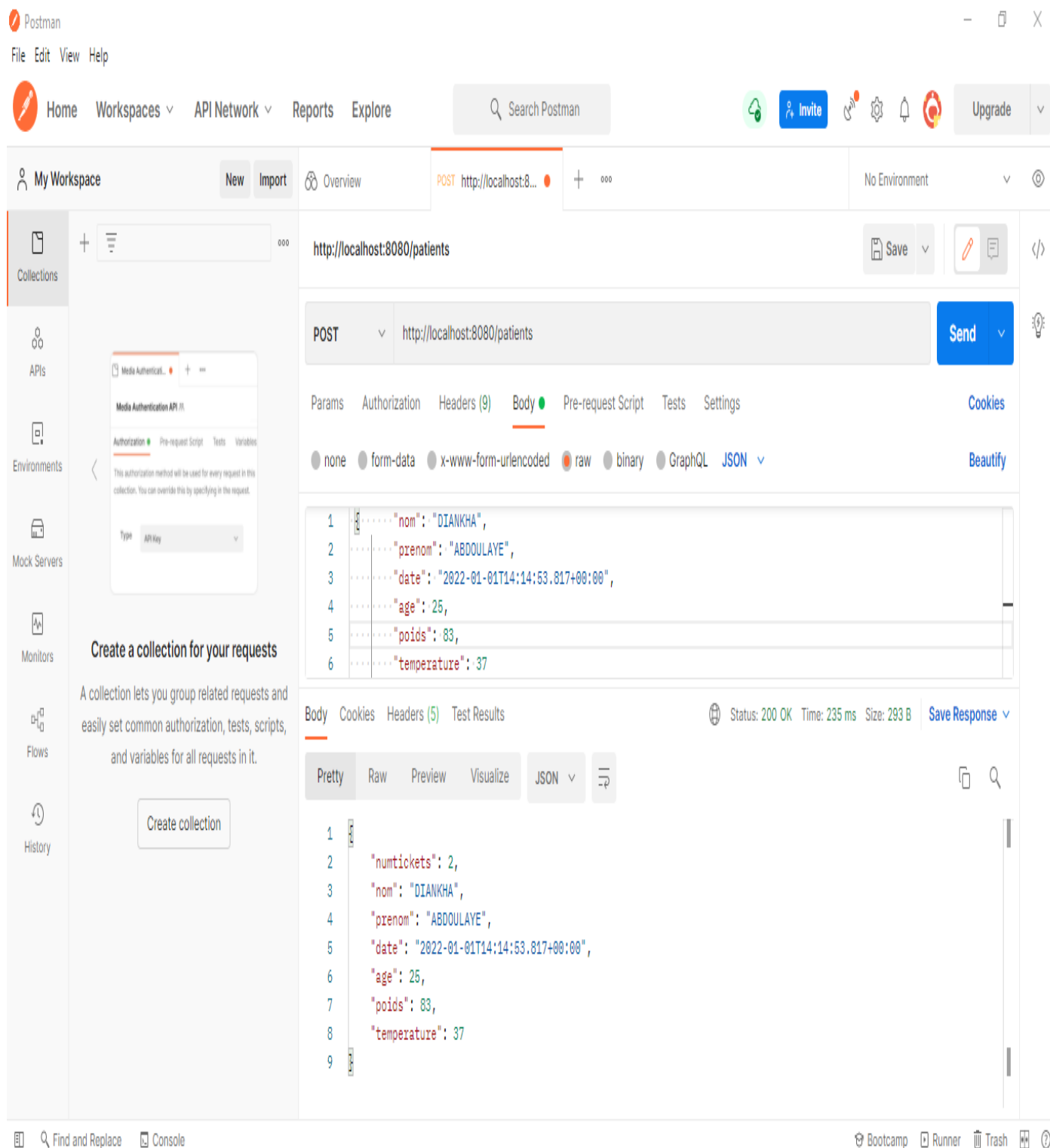
- Affichage



- ENREGISTREMENT

L'enregistrement d'un patient se fait en saisissant « `http://localhost:8080/patients` » dans Postman avec comme type de requête « POST ». La saisie se fait sous format **json**.

Image illustrative de l'insertion d'un patient :



- Résultat dans la base de données

```
| performance_schema |
| sys                |
+-----+
7 rows in set (0.60 sec)

mysql> use bdtickets
No connection. Trying to reconnect...
Enter password: *****
Connection id: 2041
Current database: *** NONE ***

Database changed
mysql> select * from bdtickets.patients;
+-----+-----+-----+-----+-----+-----+
| numtickets | age | date                | nom   | poids | prenom | temperature |
+-----+-----+-----+-----+-----+-----+
|          1 | 18 | 2022-01-01 14:14:53.817000 | Diop | 73 | Modou | 37 |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.02 sec)

mysql> select * from bdtickets.patients;
+-----+-----+-----+-----+-----+-----+
| numtickets | age | date                | nom   | poids | prenom | temperature |
+-----+-----+-----+-----+-----+-----+
|          1 | 18 | 2022-01-01 14:14:53.817000 | Diop | 73 | Modou | 37 |
|          2 | 25 | 2022-01-01 14:14:53.817000 | DIANKHA | 83 | ABDOULAYE | 37 |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql>
```

4. Réalisation de l'enregistrement des informations de paiements dans MySQL

Ce micro-service permet d'afficher l'ensemble les informations des paiements dans MySQL et d'en enregistrer plusieurs. Il contient entre autres un contrôleur **payementRestControllers**, un repository **payementRepositories**, une classe **payement** etc.

- La classe **payement** : contient les champs, *numpayement*, *date*, *numtickets*, *nom*, *prenom*, *montant* ayant chacun un constructeur et un getter et setter pour chacun des champs.
- L'interface **payementRepositories** : qui étend *JpaRepository<payement, Integer>* nous permettant de définir une interface de référentiel pour chaque entité de domaine de l'application.
- La classe **payementRestControllers** qui contient :
 - La fonction *listepatients ()* : permettant ainsi de renvoyer la liste de tous les patients grâce à l'annotation *@GetMapping* qui mappe les requêtes http GET sur des méthodes de gestionnaire spécifiques.
 - La fonction *save ()* : qui permet d'ajouter des patients dans MySQL grâce à l'annotation *@PostMapping* pour gérer le type de méthode de requête POST etc.

a. DESCRIPTION DE LA BASE DE DONNEES PAYEMENT

La base de données de paiement est nommée bdpayements qui dispose d'une table paiement contenant un certain nombre d'enregistrements

```
MySQL 8.0 Command Line Client - Unicode
Database changed
mysql> select * from bdtickets.patients;
+-----+-----+-----+-----+-----+-----+-----+
| numtickets | age | date                | nom | poids | prenom | temperature |
+-----+-----+-----+-----+-----+-----+-----+
|          1 | 18 | 2022-01-01 14:14:53.817000 | Diop | 73 | Modou |          37 |
+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.02 sec)

mysql> select * from bdtickets.patients;
+-----+-----+-----+-----+-----+-----+-----+
| numtickets | age | date                | nom | poids | prenom | temperature |
+-----+-----+-----+-----+-----+-----+-----+
|          1 | 18 | 2022-01-01 14:14:53.817000 | Diop | 73 | Modou |          37 |
|          2 | 25 | 2022-01-01 14:14:53.817000 | DIANKHA | 83 | ABDOULAYE |          37 |
+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

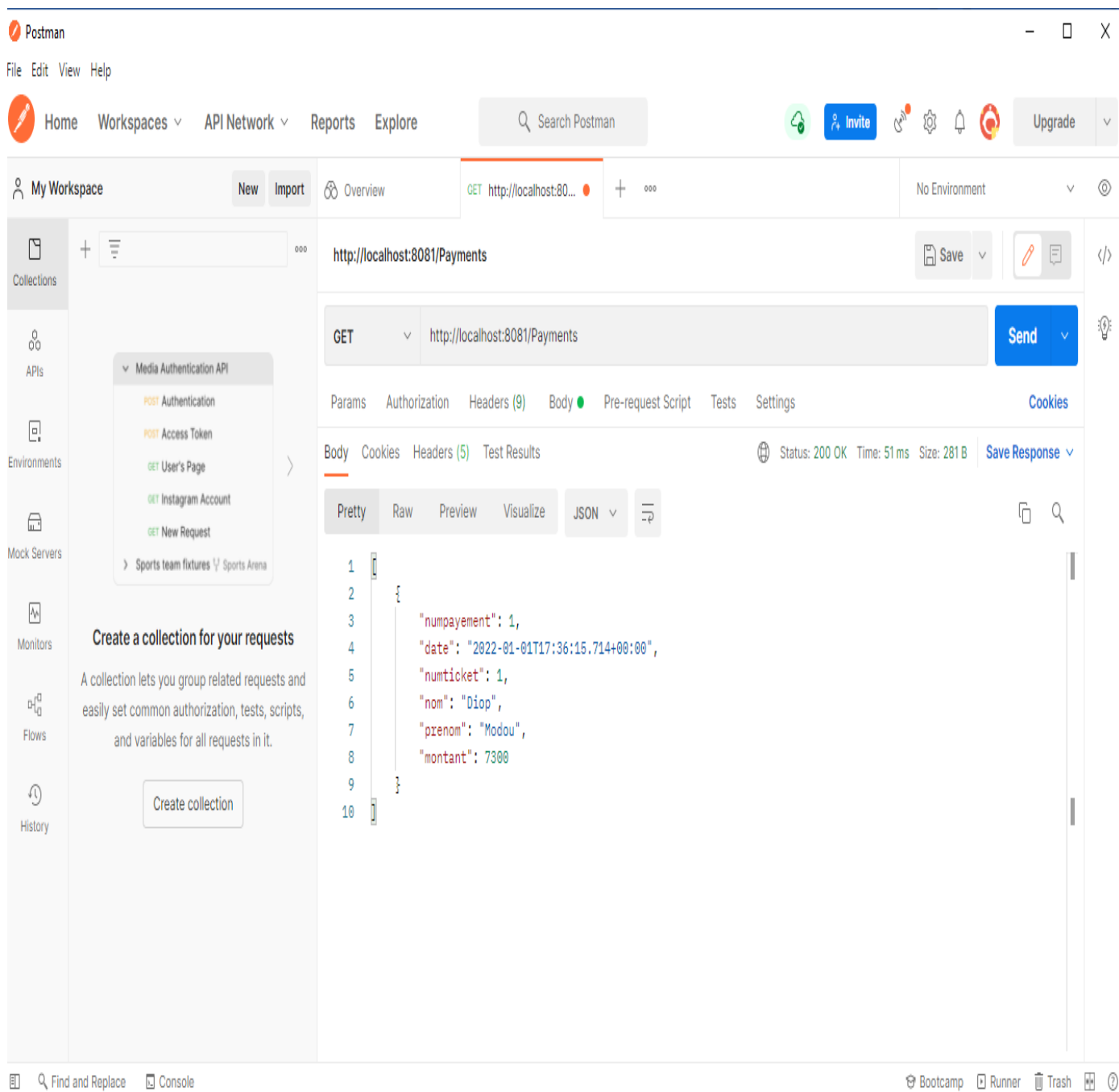
mysql> use bdpayements
Database changed
mysql> SELECT * FROM bdpayements.paiement;
+-----+-----+-----+-----+-----+-----+-----+
| numpaiement | date                | montant | nom | numticket | prenom |
+-----+-----+-----+-----+-----+-----+-----+
|           1 | 2022-01-01 14:15:52.509000 |    7300 | Diop |          1 | Modou |
+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql>
```

c. FONCTIONNEMENT DE L’AFFICHAGE ET DE L’ENREGISTREMENT DES PAYEMENT SUR POSTMAN

• AFFICHAGE

Ainsi la saisie de l’url « **http://localhost:8081/Payments** » avec comme type de requête « GET » dans Postman, renvoie les enregistrements de la table bdpayments. payment sous format **Json**.



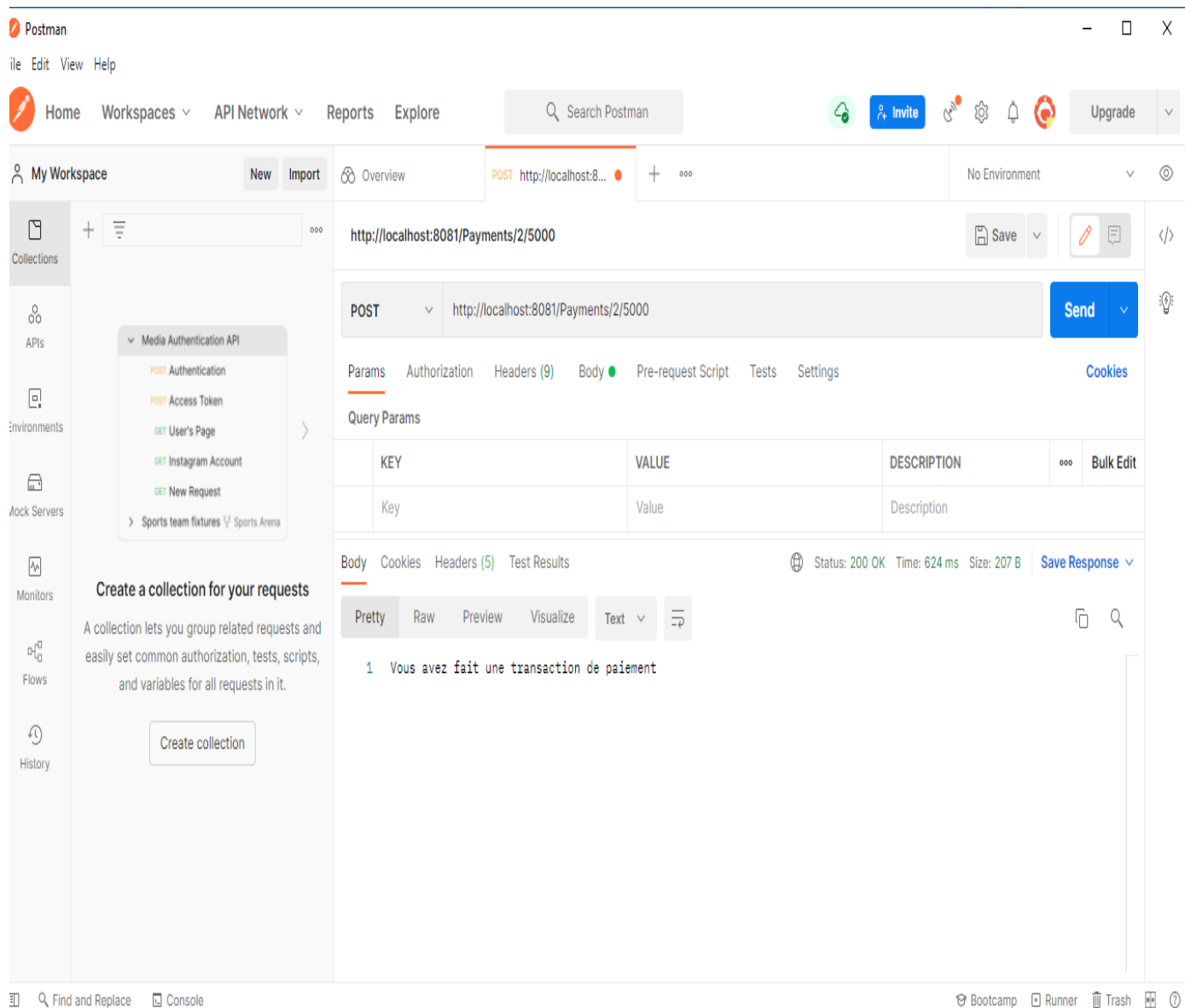
- ENREGISTREMENT

L'enregistrement d'un se patient fait en saisissant « **http://localhost:8081/Payments/{{numticket}} /{{montant}}** » dans Postman avec comme type de requête « POST ». La saisie se fait sous format **json**.

NB : pour numticket on fait référence au numticket de la table patient et pour montant on saisit le montant de notre choix.

Image illustrative de l'insertion d'un Payement

Avec comme url : **http://localhost:8081/Payments/2/5000** :



- Résultat dans la base de données après insertion

MySQL 8.0 Command Line Client - Unicode

```
mysql>
mysql>
mysql> use bdtickets
Database changed
mysql> select * from bdtickets.patients;
```

numtickets	age	date	nom	poids	prenom	temperature
1	18	2022-01-01 14:14:53.817000	Diop	73	Modou	37
2	25	2022-01-01 14:14:53.817000	DIANKHA	83	ABDOULAYE	37

```
2 rows in set (0.03 sec)

mysql> use bdpayements
Database changed
mysql> SELECT * FROM bdpayements.payement;
```

numpayment	date	montant	nom	numticket	prenom
1	2022-01-01 17:59:39.857000	7300	Diop	1	Modou

```
1 row in set (0.00 sec)

mysql> /*after insersion*/
mysql> SELECT * FROM bdpayements.payement;
```

numpayment	date	montant	nom	numticket	prenom
1	2022-01-01 17:59:39.857000	7300	Diop	1	Modou
2	2022-01-01 14:14:53.817000	5000	DIANKHA	2	ABDOULAYE

```
2 rows in set (0.05 sec)

mysql>
```

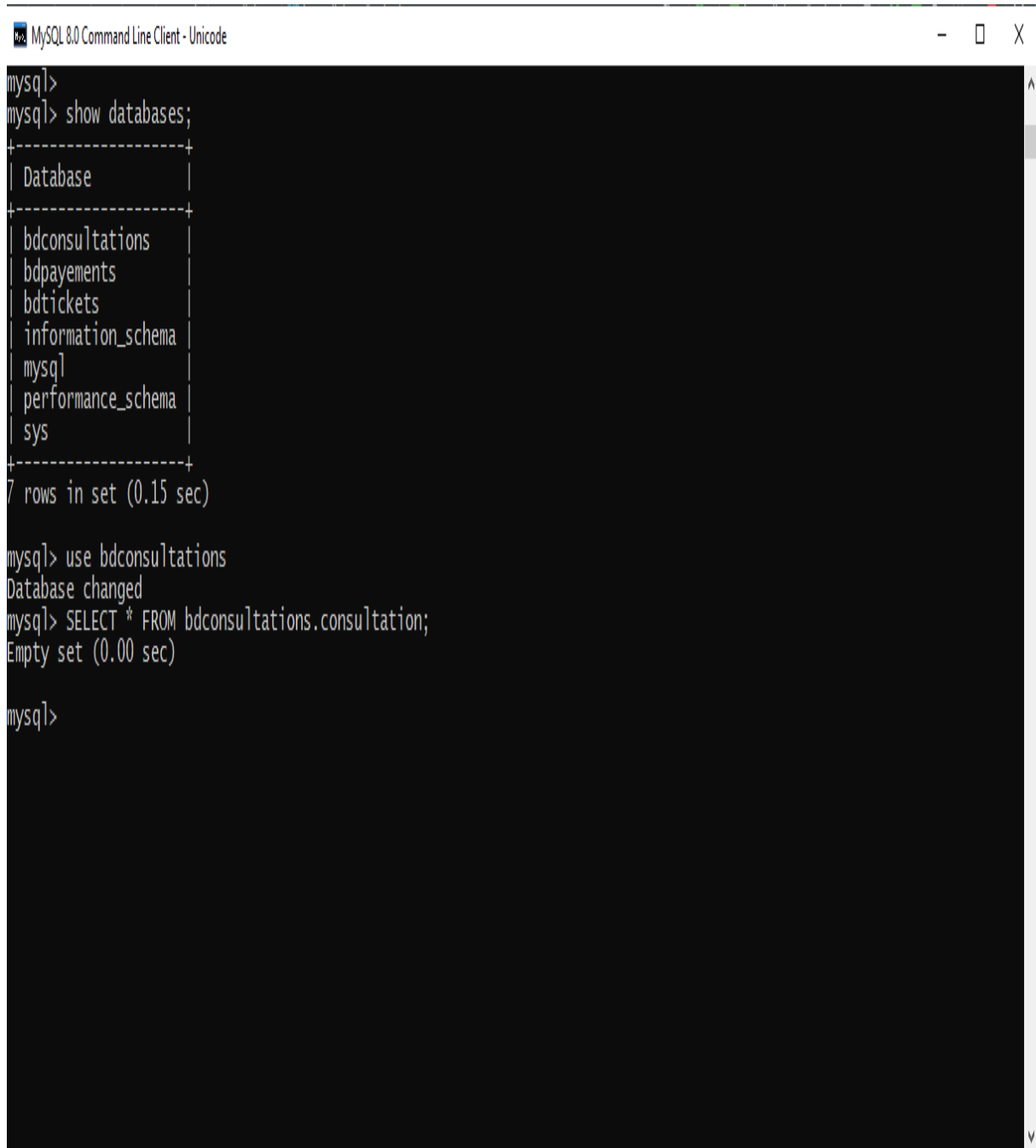

5. Réalisation de l'enregistrement des informations de consultation dans MySQL

Ce micro-service permet d'afficher l'ensemble les informations des consultations dans MySQL et d'en enregistrer plusieurs. Il contient entre autres un contrôleur **consultationController**, un repository **consultationRestRepositories**, une classe **Consultation** etc.

- La classe **Consultation** : contient les champs, *numconsultation*, *age*, *date*, *montant*, *nom*, *prenom*, *numtickets* ayant chacun un constructeur et un getter et setter pour chacun des champs.
- L'interface **consultationRestRepositories** : qui étend *JpaRepository<Consultation, Integer>* nous permettant de définir une interface de référentiel pour chaque entité de domaine de l'application.
- La classe **consultationController** qui contient :
 - La fonction *listepatients ()* : permettant ainsi de renvoyer la liste de tous les patients grâce à l'annotation *@GetMapping* qui mappe les requêtes http GET sur des méthodes de gestionnaire spécifiques.
 - La fonction *save ()* : qui permet d'ajouter des patients dans MySQL grâce à l'annotation *@PostMapping* pour gérer le type de méthode de requête POST etc.

a. DESCRIPTION DE LA BASE DE DONNEES Consultations

La base de données consultations est nommée bdconsultation qui dispose d'une table consultation contenant un certain nombre d'enregistrements



```
mysql>
mysql> show databases;
+-----+
| Database |
+-----+
| bdconsultations |
| bdpayements |
| bdtickets |
| information_schema |
| mysql |
| performance_schema |
| sys |
+-----+
7 rows in set (0.15 sec)

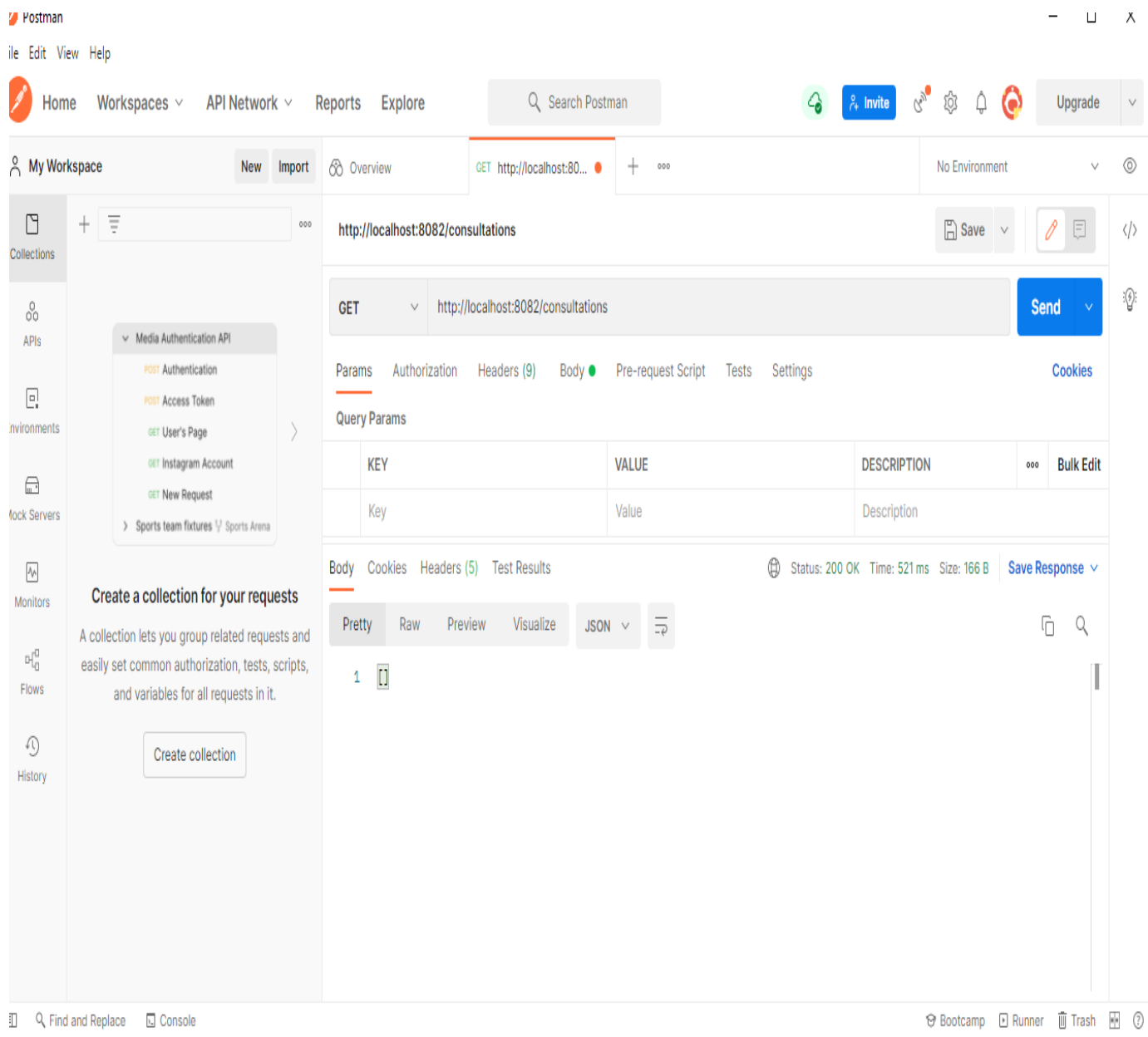
mysql> use bdconsultations
Database changed
mysql> SELECT * FROM bdconsultations.consultation;
Empty set (0.00 sec)

mysql>
```

d. FONCTIONNEMENT DE L’AFFICHAGE ET DE L’ENREGISTREMENT DES Consultations SUR POSTMAN

- AFFICHAGE

Ainsi la saisie de l’url « **http://localhost:8082/consultations** » avec comme type de requête « GET » dans Postman, renvoie les enregistrements de la table bdconsultations.consultation sous format **Json**.

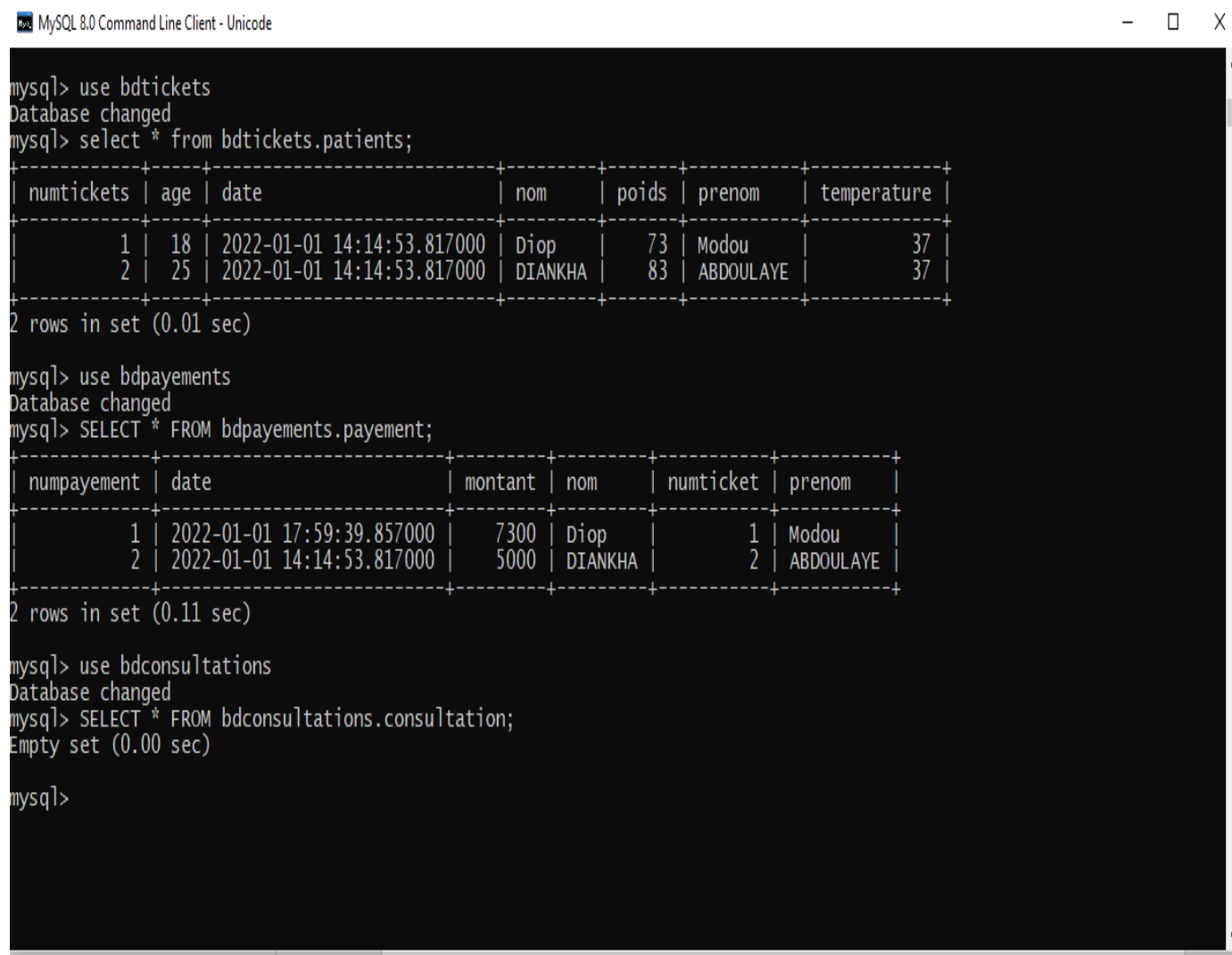


- ENREGISTREMENT

L'enregistrement d'un se patient fait en saisissant « **http://localhost:8082/consultations/{{numticket}}/{{numpayement}}** » dans Postman avec comme type de requête « POST ». La saisie se fait sous format **json**.

NB : pour numticket on fait référence au numticket de la table patient et pour numpayement on fait référence au numpayement de la table paiement.

Image illustrative des bases de données avant insertion :



```
mysql> use bdtickets
Database changed
mysql> select * from bdtickets.patients;
+-----+-----+-----+-----+-----+-----+-----+
| numtickets | age | date                | nom    | poids | prenom | temperature |
+-----+-----+-----+-----+-----+-----+-----+
|          1 | 18 | 2022-01-01 14:14:53.817000 | Diop   | 73    | Modou  | 37           |
|          2 | 25 | 2022-01-01 14:14:53.817000 | DIANKHA | 83    | ABDOULAYE | 37           |
+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.01 sec)

mysql> use bdpayements
Database changed
mysql> SELECT * FROM bdpayements.paiement;
+-----+-----+-----+-----+-----+-----+-----+
| numpayement | date                | montant | nom    | numticket | prenom |
+-----+-----+-----+-----+-----+-----+-----+
|          1 | 2022-01-01 17:59:39.857000 | 7300    | Diop   | 1          | Modou  |
|          2 | 2022-01-01 14:14:53.817000 | 5000    | DIANKHA | 2          | ABDOULAYE |
+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.11 sec)

mysql> use bdconsultations
Database changed
mysql> SELECT * FROM bdconsultations.consultation;
Empty set (0.00 sec)

mysql>
```

- Maintenant insérons

Avec comme `http://localhost:8082/consultations/2 /2` en utilisant la méthodes POST

The screenshot shows the Postman application interface. The top bar includes the Postman logo, a menu (File, Edit, View, Help), navigation tabs (Home, Workspaces, API Network, Reports, Explore), a search bar, and utility buttons (Invite, settings, notifications). The left sidebar contains a 'My Workspace' section with 'New' and 'Import' buttons, and a 'Collections' section with a search bar and a list of collections. The main panel displays a POST request to `http://localhost:8082/consultations/2/2`. The request is configured with the following settings:

- Method: POST
- URL: `http://localhost:8082/consultations/2/2`
- Params: Authorization, Headers (9), Body, Pre-request Script, Tests, Settings
- Query Params: A table with columns KEY, VALUE, and DESCRIPTION. The first row shows 'Key' and 'Value'.
- Body: A text box containing the message '1 Vous avez fait une transaction de consultation'.
- Status: 200 OK, Time: 556 ms, Size: 211 B

The bottom of the interface shows a 'Find and Replace' bar and a 'Console' tab.

- Résultat dans la base de données après insertion

```
MySQL 8.0 Command Line Client - Unicode
mysql> use bdtickets
Database changed
mysql> select * from bdtickets.patients;
+-----+-----+-----+-----+-----+-----+
| numtickets | age | date                | nom    | poids | prenom  | temperature |
+-----+-----+-----+-----+-----+-----+
|          1 | 18 | 2022-01-01 14:14:53.817000 | Diop   | 73    | Modou   | 37           |
|          2 | 25 | 2022-01-01 14:14:53.817000 | DIANKHA | 83    | ABDOULAYE | 37           |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.01 sec)

mysql> use bdpayements
Database changed
mysql> SELECT * FROM bdpayements.payement;
+-----+-----+-----+-----+-----+-----+
| numpayement | date                | montant | nom    | numticket | prenom  |
+-----+-----+-----+-----+-----+-----+
|          1 | 2022-01-01 17:59:39.857000 | 7300    | Diop   |          1 | Modou   |
|          2 | 2022-01-01 14:14:53.817000 | 5000    | DIANKHA |          2 | ABDOULAYE |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.11 sec)

mysql> use bdconsultations
Database changed
mysql> SELECT * FROM bdconsultations.consultation;
Empty set (0.00 sec)

mysql> /*Apres INSERTION*/
mysql> SELECT * FROM bdconsultations.consultation;
+-----+-----+-----+-----+-----+-----+
| numconsultation | age | date                | montant | nom    | numticket | prenom  |
+-----+-----+-----+-----+-----+-----+
|              1 | 25 | 2022-01-01 14:14:53.817000 | 5000    | DIANKHA |          2 | ABDOULAYE |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.01 sec)
```

Conclusion :

L'architecture des microservices est une approche de conception distribuée destinée à surmonter les limites des architectures monolithiques traditionnelles. Les microservices aident à faire évoluer les applications et les organisations tout en améliorant les temps de cycle. Cependant, ils présentent également quelques défis qui pourraient ajouter une complexité architecturale et une charge opérationnelle supplémentaires. AWS propose un large portefeuille de services gérés qui peuvent aider les équipes produit à créer des architectures de microservices et à minimiser la complexité architecturale et opérationnelle.