

RAPPORT BACK-END



Université Assane SECK de Ziguinchor

UFR SCIENCES ET TECHNOLOGIES

Département : Informatique

Filière : L3-2i option GL

Projet :

GESTION EVENEMENT

Année Scolaire 2022-2023

Professeur : El Bachir TOURE

SOMMAIRE

I) Fonctionnement du Back-End

- 1) Création d'évènement par les clients
- 2) Ajout de prestataires à l'évènement
- 3) Sécurisation du Système
- 4) CRUD des fonctionnalités

II) Diagramme de Classe

III) Capture D'image des fonctionnalités

(Postman)

- 1) Image Postman Clients
- 2) Image Postman Prestataires
- 3) Image Postman Evènements

IV) Auteur du projet

I). Fonctionnement du Back-End

Résumé : La plateforme de gestion d'événements a été développée pour permettre aux clients de créer, personnaliser et gérer leurs propres événements. Ce rapport présente les principales fonctionnalités implémentées, notamment la création d'événements, l'ajout de prestataires, la sécurisation du système, et la vérification du fonctionnement du CRUD

1) Création d'événements par les clients :

Les clients ont la possibilité d'initier la création d'événements en envoyant des requêtes au backend. Les informations essentielles, telles que le nom de l'événement, la date, la description, et le lieu, sont incluses dans ces requêtes. Le backend est conçu pour recevoir, valider et enregistrer ces données dans une base de données dédiée. Cette fonctionnalité offre aux clients la flexibilité nécessaire pour personnaliser leurs événements selon leurs besoins

2) Ajout de prestataires à l'événement :

Une fois l'événement créé, les clients ont la possibilité d'ajouter des prestataires en fonction des besoins spécifiques de leur événement. Ces prestataires peuvent être des fournisseurs de services variés, tels que des traiteurs, des photographes, des DJ, etc. Cette fonctionnalité enrichit l'expérience des clients en leur permettant de coordonner divers aspects de leur événement à partir de la même plateforme.

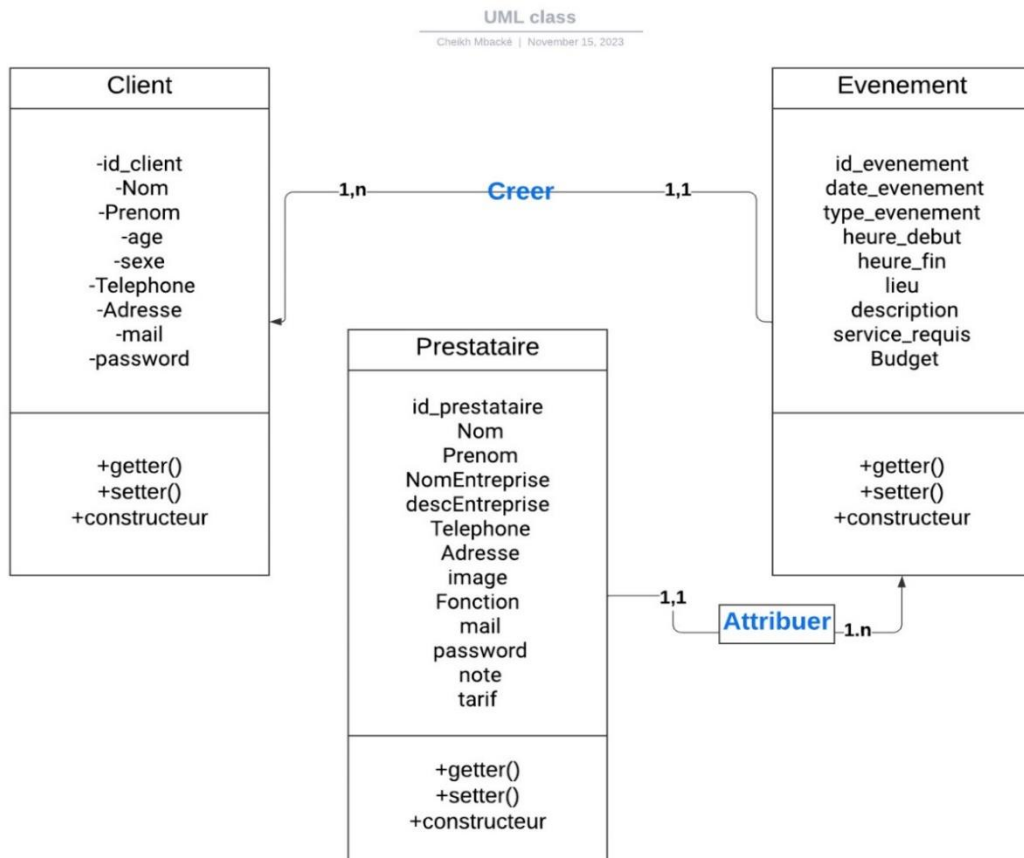
3) Sécurisation du Système :

Spring Security a été configuré pour gérer l'authentification des utilisateurs, assurant une vérification stricte de leurs identités. De plus, le système d'autorisation a été mis en place pour déterminer avec précision les actions et les ressources auxquelles chaque utilisateur a accès. Ceci garantit une sécurité granulaire alignée sur les besoins spécifiques de chaque rôle utilisateur.

4) CRUD des Fonctionnalités :

Le système a été testé et validé pour confirmer le bon fonctionnement des opérations **CRUD** (Create, Read, Update, Delete) sur les différentes fonctionnalités. Cela garantit que les utilisateurs peuvent créer, visualiser, mettre à jour et supprimer les événements ainsi que les informations associées de manière fiable

II). Diagramme de Classe



➤ Base de données [gestEvenBD]

Cette structure de base permet de stocker les informations essentielles pour les clients, les événements et les prestataires, tout en établissant des relations significatives entre ces entités.

```

pegasus77@pegasus77-Latitude-7490 ~$
pegasus77@pegasus77-Latitude-7490 ~$ mysql Mariadb -u root -p
Enter password:
Welcome to the Mariadb monitor. Commands end with ; or \g.
Your Mariadb connection id is 661
Server version: 10.6.12-MariaDB-Debian-0.0.1-QA0 2023-02-04

Copyright (c) 2000, 2015, Oracle, MariaDB Corporation AB and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> use gestEvenBD;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

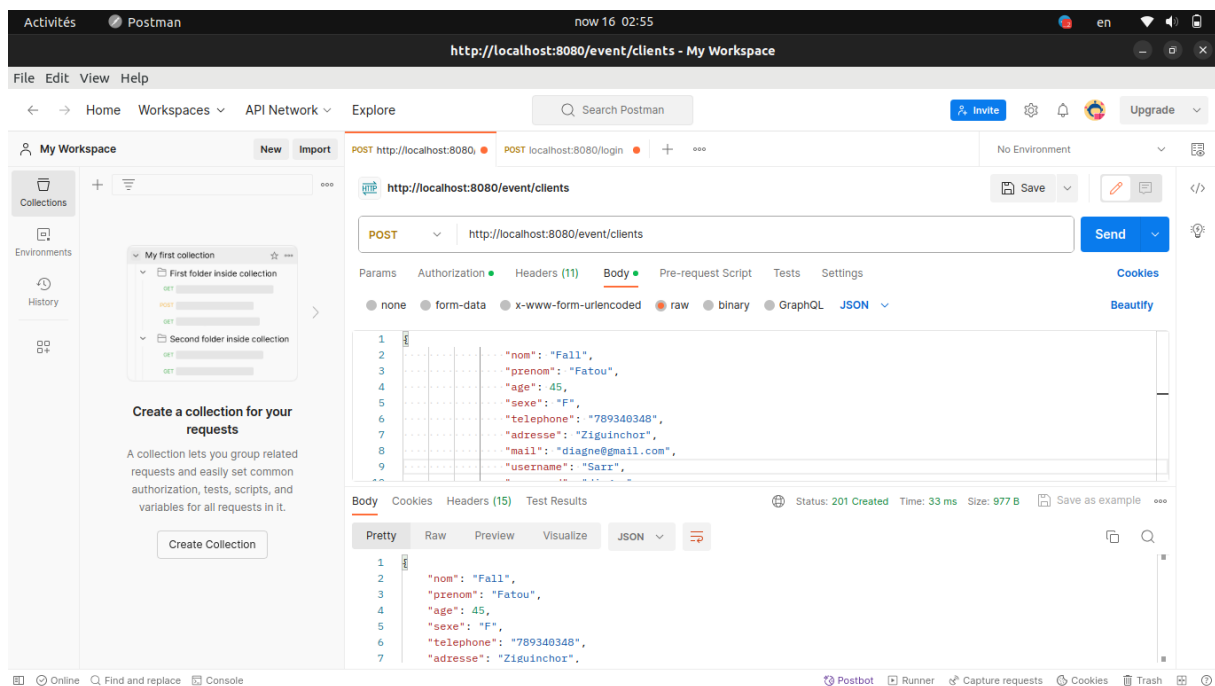
Database changed
MariaDB [(none)]> show tables;
+-----+
| Tables_in_gestEvenBD |
+-----+
| client |
| evenement |
| prestataire |
| service_requis |
+-----+
1 row in set (0.000 sec)
    
```

III). Capture D'image des fonctionnalités (Postman)

1). Image Postman Client

Voici une capture d'écran à partir de Postman illustrant les opérations CRUD

➤ POST : l'opération POST



- GET : Nous disposons des informations clients, y compris son adresse e-mail et son mot de passe, nécessaires à son authentification, accessibles via la méthode GET. L'URL ci-dessous nous indique que le client a la possibilité d'organiser un ou plusieurs événements`[http://localhost:8080/event/clients/2/evenements`](http://localhost:8080/event/clients/2/evenements)

Activités Postman now 16 03:01

http://localhost:8080/login - My Workspace

File Edit View Help

← → Home Workspaces API Network Explore

Search Postman

My Workspace New Import

POST http://localhost:8080/ POST localhost:8080/login GET http://localhost:8080/

No Environment

http://localhost:8080/login

GET http://localhost:8080/event/clients

Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

This request does not have a body

Status: 200 OK Time: 43 ms Size: 1.97 KB Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "_embedded": {
3     "clients": [
4       {
5         "nom": "Dombia",
6         "prenom": "Fatou",
7         "age": 45,
8         "sexe": "F",
9         "telephone": "789340348",
10        "adresse": "Ziguinchor",
11        "mail": "diagne@gmail.com",
12        "username": "user",
13        "password": "$2a$10$NVm0n8ElaRgg7zW01CxUde17vWoPg91lz2aYavh9.f9q0e4bRadue",
14        "role": "USER",
15        "_links": {
```

PUT:

Activités Postman now 16 03:17

http://localhost:8080/login - My Workspace

File Edit View Help

← → Home Workspaces API Network Explore

Search Postman

My Workspace New Import

POST http://localhost:8080/ POST localhost:8080/login PATCH http://localhost:8080/

No Environment

http://localhost:8080/login

PATCH http://localhost:8080/event/clients/1

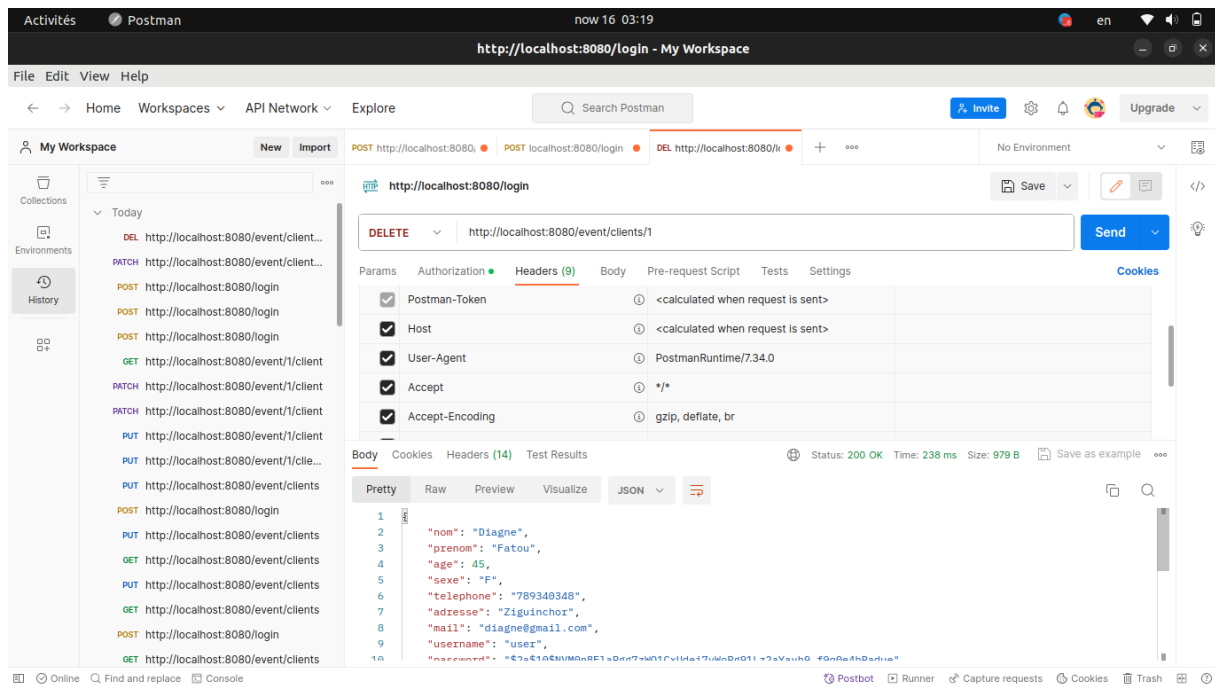
Send

Params Authorization Headers (11) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
1 {
2   "nom": "Diagne",
3   "prenom": "Fatou",
4   "age": 45,
5   "sexe": "F",
6   "telephone": "789340348",
7   "adresse": "Ziguinchor",
8   "mail": "diagne@gmail.com",
9   "username": "user",
10  "password": "$2a$10$NVm0n8ElaRgg7zW01CxUde17vWoPg91lz2aYavh9.f9q0e4bRadue"
```

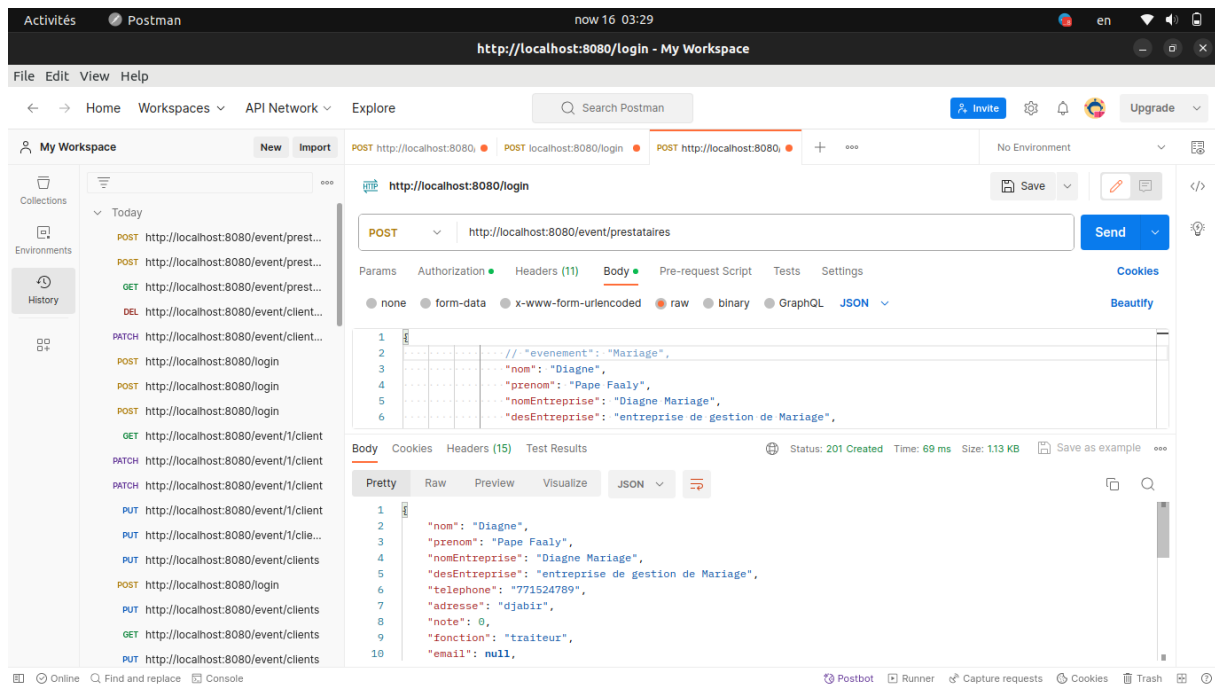
➤ DELETE :



2). Image Postman Prestataire

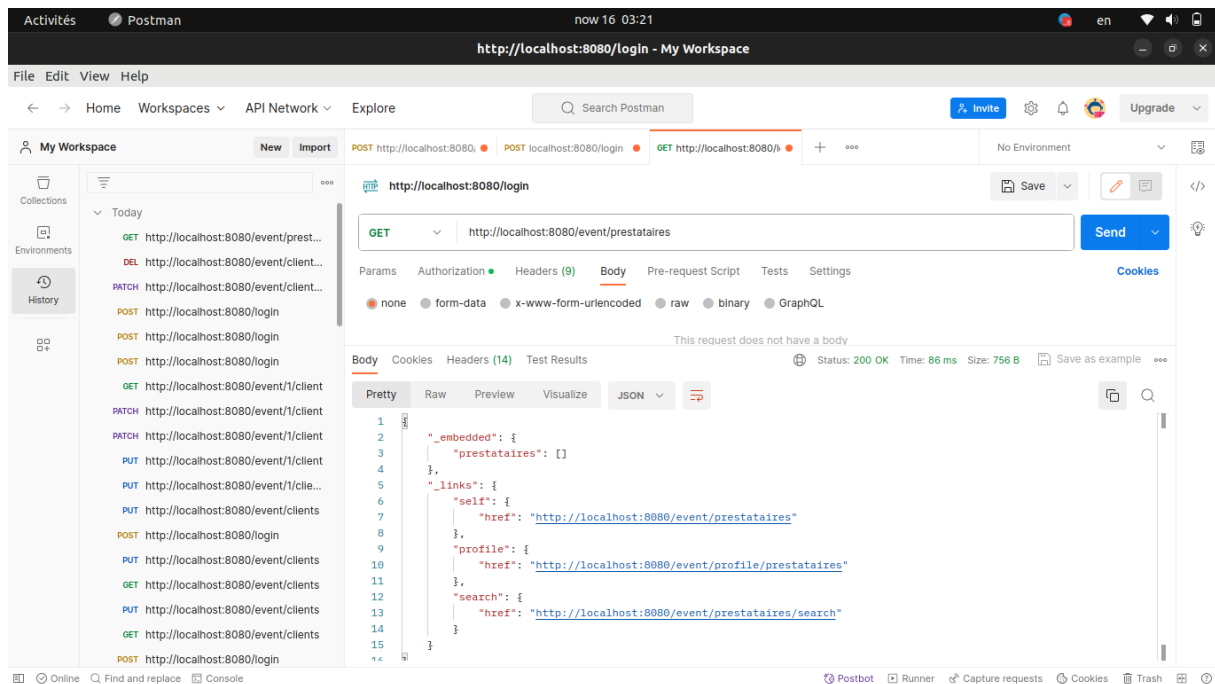
Voici une capture d'écran issue de Postman, présentant de manière visuelle les opérations CRUD.

➤ POST :

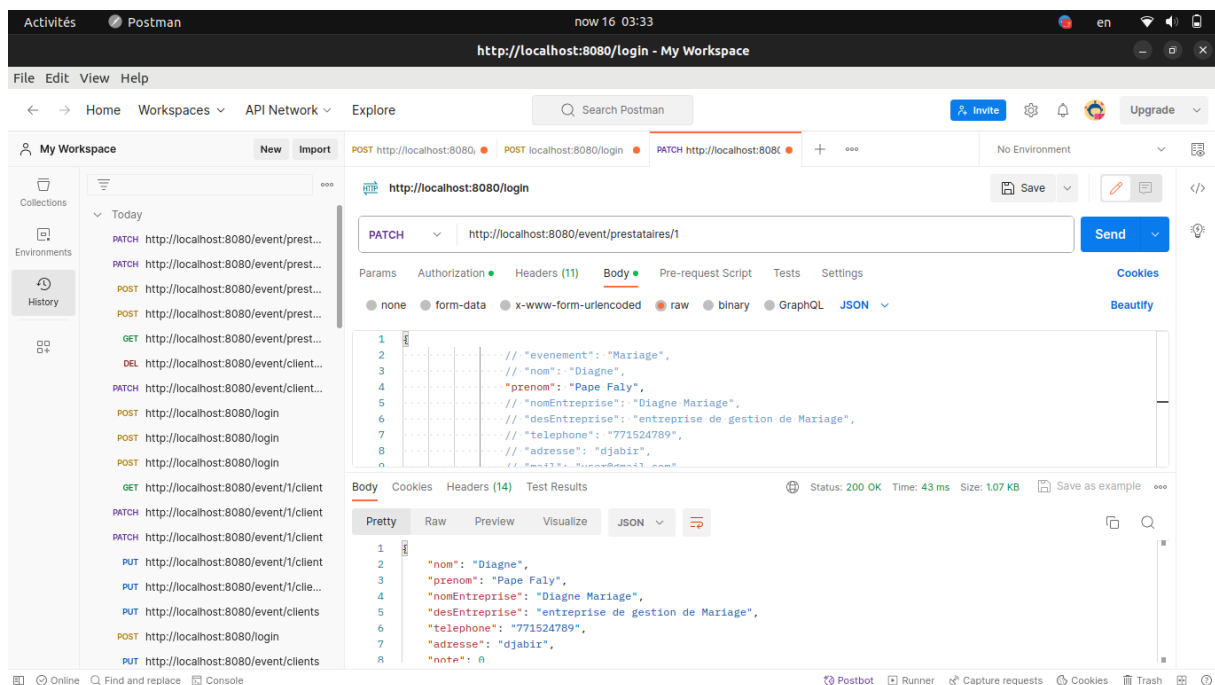


➤ GET :

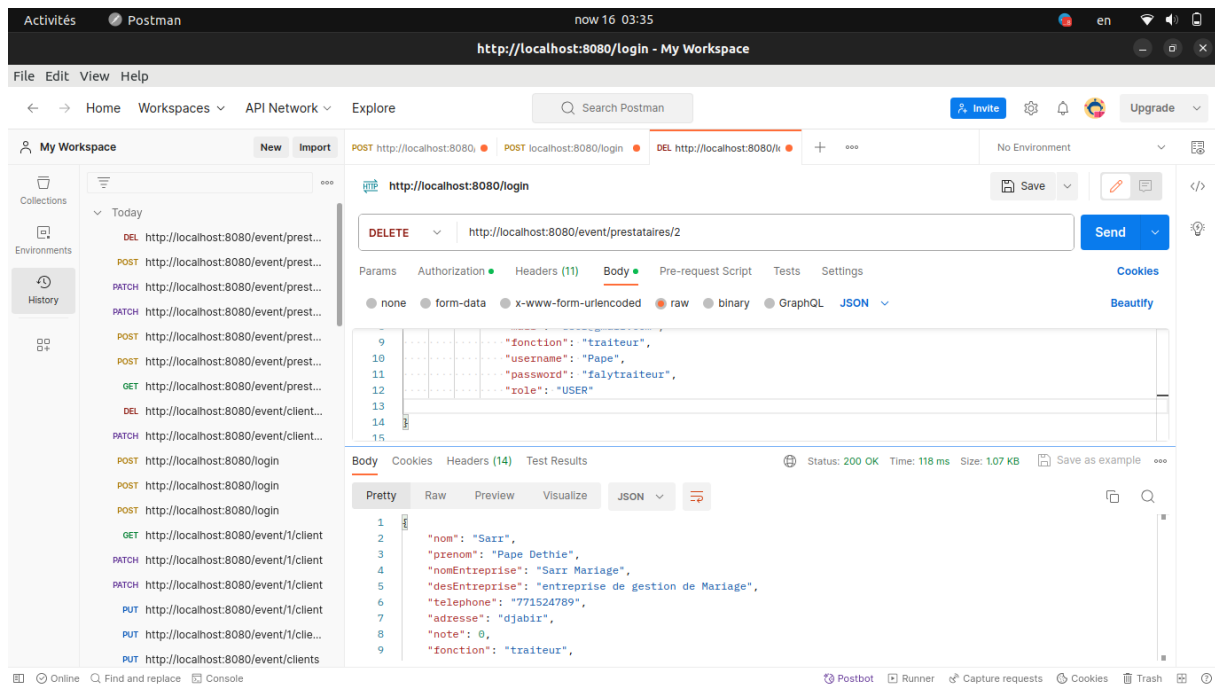
Nous avons également accès aux informations relatives aux prestataires. Le lien '<http://localhost:8080/event/prestataire/1/evenements>' indique qu'un prestataire est associé à un événement spécifique



➤ PUT :

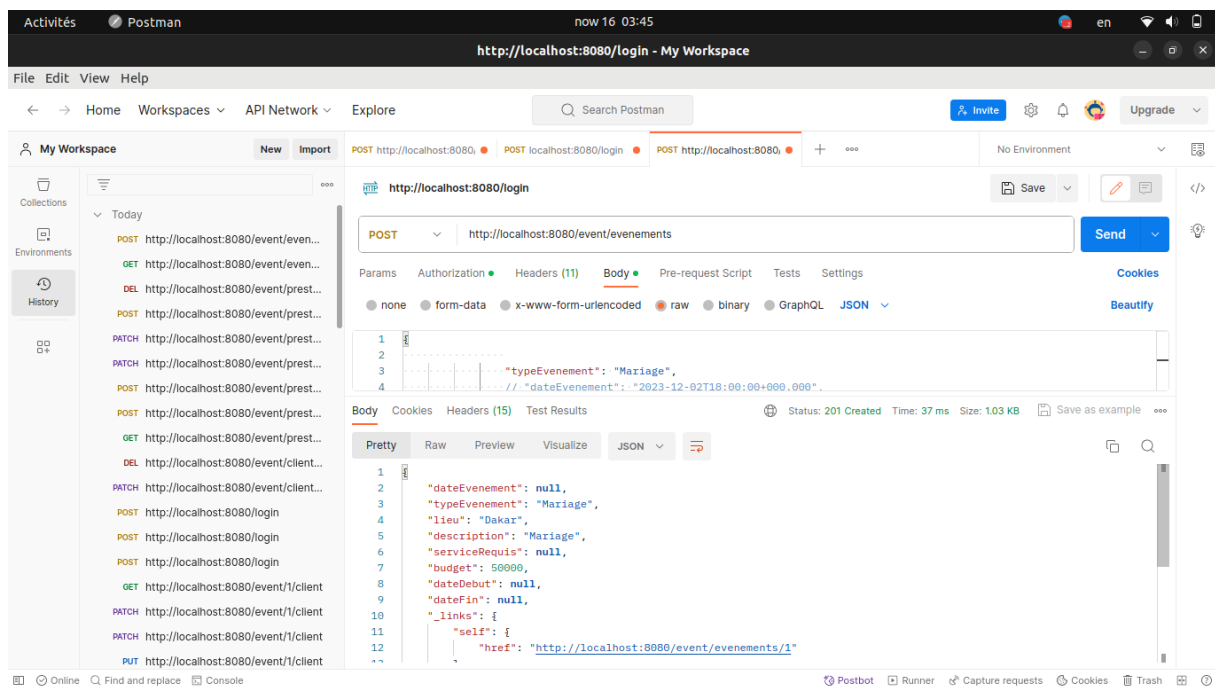


➤ DELETE :



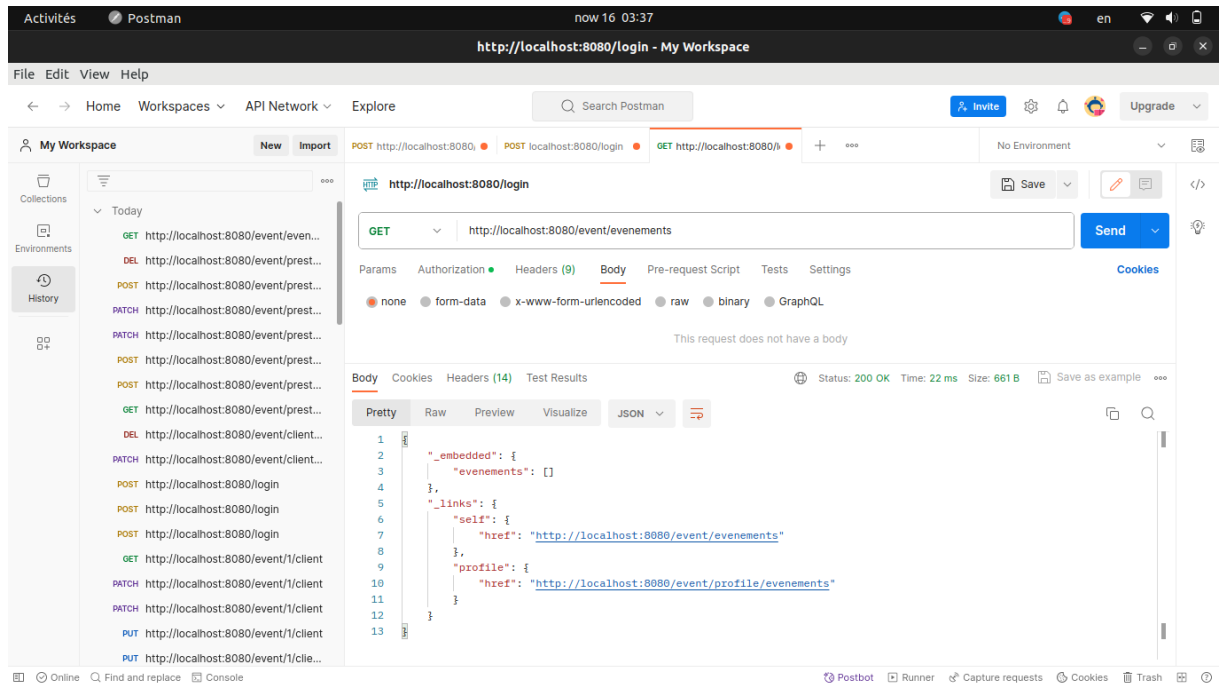
3). Image Postman Evènement

➤ POST :



➤ GET :

En ce qui concerne les événements, nous disposons également de leurs informations détaillées. Il est à noter qu'un événement peut être associé à un ou plusieurs prestataires, comme indiqué par le lien '<http://localhost:8080/event/prestataire/1/evenements>'.



Activités Postman now 16 03:37

http://localhost:8080/login - My Workspace

File Edit View Help

← → Home Workspaces API Network Explore

Search Postman

My Workspace

Today

- GET http://localhost:8080/event/evenements
- DEL http://localhost:8080/event/prest...
- POST http://localhost:8080/event/prest...
- PATCH http://localhost:8080/event/prest...
- PATCH http://localhost:8080/event/prest...
- POST http://localhost:8080/event/prest...
- POST http://localhost:8080/event/prest...
- GET http://localhost:8080/event/prest...
- DEL http://localhost:8080/event/client...
- PATCH http://localhost:8080/event/client...
- POST http://localhost:8080/login
- POST http://localhost:8080/login
- POST http://localhost:8080/login
- GET http://localhost:8080/event/1/client...
- PATCH http://localhost:8080/event/1/client...
- PATCH http://localhost:8080/event/1/client...
- PUT http://localhost:8080/event/1/client...
- PUT http://localhost:8080/event/1/client...

http://localhost:8080/login

GET http://localhost:8080/event/evenements

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

This request does not have a body

Body Cookies Headers (14) Test Results

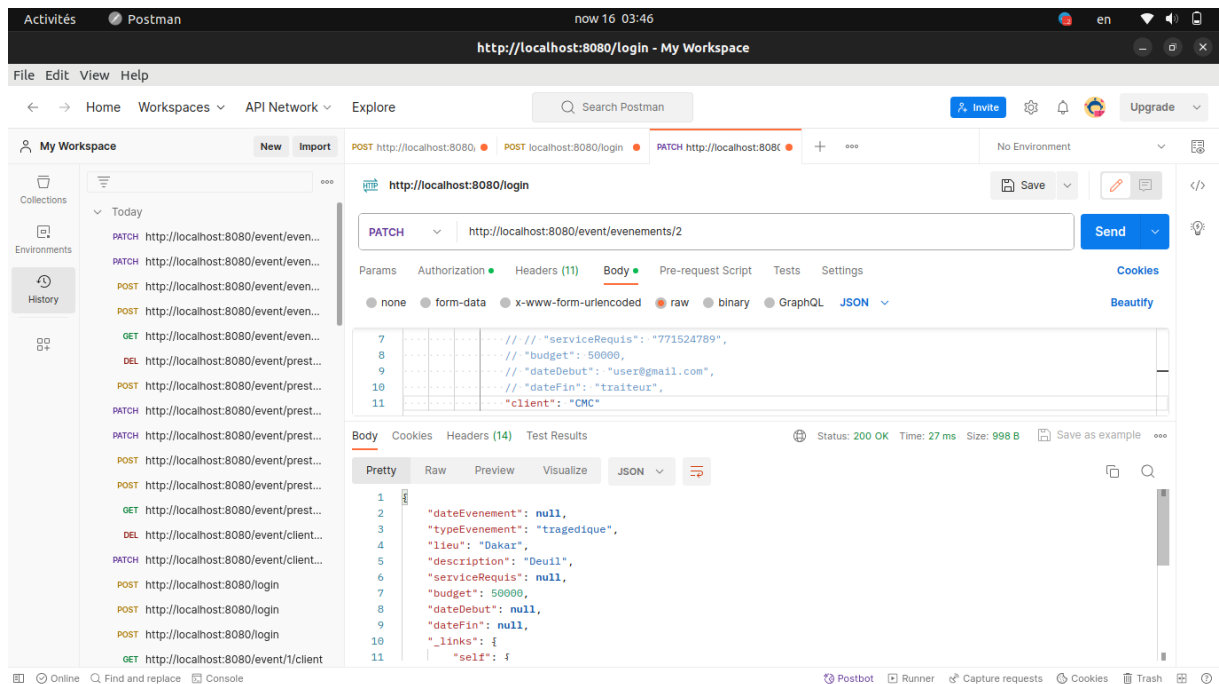
Status: 200 OK Time: 22 ms Size: 661 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "_embedded": {
3     "evenements": []
4   },
5   "_links": {
6     "self": {
7       "href": "http://localhost:8080/event/evenements"
8     },
9     "profile": {
10      "href": "http://localhost:8080/event/profile/evenements"
11    }
12  }
13 }
```

Postbot Runner Capture requests Cookies Trash

➤ PUT :



Activités Postman now 16 03:46

http://localhost:8080/login - My Workspace

File Edit View Help

← → Home Workspaces API Network Explore

Search Postman

My Workspace

Today

- PATCH http://localhost:8080/event/even...
- PATCH http://localhost:8080/event/even...
- POST http://localhost:8080/event/even...
- POST http://localhost:8080/event/even...
- DEL http://localhost:8080/event/prest...
- POST http://localhost:8080/event/prest...
- PATCH http://localhost:8080/event/prest...
- PATCH http://localhost:8080/event/prest...
- POST http://localhost:8080/event/prest...
- POST http://localhost:8080/event/prest...
- GET http://localhost:8080/event/prest...
- DEL http://localhost:8080/event/client...
- PATCH http://localhost:8080/event/client...
- POST http://localhost:8080/login
- POST http://localhost:8080/login
- POST http://localhost:8080/login
- GET http://localhost:8080/event/1/client...
- PATCH http://localhost:8080/event/1/client...
- PATCH http://localhost:8080/event/1/client...
- PUT http://localhost:8080/event/1/client...
- PUT http://localhost:8080/event/1/client...

http://localhost:8080/login

PUT http://localhost:8080/event/evenements/2

Params Authorization Headers (11) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

Beautify

```
7 {
8   "serviceRequis": "771524789",
9   "budget": 50000,
10  "dateDebut": "user@gmail.com",
11  "dateFin": "traiteur",
12  "client": "CMC"
13 }
```

Body Cookies Headers (14) Test Results

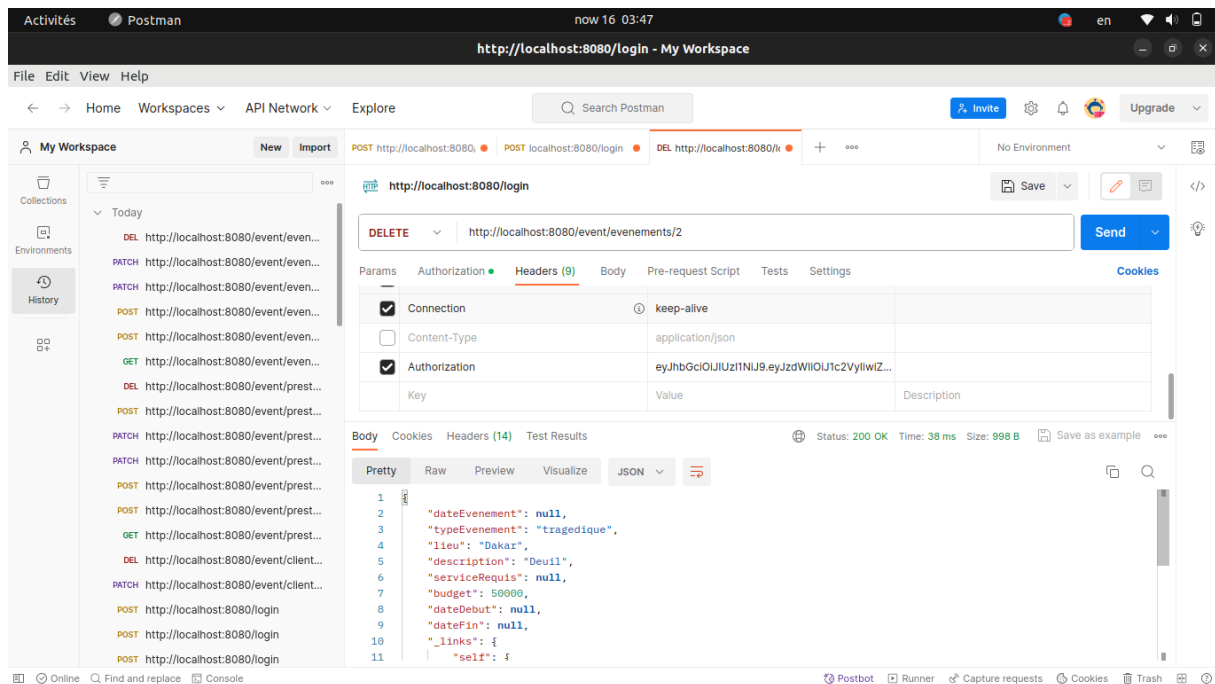
Status: 200 OK Time: 27 ms Size: 998 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "dateEvenement": null,
3   "typeEvenement": "tragedique",
4   "lieu": "Dakar",
5   "description": "Deuil",
6   "serviceRequis": null,
7   "budget": 50000,
8   "dateDebut": null,
9   "dateFin": null,
10  "_links": {
11    "self": {
12      "href": "http://localhost:8080/event/evenements/2"
13    }
14  }
15 }
```

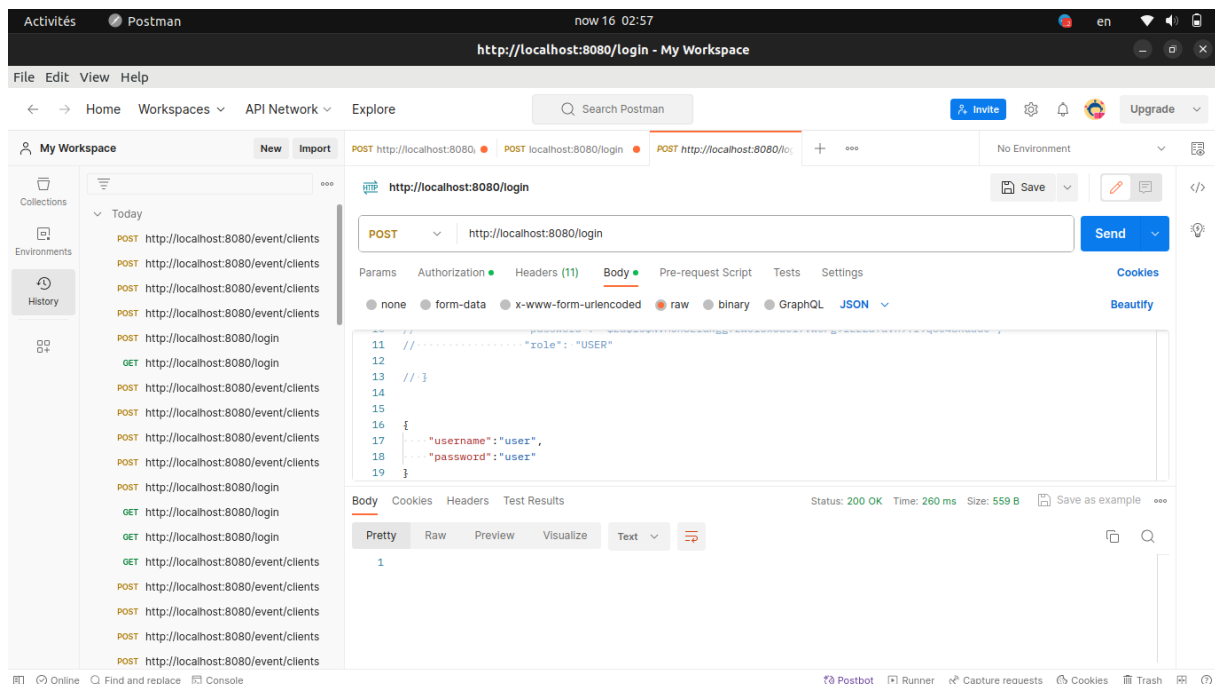
Postbot Runner Capture requests Cookies Trash

➤ DELETE :



4). Image Postman sécurisation (Login)

Pour Postman, en ce qui concerne la sécurisation, nous avons mis en place deux utilisateurs distincts : un utilisateur standard ("user") et un administrateur ("admin"). Ces identifiants sont utilisés pour tester et valider les différents scénarios de sécurité dans le contexte de notre application. Les autorisations et les accès sont différenciés entre ces deux rôles pour refléter de manière réaliste les divers niveaux de privilèges au sein de la plateforme. Vous trouverez ci-dessous des captures d'écran spécifiques à chaque utilisateur pour illustrer leurs interactions respectives avec l'application.



IV). Auteur du projet



Cheikh Mbacke COLY



202000142



<https://github.com/CMCode2001>



Niako KEBE



202001930



<https://github.com/NiakoDev2i>



Pape Dethie SARR



202002010



<https://github.com/BfdCode>



Mouhamet DIAGNE



202000142



<https://github.com/Mhdiagne>



Pape Faly DIAGNE



202001957



<https://github.com/PapaFaly666>