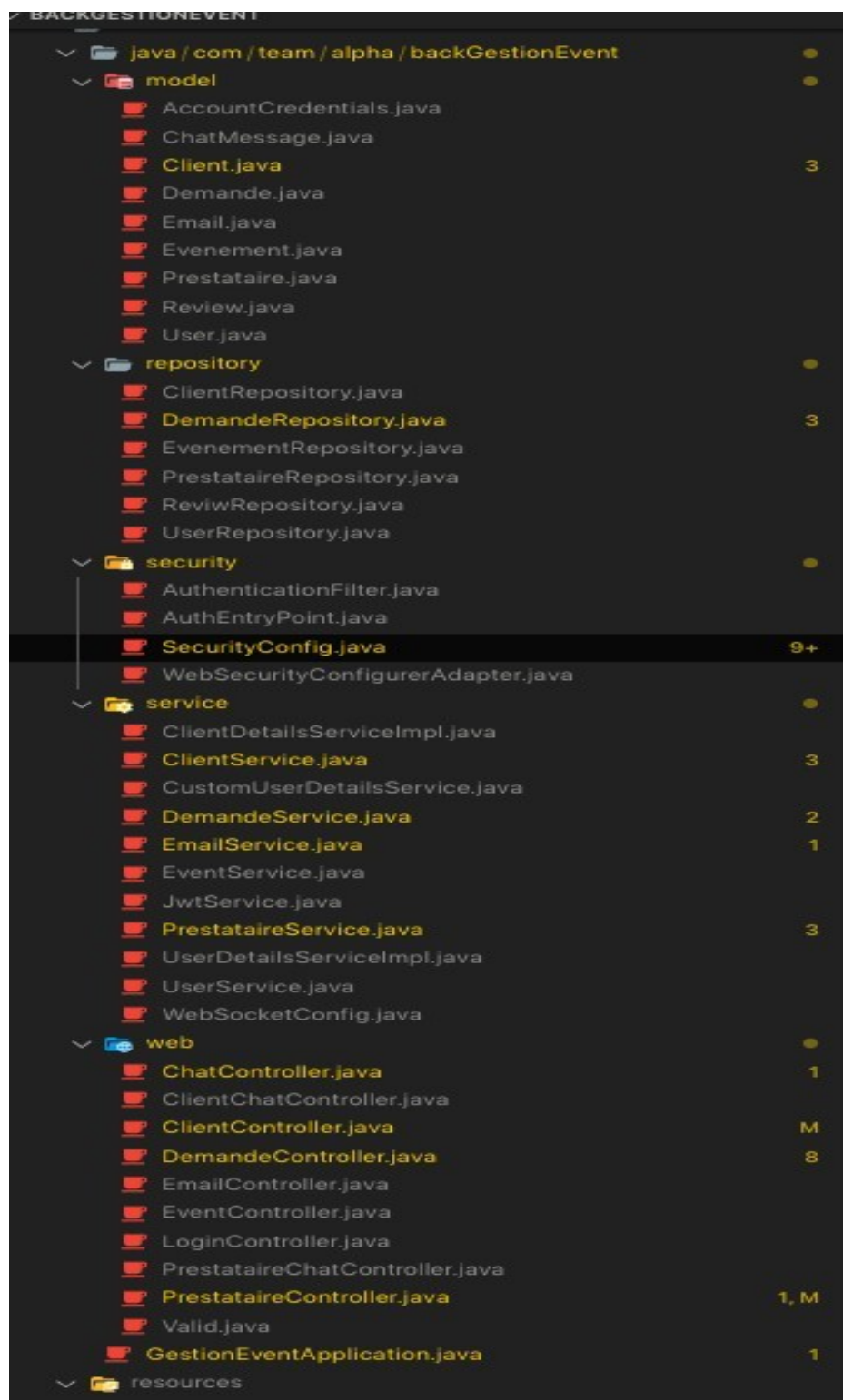


RAPPORT DU PROJET GESTION EVENEMENT

Dans le cadre de ce projet portant sur la gestion des événements, nous avons développé un système robuste permettant aux clients de créer et organiser des événements de manière efficace et personnalisée. Dans les lignes qui suivront nous allons exposer les fonctionnalités du back-end.

Arborescence des classes.



Nous avons eu a ajouter les dépendances suivantes :

1. ****Spring Boot Starter Web:**

Simplifie le développement d'une application web en utilisant Spring MVC, en fournissant des dépendances et des configurations essentielles pour la création de services web.

2. ****Spring Boot Starter Data REST:****

Facilite la création d'API REST en intégrant les dépôts Spring Data dans l'application, exposant automatiquement les données sous forme de points d'accès REST.

3. ****MariaDB JDBC Driver:****

Permet la communication entre l'application Java et la base de données MariaDB, assurant la connectivité et les interactions avec la base de données.

4. ****Spring Boot Starter Data JPA:****

Fournit une série de configurations par défaut pour utiliser Spring Data JPA, simplifiant la mise en œuvre de l'accès aux données via l'API de persistance Java (JPA).

5. ****Spring Boot Starter Security:****

Intègre Spring Security dans le projet, offrant des solutions prêtes à l'emploi pour l'authentification et l'autorisation dans une application Spring Boot.

6. ****Dépendances JWT (JSON Web Token):****

- ****jjwt-api : **** API pour la manipulation des JSON Web Tokens.
- ****jjwt-impl et jjwt-jackson : **** Implémentations et prise en charge de Jackson pour les JSON Web Tokens, utilisés pour l'échange d'informations sécurisé.

7. ****Spring Boot Starter Test:****

Inclut des dépendances de test pour les tests unitaires et d'intégration dans une application Spring Boot.

8. ****Spring Boot Starter Mail:****

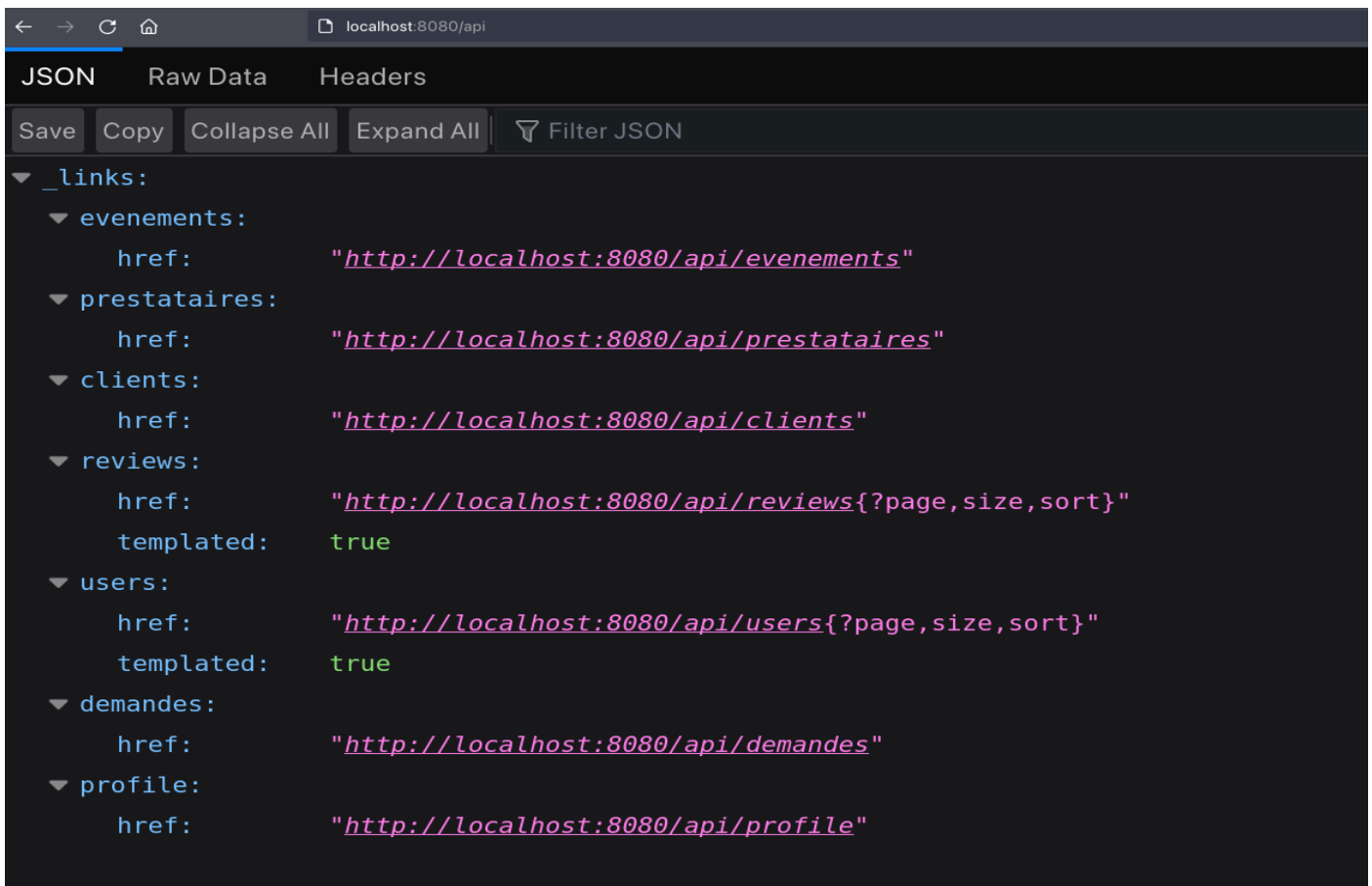
Simplifie l'intégration des e-mails dans les applications Spring Boot en fournissant des paramètres préconfigurés pour l'envoi d'e-mails.

9. ****Dépendances WebSocket:****

- ****Spring Boot Starter WebSocket :** Simplifie l'intégration des WebSockets dans Spring Boot.
- ****sockjs-client :** Bibliothèque JavaScript pour la communication WebSocket.
- ****stomp-websocket :** Fournit un support pour le protocole de messagerie orienté texte simple (STOMP) dans la communication WebSocket.

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-rest</artifactId>
</dependency>
```

Ce qui nous permet d'obtenir les points de terminaison suivants



The screenshot shows a web browser window with the address bar displaying `localhost:8080/api`. The browser's developer tools are open, showing the JSON response of the API. The response is a list of links for various API endpoints.

```
{
  "_links": {
    "evenements": {
      "href": "http://localhost:8080/api/evenements"
    },
    "prestataires": {
      "href": "http://localhost:8080/api/prestataires"
    },
    "clients": {
      "href": "http://localhost:8080/api/clients"
    },
    "reviews": {
      "href": "http://localhost:8080/api/reviews{?page,size,sort}"
    },
    "users": {
      "href": "http://localhost:8080/api/users{?page,size,sort}"
    },
    "demandes": {
      "href": "http://localhost:8080/api/demandes"
    },
    "profile": {
      "href": "http://localhost:8080/api/profile"
    }
  }
}
```

Remarque Pour que les clients et les prestataires puissent se connecter dans la plateforme on les a associés à une classe User

à chaque fois qu'un Client ou un Prestataire est créé, un utilisateur (User) est créé. Nous allons y revenir plus tard.

Même si on a utilisé `spring-boot-starter-data-rest` pour automatiser la création des end-points, pour certaines opérations on a jugé nécessaire de créer des end-points personnalisés.

Dans `ClientController` on a

```
@RequestMapping("/client")
```

```
//Pour récupérer le client associé à User grâce à son email
```

```
@GetMapping("/mail")
public Client getClientByMail(@RequestParam String mail) {
    return clientService.getClientByMail(mail).get();
}
```

Quelques Points de terminaison personnalisés

```
http://localhost:8080/client/mail
```

```
http://localhost:8080/client/{id}
```

```
http://localhost:8080/client/profile
```

```
http://localhost:8080/prestataires/mail
```

```
http://localhost:8080/prestataires/{id}
```

```
http://localhost:8080/prestataires/profile
```

```
http://localhost:8080/event/{idEvent}/prestataire/{idp}
```

```
http://localhost:8080/prestataires/event/{idp}/{idEvent}
```

//Testons les End-Points

Création de Prestataire

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** localhost:8080/prestataires
- Body (raw):**

```
2 ..... "nom": "Aminata",
3 ..... "prenom": "Diallo",
4 ..... "service": "Restauratrice",
5 ..... "password": "password",
6 ..... "mail": "ami@gmail.com",
7 ..... "photo": null,
```
- Status:** 200 OK
- Time:** 263 ms
- Size:** 650 B
- Body (pretty):**

```
1
2   "idp": 4,
3   "nom": "Aminata",
4   "prenom": "Diallo",
5   "service": "Restauratrice",
6   "password": "$2a$10$NDBcLruwB4SRuJC7BAxjz1t1iPn00VXvQHdKRwPzYLMenzqy3sFFy",
7   "mail": "ami@gmail.com",
```

Creation de Client

POST localhost:8080/client

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies Beautify

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

```
3 {
4   "nom": "GAYE",
5   "prenom": "Abdoulaye",
6   "password": "$2a$10$AkBZ8jP0ok1108gJ1zIfj.RUFaqPRX3M/FwdJRCr2T.eEuwwWwxou",
7   "mail": "gayeadbdoulaye163@gmail.com",
8   "photo": null
9 }
```

Body Cookies Headers (14) Test Results Status: 200 OK Time: 255 ms Size: 590 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "idc": 3,
3   "nom": "GAYE",
4   "prenom": "Abdoulaye",
5   "password": "$2a$10$JY/E79MY9FS/y1U3mXCTnuniMPJk9XRS38ByIz6X/ARQk0r3XPEW",
6   "mail": "gayeadbdoulaye163@gmail.com",
7   "photo": null
8 }
```

Confirmation de l'ajout avec la methode get sur le meme end-point

GET localhost:8080/client

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Body Cookies Headers (14) Test Results Status: 200 OK Time: 11 ms Size: 919 B Save as example

Pretty Raw Preview Visualize JSON

```
11 {
12   "idc": 2,
13   "nom": "NDIAYE",
14   "prenom": "Mouhamadou",
15   "password": "$2a$10$17PgMi2c9aFjc.1/EiJnWeHT0Q0AMmVF9x8B1K7GQp8t8.nf4bkCK",
16   "mail": "smah.n@univ.zig.sn",
17   "photo": null
18 },
19 {
20   "idc": 3,
21   "nom": "GAYE",
22   "prenom": "Abdoulaye",
23   "password": "$2a$10$JY/E79MY9FS/y1U3mXCTnuniMPJk9XRS38ByIz6X/ARQk0r3XPEW",
24   "mail": "gayeadbdoulaye163@gmail.com",
25   "photo": null
26 }
```

Remarquons qu'après la création des deux instances de Client et Prestataire, on a l'apparition de deux instances de User

```
MariaDB [eventdb]> SELECT * FROM user;
```

id	mail	password	photo	role
1	oriontheroot@gmail.com	\$2a\$10\$Iy2WANR6E5j5WEH9hS1mL.Wvxox12iytZyjq1l/d3MqkX8nGHncy	NULL	client
2	smah.n@univ.zig.sn	\$2a\$10\$5bq/K1YUgjYr.cVbCnVx0CuaqJA11R3BQRN24F3G0a6xaydZ4HhJcK	NULL	client
3	diop@gmail.com	\$2a\$10\$5qaiRmWkMaqAI8onYHN04VuefLNqpap3ox9q9cReJzTcfr58pRKdYq	NULL	prestataire
4	keba@gmail.com	\$2a\$10\$1vt.Fyc1WHx0040RZPGzCe5QZ5HCQNJ14/dy1Qf0xAQyCgYlabFCy	NULL	prestataire
5	nagato@gmail.com	\$2a\$10\$X0kz.KMCgAQVgP6sC10IX.tox3Ne.5ECaF9vqDmKK0VdLvt4bhXTm	NULL	prestataire
6	gayeadbdoulaye163@gmail.com	\$2a\$10\$0zXuEioTr998UDMWhCFw0.ujrdW73q.ax3IuV0ncmoTnHpiKKWxK	NULL	client
8	ami@gmail.com	\$2a\$10\$5siFlmhXjNOT.ER6KYHBXKe0JwAIi.9hyiD54KI1pZ.IRinPcybf2	NULL	prestataire

```
7 rows in set (0.000 sec)
```

```
MariaDB [eventdb]>
```

Retrouver un Client avec son Mail

```
(base) tinkin-djeeri@tinkin:~$ http :8080/client/mail?mail=oriontheroot@gmail.com
HTTP/1.1 200
```

```
{
  "idc": 1,
  "mail": "oriontheroot@gmail.com",
  "nom": "NDIAYE",
  "password": "$2a$10$R/XSHMJ9VC1s0cVKNJTsceRW/ZJ.15JcUvOX6sXgF5ig0d2DYGyd2",
  "photo": null,
  "prenom": "Al Hamine"
}
```

Maintenant attaquons la Creation de Evenement

POSTlocalhost:8080/event

Send

ParamsAuthorizationHeaders (8)BodyPre-request ScriptTestsSettings

Cookies

noneform-datax-www-form-urlencodedrawbinaryGraphQLJSON

Beautify

```
1 { "nomEvent": "Priere de nuit",
2   "date": "2023-11-10T16:02:02.623+00:00",
3   "description": "S'eloigner de la vie ephemere",
4   "lieu": "Ziguichor",
5   "organisateur": { "idc": 3 }
6 }
```

BodyCookiesHeaders (14)Test Results

Status: 200 OKTime: 19 msSize: 658 BSave as example

PrettyRawPreviewVisualizeJSON

```
1 {
2   "idEvent": 6,
3   "nomEvent": "Priere de nuit",
4   "date": "2023-11-10T16:02:02.623+00:00",
5   "description": "S'eloigner de la vie ephemere",
6   "lieu": "Ziguichor",
7   "organisateur": {
8     "idc": 3,
```

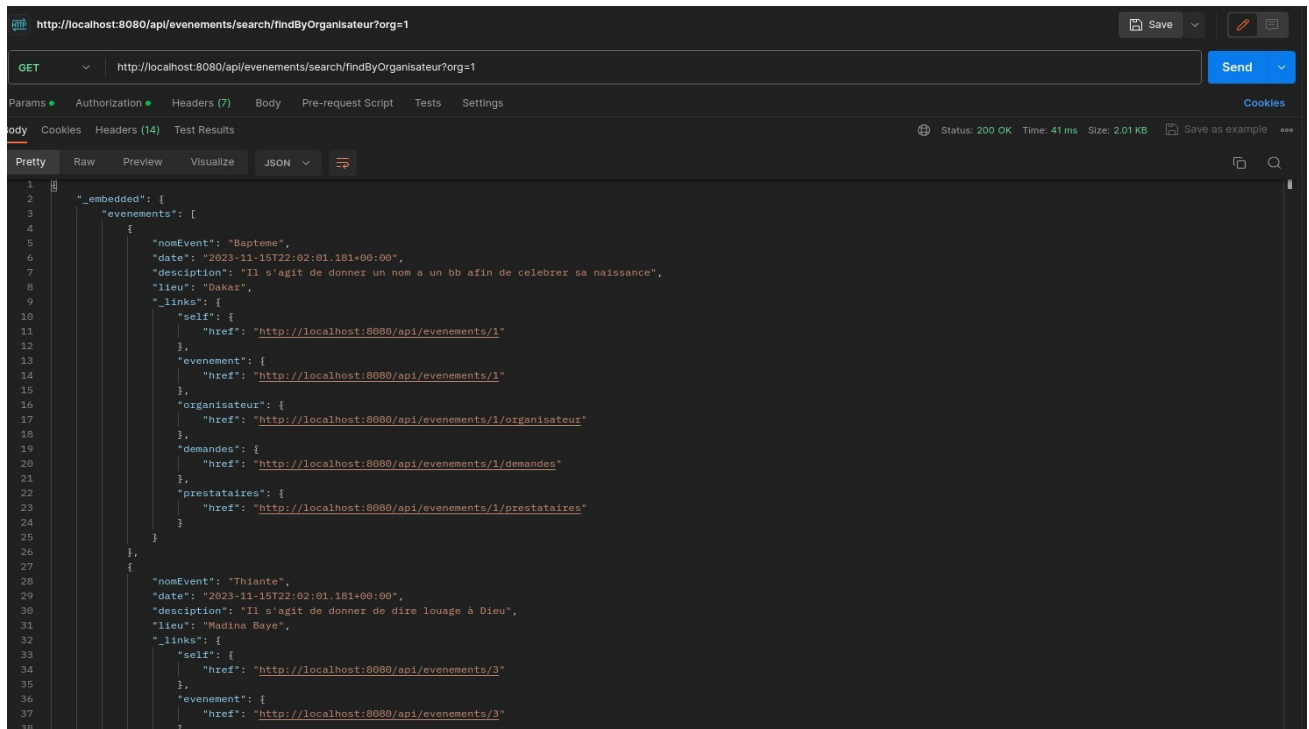
Evenement ajoute avec succes

```
5:
  nomEvent: "Priere de nuit"
  date: "2023-11-10T16:02:02.623+00:00"
  description: "S'eloigner de la vie ephemere"
  lieu: "Ziguichor"
  _links:
    self: "http://localhost:8080/api/evenements/6"
    evenement: "http://localhost:8080/api/evenements/6"
    prestataires:
      href: "http://localhost:8080/api/evenements/6/prestataires"
    organisateur:
      href: "http://localhost:8080/api/evenements/6/organisateur"
```

Quels sont les evenement crees par un Client donne

```
@Query("SELECT e FROM Evenement e WHERE e.organisateur.id = :org")
```

```
List<Evenement> findByOrganisateur(@Param("org") Long id);
```



Pour eviter la sérialisation continuelle des objets Client, Prestataire, et Evenement entrainant une inclusion infinie. On n'a introduit

```
@JsonIgnoreProperties({"hibernateLazyInitializer", "handler"})
```

Avant de déclarer la classe `Owner` dans le fichier `Prestataire.java`

Et avant de déclarer la relation (1-*) on met dans `Client` et `Evenement`

```
@JsonIgnore
```

Quand un evenement est crée la liste des prestataire est vide.
Donc si le Client envoi une demande (contrat) a un prestataire, et que
Ce prestataire accepte la demande la variable eventActuel qui est dans
Prestataire sera mis a jour automatiquement et en meme tant le prestataire
Ajoute a la liste des prestataires de evenement.

La mise a jour est tres simple mais impossible a comprendre sans le code

```
@PutMapping("/event/{id}/{idE}")
public ResponseEntity<Prestataire> updatePrestataireE(@PathVariable Long id, @PathVariable Long idE,
    @RequestBody Prestataire prestataire) {

    // Vérifiez si le prestataire existe
    Prestataire prestataireExistant = prestataireRepository.findById(id).orElseThrow();

    // Récupérez l'événement existant de la base de données
    Evenement evenement = eService.getEvenementById(idE);

    // Définissez l'événement sur la propriété evenement du prestataire
    // prestataireExistant.setEvenement(evenement);
    prestataireExistant.ajoutEvenement(evenement);
    evenement.ajouterPrestataire(prestataireExistant);

    // Enregistrez le prestataire
    prestataireRepository.save(prestataireExistant);

    return ResponseEntity.ok(prestataireExistant);
}
```

Du coup tout ce qu'on aura besoin pour ce sont les id du prestataire et
De l'événement.

Ce qui suit est une démonstration

The screenshot shows a REST client interface with a PUT request to `http://localhost:8080/prestataires/event/1/1`. The request body is in JSON format and contains the following data:

```
{
  "idp": 1,
  "nom": "DIOPI",
  "prenom": "May",
  "service": "Restauration",
  "password": "$2a$10$hwQ5x1zsmkUhlEGDTd2Ze.kIFOT3ITmpumjX8K5fOM1QFhQ.7zxBi",
  "mail": "diop@gmail.com",
  "photo": null,
  "note": null,
  "evenement": [
    {
      "idEvent": 2,
      "nomEvent": "Fineraille",
      "date": "2023-11-17T19:17:23.745+00:00",
      "description": "Il s'agit d'inhumer un corps humain afin de lui rendre à Dieu",
      "lieu": "Voif",
      "organisateur": {
        "idc": 2,
        "nom": "NDIAYE",
        "prenom": "Mouhamadou",
        "password": "$2a$10$cXgT0JH9HQC.yz93gcJxiudtz1VipSxU31g8nD5LsIKKCNh6qFUee",
        "mail": "smah.n@univ.zig.sn",
        "photo": null
      }
    },
    {
      "idEvent": 1,
      "nomEvent": "Baptême",
      "date": "2023-11-17T19:17:23.745+00:00",
      "description": "Il s'agit de donner un nom à un bb afin de célébrer sa naissance"
    }
  ]
}
```

Ainsi même avec un body vide {} on peut effectuer le travail

Maintenant le prestataire qui id=1 doit prester dans 2 evenements

```
localhost:8080/api/prestataires/1/evenement

JSON Raw Data Headers
Save Copy Collapse All Expand All Filter JSON

{
  "_embedded": {
    "evenements": [
      {
        "0": {
          "nomEvent": "Fineraille",
          "date": "2023-11-17T19:17:23.745+00:00",
          "description": "Il s'agit d'inhumer un corps humain afin de lui rendre à Dieu",
          "lieu": "Yoff",
          "_links": {
            "self": {
              "href": "http://localhost:8080/api/evenements/2"
            },
            "evenement": {
              "href": "http://localhost:8080/api/evenements/2"
            },
            "demandes": {
              "href": "http://localhost:8080/api/evenements/2/demandes"
            },
            "prestataires": {
              "href": "http://localhost:8080/api/evenements/2/prestataires"
            },
            "organisateur": {
              "href": "http://localhost:8080/api/evenements/2/organisateur"
            }
          }
        },
        "1": {
          "nomEvent": "Bapteme",
          "date": "2023-11-17T19:17:23.745+00:00",
          "description": "Il s'agit de donner un nom a un bb afin de celebrer sa naissance",
          "lieu": "Dakar",
          "_links": {}
        }
      ]
    }
  }
}
```

Et cote evenement aussi on peut voir la liste des prestatatires
Exemple evenement idEvent=1

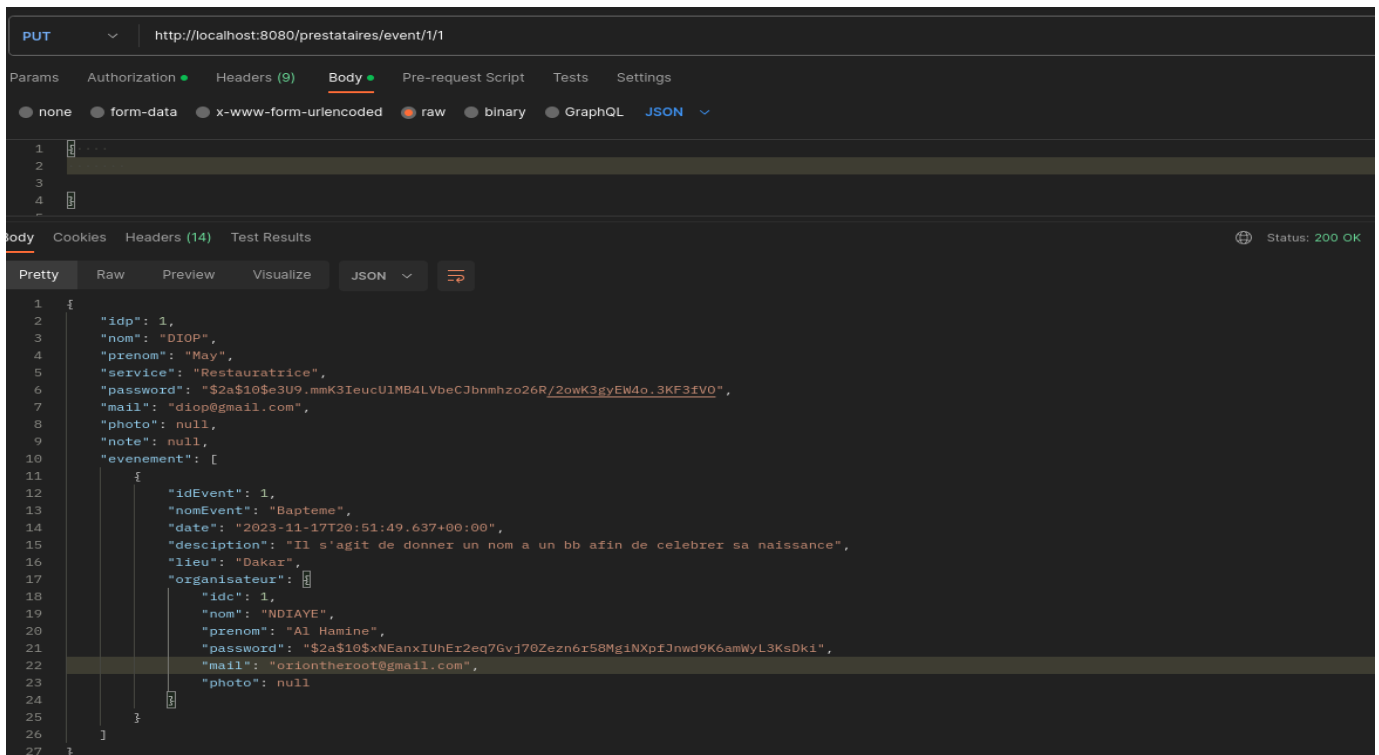
```
localhost:8080/api/evenements/1/prestataires

JSON Raw Data Headers
Save Copy Collapse All Expand All Filter JSON

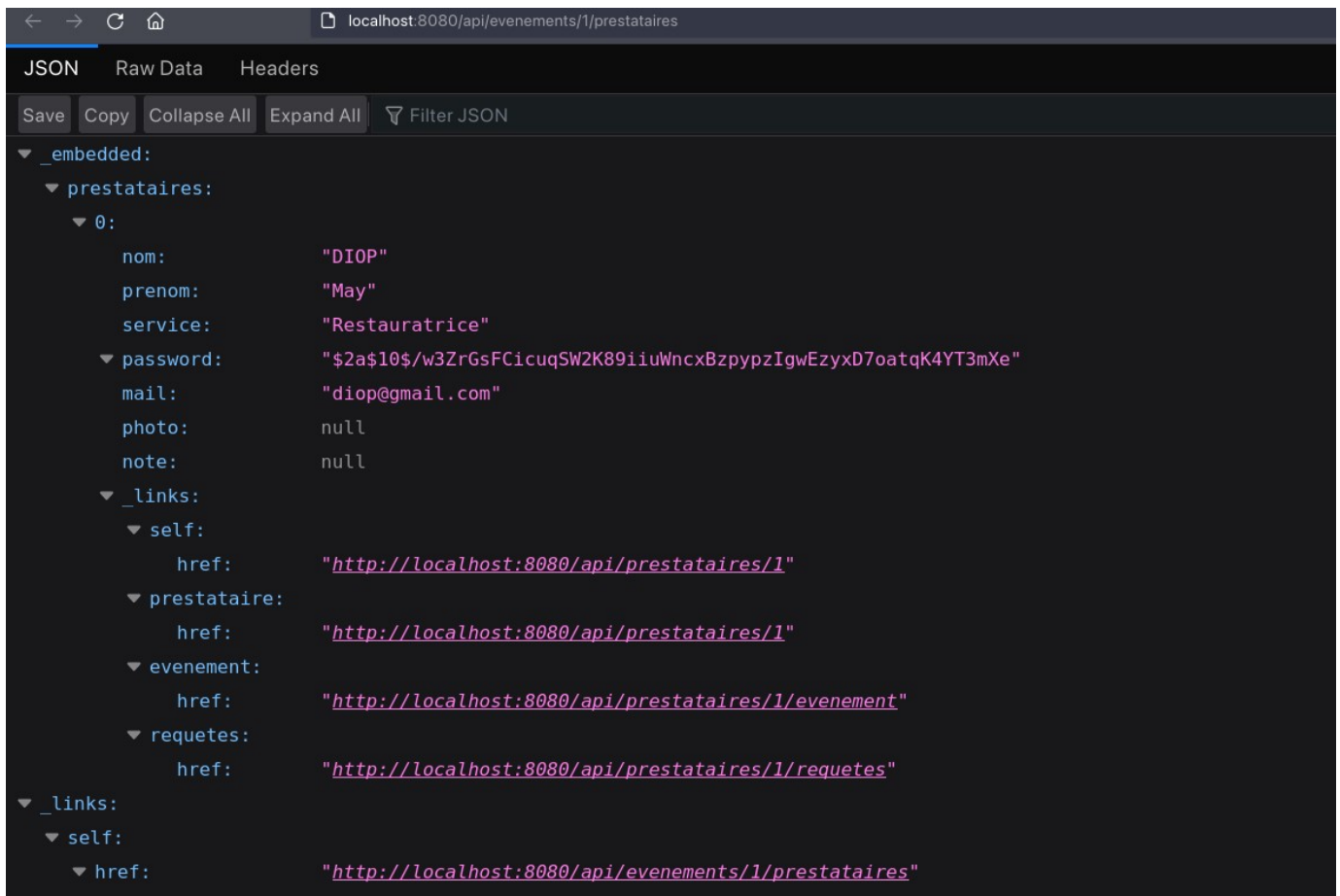
{
  "_embedded": {
    "prestataires": [
      {
        "0": {
          "nom": "DIOP",
          "prenom": "May",
          "service": "Restauratrice",
          "password": "$2a$10$Hw.Iao8pxto/j0qzsbB0lu01aurjUnMWNh1a989S9EtM6xcwPZAqu",
          "mail": "diop@gmail.com",
          "photo": null,
          "note": null,
          "_links": {
            "self": {
              "href": "http://localhost:8080/api/prestataires/1"
            },
            "prestataire": {
              "href": "http://localhost:8080/api/prestataires/1"
            },
            "requetes": {
              "href": "http://localhost:8080/api/prestataires/1/requetes"
            },
            "evenement": {
              "href": "http://localhost:8080/api/prestataires/1/evenement"
            }
          }
        },
        "1": {
          "nom": "SQUARE",
          "prenom": "KEBA",
          "service": "Receptrice",
          "password": "$2a$10$DEPYw3RNOBcMC3F7bDdpNuyxnKurLy28uvoAN9xE0Wnh5itnzmS82",
          "mail": "keba@gmail.com",
          "photo": null,
          "note": null
        }
      ]
    }
  }
}
```

Test la suppression de prestataires sur un Evenement.

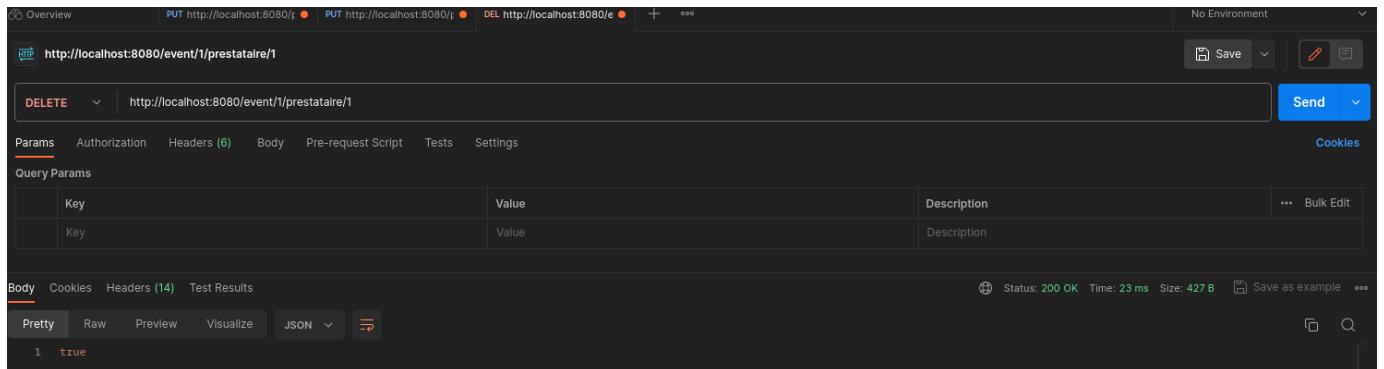
Pour ce faire on ajoute d'abord prestataire id=1 dans evenement idEvent=1



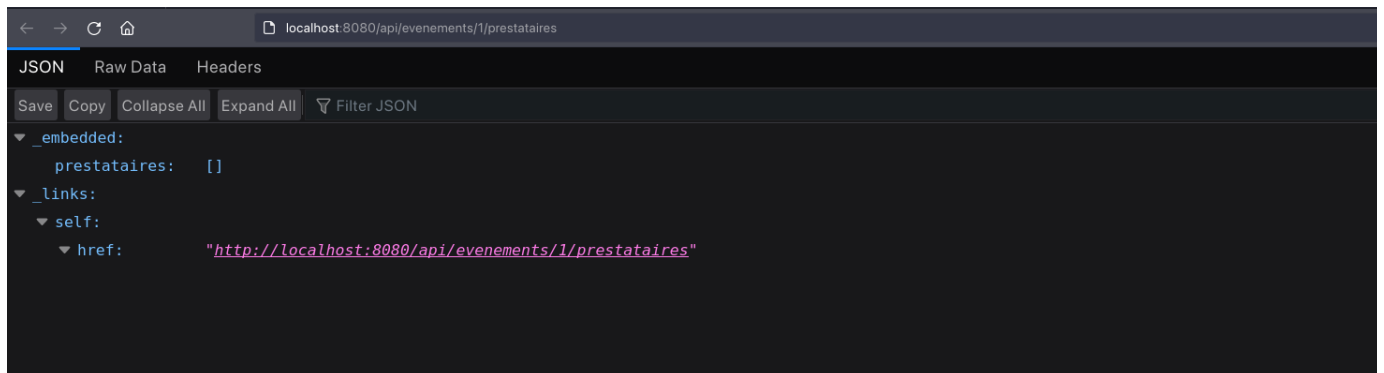
On voit nettement que l'ajout a reussit



Maintenant eliminons le. Si tout va bien la methode renvoie true

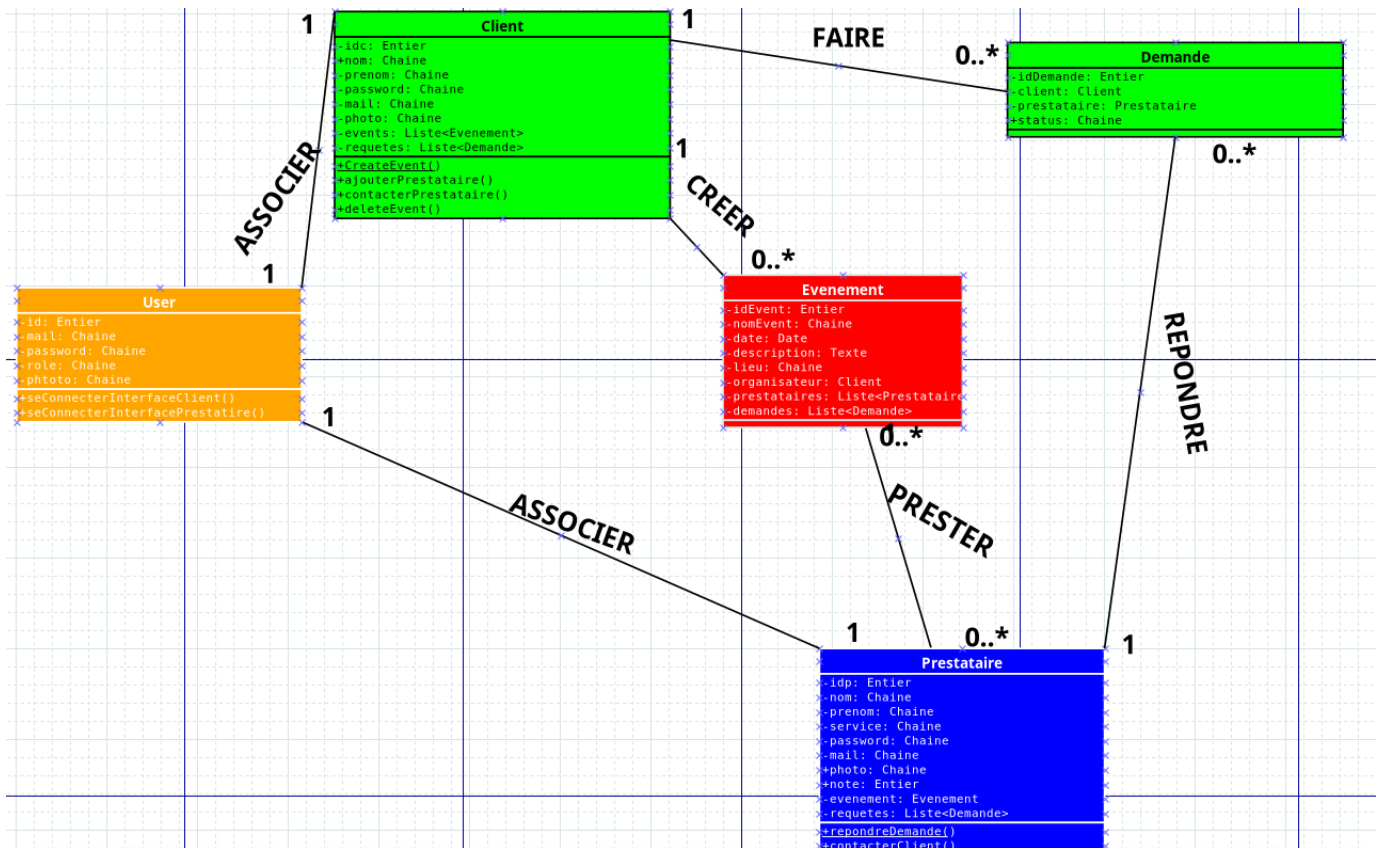


Si on regarde de nouveau la liste des prestataires dans evenement idEvent=1 On a :



Fin des Tests

Diagramme de Classe



Auteurs



Seydina Mouhamadou Al Hamine NDIAYE



Abdoulaye GAYE



Fatoumata Bah



Abdourahamane DIALLO 202002087



Abdourahamane DIALLO 202002087