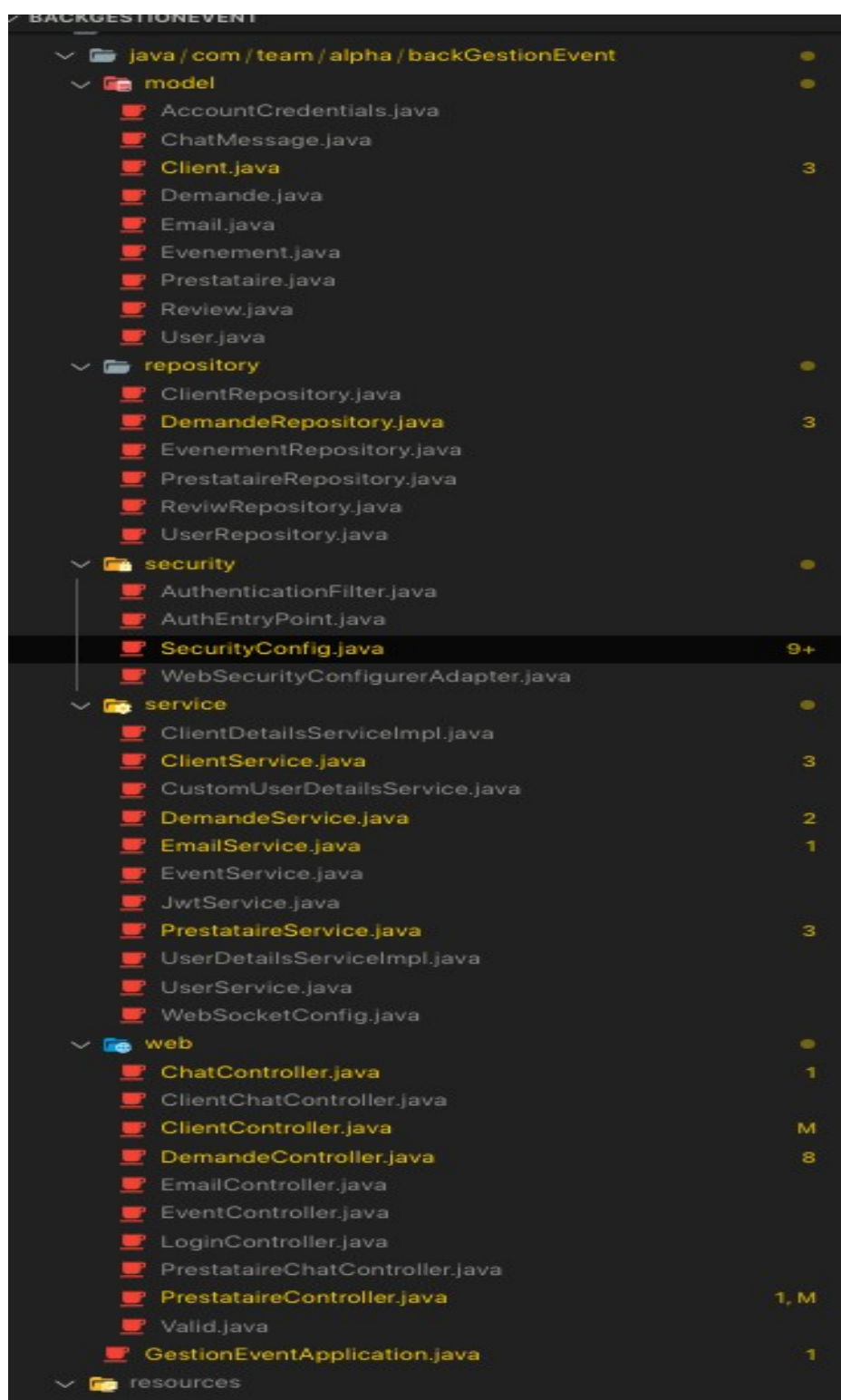


RAPPORT DU PROJET GESTION EVENEMENT

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:****

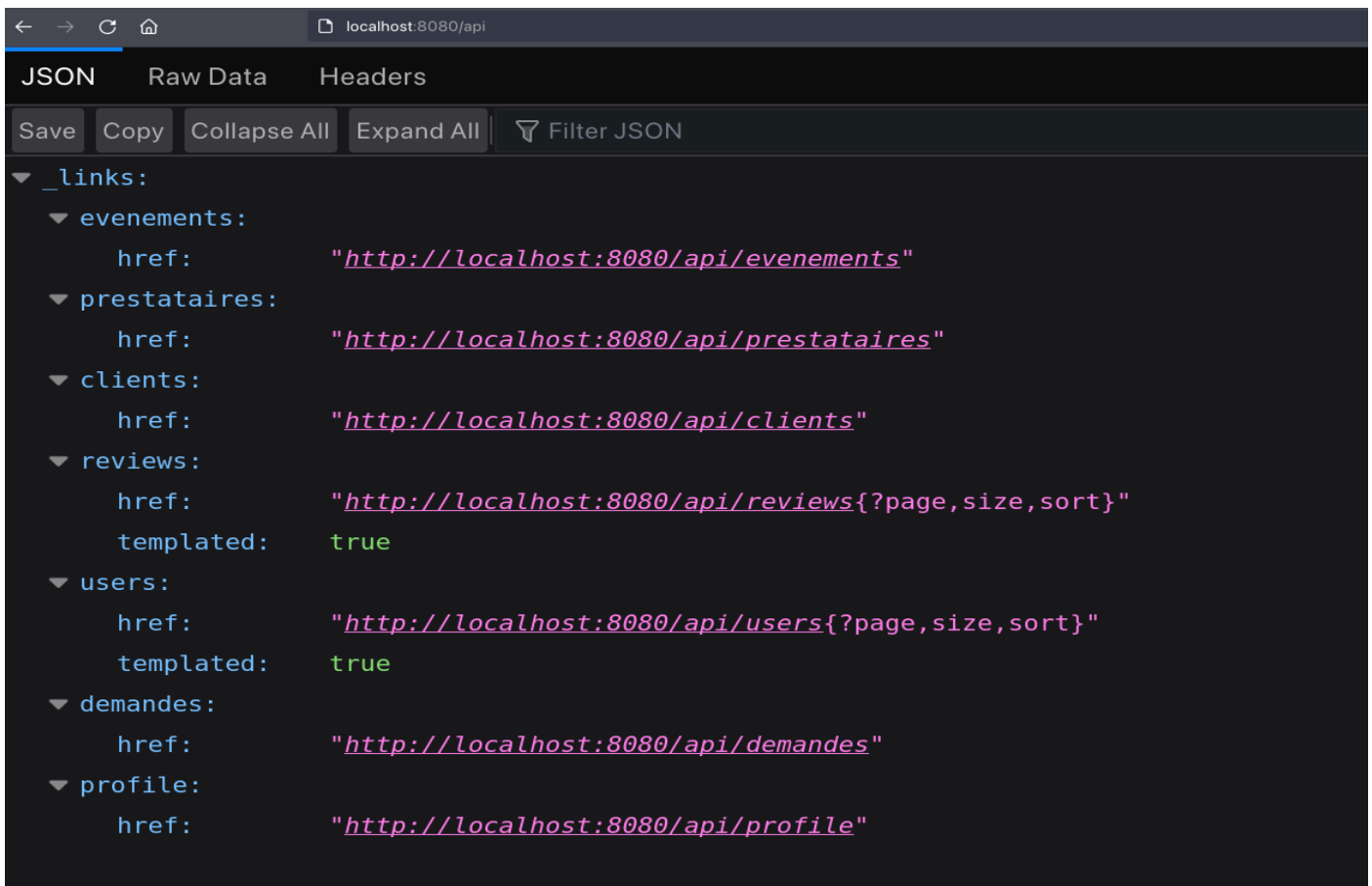
Simplifie l'intégration des e-mails dans les applications Spring Boot en fournissant des paramètres préconfigurés pour l'envoi d'e-mails.

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' tab. The JSON response is a dictionary with a '_links' key. This key points to another dictionary containing several links, each with a 'href' and a 'templated' status. The links are for 'evenements', 'prestataires', 'clients', 'reviews', 'users', 'demandes', and 'profile'. The 'reviews' and 'users' links are marked as 'templated: true'.

```
{
  "_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}"
      "templated": true
    },
    "users": {
      "href": "http://localhost:8080/api/users{?page,size,sort}"
      "templated": true
    },
    "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
```

//Testons les End-Points

Création de Prestataire

The screenshot shows a REST client interface with a POST request to `localhost:8080/prestataires`. The request body is a JSON object with the following fields: `nom` (Aminata), `prenom` (Diallo), `service` (Restauratrice), `password` (password), `mail` (ami@gmail.com), and `photo` (null). The response status is 200 OK, and the response body is a JSON object with the following fields: `idp` (4), `nom` (Aminata), `prenom` (Diallo), `service` (Restauratrice), `password` (\$2a\$10\$NDBcLruwB4SRuJC7BAxjruti1Pn00VXvQHdKRwPzYLMenzqy3sFFy), and `mail` (ami@gmail.com).

```
POST localhost:8080/prestataires
```

```
{  
  "nom": "Aminata",  
  "prenom": "Diallo",  
  "service": "Restauratrice",  
  "password": "password",  
  "mail": "ami@gmail.com",  
  "photo": null,  
}
```

```
{  
  "idp": 4,  
  "nom": "Aminata",  
  "prenom": "Diallo",  
  "service": "Restauratrice",  
  "password": "$2a$10$NDBcLruwB4SRuJC7BAxjruti1Pn00VXvQHdKRwPzYLMenzqy3sFFy",  
  "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 ..... "nom": "GAYE",
4 ..... "prenom": "Abdoulaye",
5 ..... "password": "$2a$10$AkBZ8jP0ok1108gJ1zIfj.RUFaQPRX3M/FwdJRCr2T.eEuwwWwxou",
6 ..... "mail": "gayeadbdoulaye163@gmail.com",
7 ..... "photo": null
```

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
```

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 "idc": 2,
12 "nom": "NDIAYE",
13 "prenom": "Mouhamadou",
14 "password": "$2a$10$17PgMi2c9aFjc.1/EiJnWeHT0QOAMmVF9x8B1K7GQp8t8.nf4bkCK",
15 "mail": "smah.n@univ.zig.sn",
16 "photo": null
17 },
18 {
19 "idc": 3,
20 "nom": "GAYE",
21 "prenom": "Abdoulaye",
22 "password": "$2a$10$JY/E79MY9FS/y1U3mXCTnuniMPJk9XRS38ByIz6X/ARQk0r3XPEW",
23 "mail": "gayeadbdoulaye163@gmail.com",
24 "photo": null
25 }
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$5qaiRmWkMaqAI8onYHN04VuefLNqpap3ox9q9cReJzTcf58pRKdYq | 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 | gayeaddoulaye163@gmail.com | $2a$10$0zXuEioTr998UDMWhCFw0.ujrdW73q.ax3IuV0ncmoTnHpiKKWxK | NULL | client |
8 | ami@gmail.com | $2a$10$5siFmhXjNOT.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

POST

localhost:8080/event

Send

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Cookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

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}

```

Body

Cookies

Headers (14)

Test Results

Status: 200 OK

Time: 19 ms

Size: 658 B

Save as example

Pretty

Raw

Preview

Visualize

JSON

```

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

```

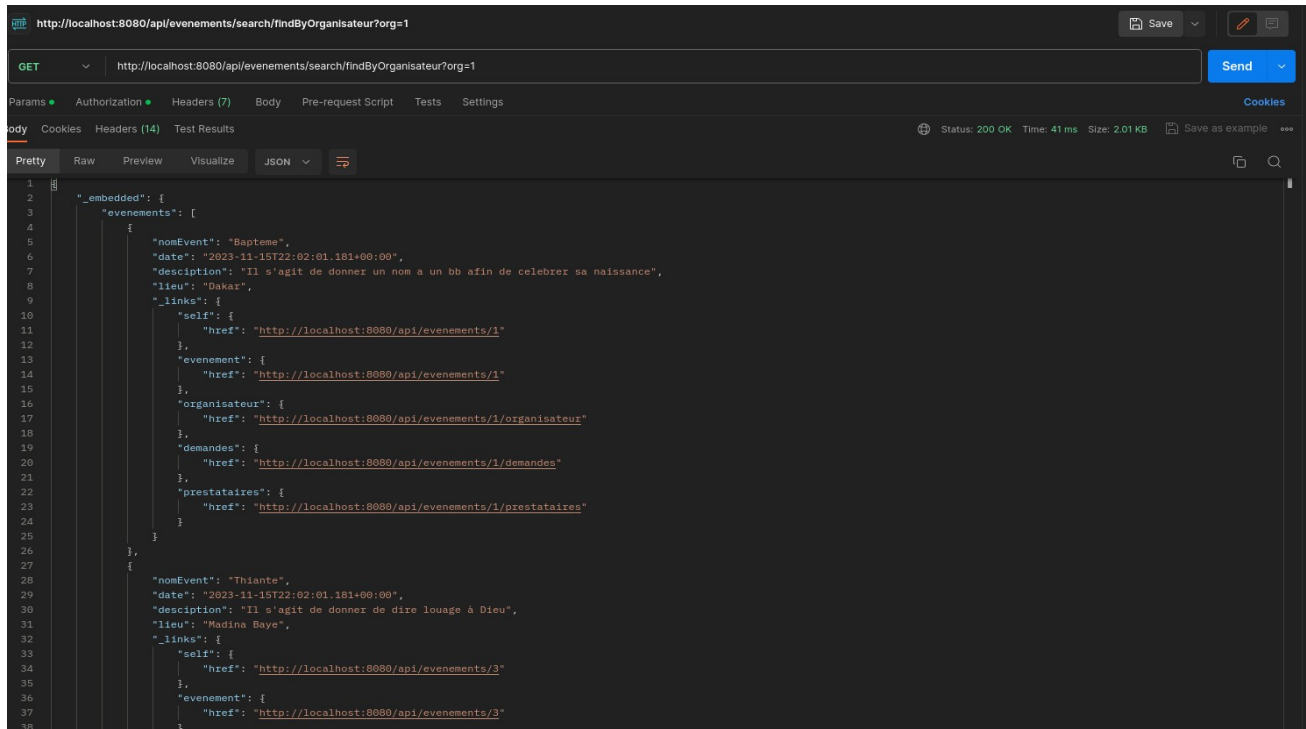
5:
  nomEvent: "Priere de nuit"
  date: "2023-11-10T16:02:02.623+00:00"
  description: "S'eloigner de la vie ephemere"
  lieu: "Ziguichor"
  _links:
    self: "http://localhost:8080/api/evenements/6"
    evenement: "http://localhost:8080/api/evenements/6"
    prestataires: "http://localhost:8080/api/evenements/6/prestataires"
    organisateur: "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 éviter la sérialisation continuelle des objets Client, Prestataire, et Evenement entraînant une inclusion infinie. On n'a introduit

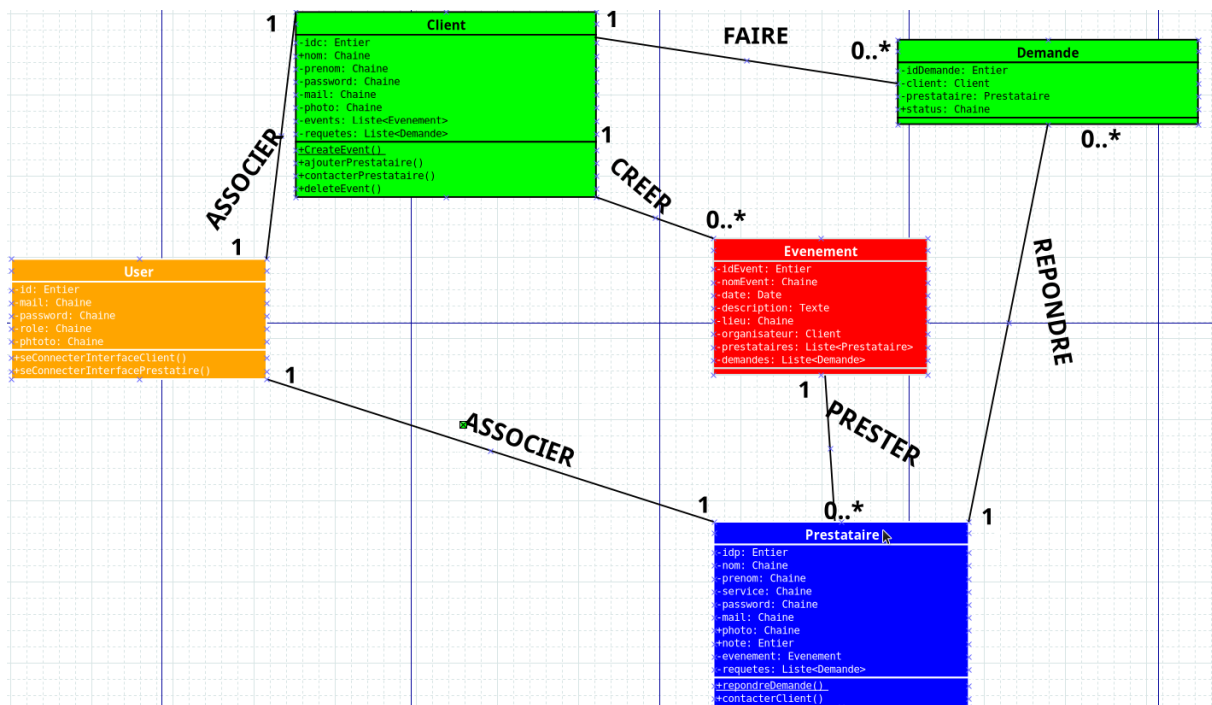
```
@JsonIgnoreProperties({"hibernateLazyInitializer", "handler"})
```

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
```

Diagramme de Classe



Auteurs



Seydina Mouhamadou Al Hamine NDIAYE

Seydina Mouhamadou Al Hamine NDIAYE



Abdoulaye GAYE



Abdourahamane DIALLO 202002087



Abdourahamane DIALLO 202000272

Fatoumata Bah

