Rapport TP Techno Web 2022/2023

Binôme: GUEYE Serigne Saliou & DIALLO Ismael

Table of contents

I. Démarrage de l'application

Cette application utilise le framework Spring Boot avec maven comme environnemnt d'exécution/compilation.Vous pouvez lancer l'application automatiquement sans ligne de commande si vous utilisez un éditeur de texte comme `IntelliJ` .Sinon, vous pouvez utiliser les lignes de commandes.

- Pour lancer l'application :

```
$ ./mvnw spring-boot:run
```

- Pour lancer les tests

```
$ ./mvnw test
```

- Pour générer les couvertures de test :

```
$ ./mvnw surefire-report:report
```

Un rapport au format HTML est normalement généré dans `${basedir}/target/site/surefire-report.html` .

NB: Pour un meilleur visuel du rapport de couverture des tests, vous pouvez lancer le fichier `index` dans le dossier `test/testCoverage` .
Attention: Pour une raison de temps, on n'a pas pu faire tous les tests.

Current scope: all classes

Overall Coverage Summary

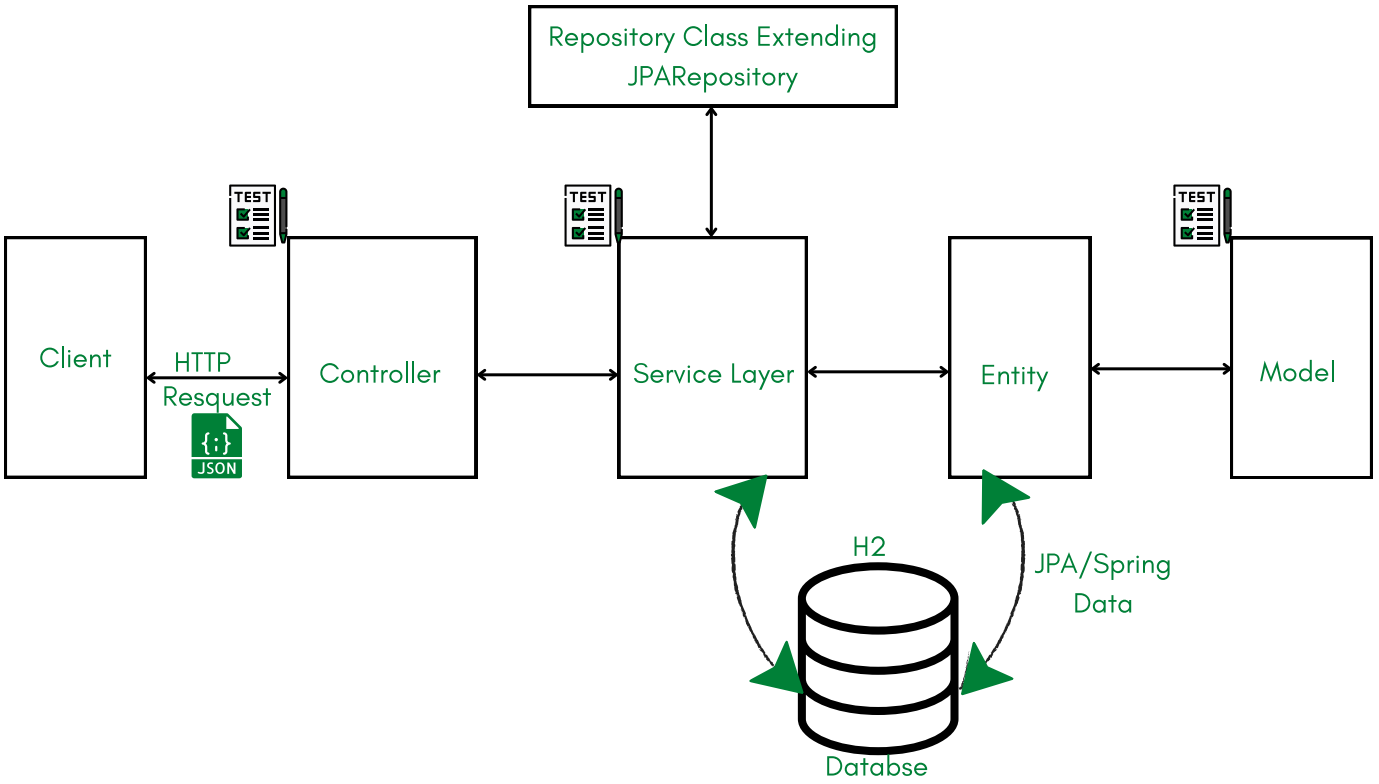
Package	Class, %	Method, %	Line, %
all classes	66,7% (10/15)	51,5% (35/68)	49,1% (181/369)

Coverage Breakdown

Package	Class, %	Method, %	Line, %
com.uca.series_temporelles.controller	20% (1/5)	13,8% (4/29)	20% (21/105)
com.uca.series_temporelles.model	100% (5/5)	100% (5/5)	100% (33/33)
com.uca.series_temporelles.service	80% (4/5)	76,5% (26/34)	55% (127/231)

generated on 2022-12-03 23:

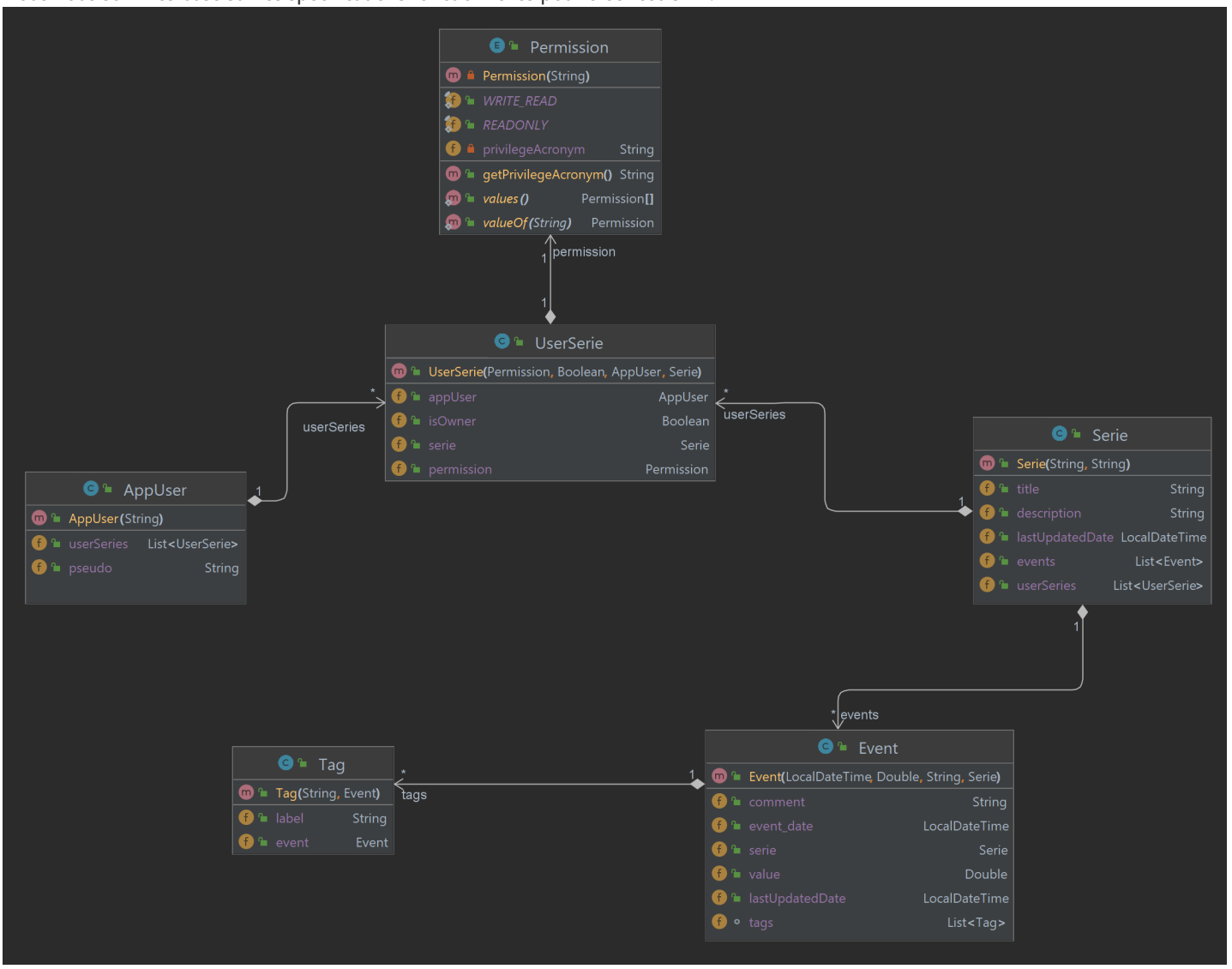
II. Architecture générale



III. UML et Modèle de données

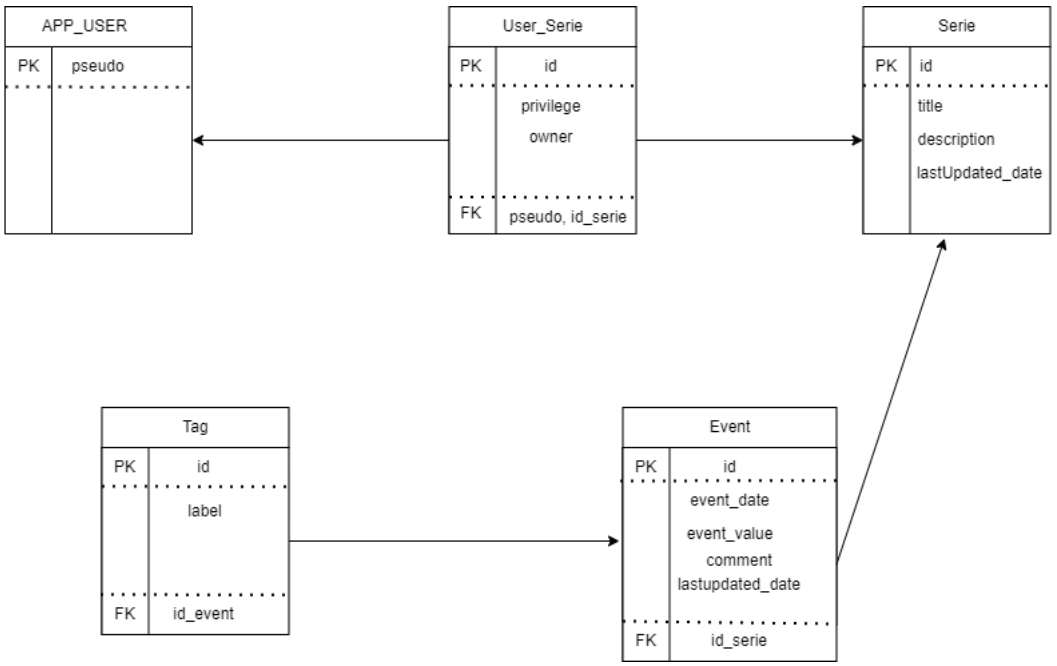
A. UML

Nous nous sommes basé sur les spécifications fonctionnelles pour créer cet UML:



B. Modèle de données

Voici notre modèle de donnée:



Compraison Mapping unidirectionnel `@ManyToOne` et Mapping bidirectionnel `@OneToMany` et `@ManyToOne` :

Chacun des deux, a des avantages et des inconvénients :

- Le *mapping unidirectionnel* avec `@ManyToOne` , peut éviter le problème de performance potentiel d'un `@OneToMany` .
  Mais il ne peut pas naviguer ou cascader les opérations CRUD vers les collections enfants vers la collection de base. Cependant, cela peut être fait manuellement.
- Le *mappage bidirectionnel* avec `@OneToMany` et `@ManyToOne` peut permettre aux deux entités de la relation d'accéder rapidement et de réaliser des opérations CRUD en cascade.
  Cependant, cela peut causer un problème de performance sur une grande collection d'enfants.

Selon ces deux comparaisons, nous avons décidé d'utiliser le *mapping unidirectionnel*, car nous préférons éviter les problèmes de performance. Et pour la cascade, nous allons la faire manuellement.

IV. Gestion des ressources

A. Schémas URL

Pour l'API, nous avons utilisé REST avec les méthodes `GET` , `POST` , `PUT` , et `DELETE` .

1. Pour gérer les utilisateurs (Création, Affichage)

- **Pour créer un utilisateur**

```
POST http://localhost:8080/api/users/add


## Exemple

POST http://localhost:8080/api/users/add
Content-Type: application/json

{
  "pseudo":"ssgueye"
}
```

- **Pour afficher tous les utilisateurs**

```
GET http://localhost:8080/api/users
```

- **Pour afficher un utilisateur (en se basant sur son pseudo)**

```
GET http://localhost:8080/api/users/{{pseudo}}
```

```
## Exemple

GET http://localhost:8080/api/users/ssgueye
```

2. Pour gérer les séries (Création, Affichage, Modification et Suppression)

- **Pour créer une série**

```
POST http://localhost:8080/api/series/add/{{pseudo}}


## Exemple

POST http://localhost:8080/api/series/add/ssgueye
Content-Type: application/json

{
  "title": "Météo Clermont-Ferrand",
  "description": "Visualisez toute la météo de Clermont-Ferrand"
}
```

- **Pour afficher toutes les séries (créées et partagées) d'un utilisateur**

```
GET http://localhost:8080/api/series/{{pseudo}}


##Exemple: Afficher toutes les séries créées par "ssgueye" et partagées à "ssgueye"

GET http://localhost:8080/api/series/ssgueye
```

- **Pour afficher toutes les séries créées par un utilisateur**

```
GET http://localhost:8080/api/series/OwnSeries/{{pseudo}}


## Exemple: Afficher toutes les séries créées par "ssgueye"

GET http://localhost:8080/api/series/OwnSeries/ssgueye
```

- **Pour afficher toutes les séries qui nous ont été partagées**

```
GET http://localhost:8080/api/series/sharedSeries/{{pseudo}}


## Exemple: Afficher toutes les séries partagées à "ssgueye"

GET http://localhost:8080/api/series/sharedSeries/ssgueye
```

- **Pour modifier une série**

```
PUT http://localhost:8080/api/series/update/{{serie_id}}/{{pseudo}}


## Exemple: modifier la série avec l'id 1 de "ssgueye"

PUT http://localhost:8080/api/series/update/1/ssgueye
Content-Type: application/json

{
  "title": "Météo Clermont-Ferrand Semaine 46",
  "description": "Visualisez toute la météo de Clermont-Ferrand"
}
```

- **Pour supprimer une série**

```
DELETE http://localhost:8080/api/series/delete/{{serie_id}}/{{pseudo}}


## Exemple: supprimer la série avec l'id 1 de "ssgueye"

DELETE http://localhost:8080/api/series/delete/1/ssgueye
```

3. Pour gérer les séries et les permissions (Partage & Affichage)

- **Pour partager une série**

```
POST http://localhost:8080/api/user_series/share/{{serie_id}}/{{pseudoOwner}}/{{pseudoReceiver}}?permission={{permission}}

## Exemple: Le user "ssgueye" partage la série 1 à "dapieu" en mode "READONLY"

POST http://localhost:8080/api/user_series/share/1/ssgueye/dapieu?permission=READONLY
```

Précision 1: Si l'utilisateur qui partage la série n'est pas le propriétaire de cette série, il y aura une erreur 500 qui précise que l'utilisateur n'a pas la permission de partager cette série.

Précision 2: Si l'utilisateur partage la série à lui même, on lui retourne la série sans rien modifier.

Précision 3: Si l'utilisateur partage deux fois la même série à un utilisateur, on vérifie s'il a changé le mode partage ou non. Si oui on met à jour le mode de partage. Sinon on garde l'ancien mode.

- **Pour afficher une série et sa permission**

```
GET http://localhost:8080/api/user_series/{{pseudo}}/{{serie_id}}

## Exemple: Afficher la série avec id "1" de "ssgueye" et sa permission.

GET http://localhost:8080/api/user_series/ssgueye/1
```

- **Pour afficher toutes les séries et leur permission**

```
GET http://localhost:8080/api/user_series/{{pseudo}}

## Exemple: Afficher toutes les séries de "ssgueye" et leur permission

GET http://localhost:8080/api/user_series/ssgueye
```

4. Pour gérer les events (Création, Affichage, Modification et Suppression)

- P**our ajouter un event dans une série**

```
POST http://localhost:8080/api/events/add?pseudo={{pseudo}}&serie_id={{serie_id}}

## Exemple: créer et Ajouter un event dans la série "1" de "ssgueye"

POST http://localhost:8080/api/events/add?pseudo=ssgueye&serie_id=1
Content-Type: application/json

{
  "event_date": "2022-11-27T15:00",
  "value": 12,
  "comment": "Du vent frais"
}
```

Précision : Le format de la date est: `yyyy-mm-ddTHH:mm:ss`

- **Pour afficher tous les events d'une série**

```
GET http://localhost:8080/api/events/all?pseudo={{pseudo}}&serie_id={{serie_id}}

## Exemple: Afficher tous les events de la série "1"

GET http://localhost:8080/api/events/all?pseudo=ssgueye&serie_id=1
```

- **Pour afficher un event d'une série**

```
GET http://localhost:8080/api/events/one?pseudo={{pseudo}}&serie_id={{serie_id}}&event_id={{event_id}}

## Exemple: Afficher l'event "1" de la serie "1" de "ssgueye"

GET http://localhost:8080/api/events/one?pseudo=ssgueye&serie_id=1&event_id=1
```

- **Pour Filtrer les events par tag**

```
GET http://localhost:8080/api/events/filter/tag/{{pseudo}}?label={{label}}

## Exemple: Filtrer les events de l'utilisateur "ssgueye" par Tag

GET http://localhost:8080/api/events/filter/tag/ssgueye?tag=FROID
```

- **Pour calculer la fréquence d'une étiquette dans une fenêtre de temps donnée**

```
GET http://localhost:8080/api/events/frequency/tag/{{pseudo}}?tag={{tag}}&startDate={{startDate}}&endDate={{endDate}}

## Exemple: Fréquence de l'étiquette 'FROID' dans l'intervalle de temps [2022-09-01T20:00;2022-12-01T12:00]

GET http://localhost:8080/api/events/frequency/tag/ssgueye?label=FROID&startDate=2022-09-01T20:00&endDate=2022-12-01T12:00:00
```

- **Pour modifier un event**

```
PUT http://localhost:8080/api/events/update?pseudo={{pseudo}}&serie_id={{serie_id}}&event_id={{event_id}}

## Exemple: Modifier l'event "1" de la série "1" de "ssgueye"

PUT http://localhost:8080/api/events/update?pseudo=ssgueye&serie_id=1&event_id=1
Content-Type: application/json

{
  "event_date": "2022-11-27T15:00",
  "value": 12,
  "comment": "Du vent"
}
```

- **Pour supprimer un event**

```
DELETE http://localhost:8080/api/events/delete?pseudo={{pseudo}}&serie_id={{serie_id}}&event_id={{event_id}}

## Exemple: Supprimer l'event "1" de la série "1" de "ssgueye"

DELETE http://localhost:8080/api/events/delete?pseudo=ssgueye&serie_id=1&event_id=1
```

5. Pour gérer les tags

- **Pour ajouter un tag à un event**

```
POST http://localhost:8080/api/tags/add?pseudo={{pseudo}}&serieId={{serieId}}&eventId={{eventId}}

## Exemple: ajouter le tag à l'event "1" de la série "1" créé par "ssgueye"

POST http://localhost:8080/api/tags/add?pseudo=ssgueye&serieId=1&eventId=1
Content-Type: application/json

{
  "label": "Froid"
}
```

- **Pour lister les tags d'un event**

```
GET http://localhost:8080/api/tags?pseudo={{pseudo}}&serieId={{serieId}}&eventId={{eventId}}


## Exemple: lister tous les tags de l'event "1" de la série "1" créé par "ssgueye"

GET http://localhost:8080/api/tags?pseudo=ssgueye&serieId=1&eventId=1
```

- **Pour modifier un tag**

```
PUT http://localhost:8080/api/tags/update?pseudo={{pseudo}}&serieId={{serieId}}&eventId={{eventId}}&tagId={{tagId}}

## Exemple: modifier le tag "1" de l'event "1" de la série "1" créé par "ssgueye"

PUT http://localhost:8080/api/tags/update?pseudo=ssgueye&serieId=1&eventId=1&tagId=1
Content-Type: application/json
```

```
{
  "label": "Nuageux"
}
```

- **Pour supprimer un tag**

```
DELETE http://localhost:8080/api/tags/delete?pseudo={{pseudo}}&serieId={{serieId}}&eventId={{eventId}}&tagId={{tagId}}


## Exemple: supprimer le tag "1" de l'event "1" de la série "1" créé par "ssgueye"

DELETE http://localhost:8080/api/tags/delete?pseudo=ssgueye&serieId=1&eventId=1&tagId=1
```

B. Négociation de contenu

Pour la négociation de contenu, l'application accepte que le `JSON` et le `XML`.
Dans le `pom.xml`, nous avons ajouté une dépendance pour accepter le `XML`.

```
<dependency>
    <groupId>com.fasterxml.jackson.dataformat</groupId>
    <artifactId>jackson-dataformat-xml</artifactId>
</dependency>
```

Exemple d'une méthode de `GET` et `POST` qui accepte le `XML` :

```
## Ajouter un utilisateur

POST http://localhost:8080/api/users/add
Content-Type: application/xml

<AppUserEntity>
    <pseudo>ssgueye</pseudo>
</AppUserEntity>


## Afficher tous les utilisateurs

GET http://localhost:8080/api/users
Accept:application/xml
```

C. Requêtes conditionnelles

Dans le dossier `config`, se trouve le fichier qui fait la configuration des Etags, nous avons placé les Etags dans les séries, les events et les userSeries.
Un exemple d'utilisation de l'Etag:

```
## Requête pour lister toutes les séries créées par l'utilisateur "ssgueye"

GET http://localhost:8080/api/series/OwnSeries/ssgueye
```

Quand on appelle cette requête pour la première fois, on a cet en-tête de réponse avec le body correspondant :

```
HTTP/1.1 200
ETag: "0c772c0ef852e0d40b3b749d088400276"
Content-Type: application/json
Content-Length: 162
Date: Tue, 29 Nov 2022 21:17:53 GMT
Keep-Alive: timeout=60
Connection: keep-alive

[
  {
    "id_serie": 1,
    "title": "Météo Clermont-Ferrand",
    "description": "Visualisez toute la météo de Clermont-Ferrand",
    "lastUpdatedDate": "2022-11-29T22:16:58.872752"
  }
]
```

Et si on place dans le header la requête conditionnelle `If-Non-Match : "0c772c0ef852e0d40b3b749d088400276"` comme ceci :

```
## Requête pour lister toutes les séries créées par l'utilisateur "ssgueye"
```

```
GET http://localhost:8080/api/series/OwnSeries/ssgueye
If-None-Match: "0c772c0ef852e0d40b3b749d088400276"
```

On aura un code de réponse de type `304 Not Modified`

```
HTTP/1.1 304
ETag: "0c772c0ef852e0d40b3b749d088400276"
Date: Tue, 29 Nov 2022 21:56:14 GMT
Keep-Alive: timeout=60
Connection: keep-alive

<Response body is empty>
```

Cela signifie que la ressource n'a pas été modifié. Et Si elle est modifiée, un autre Etag va être généré.