

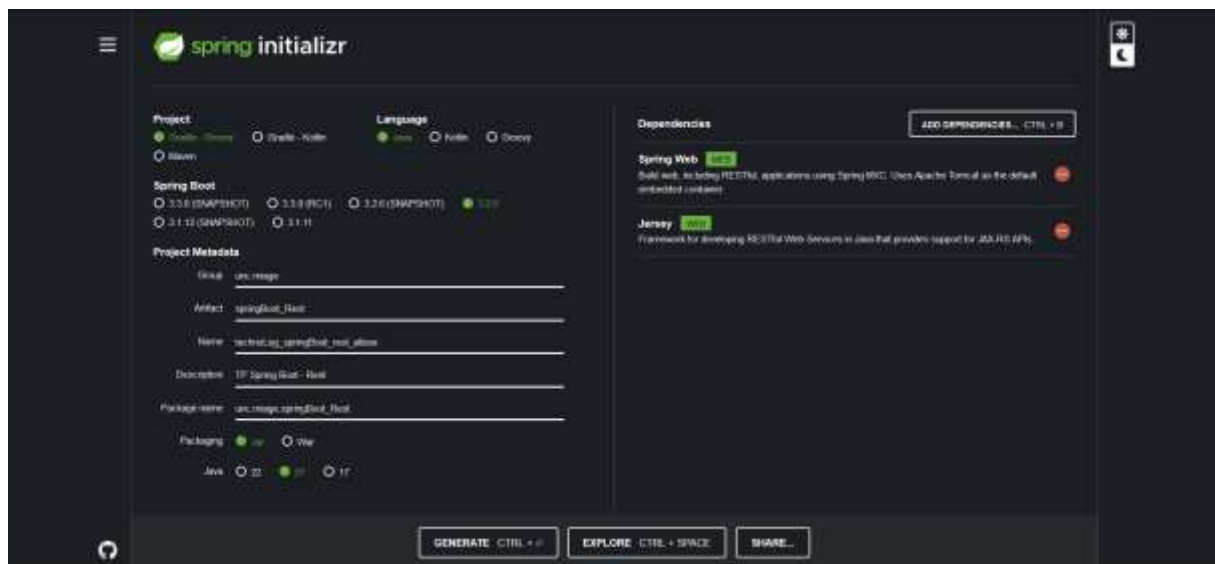
Spring Boot – Rest - 1

Liste d'exécution des fichiers

1. technoLog_SpringBootTest_alizee (sujet pages 2-7)
2. technoLog_SpringBootTest_2_alizee (sujet pages 7-12)
3. technoLog_SpringBootTest_3_alizee (sujet pages 12-22)
 - ✓ **Exécuter la classe principale SwaggerApplication.java**
4. technoLog_SpringBootTest_SpringWeb (suejt pages 22-23)
5. technoLog_SpringBootTest_Thymeleaf (sujet pages 23-28)
6. technoLog_SpringBootTest_Actuator (sujet pages 28-38)

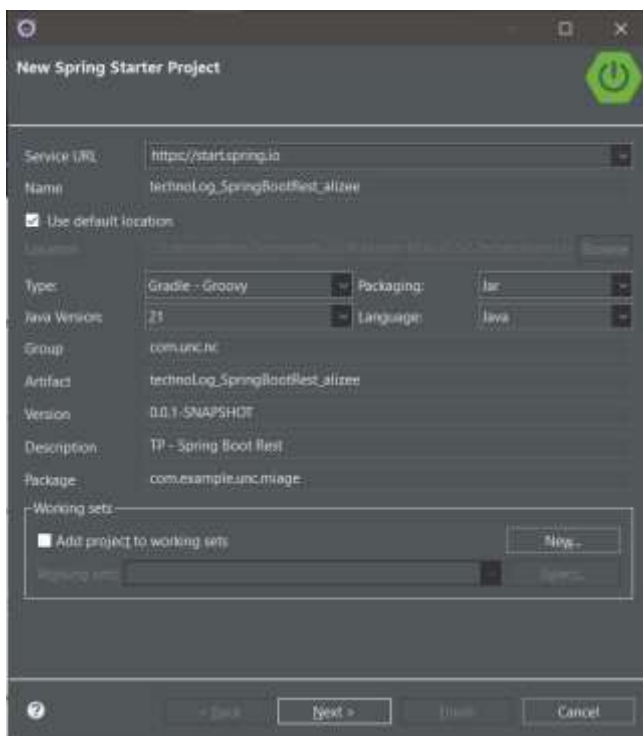
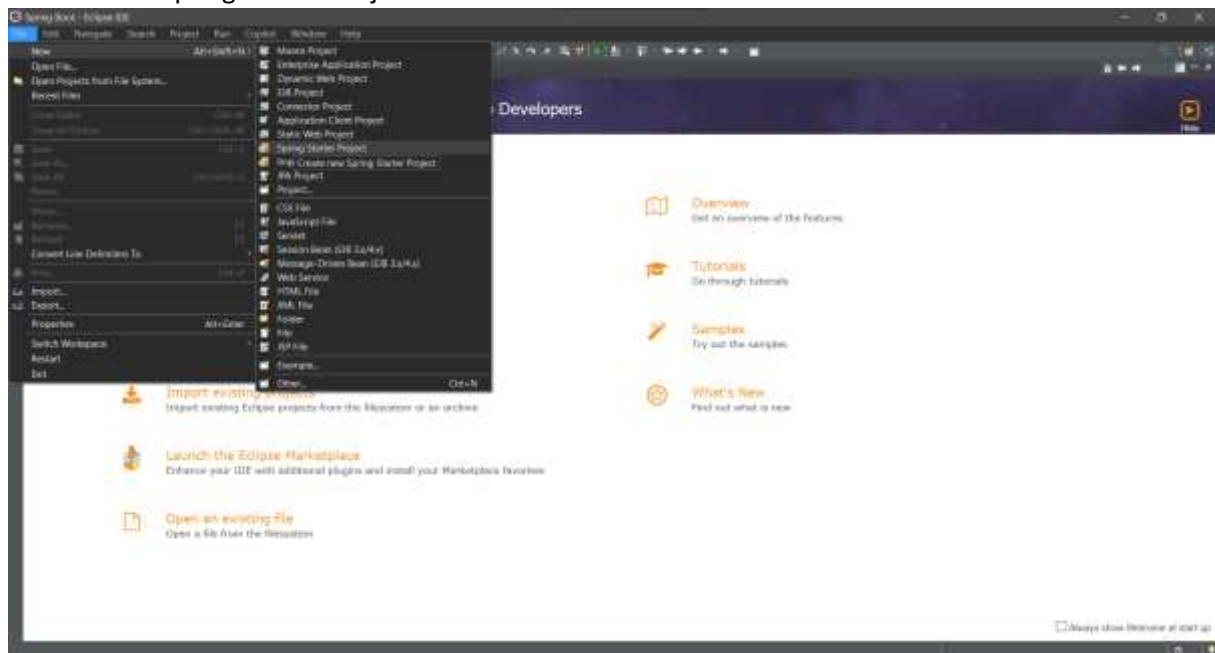
Un projet WebService REST

Pour créer un nouveau projet Spring Boot : <https://start.spring.io/>



Ou autre moyen (directement sous Eclipse) :

File > New > Spring Starter Project



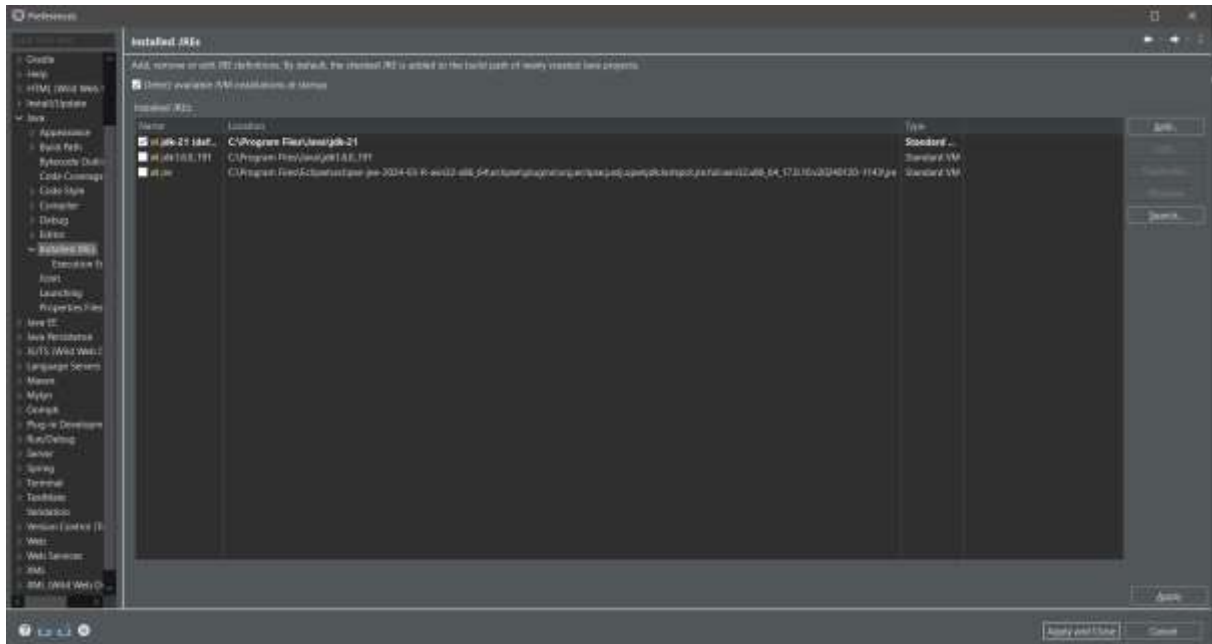
Étant donné que j'ai choisi pour version Java 21

1. Vérifiez que JDK 21 est installé

Assurez-vous que le JDK 21 est bien installé sur votre machine.

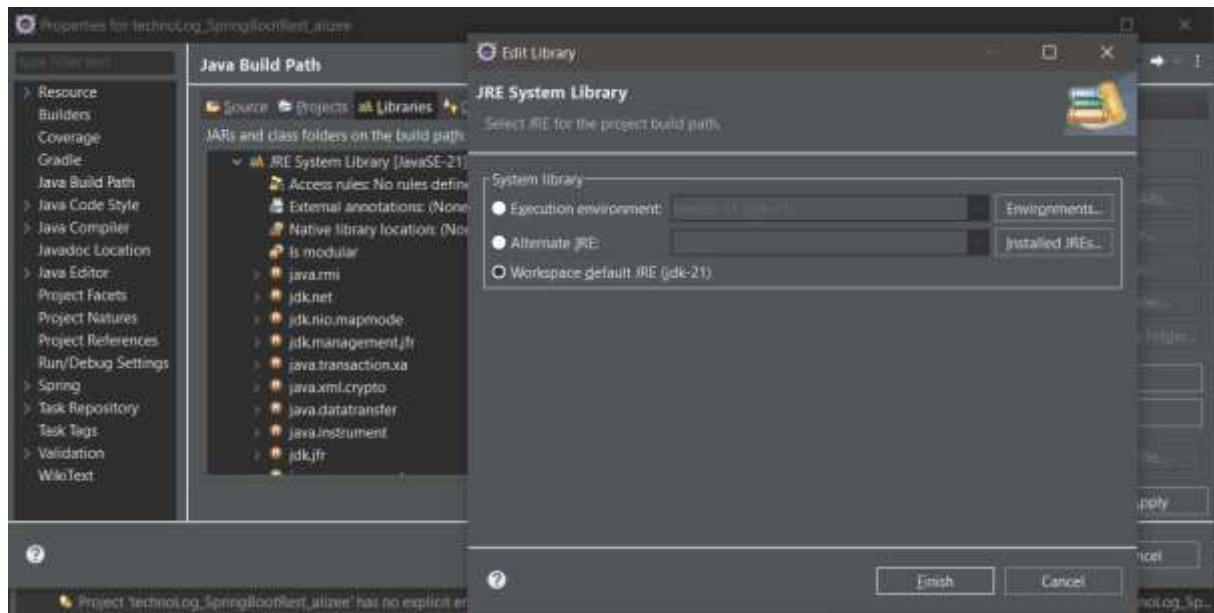
2. Ajoutez le JDK 21 dans Eclipse

1. Ouvrez Eclipse.
2. Allez dans Window > Preferences (ou Eclipse > Preferences sur macOS).
3. Dans la section Java, sélectionnez Installed JREs.
4. Cliquez sur Add... pour ajouter un nouveau JRE.
5. Sélectionnez Standard VM et cliquez sur Next >.
6. Cliquez sur Directory... et naviguez jusqu'au répertoire où le JDK 21 est installé, puis cliquez sur Finish.
7. Assurez-vous que le JDK 21 est sélectionné dans la liste des JRE installés.



3. Modifiez le JRE System Library du projet

1. Dans l'Explorateur de projets, faites un clic droit sur votre projet (technoLog_SpringBootRest_alizee).
2. Sélectionnez Properties.
3. Dans la section Java Build Path, allez à l'onglet Libraries.
4. Trouvez JRE System Library [JavaSE-21] dans la liste et sélectionnez-la.
5. Cliquez sur Edit....
6. Dans la fenêtre JRE System Library, choisissez Workspace default JRE ou sélectionnez le JDK 21 que vous avez ajouté précédemment.
7. Cliquez sur Finish et Apply and Close.



4. Nettoyez et reconstruisez le projet

1. Allez dans Project > Clean....
2. Sélectionnez votre projet et cliquez sur Clean.

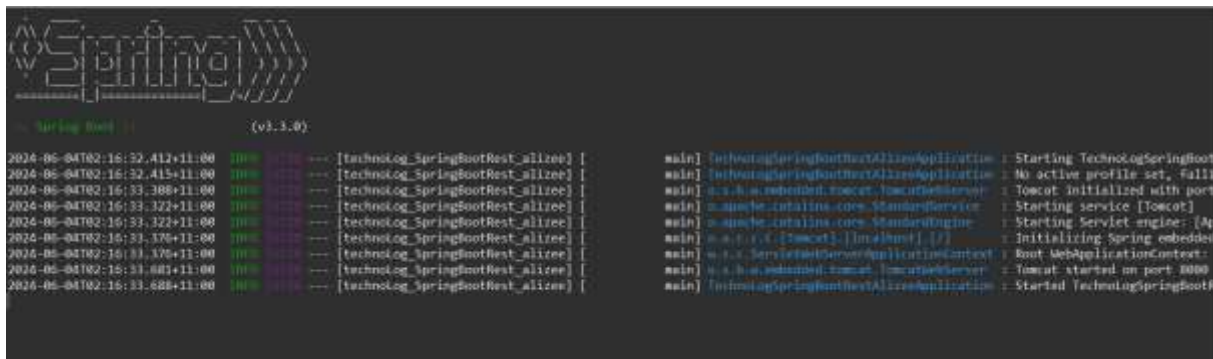
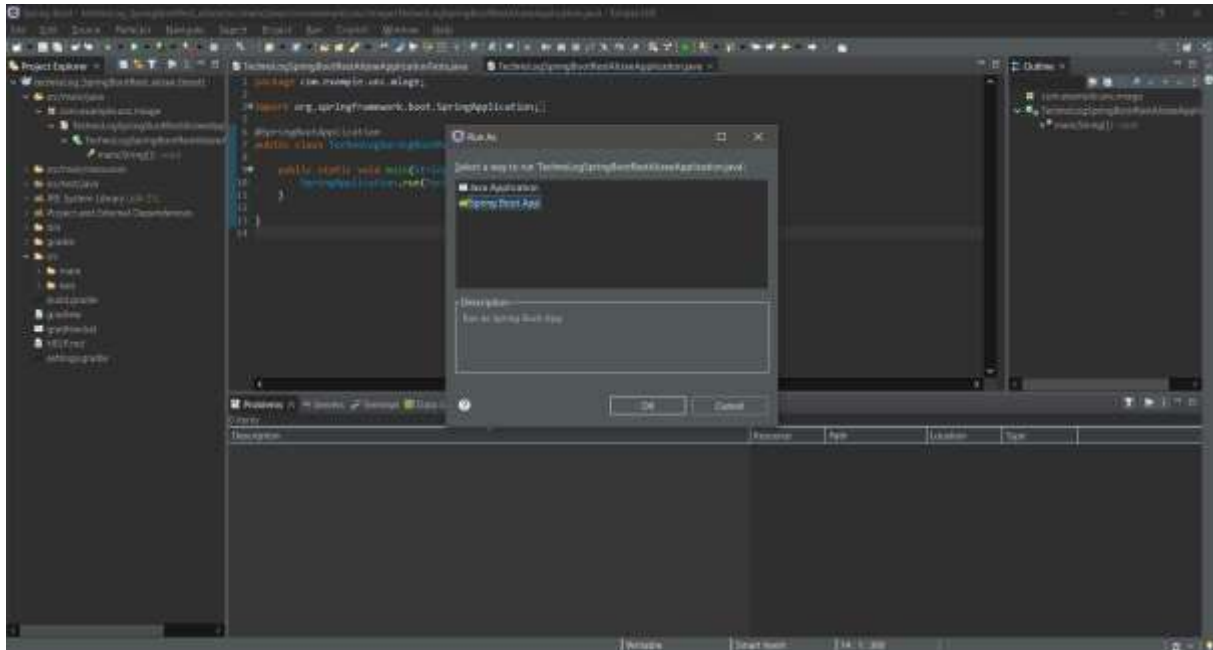
5. Problème d'encodage

Pour résoudre cette erreur : **Project 'technoLog_SpringBootRest_alizee' has no explicit encoding set technoLog_SpringBootRest_alizee /technoLog_SpringBootRest_alizee No explicit project encoding**

Le message d'erreur indique que votre projet n'a pas d'encodage explicite défini. Cela peut entraîner des problèmes de compatibilité lorsque vous travaillez avec des fichiers texte, car l'encodage par défaut peut varier selon l'environnement de développement. Pour résoudre ce problème, vous pouvez spécifier explicitement l'encodage dans les paramètres du projet Eclipse. Voici comment procéder :

1. Cliquez avec le bouton droit sur votre projet dans l'Explorateur de projets d'Eclipse.
2. Sélectionnez Properties (Propriétés).
3. Dans la fenêtre des propriétés du projet, accédez à Resource (Ressource).
4. Recherchez l'option Text file encoding (Encodage des fichiers texte).
5. Cochez la case Other (Autre).
6. Dans le menu déroulant, sélectionnez l'encodage que vous souhaitez utiliser. Par exemple, vous pouvez choisir UTF-8 qui est largement utilisé et prend en charge de nombreux caractères spéciaux.
7. Cliquez sur Apply and Close (Appliquer et fermer) pour enregistrer les modifications.

Après avoir lancé l'application :



Création du modèle Invite.java

```
package com.example.unc.miage;

import java.io.Serializable;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.Date;
import com.fasterxml.jackson.annotation.JsonFormat;

public class Invite implements Serializable {
    private static final long serialVersionUID = 1L;
    private int id;
    @JsonFormat(shape = JsonFormat.Shape.STRING, pattern = "dd-MM-yyyy")
    private Date date;
    private String nom;
    private String prenom;
    private String email;

    public Invite() {
    }

    public int getId() {
        return this.id;
    }
}
```

```
}

public void setId(int id) {
    this.id = id;
}

public Date getDate() {
    return this.date;
}

public void setDate(Date date) {
    this.date = date;
}

public String getNom() {
    return this.nom;
}

public void setNom(String nom) {
    this.nom = nom;
}

public String getPrenom() {
    return this.prenom;
}

public void setPrenom(String prenom) {
    this.prenom = prenom;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}

public Invite(int id, String nom, String prenom, String email) {
    this.id = id;
    this.nom = nom;
    this.prenom = prenom;
    this.email = email ;
    LocalDate localdate = LocalDate.now();
    date = new Date(1000 * 24 * 3600 * localdate.toEpochDay());
    System.out.println("date de " + nom + " = " + date);
}

public String toString() {
    String ladate = LocalDate.ofEpochDay(
        (long) (Math.ceil((double) date.getTime() / (double)
(1000 * 3600 * 24))))
        .format(DateTimeFormatter.ofPattern("dd-MM-
yyyy"));
    return nom + " " + " " + prenom + " " + ladate;
}
}
```

Implémentation du contrôleur REST

J'ai une erreur au niveau de cette ligne :

```
return new Invite(counter.incrementAndGet(), =nom, prenom, email);
```

Indiquant : The constructor Invite(long, String, String, String) is undefined

Dans la classe Invite.java voici la modification à apporter :

```
public Invite(long id, String nom, String prenom, String email) {  
    this.id = (int) id;  
    this.nom = nom;  
    this.prenom = prenom;  
    this.email = email;  
    LocalDate localdate = LocalDate.now();  
    date = new Date(1000 * 24 * 3600 * localdate.toEpochDay());  
    System.out.println("date de " + nom + " = " + date);  
}
```

Exécuter le projet. Tester avec un navigateur les différents URL :

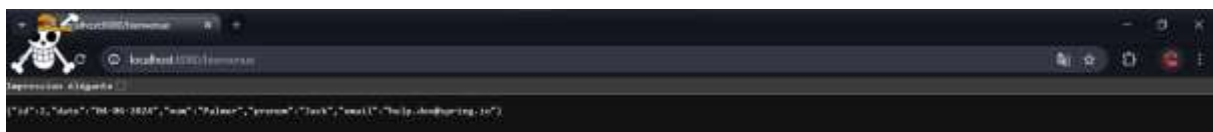
- localhost:8080



- localhost:8080/Dupond



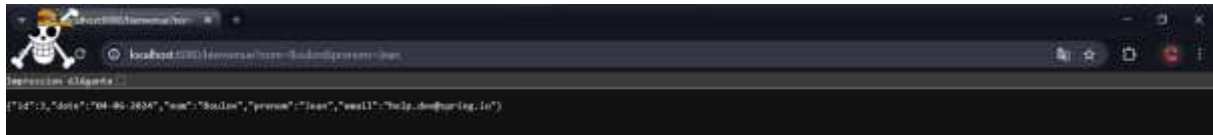
- localhost:8080/bienvenue



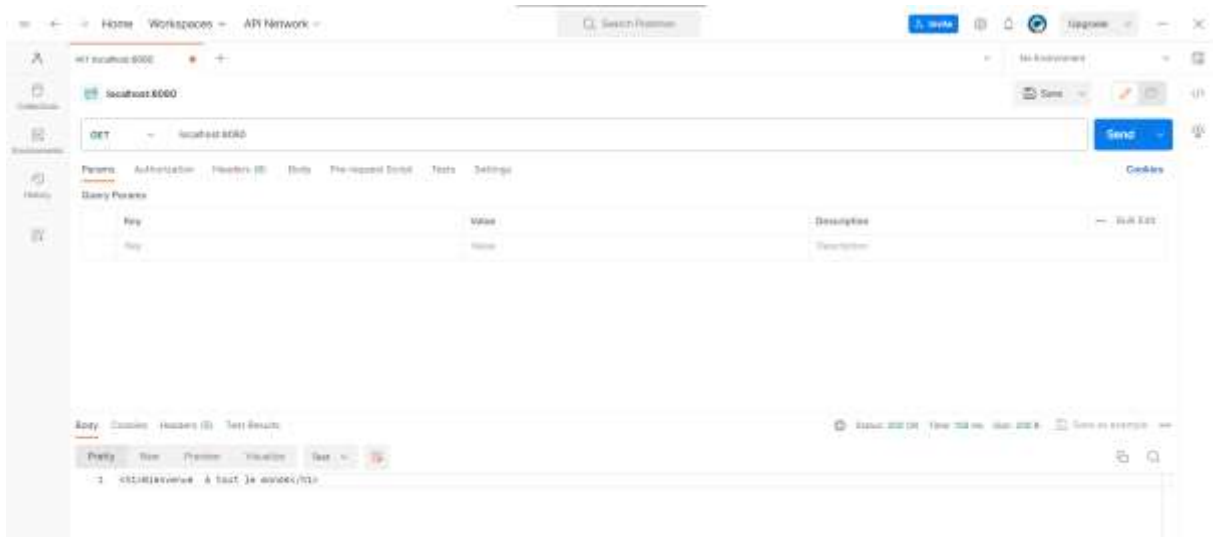
- localhost:8080/test/100



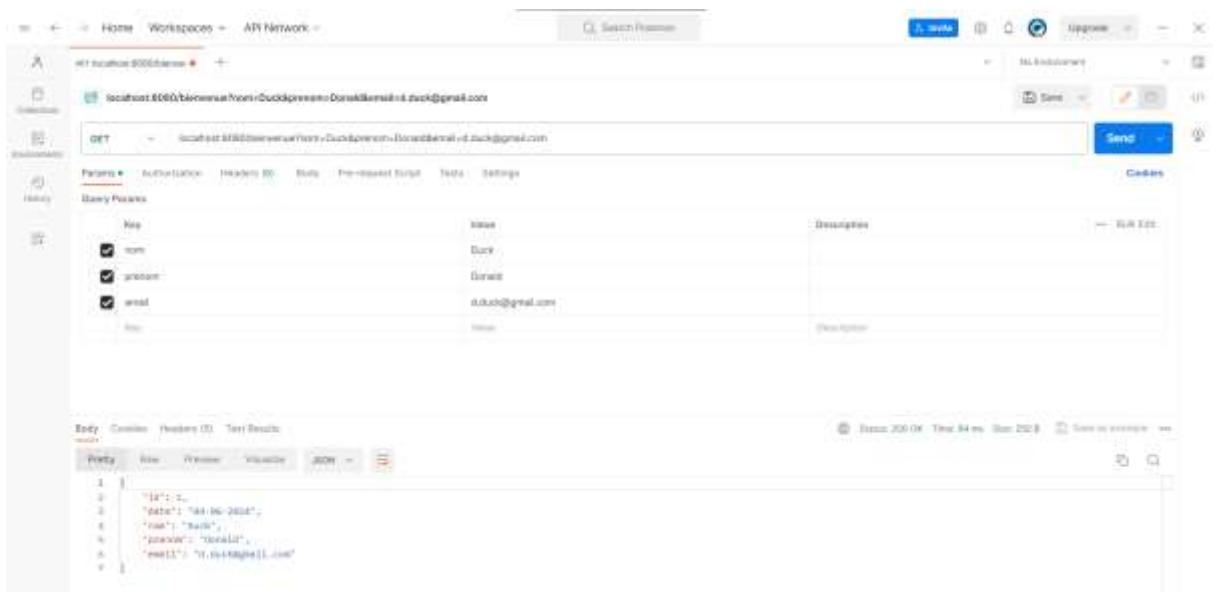
- localhost:8080/bienvenue?nom=Boulon&prenom=Jean



Utilisation d'un client REST : Postman



- GET localhost:8080/bienvenue?nom=Duck&prenom=Donald&email=d.duck@gmail.com



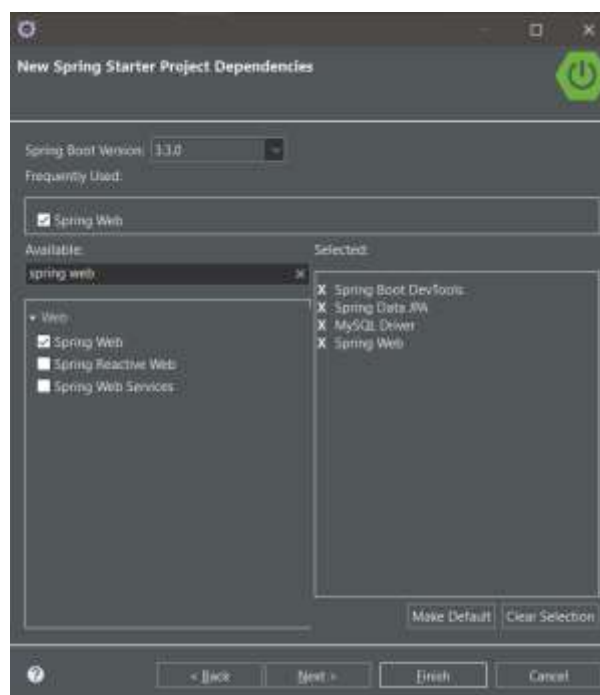
Projet WebService REST avec JPA et MySQL

La base de données : lesinvites

```
CREATE TABLE IF NOT EXISTS invites (  
  id INT(11) NOT NULL AUTO_INCREMENT,  
  nom VARCHAR(255),  
  prenom VARCHAR(255),  
  email VARCHAR(255),  
  date DATE,  
  PRIMARY KEY (id)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

Création du projet

Nom du projet : technoLog_SpringBootRest_2_alizee



Remplissage le fichier src/main/resources/application.properties :

```
spring.application.name=technoLog_SpringBootRest_2_alizee  
  
spring.jpa.hibernate.ddl-auto=none  
spring.datasource.url=jdbc:mysql://localhost:3306/lesinvites2?serverTimezone=UTC  
spring.datasource.username=root  
spring.datasource.password=Alizee.2024
```

Autres valeurs pour cette ligne : spring.jpa.hibernate.ddl-auto = none

1. **validate** : Hibernate valide le schéma existant par rapport aux entités définies. Si les schémas ne correspondent pas, une exception est levée. Aucune modification n'est apportée à la base de données.

2. **update** : Hibernate met à jour le schéma de la base de données pour qu'il corresponde aux entités définies. Les modifications non destructives sont appliquées, comme l'ajout de nouvelles colonnes ou de nouvelles tables.
3. **create** : Hibernate crée le schéma de la base de données en supprimant d'abord les tables existantes (s'il y en a) et en les recréant à partir des entités définies. Cela supprime toutes les données existantes dans les tables.
4. **create-drop** : Hibernate crée le schéma de la base de données au démarrage de l'application de la même manière que create, mais il supprime également les tables au moment de l'arrêt de l'application.
5. **none** : Aucune action n'est effectuée par Hibernate sur le schéma de la base de données. C'est la valeur par défaut.

L'entité Invite

Erreur au niveau des imports que j'ai remplacé par ceci :

```
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.Table;
```

Le repository invité

On importe le modèle Invite :

```
package repository;

import java.util.ArrayList;
import org.springframework.data.repository.CrudRepository;

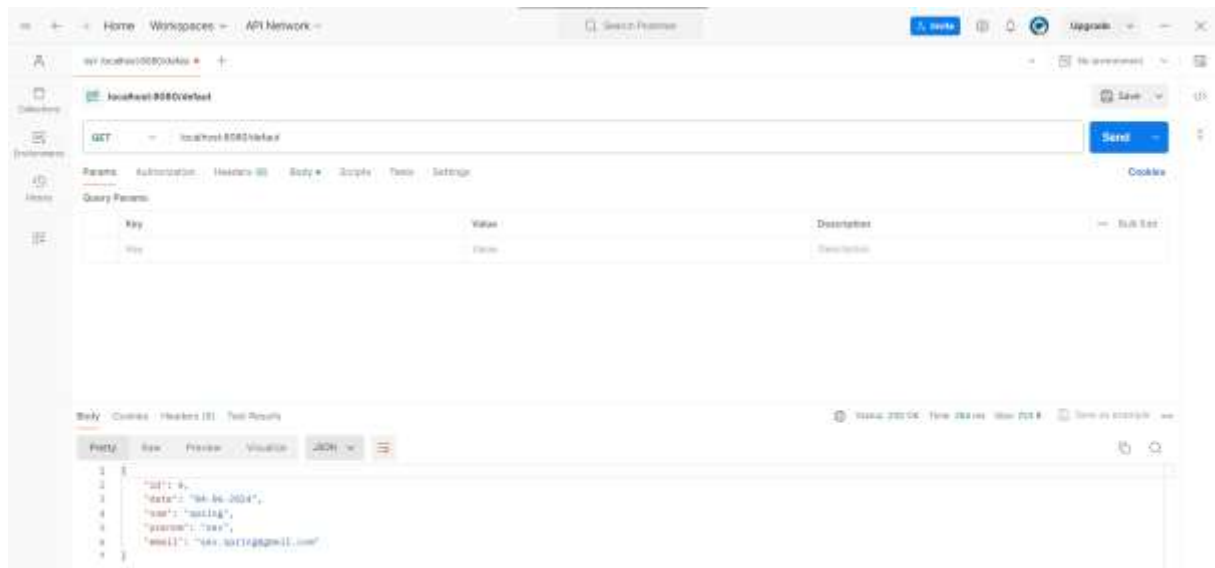
import model.Invite;
public interface InviteRepository extends CrudRepository<Invite, Long> {
    ArrayList<Invite> findByNomAndPrenom(String nom, String prenom);
    ArrayList<Invite> findByNom(String nom);
}
```

Le contrôleur REST: 1ère version simplifiée

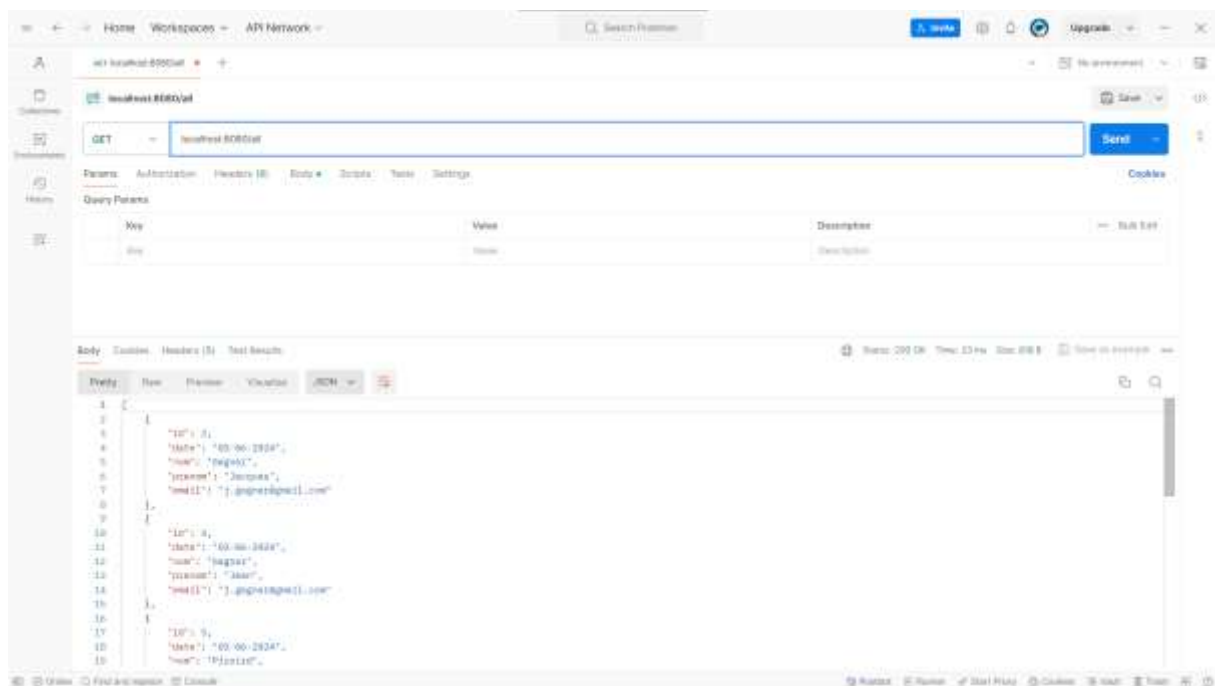
Exécution du projet .

Testez les différents URL, chaque URL fournissant un web service est appelé un end-point. Vérifiez les résultats affichés dans Postman et notamment le code de retour.

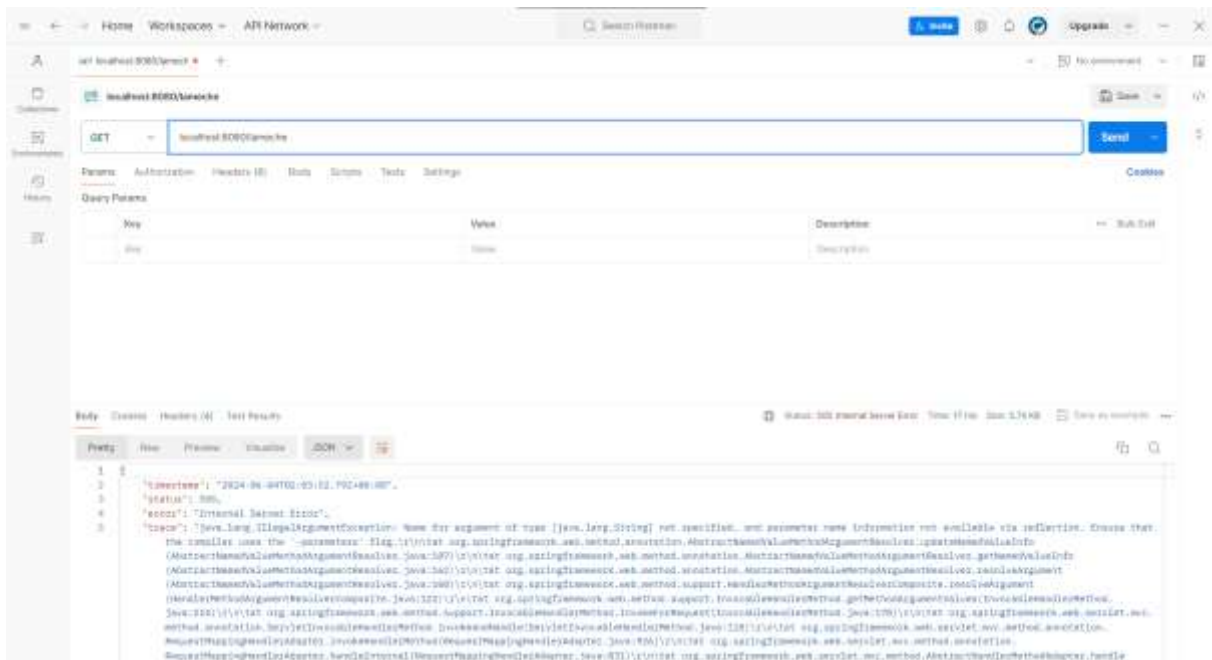
- localhost:8080/default



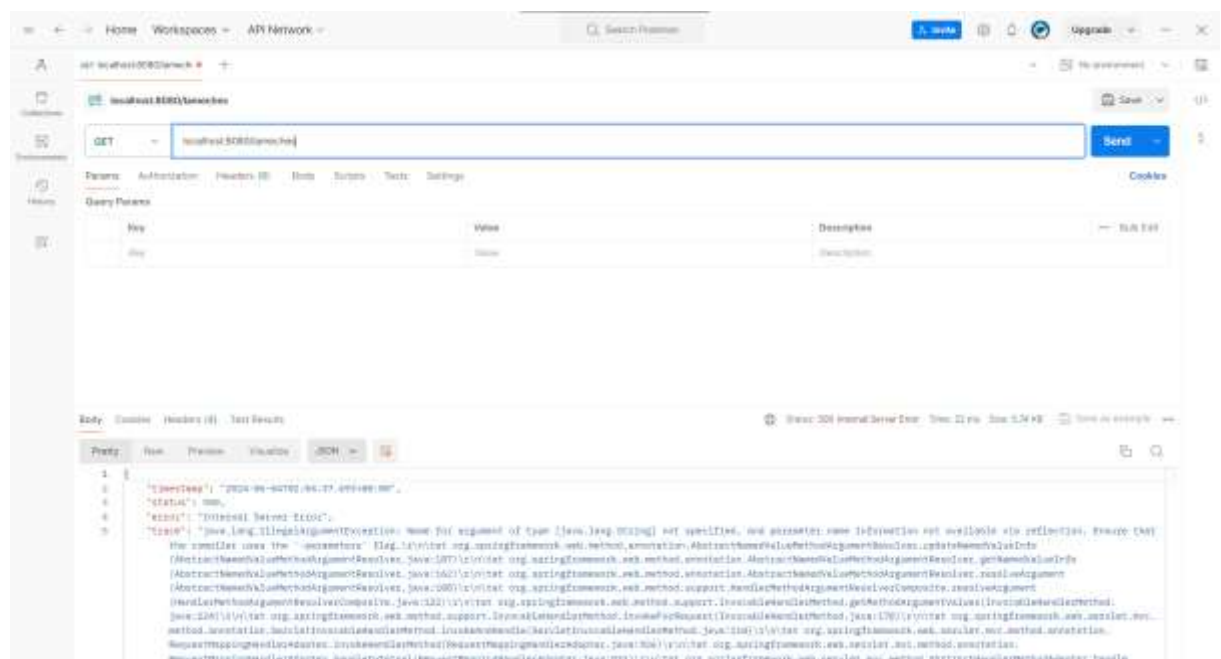
- `localhost:8080/all`



- `localhost:8080/lameche`



- localhost:8080/lameches



Conclure

Les deux premières adresses URL `localhost:8080/defaut` et `localhost:8080/all` produisent toutes deux des réponses 200, car elles sont correctement définies dans le contrôleur `InvitesController`."

```

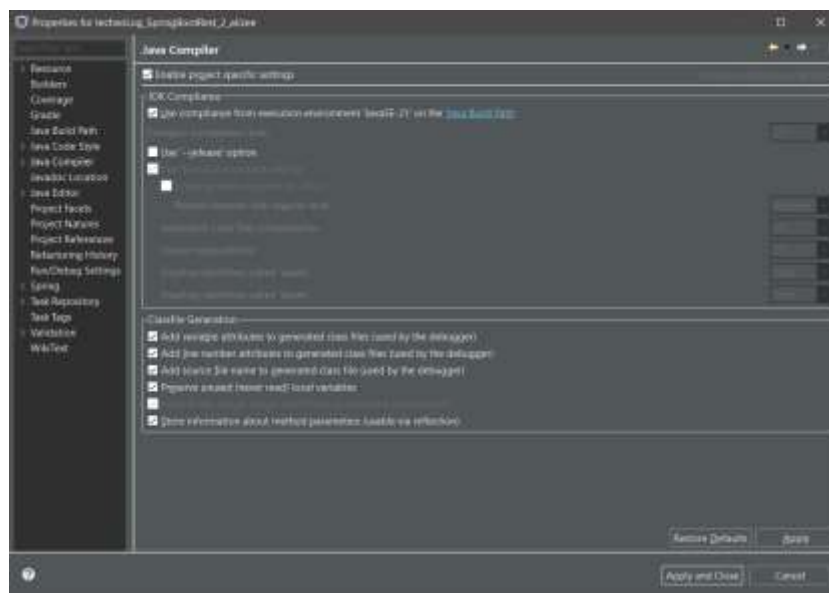
@RequestMapping("/defaut")
public Invite getDefaut() {
    return new Invite("spring", "sav", "sav.spring@gmail.com");
}

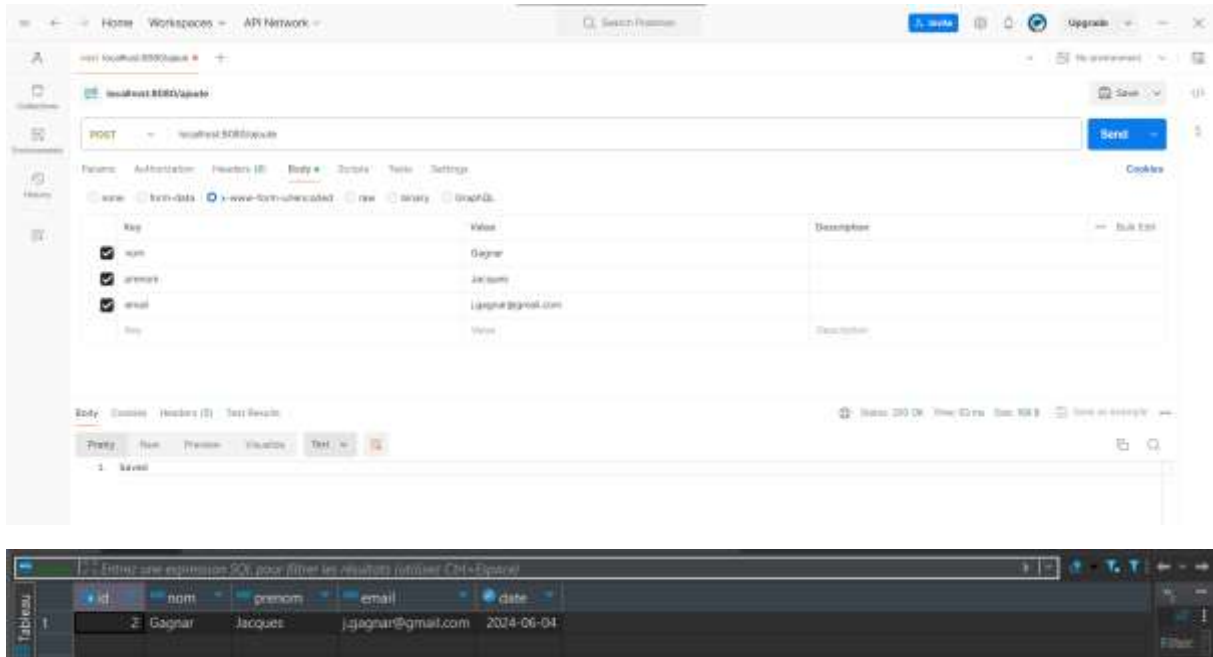
```

```
@RequestMapping("/all")
ArrayList<Invite> getAll() {
    return (ArrayList<Invite>)inviteRepository.findAll();
}
```

La requête localhost:8080/all renvoie une liste vide, car aucune donnée n'est présente en base de données. Quant aux URL localhost:8080/lameche et localhost:8080/lameches, elles génèrent des erreurs de type 500, puisqu'aucun invité portant le nom 'lameche' ou 'lameches' n'est trouvé.

Ajout d'un invité avec la commande POST





Un contrôleur REST 2ème version

Nom du projet : technolog_SpringBootRest_3_alizee

On copie les fichiers Invite.java et InviteRepository et application.properties du projet précédent dans ce projet.

La classe ResponseEntity

Spring Web propose la classe ResponseEntity pour créer la réponse renvoyée au client.

Cette classe permet de préciser tous les éléments de la réponse HTTP renvoyée au client : le code de retour (status), l'entête et le Body.

- Ajout de la méthode "findByNomAndPrenomAndEmail()" dans la classe InviteRepository

```
package com.example.unc.miage;

import java.util.ArrayList;
import org.springframework.data.repository.CrudRepository;

public interface InviteRepository extends CrudRepository<Invite, Long> {
    ArrayList<Invite> findByNomAndPrenom(String nom, String prenom);
    ArrayList<Invite> findByNom(String nom);
    Invite findByNomAndPrenomAndEmail(String nom, String prenom, String email);
}
```

Création d'une classe InvitesController

```
package com.example.unc.miage;

import java.util.ArrayList;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
```

```
import org.springframework.web.bind.annotation.RestController;

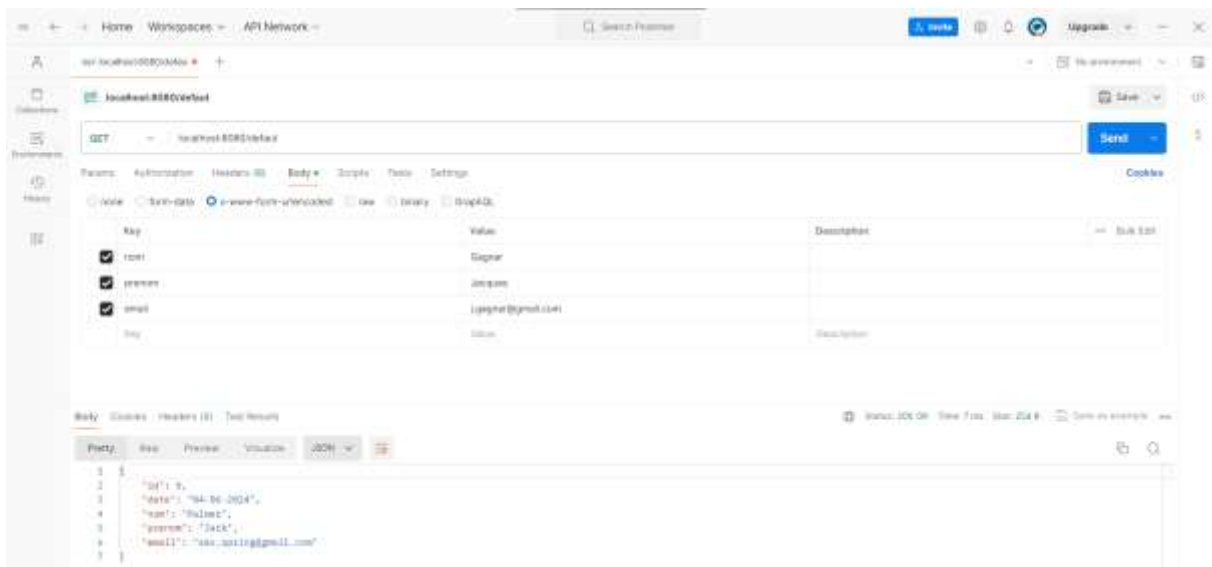
@RestController
public class InvitesController {

    @Autowired
    private InviteRepository inviteRepository;

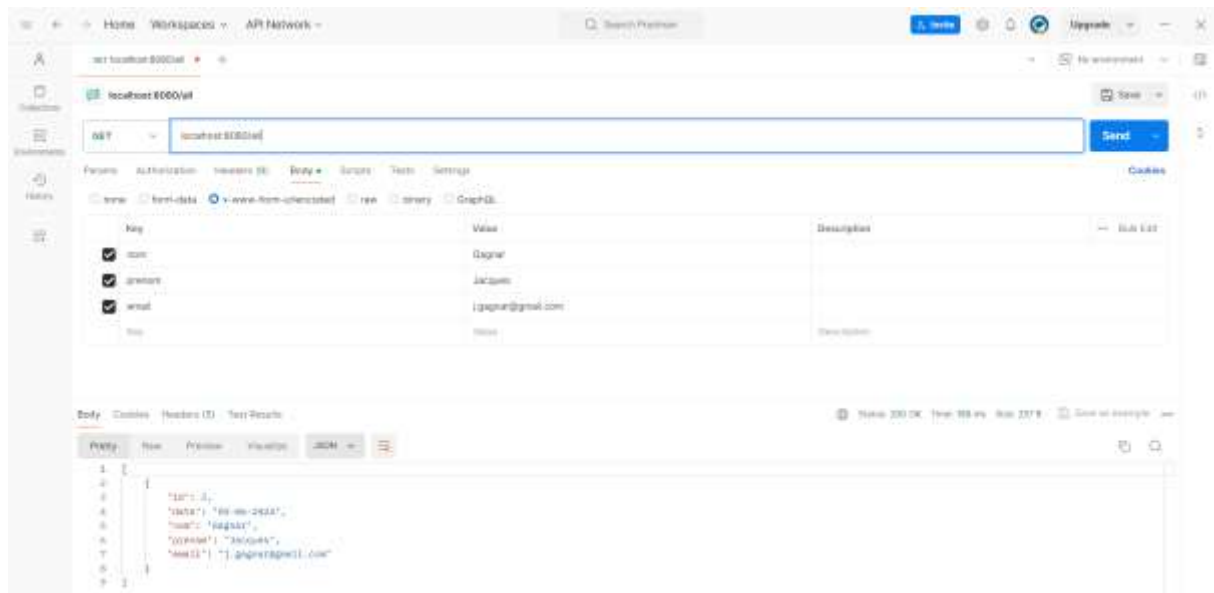
    @GetMapping("/{invite}") // Map requêtes GET
    public ResponseEntity<Invite> rechercheInviteParGetInvite
    (@PathVariable("invite") String nom) {
        ArrayList<Invite> liste = inviteRepository.findByNom(nom);
        if (liste.size()==0)
            return ResponseEntity.notFound().build();
        else
            return ResponseEntity.ok(liste.get(0));
    }
}
```

Tests des différents URL possibles

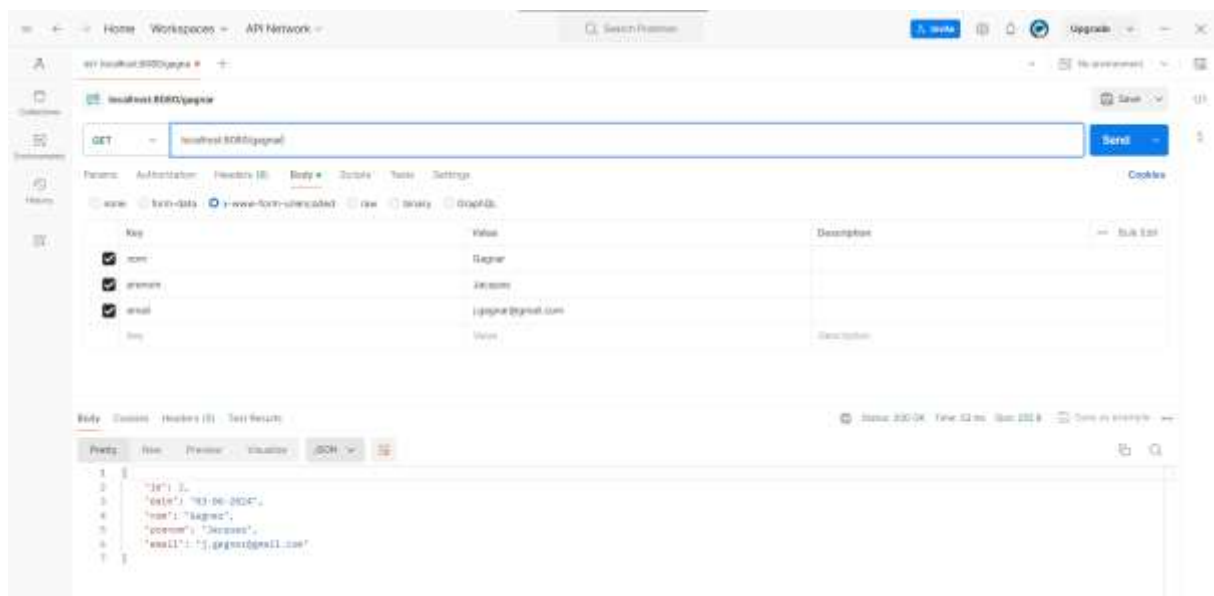
- Pour obtenir l'invité par défaut : **GET localhost:8080/default**



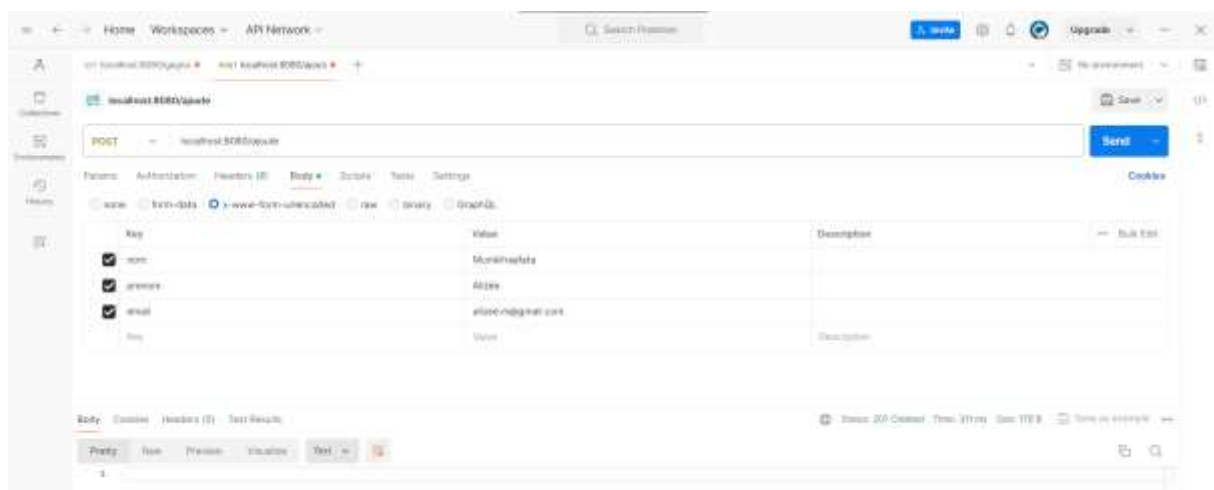
- Pour obtenir tous les invités : **GET localhost:8080/all**



- Rechercher un invité par nom : **GET localhost:8080/{invite}**

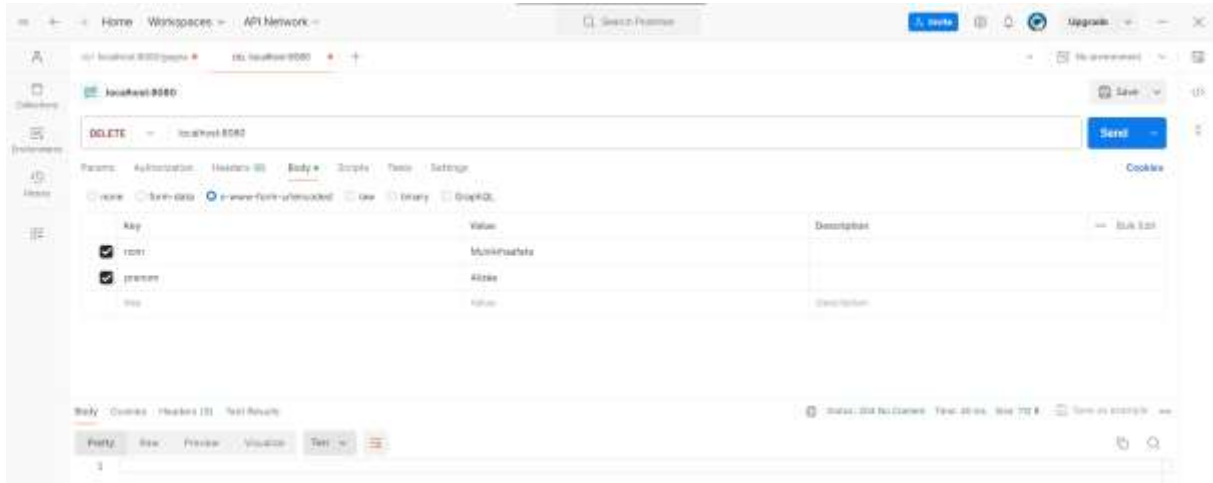


- Ajouter un invité : **POST localhost:8080/ajoute**



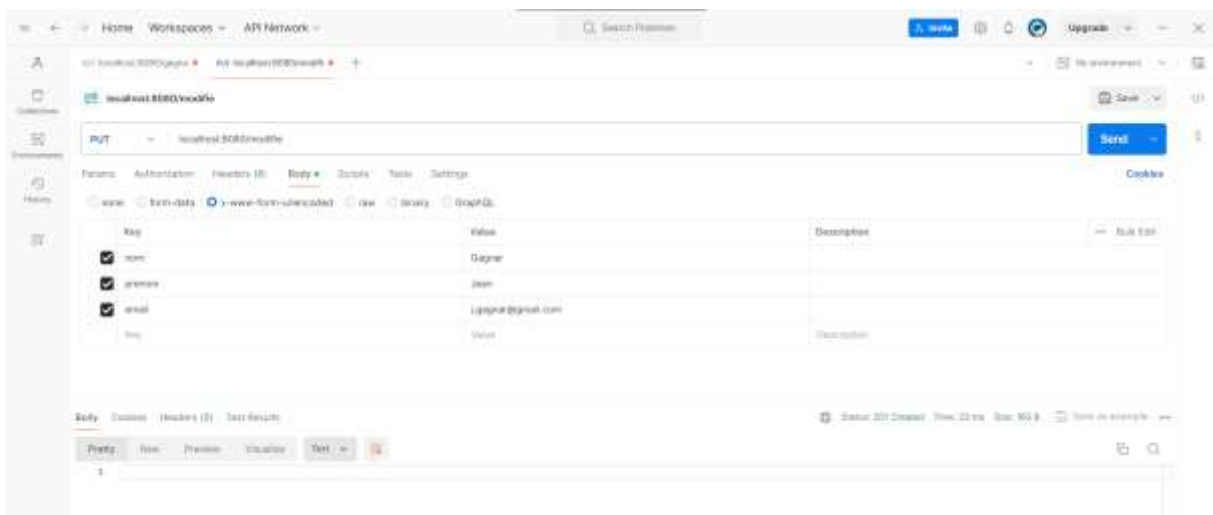
	id	nom	prenom	email	date
1	2	Gagnar	Jacques	j.gagnar@gmail.com	2024-06-04
2	3	Munkihafata	Alizée	alizee.m@gmail.com	2024-06-04

- Supprimer un invité : **DELETE localhost:8080**



	id	nom	prenom	email	date
1	2	Gagnar	Jacques	j.gagnar@gmail.com	2024-06-04

- Modifier un invité : **PUT localhost:8080/modifie**



	id	nom	prenom	email	date
1	2	Gagnar	Jacques	j.gagnar@gmail.com	2024-06-04
2	4	Gagnar	Jean	j.gagnar@gmail.com	2024-06-04

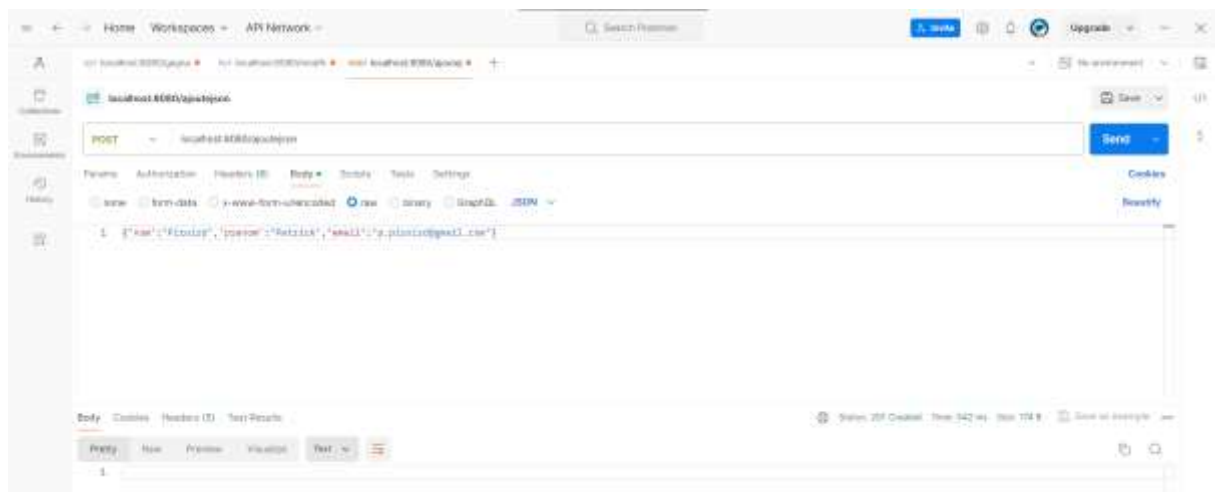
Ajout d'une méthode POST recevant un objet Invite au format Json

Ajout de ce code dans InvitesController :

```
// ajoute un invite, l'objet est reçu au format json
@PostMapping(value = "/ajoutejson")
public ResponseEntity<Void> ajouterInvite(@RequestBody Invite invite) {
```

```
        Invite r =  
inviteRepository.findByNomAndPrenomAndEmail(invite.getNom(), invite.getPrenom(),  
invite.getEmail());  
        if (r != null)  
            return ResponseEntity.noContent().build();  
        invite.setDate(Date.from(Instant.now()));  
        Invite inviteAjoute = inviteRepository.save(invite);  
        URI location = ServletUriComponentsBuilder  
            .fromCurrentRequest().path("/{id}")  
            .buildAndExpand(inviteAjoute.getId()).toUri();  
        return ResponseEntity.created(location).build();  
    }  
}
```

Voir le lien suivant pour les codes de retour des services REST : [HTTP Status Codes - REST API Tutorial \(restfulapi.net\)](https://restfulapi.net)

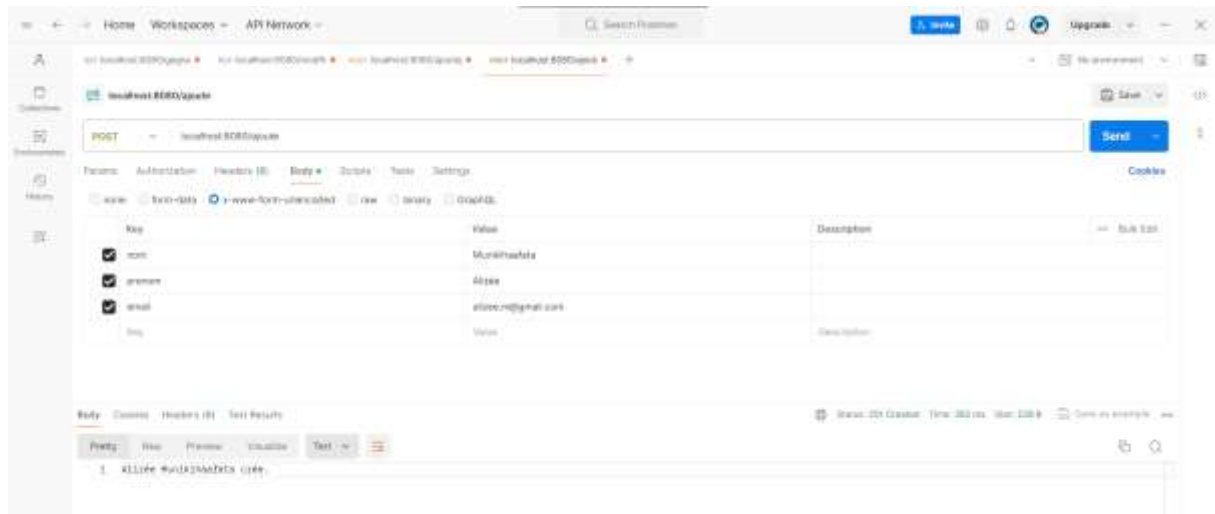


	id	nom	prenom	email	date
1	2	Gagnar	Jacques	j.gagnar@gmail.com	2024-06-04
2	4	Gagnar	Mian	j.gagnar@gmail.com	2024-06-04
3	5	Piroird	Patrick	p.piroird@gmail.com	2024-06-04

Modification de la méthode POST pour retourner en plus un message au client

```
@PostMapping(path="/ajoute")  
public ResponseEntity<String> ajouteInvite (@RequestParam String nom,  
@RequestParam String prenom, @RequestParam String email){  
    Invite r = inviteRepository.findByNomAndPrenomAndEmail(nom, prenom,  
email);  
    if(r != null)  
        return ResponseEntity.noContent().build();  
    Invite i = new Invite(nom, prenom, email);  
    i = inviteRepository.save(i);  
    URI location = ServletUriComponentsBuilder  
        .fromCurrentRequest()  
        .path("/{id}")  
        .buildAndExpand(i.getId())  
        .toUri();  
    HttpHeaders responseHeaders = new HttpHeaders();  
    responseHeaders.setLocation(location);  
}
```

```
return new ResponseEntity<String>(prenom+ " " + nom + " crée.",
responseHeaders, HttpStatus.CREATED);
}
```



Générer la documentation avec swagger

Nom du projet : technoloLog_SpringBootRest_3_alizée

Ajouter la dépendance suivante en rouge dans le fichier pom.xml

Pour un projet utilisant gradle, pom.xml n'existe pas. À la place, nous aurons des fichiers build.gradle et éventuellement settings.gradle pour gérer les dépendances et la configuration du projet.

Dans build.gradle : (Assurez-vous d'avoir ces dépendances)

```
dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
    implementation 'org.springframework.boot:spring-boot-starter-web'
    developmentOnly 'org.springframework.boot:spring-boot-devtools'
    runtimeOnly 'com.mysql:mysql-connector-j'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
    testRuntimeOnly 'org.junit.platform:junit-platform-launcher'
    implementation 'org.springdoc:springdoc-openapi-starter-webmvc-ui:2.0.2'
    implementation 'io.springfox:springfox-swagger2:3.0.0'
    implementation 'io.springfox:springfox-swagger-ui:3.0.0'
    implementation 'javax.servlet:javax.servlet-api:4.0.1'
}
```

Pour rafraichir : clic droit sur le fichier build.gradle > Gradle > Refresh Gradle Project

Ajouter la ligne suivante dans le fichier application.properties

```
spring.mvc.pathmatch.matching-strategy=ANT_PATH_MATCHER
```

Placer l'annotation @EnableSwagger2 sur la classe contenant le main

Clic droit sur build.gradle > Gradle > Refresh Gradle Projects

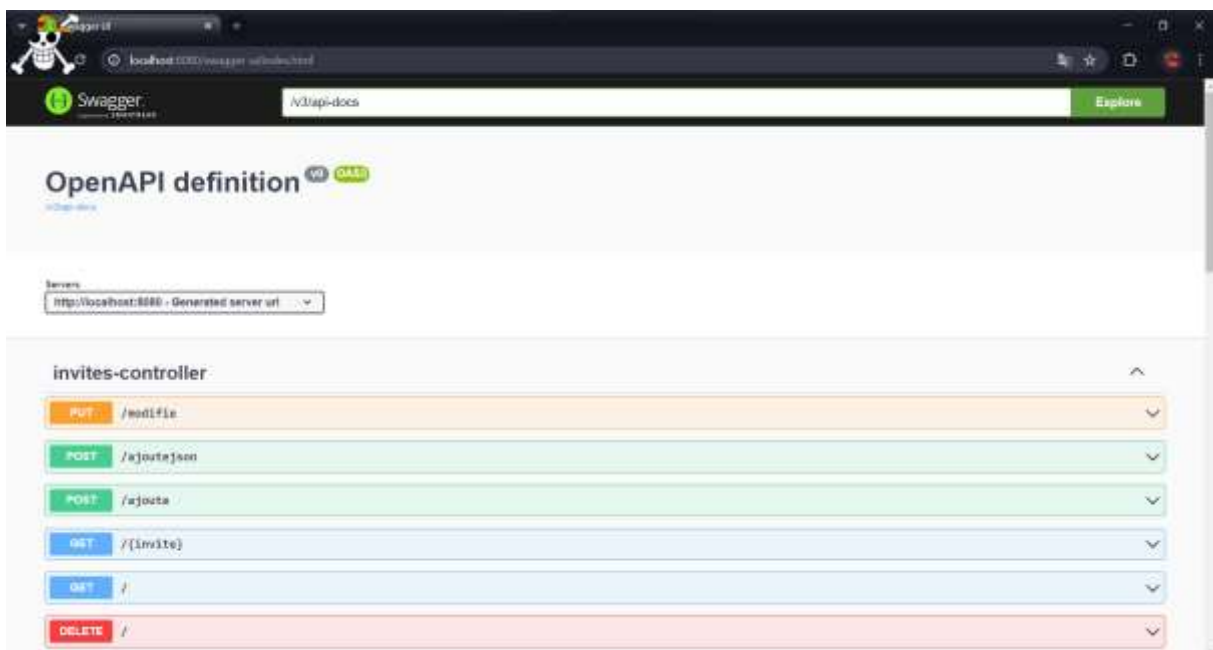
```
package com.example.unc.miage;
```

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import springfox.documentation.swagger2.annotations.EnableSwagger2;

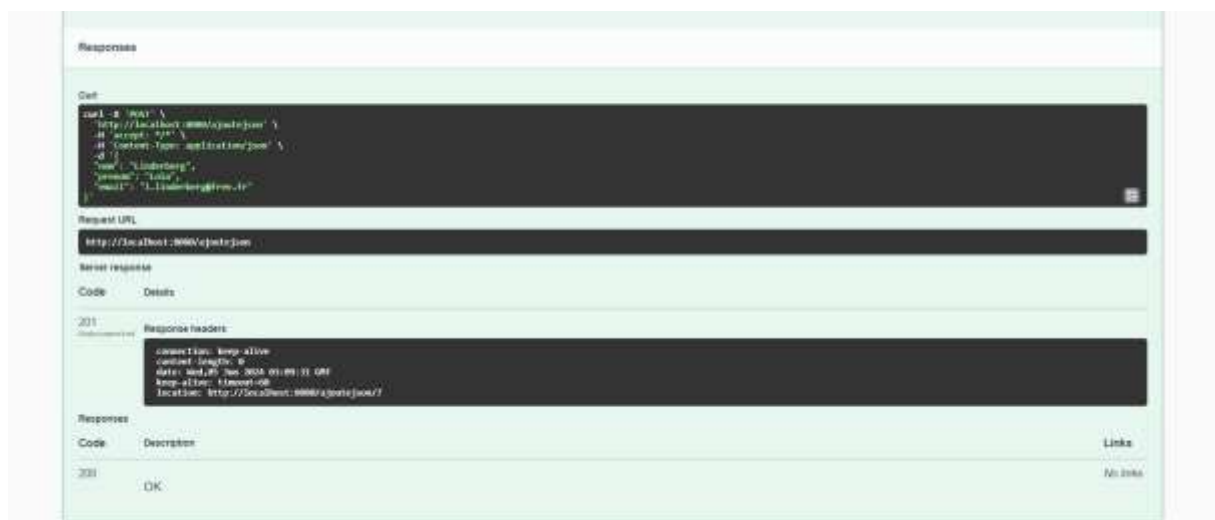
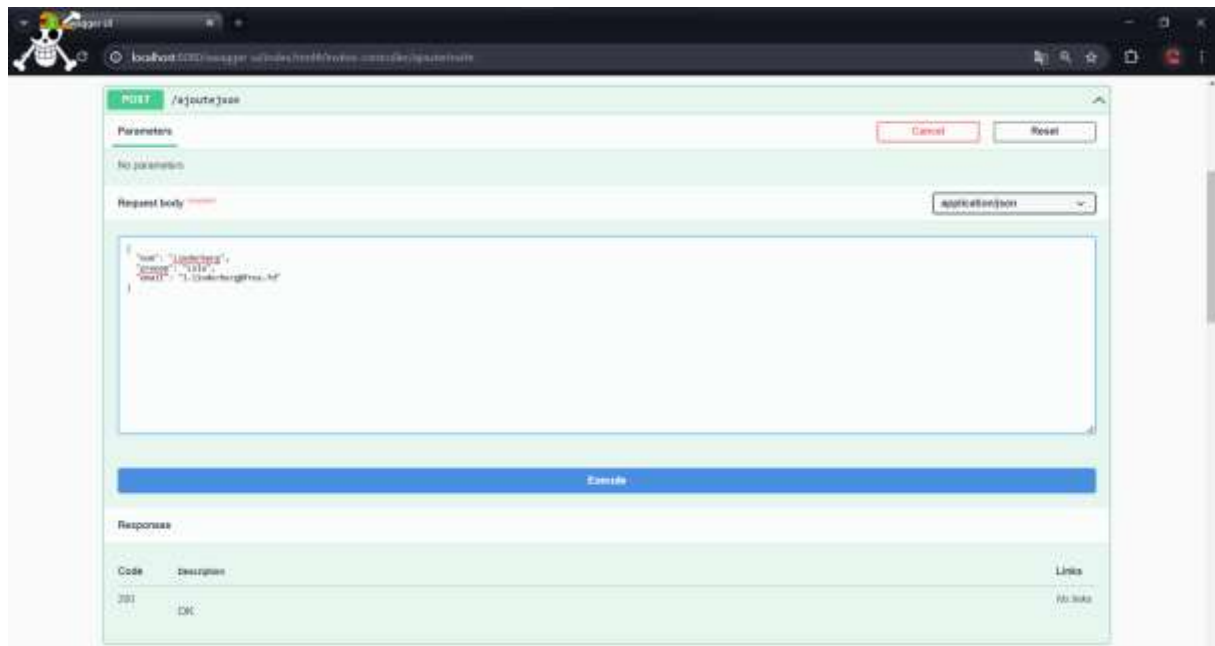
@SpringBootApplication
@EnableSwagger2
public class SwaggerApplication {
    public static void main(String[] args) {
        SpringApplication.run(SwaggerApplication.class, args);
    }
}
```

Lancez l'application SwaggerApplication

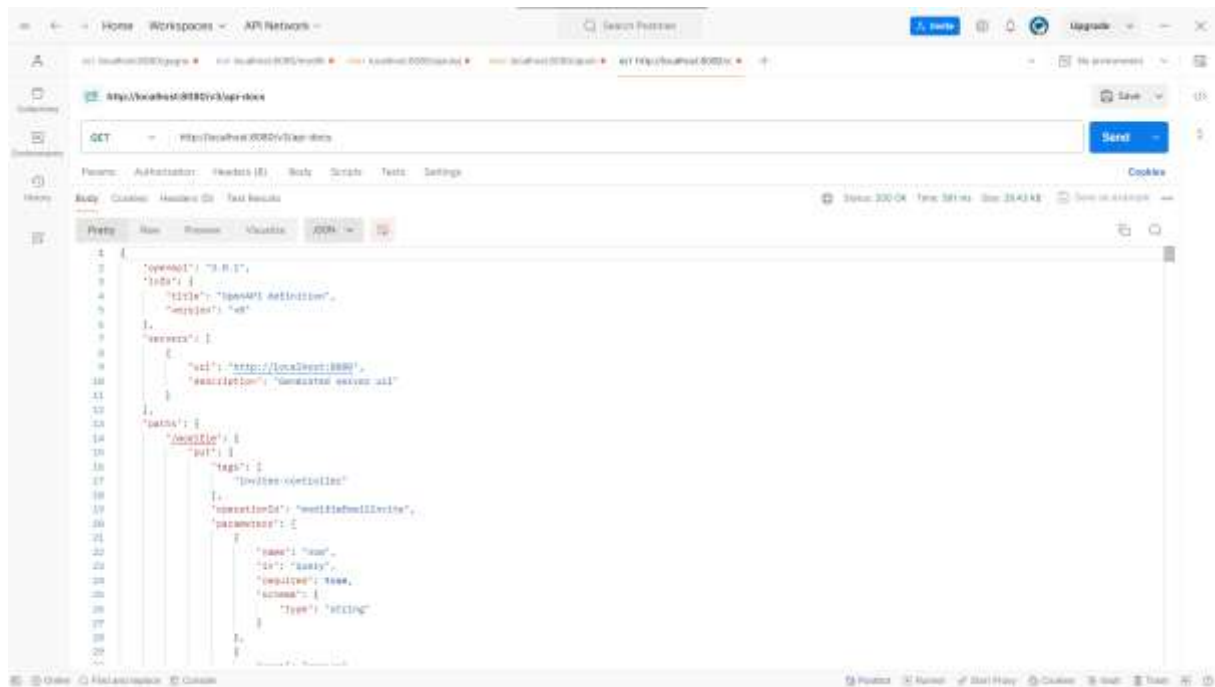
Lien à saisir dans le navigateur : <http://localhost:8080/swagger-ui/index.html>



Je n'ai pas de basic error controller mais j'ai des méthodes qui sont redondants

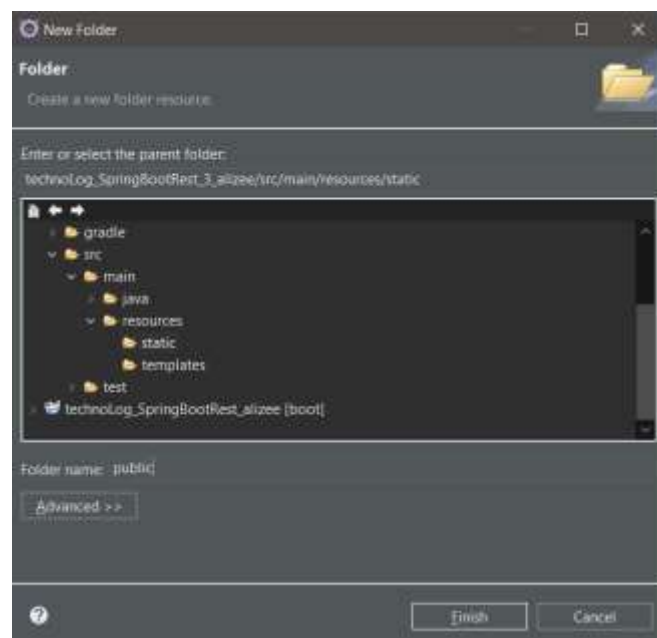


- Exécuter l'url `http://localhost:8080/v3/api-docs` dans Postman, cette requête affiche toutes les informations nécessaires pour utiliser les services concernés.

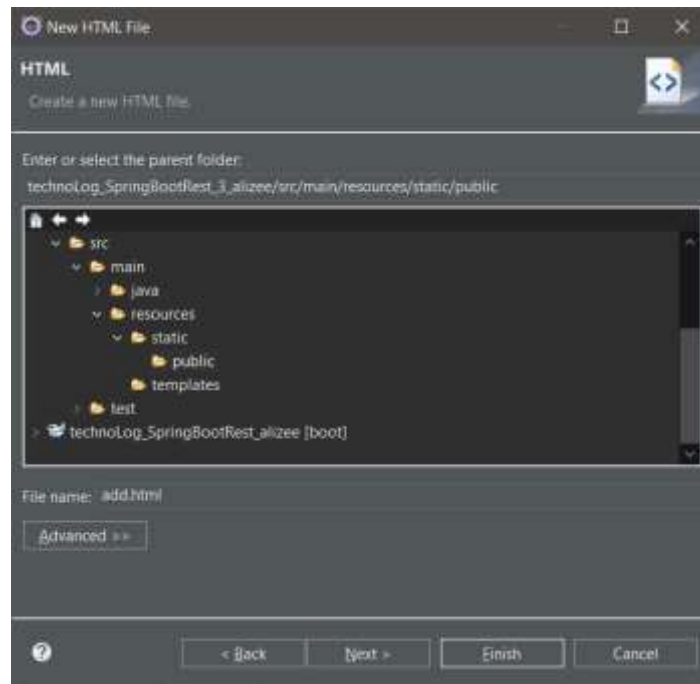


Test avec un formulaire HTML

Création du dossier public



Fichier add.html



```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Test du formulaire</title>
</head>
<body>
  <form action="../ajoute" method="post">
    <table>
      <tr>
        <td>Nom :</td>
        <td><input type="text" name="nom"></input></td>
      </tr>
      <tr>
        <td>Prénom :</td>
        <td><input type="text" name="prenom"></input></td>
      </tr>
      <tr>
        <td>Email :</td>
        <td><input type="text" name="email"></input></td>
      </tr>
      <tr>
        <td></td>
        <td><input type="submit" value="Validez"></input></td>
      </tr>
    </table>
  </form>
</body>
</html>
```

Exécution du lien : <http://localhost:8080/public/add.html>



Appeler un service REST à partir d'un autre service REST

Nom du projet : technoloLog_SpringBootRest_SpringWeb



La classe Controller

```
package com.example.unc.miage;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.client.RestTemplate;
@RestController
public class Controller {

    @GetMapping(value = "/invite")
    public Invite getStudent() {
        return new Invite(1, "Dupond", "Jean", "j.dupond");
    }

    @GetMapping(value = "/getinvitestring")
    private String getStudentString() {
        String uri = "http://localhost:8080/invite";
```



```
        RestTemplate restTemplate = new RestTemplate();
        String result = restTemplate.getForObject(uri, String.class);
        return result;
    }

    @GetMapping(value = "/getinvite")
    private Invite getStudentObject() {
        String uri = "http://localhost:8080/invite";
        RestTemplate restTemplate = new RestTemplate();
        Invite result = restTemplate.getForObject(uri, Invite.class);
        return result;
    }
}
```

La classe Invite.java

```
package com.example.unc.miage;

import java.io.Serializable;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.Date;

import com.fasterxml.jackson.annotation.JsonFormat;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.Table;

@Entity
@Table(name = "invites")
public class Invite implements Serializable {
    private static final long serialVersionUID = 1L;

    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;
    @JsonFormat(shape = JsonFormat.Shape.STRING, pattern = "dd-MM-yyyy")

    private Date date;
    private String nom;
    private String prenom;
    private String email;

    public Invite() {
    }

    public long getId() {
        return this.id;
    }

    public void setId(long id) {
        this.id = id;
    }
}
```

```
public Date getDate() {
    return this.date;
}

public void setDate(Date date) {
    this.date = date;
}

public String getNom() {
    return this.nom;
}

public void setNom(String nom) {
    this.nom = nom;
}

public String getPrenom() {
    return this.prenom;
}

public void setPrenom(String prenom) {
    this.prenom = prenom;
}

public String getEmail() {
    return this.email;
}

public void setEmail(String email) {
    this.email = email;
}

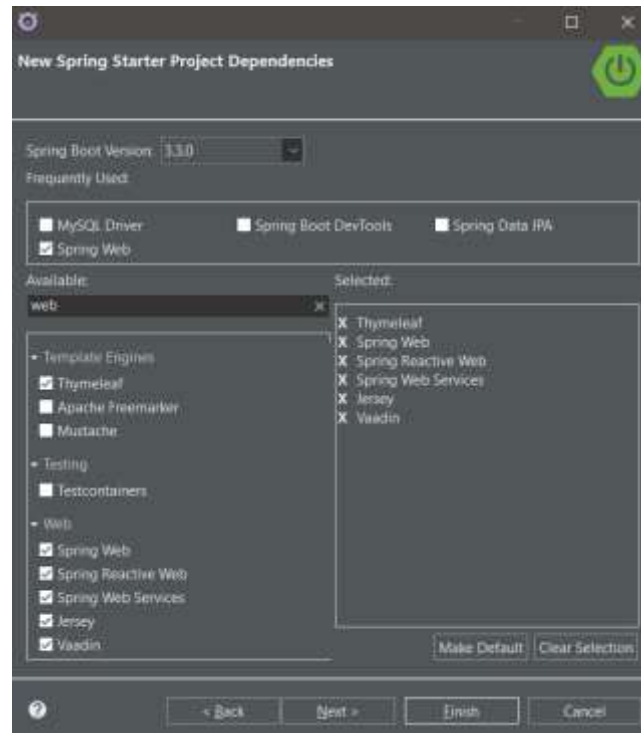
public Invite(long id, String nom, String prenom, String email) {
    this.id = id;
    this.nom = nom;
    this.prenom = prenom;
    this.email = email;
    LocalDate localdate = LocalDate.now();
    date = new Date(1000*24*3600*localdate.toEpochDay());
}

public Invite(String nom, String prenom, String email) {
    this.nom = nom;
    this.prenom = prenom;
    this.email = email;
    LocalDate localdate = LocalDate.now();
    date = new Date(1000 * 24 * 3600 * localdate.toEpochDay());
}

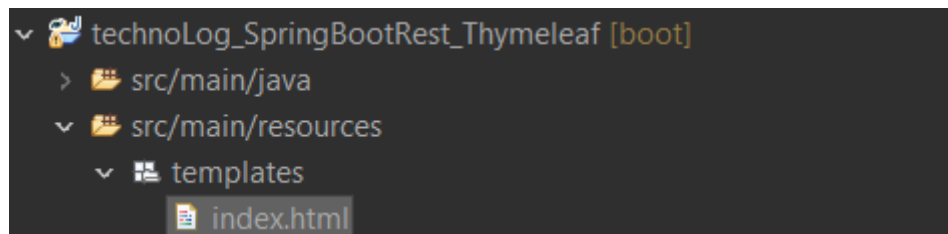
public String toString() {
    String ladata = LocalDate.ofEpochDay((long) (Math.ceil((double)
date.getTime() / (double)
(1000*24*3600))))).format(DateTimeFormatter.ofPattern("dd-MM-yyyy"));
    return nom + " " + prenom + " " + email + " " + ladata;
}
}
```

Projet avec Thymeleaf

Nom du projet : technolLog_SpringBootRest_Thymeleaf



Comme je n'ai pas Web. J'ai coché toutes les dépendances en lien avec Web.



Création du fichier templates/index.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Page d'index</title>
</head>
<body>
  <form action="save" method="post">
    <table>
      <tr>
        <td><label>Nom :</label></td>
        <td><input type="text" name="nom"></input></td>
      </tr>
      <tr>
        <td><label>Pr#233;nom :</label></td>
```

```
        <td><input type="text" name="prenom"></input></td>
    </tr>
    <tr>
        <td><label>Email :</label></td>
        <td><input type="text" name="email"></input></td>
    </tr>
    <tr>
        <td></td>
        <td><input type="submit" value="Validez"></input></td>
    </tr>
</table>
</form>

</body>
</html>
```

Créer le fichier templates/invite-data.html

```
<html xmlns:th="http://thymeleaf.org">
    <table>
        <tr>
            <td><h4>Nom :</h4></td>
            <td><h4 th:text="${invite.nom}"></h4></td>
        </tr>
        <tr>
            <td><h4>Prénom :</h4></td>
            <td><h4 th:text="${invite.prenom}"></h4></td>
        </tr>
        <tr>
            <td><h4>Email :</h4></td>
            <td><h4 th:text="${invite.email}"></h4></td>
        </tr>
    </table>
</html>
```

Créer le modèle Invite.java

```
package com.example.unc.miage;

import java.io.Serializable;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.Date;

import com.fasterxml.jackson.annotation.JsonFormat;

public class Invite implements Serializable {
    private static final long serialVersionUID = 1L;

    private long id;
    @JsonFormat(shape = JsonFormat.Shape.STRING, pattern = "dd-MM-yyyy")
    private Date date;
    private String nom;
    private String prenom;
    private String email;

    public Invite() {
    }
}
```

```
public long getId() {
    return this.id;
}

public void setId(long id) {
    this.id = id;
}

public Date getDate() {
    return this.date;
}

public void setDate(Date date) {
    this.date = date;
}

public String getNom() {
    return this.nom;
}

public void setNom(String nom) {
    this.nom = nom;
}

public String getPrenom() {
    return this.prenom;
}

public void setPrenom(String prenom) {
    this.prenom = prenom;
}

public String getEmail() {
    return this.email;
}

public void setEmail(String email) {
    this.email = email;
}

public Invite(long id, String nom, String prenom, String email) {
    this.id = id;
    this.nom = nom;
    this.prenom = prenom;
    this.email = email;
    LocalDate localdate = LocalDate.now();
    date = new Date(1000*24*3600*localdate.toEpochDay());
    System.out.println("date de " + nom + " = " + date);
}

public String toString() {
    String ldate = LocalDate.ofEpochDay((long)(Math.ceil((double)
date.getTime()/ (double)(1000*3600*24))))
        .format(DateTimeFormatter.ofPattern("dd-MM-yyyy"));
    return nom + " " + prenom + " " + ldate;
}
}
```

Créer le contrôleur InviteController.java

```
package com.example.unc.miage;

import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.stereotype.Controller;

@Controller
public class InviteController {

    @RequestMapping("/")
    public String index() {
        return "index";
    }

    @RequestMapping(value="/save", method=RequestMethod.POST)
    public ModelAndView save(@ModelAttribute Invite invite) {
        ModelAndView modelAndView = new ModelAndView();
        modelAndView.setViewName("invite-data");
        modelAndView.addObject("invite", invite);
        return modelAndView;
    }
}
```

On exécute l'application



- Améliorer l'application en gardant le même formulaire de saisie et en ajoutant dans la vue récapitulative la date de la création de l'invité

Modification des méthodes dans la classe Invite.java, dans le contrôleur InviteController et le fichier invite-data :

Invite.java

```
package com.example.unc.miage;
```

```
import java.io.Serializable;
import java.time.LocalDate;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.util.Date;

import com.fasterxml.jackson.annotation.JsonFormat;

public class Invite implements Serializable {
    private static final long serialVersionUID = 1L;

    private long id;
    @JsonFormat(shape = JsonFormat.Shape.STRING, pattern = "dd-MM-yyyy")
    private Date date;
    private String nom;
    private String prenom;
    private String email;

    public Invite() {
    }

    public long getId() {
        return this.id;
    }

    public void setId(long id) {
        this.id = id;
    }

    public Date getDate() {
        return this.date;
    }

    public void setDate(Date date) {
        this.date = date;
    }

    public String getNom() {
        return this.nom;
    }

    public void setNom(String nom) {
        this.nom = nom;
    }

    public String getPrenom() {
        return this.prenom;
    }

    public void setPrenom(String prenom) {
        this.prenom = prenom;
    }

    public String getEmail() {
        return this.email;
    }

    public void setEmail(String email) {
    }
}
```

```
        this.email = email;
    }

    public Invite(long id, String nom, String prenom, String email) {
        this.id = id;
        this.nom = nom;
        this.prenom = prenom;
        this.email = email;
        this.date = new Date();
    }

    // Méthode pour obtenir la date formatée
    public String getFormattedDate() {
        // Convertir java.util.Date en java.time.LocalDate
        LocalDate localDate =
this.date.toInstant().atZone(ZoneId.systemDefault()).toLocalDate();

        // Formater la date
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd-MMMM-yyyy");
        return localDate.format(formatter);
    }

    @Override
    public String toString() {
        // Formater la date directement dans la méthode toString()
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd-MMMM-yyyy");
        return nom + " " + prenom + " " +
formatter.format(date.toInstant().atZone(ZoneId.systemDefault()).toLocalDate());
    }
}
```

InviteController.java

```
package com.example.unc.miage;

import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.servlet.ModelAndView;

import java.time.format.DateTimeFormatter;
import java.util.Date;

import org.springframework.stereotype.Controller;

@Controller
public class InviteController {

    @RequestMapping("/")
    public String index() {
        return "index";
    }

    @RequestMapping(value="/save", method=RequestMethod.POST)
    public ModelAndView save(@ModelAttribute Invite invite) {
        ModelAndView modelAndView = new ModelAndView();
        // Initialiser la date de création
        invite.setDate(new Date());
    }
}
```



```
        modelAndView.setViewName("invite-data");
        modelAndView.addObject("invite", invite);
        return modelAndView;
    }
}
```

invite-data.html

```
<html xmlns:th="http://thymeleaf.org">
  <table>
    <tr>
      <td><h4>Nom :</h4></td>
      <td><h4 th:text="{invite.nom}"></h4></td>
    </tr>
    <tr>
      <td><h4>Prénom :</h4></td>
      <td><h4 th:text="{invite.prenom}"></h4></td>
    </tr>
    <tr>
      <td><h4>Email :</h4></td>
      <td><h4 th:text="{invite.email}"></h4></td>
    </tr>
    <tr>
      <td><h4>Date de création :</h4></td>
      <td><h4 th:text="{invite.formattedDate}"></h4></td>
    </tr>
  </table>
</html>
```



Monitorer un Webservice REST-JPA-MySQL

Nom du projet : technolog_SpringBootRest_Actuator

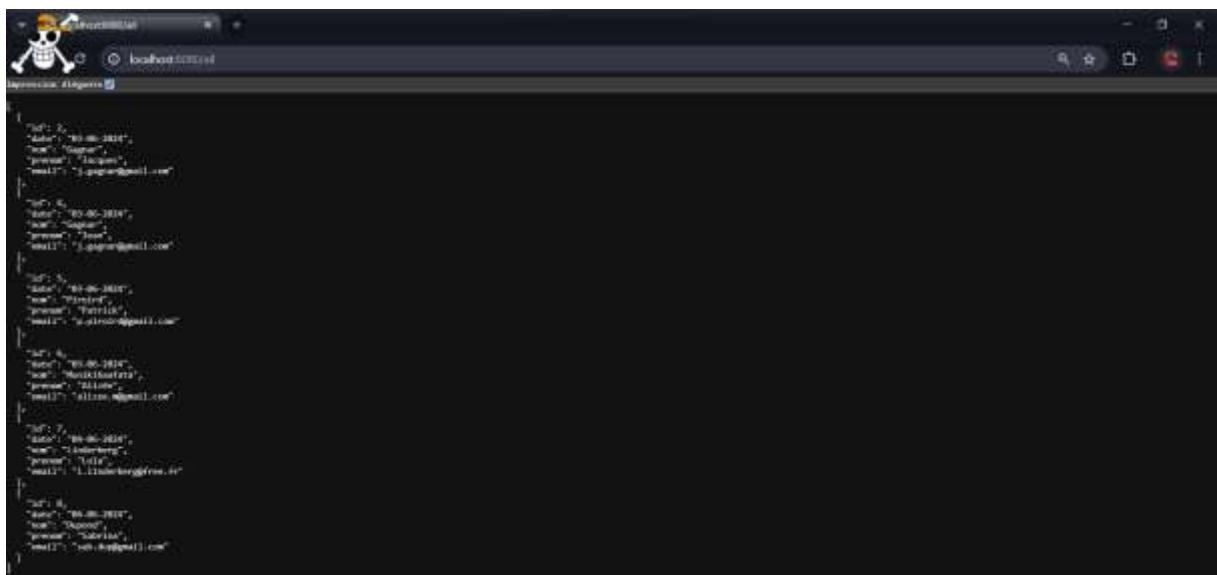


Reprendre le projet : technoLog_SpringBootRest_3_alizee en récupérant les fichiers :

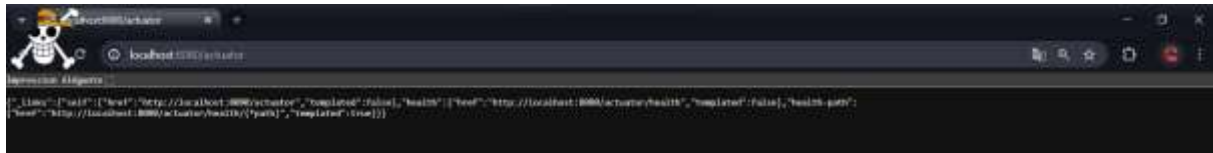
- Invite.java
- InviteRepository.java
- InviteController.java
- Application.properties

Vérifiez le fonctionnement de l'application avec l'URI

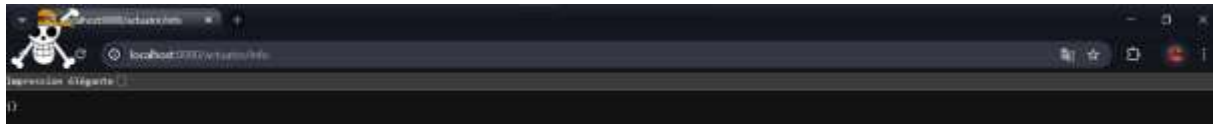
<http://localhost:8080/all>



<http://localhost:8080/actuator>



<http://localhost:8080/actuator/info>



Ajout d'informations sur l'application, le endpoint /info

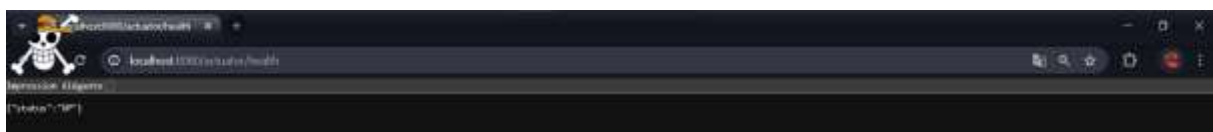
Ajout de ces lignes dans le fichier application.properties :

```
management.endpoints.web.exposure.include=health,info
management.info.env.enabled=true
info.app.name = @project.name@
info.app.groupId = @project.groupId@
info.app.artifactId = @project.artifactId@
info.app.version = @project.version@
info.app.programmeur=Jean Barre
info.app.info=source: Pierre La Penne
info.ihm=dev en cours
```



Configuration de base du service health, le endpoint /health

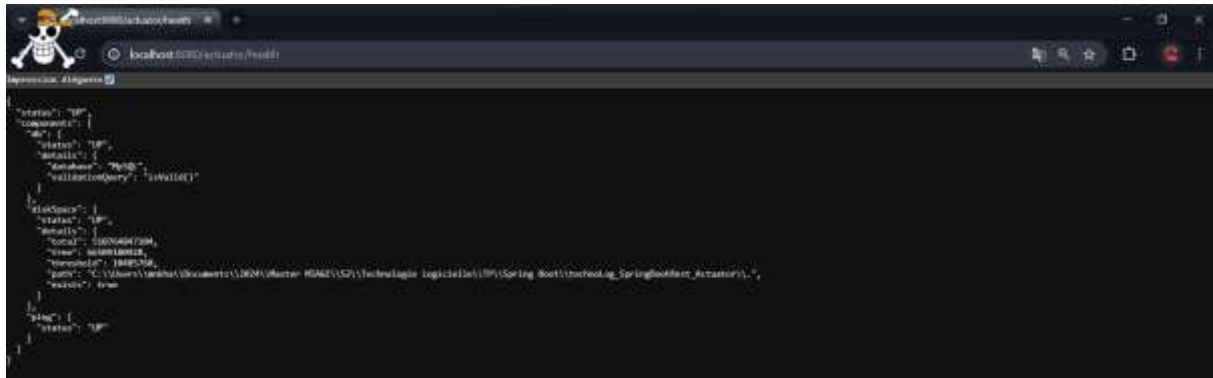
- Afficher le service health



- Ajout de cette ligne dans application.properties

```
management.endpoint.health.show-details=always
```

- Exécuter le end-point localhost:8080/actuator/health



Configuration d'un service health, le endpoint /health

- Ajout de lignes dans InviteRepository.java

```
@Query("select i from Invite i group by i.nom,i.prenom,i.email having count(*)>1")
Collection<Invite> rechercherDoublons() ;
```

Nouvelle classe HealthCheckBDD

```
package com.example.unc.miage;

import java.util.Collection;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.actuate.health.Health;
import org.springframework.boot.actuate.health.HealthIndicator;
import org.springframework.stereotype.Component;

@Component
public class HealthCheckBDD implements HealthIndicator {

    private final String message_key = "Base de données : ";
    @Autowired // Injection du repository
    private InviteRepository inviteRepository;

    @Override
    public Health health() {
        int errorCode = check();
        if (errorCode != 0) {
            return Health.down().withDetail(message_key, "Doublons détectés").build();
        }
        return Health.up().withDetail(message_key, "Pas de doublons détectés").build();
    }

    public int check() {
        Collection<Invite> doublons = inviteRepository.rechercherDoublons();
        if (doublons.size()==0)
            return 0;
        else return 1;
    }
}
```

Erreur au niveau de la requête

Explication de l'erreur

L'erreur survient parce que votre requête SQL contient des colonnes non agrégées (comme id) dans la liste SELECT qui ne sont pas incluses dans la clause GROUP BY. MySQL en mode `only_full_group_by` exige que toutes les colonnes non agrégées dans le SELECT apparaissent également dans la clause GROUP BY.

Solution

Il faut désactiver le mode “`only_full_group_by`” dans MySQL.

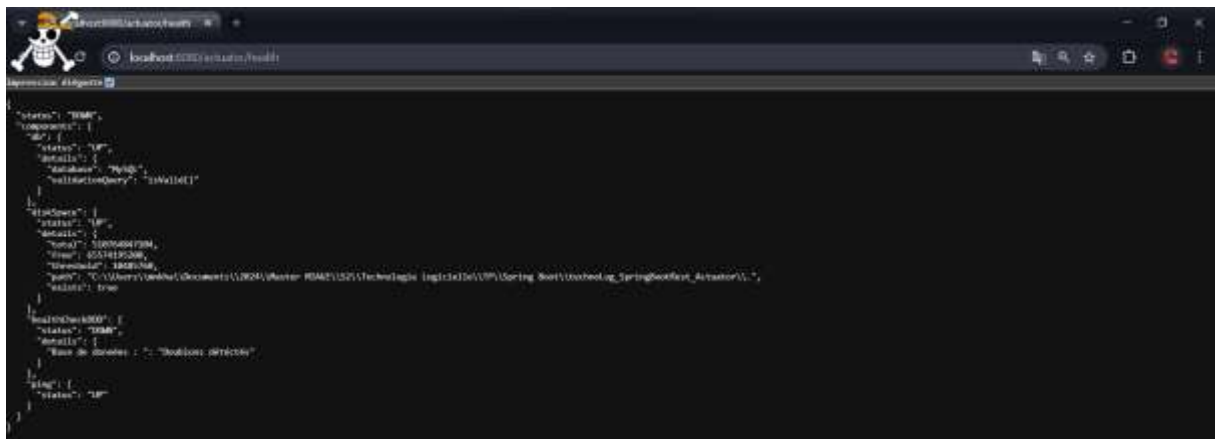
```
SET GLOBAL sql_mode=(SELECT REPLACE(@@GLOBAL.sql_mode, 'ONLY_FULL_GROUP_BY', ''));
```

- **Rafraichir la base de données MySQL**

J’ai modifié la requête sql pour qu’il cherche juste au niveau des noms sil y a des doublons :

```
@Query("SELECT i FROM Invite i GROUP BY i.nom HAVING COUNT(i) > 1")
```

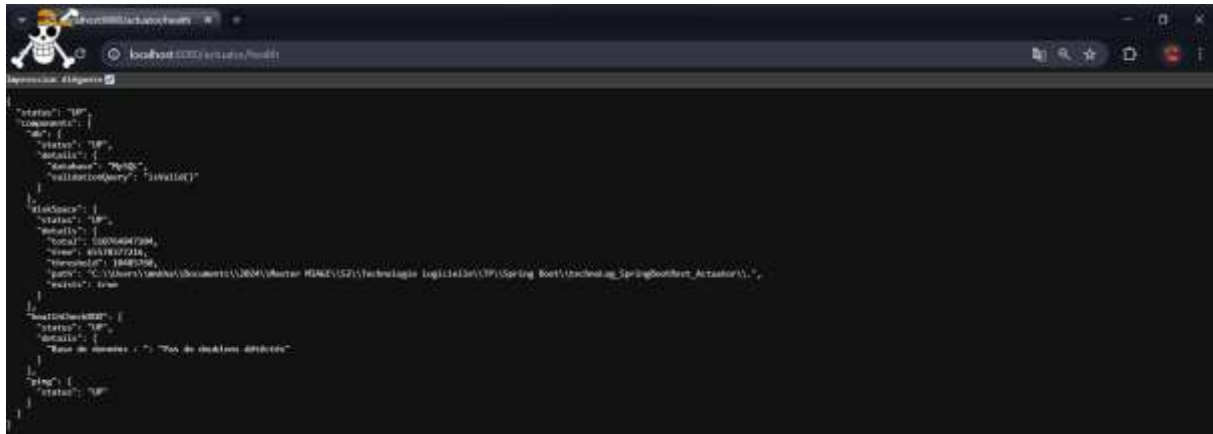
Etant donné que dans ma base de données j’ai deux fois le nom “Gagnar” voici le résultat obtenu :



Pour la requête du projet :

```
@Query("SELECT i FROM Invite i GROUP BY i.nom, i.prenom, i.email HAVING COUNT(i) > 1")
```

La requête ci-dessus donne pour réponse qu’il n’y a pas de doublons étant donné que l’on vérifie selon le nom, le prenom, et l’email. Nous pouvons remarquer que l’on n’a pas exactement la même ligne :



Création de la classe HealthCheckGetOneIndividu

```
package com.example.unc.miage;

import org.springframework.boot.actuate.health.Health;
import org.springframework.boot.actuate.health.HealthIndicator;
import org.springframework.stereotype.Component;
import org.springframework.web.client.RestTemplate;

@Component
public class HealthCheckGetOneIndividu implements HealthIndicator {

    @Override
    public Health health() {
        RestTemplate restTemplate = new RestTemplate();
        String message_key = "URL par défaut : ";
        Invite uninvite =
restTemplate.getForObject("http://localhost:8080/invite/invite", Invite.class);
        if(uninvite.getNom().equals("Palmer") &&
uninvite.getPrenom().equals("Jack")) {
            return Health.up().withDetail(message_key, "fonctionne
bien").build();
        }
        else
            return Health.up().withDetail(message_key, "erreur").build();
    }
}
```

```

{
  "status": "UP",
  "components": {
    "db": {
      "status": "UP",
      "details": {
        "database": "H2DB",
        "validationQuery": "SELECT 1"
      }
    },
    "diskSpace": {
      "status": "UP",
      "details": {
        "total": 1000000000,
        "free": 500000000,
        "threshold": 100000000,
        "path": "C:\\Users\\Admin\\Documents\\M2024\\Technologies Logicielles\\TP\\Spring Boot\\src\\test\\resources\\application.properties",
        "exists": true
      }
    },
    "healthCheck": {
      "status": "UP",
      "details": {
        "base de données : ": "Pas de doublon détecté"
      }
    }
  }
}

```

J’ai modifié le prénom par “Alizée” l’utilisateur par défaut ne porte pas ce prénom du coup une erreur est levée :

```

{
  "status": "UP",
  "components": {
    "db": {
      "status": "UP",
      "details": {
        "database": "H2DB",
        "validationQuery": "SELECT 1"
      }
    },
    "diskSpace": {
      "status": "UP",
      "details": {
        "total": 1000000000,
        "free": 500000000,
        "threshold": 100000000,
        "path": "C:\\Users\\Admin\\Documents\\M2024\\Technologies Logicielles\\TP\\Spring Boot\\src\\test\\resources\\application.properties",
        "exists": true
      }
    },
    "healthCheck": {
      "status": "UP",
      "details": {
        "base de données : ": "Pas de doublon détecté"
      }
    }
  }
}

```

- Pour exposer tous les endpoints :

Ajoutez cette ligne dans application.properties : `management.endpoints.web.exposure.include=*`

```

{
  "links": [
    {
      "rel": "self",
      "href": "http://localhost:8080/actuator",
      "templated": false,
      "media": "application/json",
      "cached": false,
      "cache-control": "no-cache"
    },
    {
      "rel": "http://localhost:8080/actuator/health",
      "templated": true,
      "cache-control": "no-cache",
      "media": "application/json"
    },
    {
      "rel": "http://localhost:8080/actuator/info",
      "templated": false,
      "condition": "true",
      "href": "http://localhost:8080/actuator/info",
      "templated": true,
      "cache-control": "no-cache"
    },
    {
      "rel": "http://localhost:8080/actuator/conditions",
      "templated": false,
      "condition": "true",
      "href": "http://localhost:8080/actuator/conditions",
      "templated": true,
      "cache-control": "no-cache"
    },
    {
      "rel": "http://localhost:8080/actuator/configprops",
      "templated": false,
      "condition": "true",
      "href": "http://localhost:8080/actuator/configprops",
      "templated": true,
      "cache-control": "no-cache"
    },
    {
      "rel": "http://localhost:8080/actuator/configprops/{name}",
      "templated": true,
      "cache-control": "no-cache"
    },
    {
      "rel": "http://localhost:8080/actuator/info",
      "templated": false,
      "condition": "true",
      "href": "http://localhost:8080/actuator/info",
      "templated": true,
      "cache-control": "no-cache"
    },
    {
      "rel": "http://localhost:8080/actuator/metrics",
      "templated": false,
      "condition": "true",
      "href": "http://localhost:8080/actuator/metrics",
      "templated": true,
      "cache-control": "no-cache"
    },
    {
      "rel": "http://localhost:8080/actuator/metrics/{name}",
      "templated": true,
      "cache-control": "no-cache"
    },
    {
      "rel": "http://localhost:8080/actuator/show",
      "templated": false,
      "condition": "true",
      "href": "http://localhost:8080/actuator/show",
      "templated": true,
      "cache-control": "no-cache"
    },
    {
      "rel": "http://localhost:8080/actuator/show/{id}",
      "templated": true,
      "cache-control": "no-cache"
    },
    {
      "rel": "http://localhost:8080/actuator/show/{id}/metrics",
      "templated": true,
      "cache-control": "no-cache"
    },
    {
      "rel": "http://localhost:8080/actuator/show/{id}/metrics/{name}",
      "templated": true,
      "cache-control": "no-cache"
    }
  ]
}

```

On peut limiter aux endpoint voulus en modifiant l’instruction précédente dans le fichier application.properties pour exposer seulement, par exemple, les endpoint health, info et metrics: `management.endpoints.web.exposure.include=health,info,metrics`.



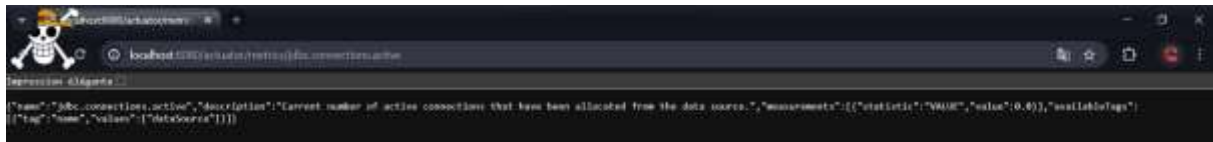
Le endpoint /metrics

Lien URL : <http://localhost:8080/actuator/metrics>



On peut visualiser le endpoint /metrics/jdbc.connections.active :

<http://localhost:8080/actuator/metrics/jdbc.connections.active>



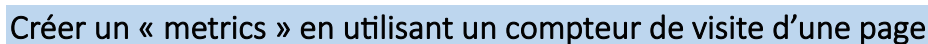
Le endpoint /env

<http://localhost:8080/actuator/env>



Le endpoint /beans

<http://localhost:8080/actuator/beans>



- ```
package com.example.unc.miage;

import java.net.URI;
import java.sql.Date;
import java.time.Instant;
import java.util.ArrayList;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.servlet.support.ServletUriComponentsBuilder;
import io.micrometer.core.instrument.Counter;
import io.micrometer.core.instrument.MeterRegistry;

@RestController
public class InvitesController {

 @Autowired
 private InviteRepository inviteRepository;

 Counter compteur;

 public InvitesController(MeterRegistry registry) {
```

```
 compteur = registry.counter("access.all");
 }

 @RequestMapping("/default")
 public Invite getDefault() {
 return new Invite("Palmer", "Jack", "sav.spring@gmail.com");
 }

 @RequestMapping("/all")
 ArrayList<Invite> getAll(){
 compteur.increment();
 return (ArrayList<Invite>) inviteRepository.findAll();
 }

 @GetMapping("/{invite}") // Map requêtes GET
 public ResponseEntity<Invite> rechercheInviteParGetInvite
 (@PathVariable("invite") String nom) {
 ArrayList<Invite> liste = inviteRepository.findByNom(nom);
 if (liste.size()==0)
 return ResponseEntity.notFound().build();
 else
 return ResponseEntity.ok(liste.get(0));
 }

 @GetMapping
 public ResponseEntity<Invite> rechercheInviteParGet(@RequestParam String
nom, @RequestParam String prenom){
 ArrayList<Invite> liste = inviteRepository.findByNomAndPrenom(nom,
prenom);

 if(liste.size()==0)
 return ResponseEntity.notFound().build();
 else
 return ResponseEntity.ok(liste.get(0));
 }

 @PostMapping(path="/ajoute")
 public ResponseEntity<String> ajouteInvite (@RequestParam String nom,
 @RequestParam String prenom, @RequestParam String email){
 Invite r = inviteRepository.findByNomAndPrenomAndEmail(nom, prenom,
email);

 if(r != null)
 return ResponseEntity.noContent().build();
 Invite i = new Invite(nom, prenom, email);
 i = inviteRepository.save(i);
 URI location = ServletUriComponentsBuilder
 .fromCurrentRequest()
 .path("/{id}")
 .buildAndExpand(i.getId())
 .toUri();

 HttpHeaders responseHeaders = new HttpHeaders();
 responseHeaders.setLocation(location);
 return new ResponseEntity<String>(prenom+ " " + nom + " crée.",
responseHeaders,HttpStatus.CREATED);
 }

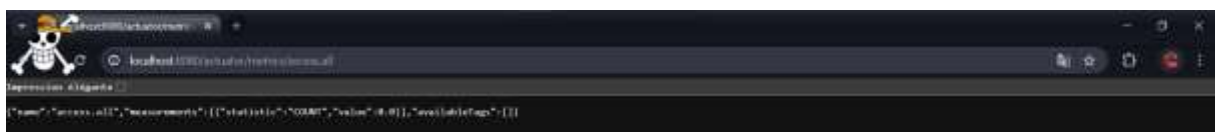
 // ajoute un invite, l'objet est reçu au format json
 @PostMapping(value = "/ajoutejson")
 public ResponseEntity<Void> ajouterInvite(@RequestBody Invite invite) {
```

```
 Invite r =
inviteRepository.findByNomAndPrenomAndEmail(invite.getNom(), invite.getPrenom(),
invite.getEmail());
 if (r != null)
 return ResponseEntity.noContent().build();
 invite.setDate(Date.from(Instant.now()));
 Invite inviteAjoute = inviteRepository.save(invite);
 URI location = ServletUriComponentsBuilder
 .fromCurrentRequest().path("/{id}")
 .buildAndExpand(inviteAjoute.getId()).toUri();
 return ResponseEntity.created(location).build();
 }

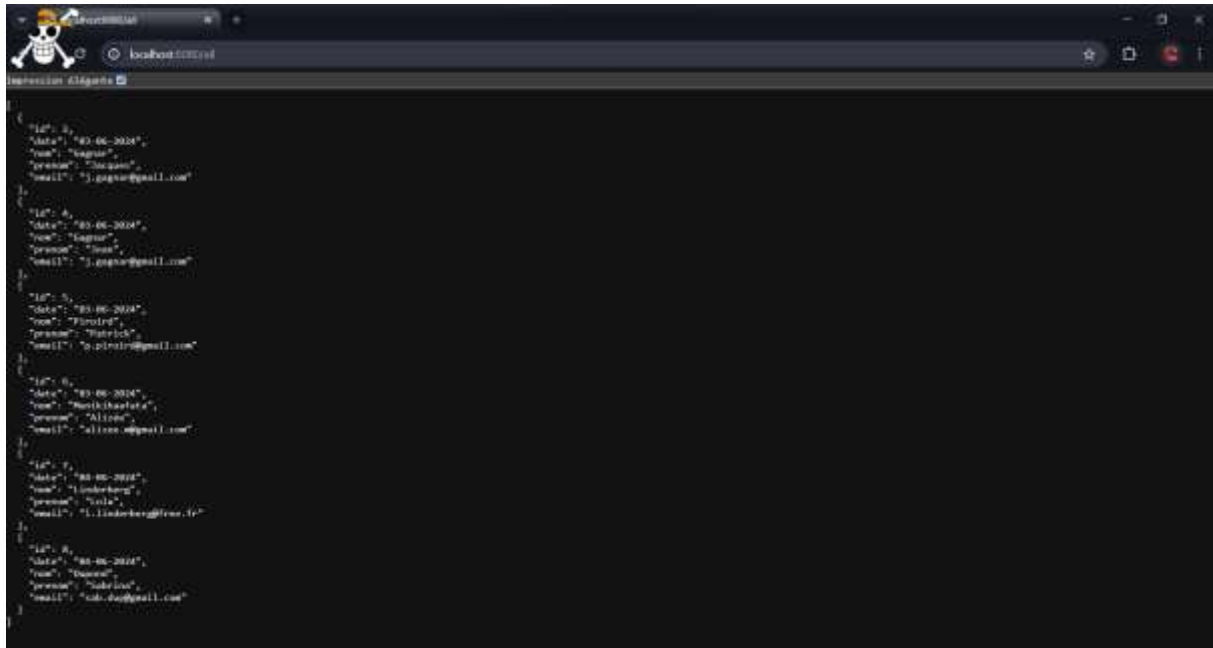
 @DeleteMapping
 public ResponseEntity<Void> supprimerInvite(@RequestParam String nom,
 @RequestParam String prenom){
 ArrayList<Invite> liste = inviteRepository.findByNomAndPrenom(nom,
 prenom);
 if(liste.size()==0)
 return ResponseEntity.notFound().build();
 else {
 inviteRepository.delete(liste.get(0));
 return ResponseEntity.status(HttpStatus.NO_CONTENT).build();
 }
 }

 @PutMapping(path="/modifier")
 public ResponseEntity<Void> modifierEmailInvite(@RequestParam String nom,
 @RequestParam String prenom, @RequestParam String email){
 ArrayList<Invite> liste = inviteRepository.findByNomAndPrenom(nom,
 prenom);
 if(liste.size() !=0) {
 Invite i = liste.get(0);
 i.setEmail(email);
 inviteRepository.save(i);
 return ResponseEntity.ok().build();
 }
 Invite i = new Invite(nom, prenom, email);
 i = inviteRepository.save(i);
 URI location = ServletUriComponentsBuilder
 .fromCurrentContextPath()
 .path("/{id}")
 .buildAndExpand(i.getId())
 .toUri();
 return ResponseEntity.created(location).build();
 }
}
```

<http://localhost:8080/actuator/metrics/access.all>

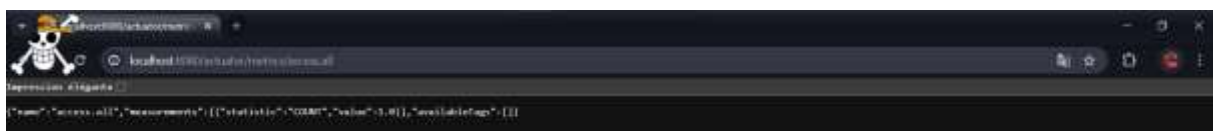


<http://localhost:8080/all>



```
{
 "id": 1,
 "date": "20-06-2024",
 "nom": "Jaguar",
 "prenom": "Jacques",
 "email": "j.jaguar@gmail.com"
},
{
 "id": 2,
 "date": "20-06-2024",
 "nom": "Jaguar",
 "prenom": "Jean",
 "email": "j.jaguar@gmail.com"
},
{
 "id": 3,
 "date": "20-06-2024",
 "nom": "Fouled",
 "prenom": "Patrick",
 "email": "p.pouled@gmail.com"
},
{
 "id": 4,
 "date": "20-06-2024",
 "nom": "Munkihafata",
 "prenom": "Alizee",
 "email": "alizee.m@gmail.com"
},
{
 "id": 5,
 "date": "20-06-2024",
 "nom": "Lindenberg",
 "prenom": "Sila",
 "email": "l.lindenberg@free.fr"
},
{
 "id": 6,
 "date": "20-06-2024",
 "nom": "Ducard",
 "prenom": "Cubius",
 "email": "cub.ducard@gmail.com"
}
}
```

<http://localhost:8080/actuator/metrics/access.all>



```
{
 "name": "access.all",
 "measurements": [
 {
 "statistic": "COUNT",
 "value": 1.0,
 "availableTag": []
 }
]
}
```

Et le compteur va augmenter en fonction de l'utilisation de /all

## Création du jar exécutable : le microservice

### Génération du jar

- **Clic droit sur le projet > Run As > Run Configurations > taper Maven dans la barre de recherches > Maven Build**

Etant donné que j'utilise Gradle

- **On modifie le fichier build.gradle :**

```
jar {
 manifest {
 attributes 'Main-Class':
 'com.example.unc.miage.TechnoLogSpringBootRestActuatorApplication'
 }
}
```

- **Ajout de cette ligne dans les dépendances :**

```
implementation 'org.springframework.boot:spring-boot-starter'
```

Problème au niveau de ma version Java. Il faut une version = ou supérieure à 17

Modification des variables d'environnement

