TP Projet « Foyer »

UP ASI
Bureau E204

Diagramme de classes

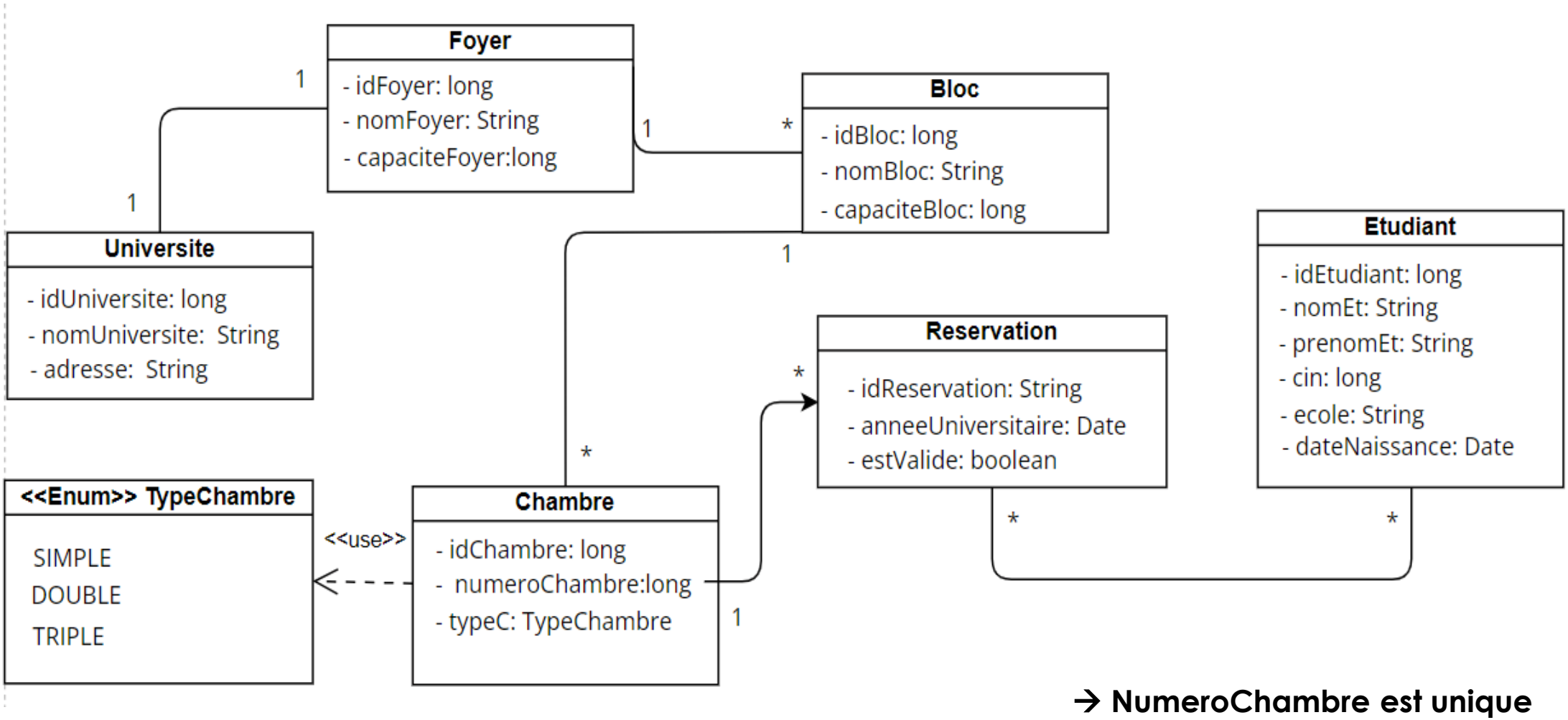


Figure 1 : Diagramme de classes

Travail à faire

Partie 1 Spring Data JPA – Première entité

- Créer les entités se trouvant dans le diagramme des classes (sans les associations) et vérifier qu'ils ont été ajoutés avec succès dans la base de données.

Partie 2 Spring Data JPA – Le mapping des différentes associations

- Supprimer les tables existantes dans la base de données.
- Créer les associations entre les différentes entités.
- Générer la base de données de nouveau et vérifier que le nombre de tables créées est correct.

NB:

- **Dans l'association Universite – Foyer → Université est le parent**
- **Dans l'association Reservation – Etudiant → Réservation est le parent**
- **Initialiser toutes les instances « List ou Set » créées pour créer l'association**

Travail à faire

Partie 3 Spring Data JPA – Crud

- Créer les CRUD des différentes **entités.**

NB:

- Pour l'ajout l'**université**, il faut créer en même temps son **foyer**.
- Pour la suppression de l'**université**, il faut supprimer en même temps son **foyer**.

➡ **Indication:** Il faut utiliser le paramètre « Cascade »

- Pour l'ajout d'un **bloc**, il faut créer en même temps ses **chambres**.
- Pour la suppression d'un **bloc**, il faut supprimer en même temps ses **chambres**.

Cascade

- Le comportement en cascade consiste à compléter ce qui se passe pour une entité en relation avec une entité père, lorsque cette entité père subit une des opérations suivantes (Ajout, modification, suppression, ...)
- Prenant l'exemple suivant :

- la relation Personne-Adresse. Sans la personne, l'entité Adresse n'a pas de signification propre. Lorsque nous supprimons l'entité Personne, notre entité Adresse doit également être supprimée.
- La cascade est le moyen d'y parvenir. Lorsque nous effectuons une action sur l'entité cible(Master), la même action sera appliquée à l'entité associée (Slave).

Cascade

- Toutes les opérations en cascade spécifiques à JPA sont représentées par l'énumération **javax.persistence.CascadeType** contenant les entrées:

→ **ALL** → **PERSIST** → MERGE → **REMOVE** → REFRESH →DETACH

- Le comportement cascade est précisé par l'attribut cascade, disponible sur les annotations : **@OneToOne**, **@OneToMany** et **@ManyToMany**

Travail à faire

POST `/bloc/addOrUpdate`

Parameters

No parameters

Request body required

```
{
  "nomBloc": "Bloc A",
  "chambres": [
    {
      "numeroChambre": 1,
      "typeC": "SIMPLE"
    },
    {
      "numeroChambre": 2,
      "typeC": "DOUBLE"
    }
  ]
}
```

Execute

Server response

Code	Details
200	Response body

```
{
  "idBloc": 1,
  "nomBloc": "Bloc A",
  "capaciteBloc": 0,
  "foyer": null,
  "chambres": [
    {
      "idChambre": 1,
      "numeroChambre": 1,
      "typeC": "SIMPLE",
      "bloc": null,
      "reservations": null
    },
    {
      "idChambre": 2,
      "numeroChambre": 2,
      "typeC": "DOUBLE",
      "bloc": null,
      "reservations": null
    }
  ]
}
```

Travail à faire

POST /universite/addOrUpdate

Parameters

No parameters

Request body required

```
{
  "nomUniversite": "Sesame",
  "adresse": "Chotrana - Ariana",
  "foyer": {
    "nomFoyer": "Foyer les filles",
    "capaciteFoyer": 400
  }
}
```

Execute

Server response

Code Details

200

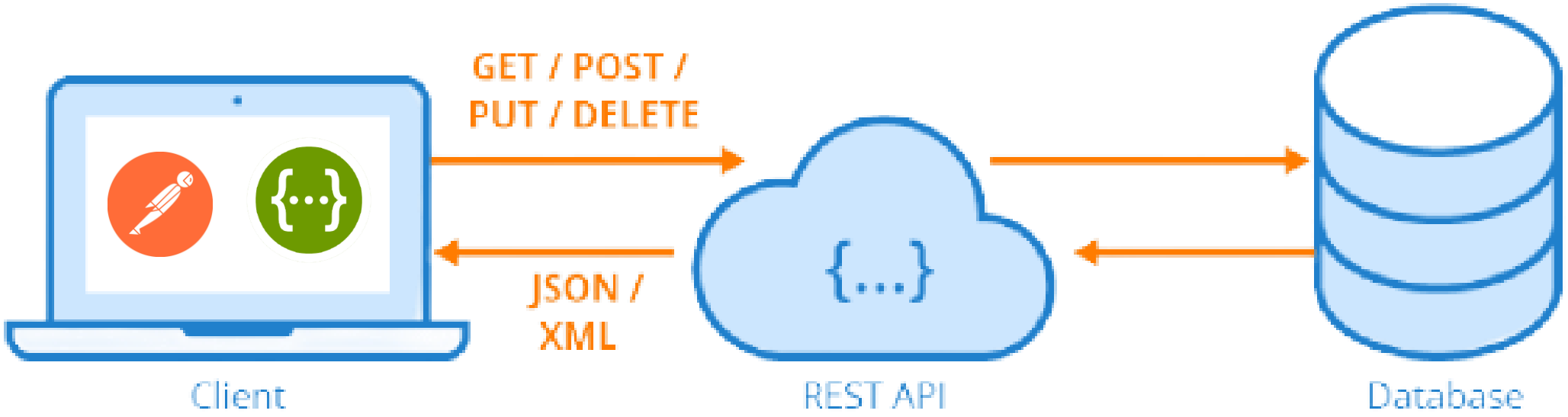
Response body

```
{
  "idUniversite": 2,
  "nomUniversite": "Sesame",
  "adresse": "Chotrana - Ariana",
  "foyer": {
    "idFoyer": 2,
    "nomFoyer": "Foyer les filles",
    "capaciteFoyer": 400,
    "universite": null,
    "blocs": null
  }
}
```

Travail à faire

Partie 4 Spring MVC

- Exposer les services implémentés dans la partie 3 avec Postman et/ou Swagger pour les tester.



Travail à faire

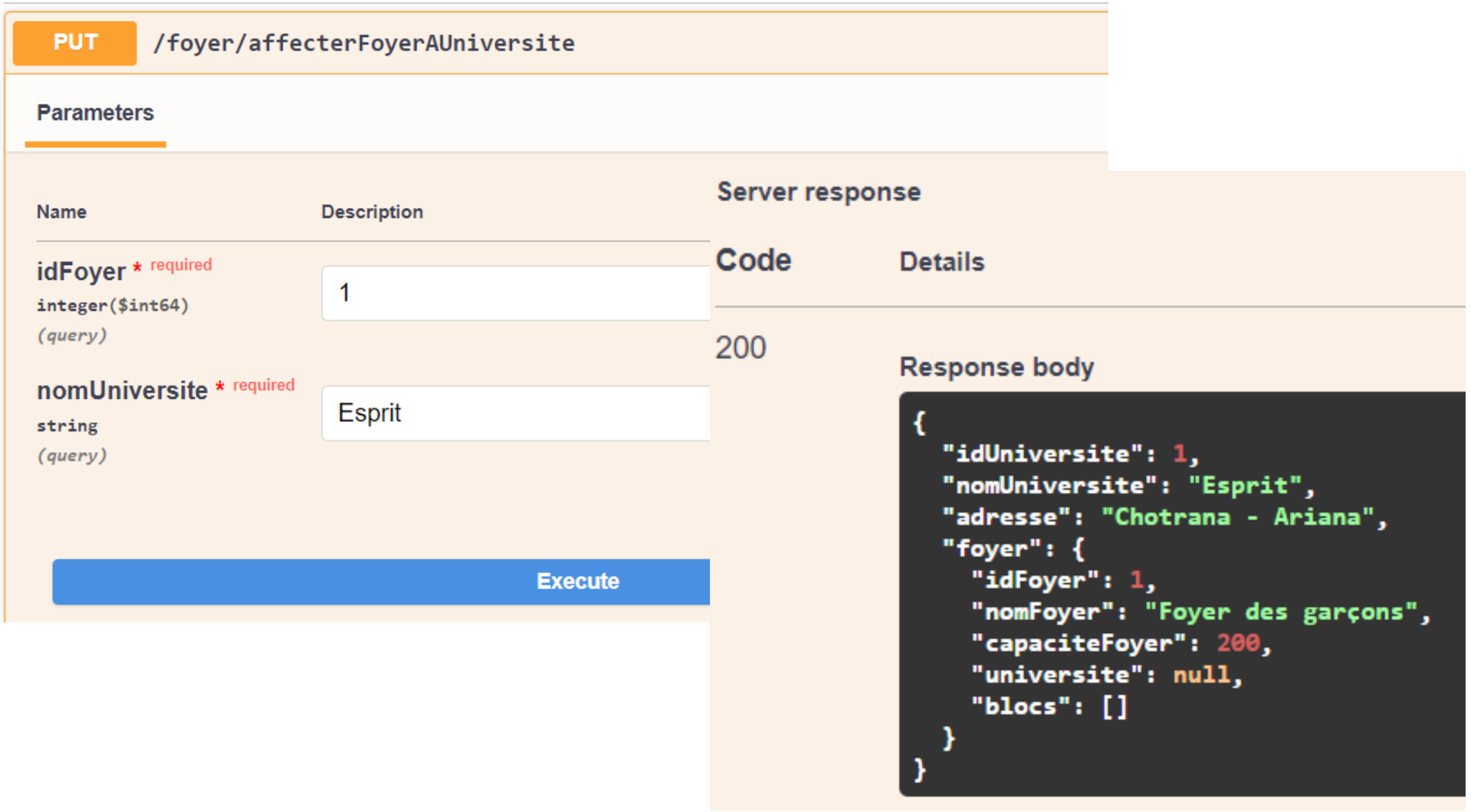
Partie 5 Services avancés

- **Service 01:** On désire affecter un Foyer à une Université.

  → Créer un service permettant l'affectation d'un Foyer à une Université et exposer le en respectant la signature suivante :

Universite affecterFoyerAUniversite (long idFoyer, String nomUniversite) ;

Travail à faire



Travail à faire

Partie 5 Services avancés

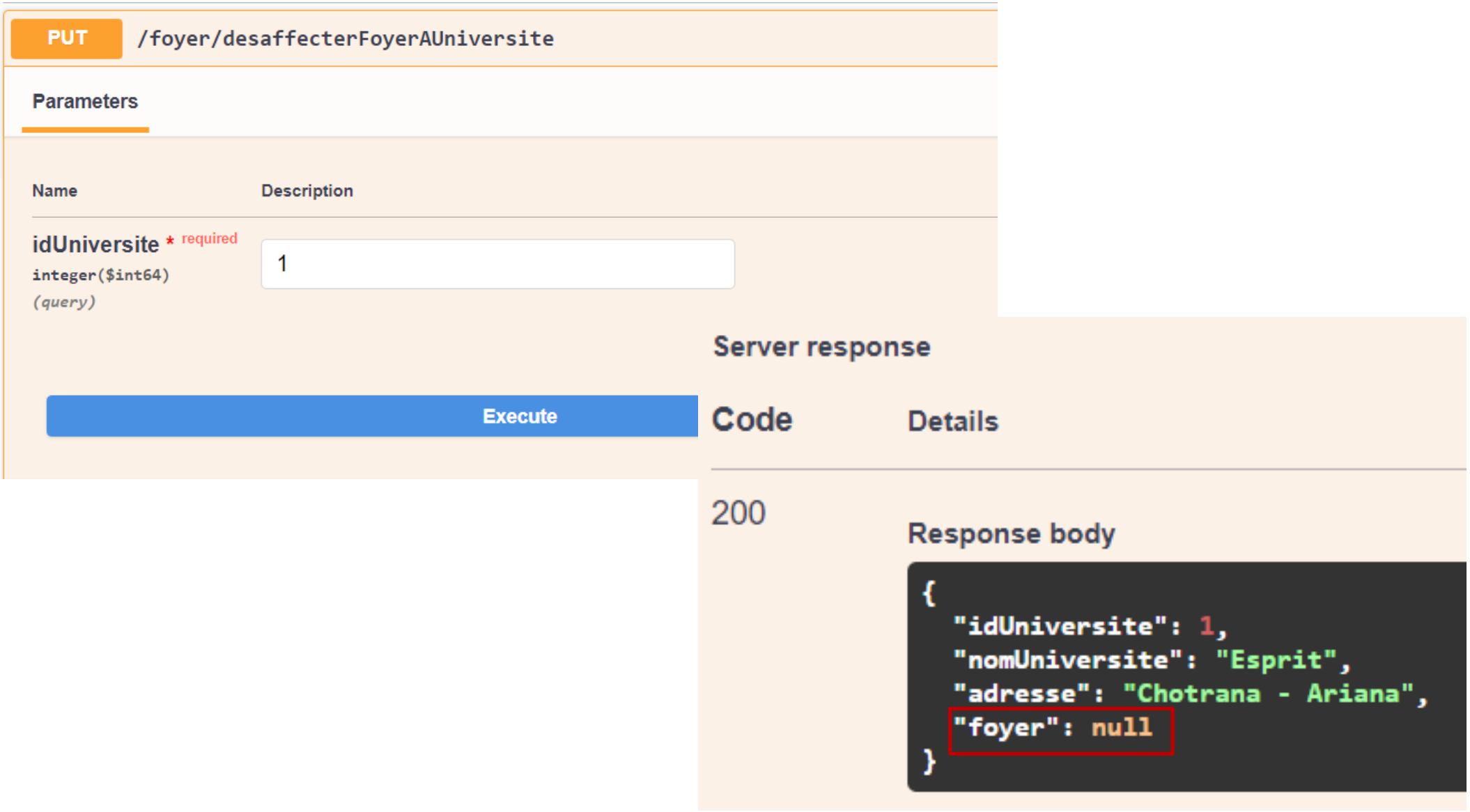
- **Service 02:** On désire désaffecter un Foyer à une Université.

  → Créer un service permettant la désaffectation d'un Foyer à une Université et exposer le en respectant la signature suivante :

  Universite desaffecterFoyerAUniversite (long idUniversite) ;

  NB: Université est le parent dans l'association entre université et Foyer

Travail à faire



Travail à faire

Partie 5 Services avancés

- **Service 03:** On désire affecter des Chambres à un Bloc.

  → Créer un service permettant l'affectation des Chambres à un Bloc et exposer le en respectant la signature suivante :

Bloc affecterChambresABloc(List<long> numChambre, String nomBloc) ;

Travail à faire

PUT `/bloc/affecterChambresABloc`

Parameters

Name	Description
nomBloc * required string *(query)*	Bloc A

Request body required

```
[
  3,4
]
```

Server response

Code	Details
200	Response body

```
{
  "idBloc": 1,
  "nomBloc": "Bloc A",
  "capaciteBloc": 0,
  "chambres": [
    {
      "idChambre": 3,
      "numeroChambre": 3,
      "typeC": "SIMPLE",
      "reservations": []
    },
    {
      "idChambre": 4,
      "numeroChambre": 4,
      "typeC": "SIMPLE",
      "reservations": []
    }
  ]
}
```

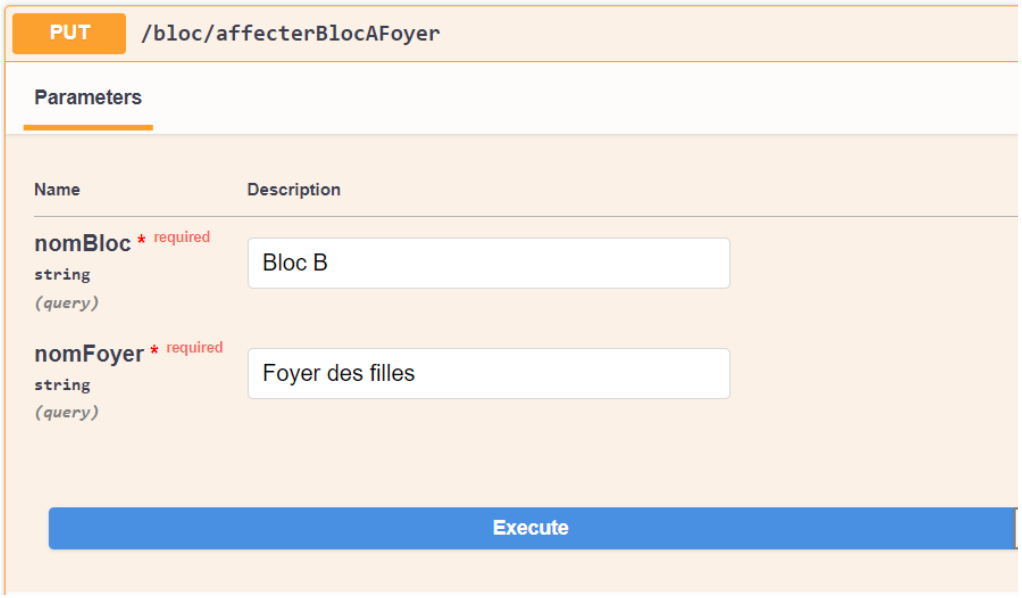
Travail à faire

Partie 5 Services avancés

- **Service 04:** On désire affecter un Bloc à un Foyer.

  → Créer un service permettant l'affectation d'un Bloc à un Foyer et exposer le en respectant la signature suivante :

  Bloc affecterBlocAFoyer(String nomBloc, String nomFoyer) ;



Travail à faire

Partie 5 Services avancés

- **Service 05:** On désire ajouter une réservation et l'assigner à un Étudiant et à une Chambre **(Indication Diapo 26)**.

  → Créer un service permettant l'ajout d'une réservation et l'assigner à une Chambre et un étudiant et exposer le en respectant la signature suivante :

Reservation ajouterReservationEtAssignerAChambreEtAEtudiant (Long numChambre, String cin) ;

NB: Pour l'ajout de Réservation, merci de prendre en considération les conditions ci-dessous:

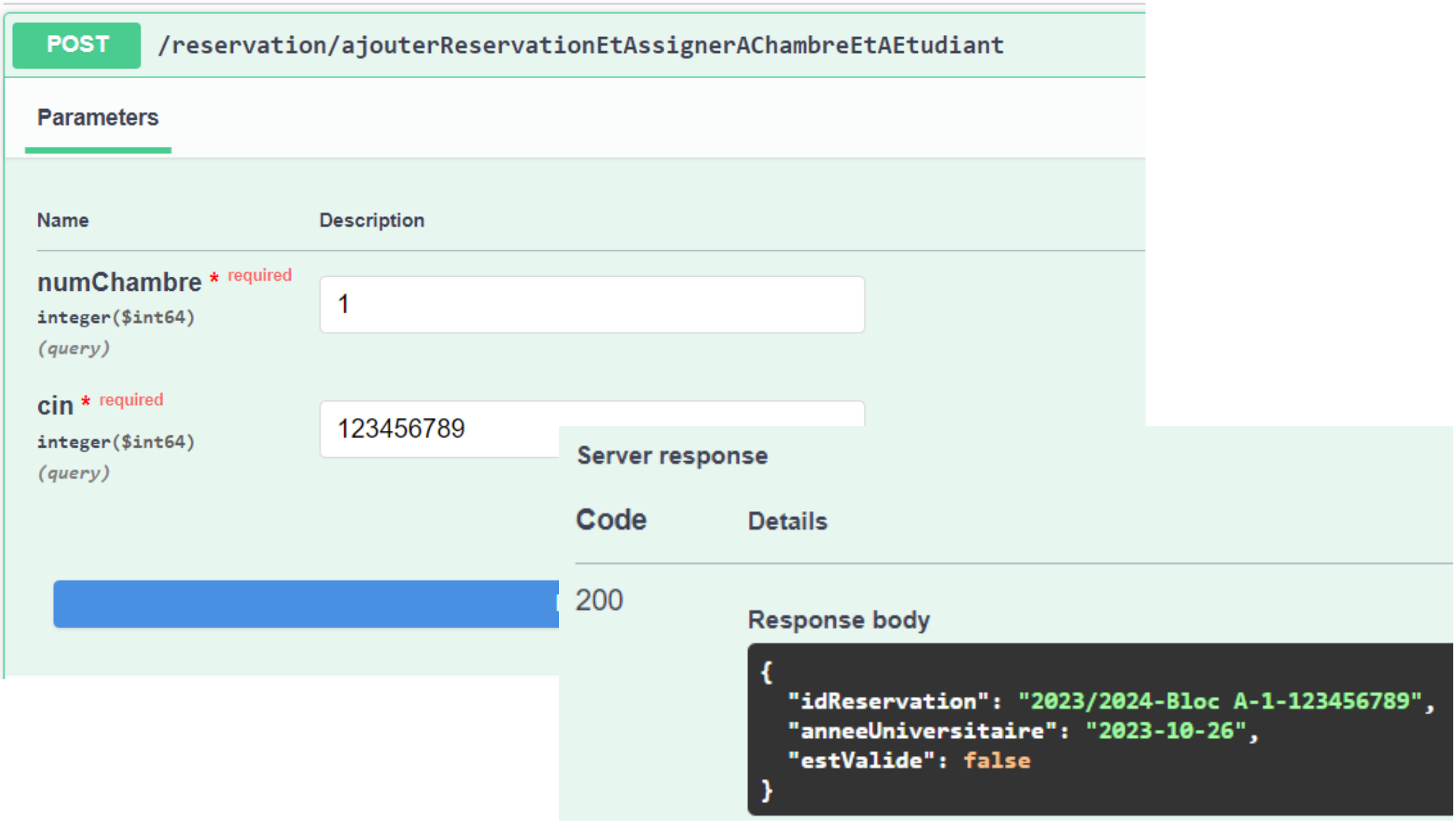
- L'id de la réservation doit respecte le format suivant:

« AnnéeUniversitaire » – « NomBloc » – « NumChambre » – « CIN »

(Exemple: 2023/2024-Bloc A-1-123456789)

- L'ajout ne se fait que si la capacite maximale de la chambre est encore non atteinte.
- La date de la réservation est égale à la date système et estValide prend par défaut la valeur true

Travail à faire



Travail à faire

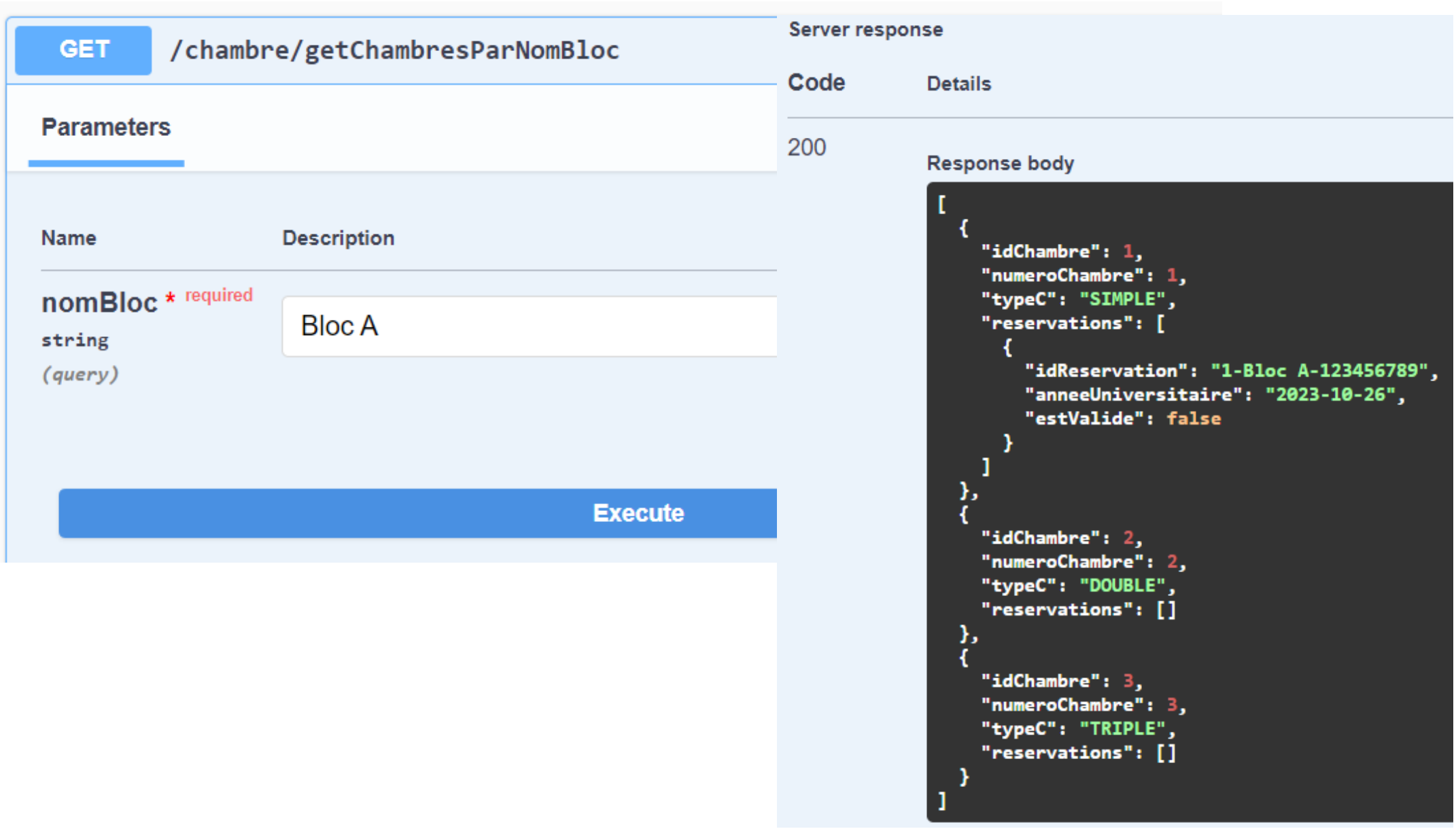
Partie 5 Services avancés

- **Service 06:** On désire afficher la liste des chambres d'un bloc donné.

  → Créer un service permettant l'affichage des chambres appartenant à un bloc donné par son nom et exposer le en respectant la signature suivante :

  List<Chambre> getChambresParNomBloc(String nomBloc) ;

Travail à faire



Travail à faire

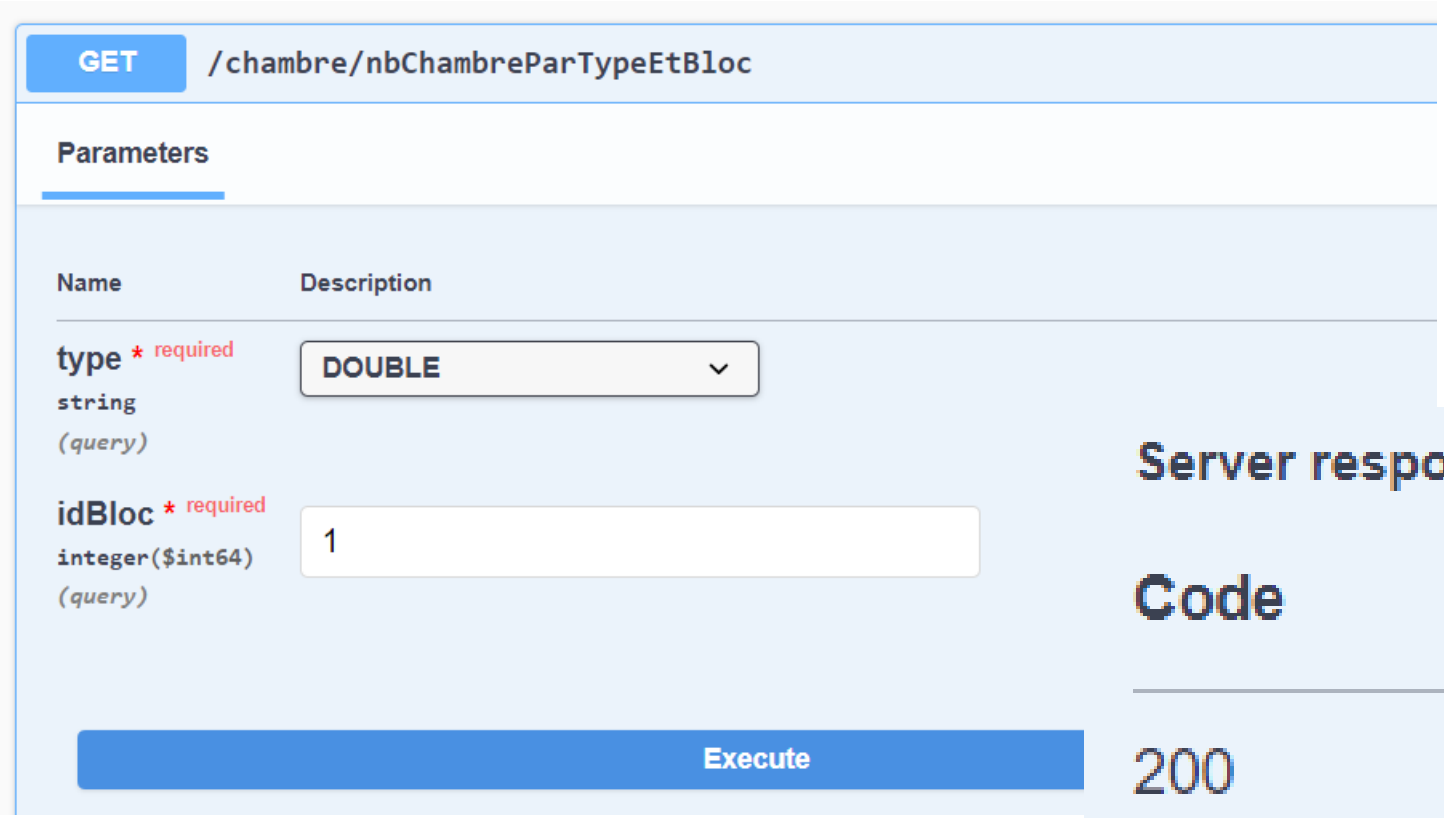
Partie 5 Services avancés

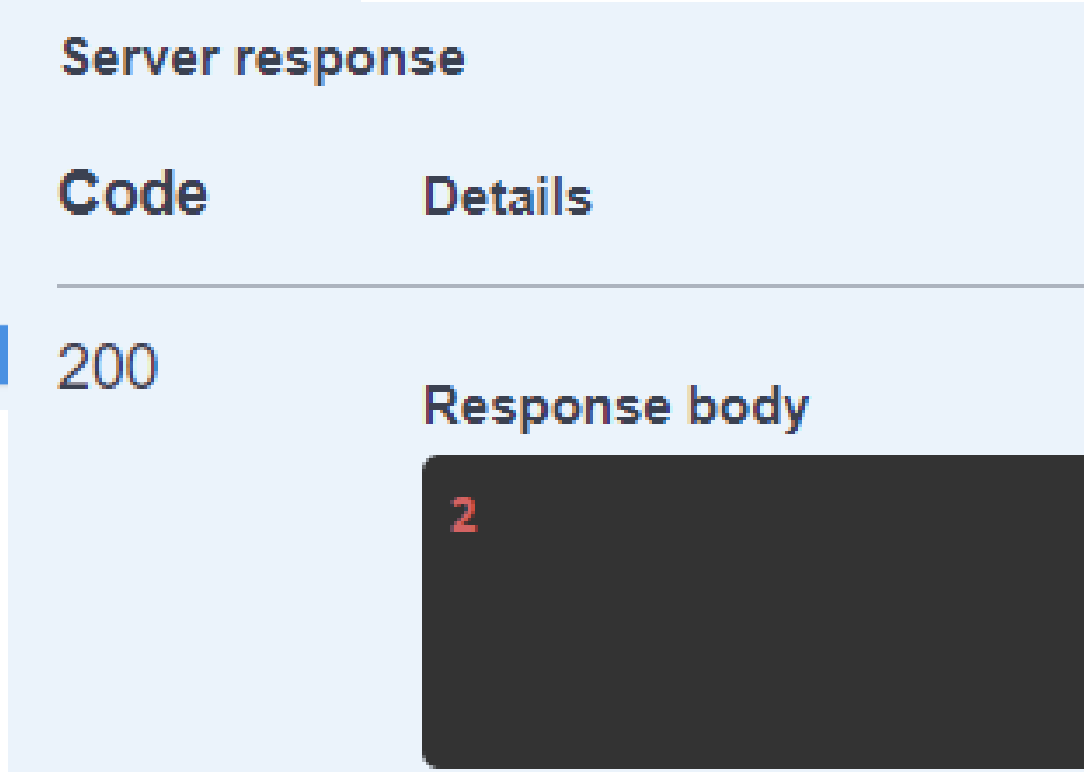
- **Service 07:** On désire afficher le nombre des chambres de typeChambre donné et appartenant à un bloc donné.

  → Créer un service permettant l'affichage le nombre des chambres de typeChambre donné et appartenant à un bloc donné par son id et exposer le en respectant la signature suivante :

  long nbChambreParTypeEtBloc(TypeChambre type, long idBloc) ;

Travail à faire





Travail à faire

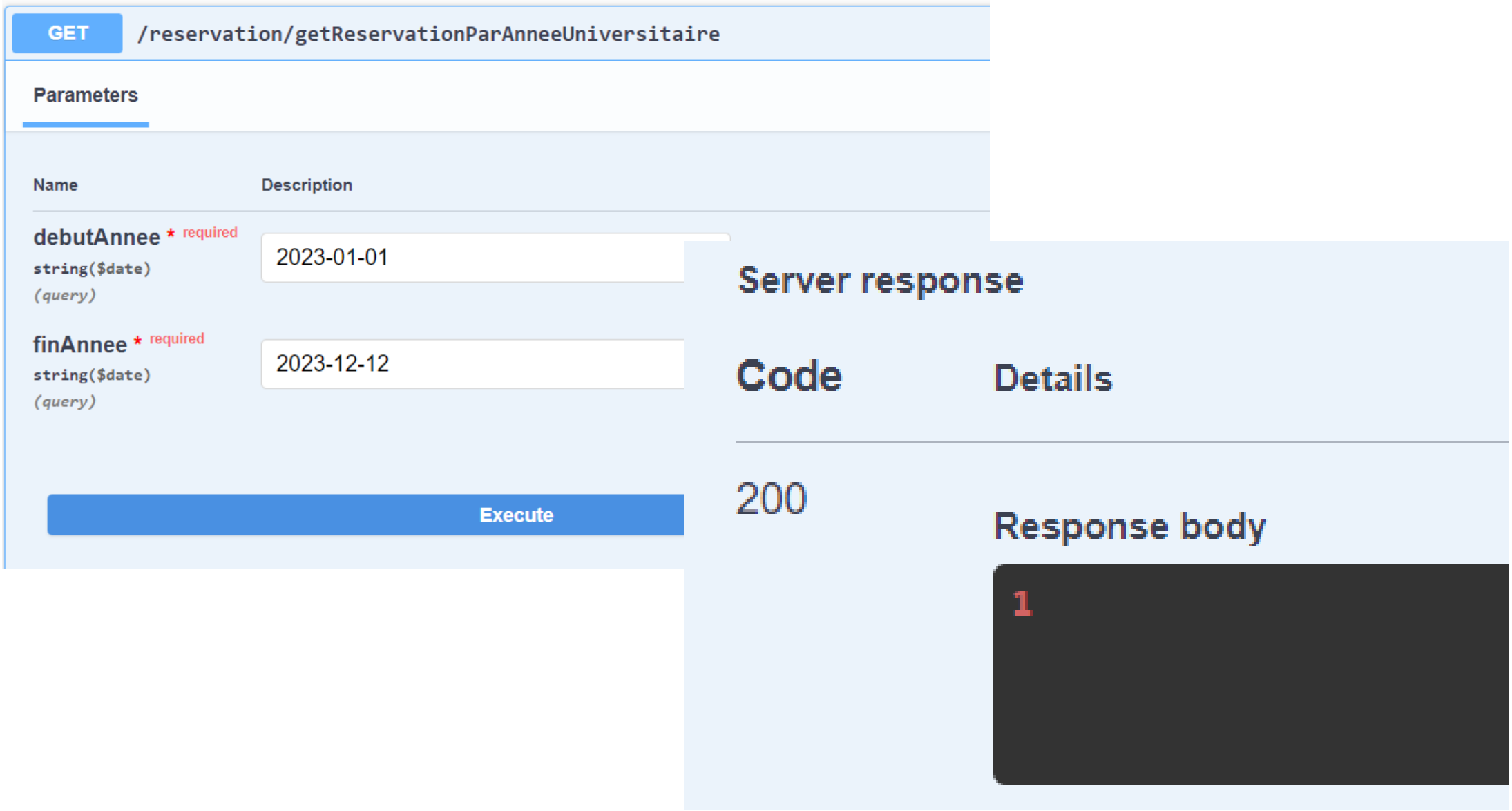
Partie 5 Services avancés

- **Service 08:** On désire afficher les réservations effectuées lors d'une année universitaire donné.

  → Créer un service permettant l'affichage des réservations effectuées lors d'une année universitaire donné et exposer le en respectant la signature suivante :

long getReservationParAnneeUniversitaire(LocalDate debutAnnee, LocalDate finAnnee) ;

Travail à faire



Travail à faire

Partie 5 Services avancés

- **Service 09:** On désire afficher les chambres non réservée, par typeChambre, appartenant à un foyer donné par son nom, effectué durant l'année universitaire actuelle **(Indication Diapo 26)**.

  → Créer un service permettant l'affichage des chambres non réservées ,par typeChambre , appartenant à un foyer donné par son nom, effectué durant l'année universitaire actuelle et exposer le en respectant la signature suivante :

List<Chambre> getChambresNonReserveParNomFoyerEtTypeChambre(String nomFoyer,TypeChambre type) ;

Travail à faire

➡ **Indication:** Pour récupérer l'année universitaire actuelle dynamiquement (On considère que l'AU commence à 15/09 et se termine à 30/06):

```
LocalDate dateDebutAU;
LocalDate dateFinAU;
int numReservation;
int year = LocalDate.now().getYear() % 100;
if (LocalDate.now().getMonthValue() <= 7) {
   dateDebutAU = LocalDate.of(Integer.parseInt("20" + (year - 1)), 9, 15);
   dateFinAU = LocalDate.of(Integer.parseInt("20" + year), 6, 30);
} else {
   dateDebutAU = LocalDate.of(Integer.parseInt("20" + year), 9, 15);
   dateFinAU = LocalDate.of(Integer.parseInt("20" + (year + 1)), 6, 30);
}
```

Travail à faire

Partie 5 Services avancés

- **Service 10:** On désire ajouter à la fois un Foyer, ses blocs associés et l'affecter à une université donnée.

  → Créer le service adéquat et exposer le en respectant la signature suivante:

Foyer ajouterFoyerEtAffecterAUniversite (Foyer foyer, long idUniversite) ;

NB: Pour l'ajout, merci de prendre en considération la condition ci-dessous:

- Il faut créer en même temps la liste des blocs (l'entité associée Bloc au Foyer) tout en assurant les affectations nécessaires.

Travail à faire

POST `/foyer/ajouterFoyerEtAffecterAUniversite`

Parameters

Name	Description
idUniversite * required integer($int64) *(query)*	1

Request body required

```
{
  "nomFoyer": "Foyer interne",
  "capaciteFoyer": 150,
  "blocs": [
    {
      "nomBloc": "A",
      "capaciteBloc": 30
    },
    {
      "nomBloc": "B",
      "capaciteBloc": 30
    }
  ]
}
```

Server response

Code	Details
200	**Response body**

```
{
  "idFoyer": 2,
  "nomFoyer": "Foyer interne",
  "capaciteFoyer": 150,
  "universite": null,
  "blocs": [
    {
      "idBloc": 1,
      "nomBloc": "A",
      "capaciteBloc": 30,
      "chambres": []
    },
    {
      "idBloc": 2,
      "nomBloc": "B",
      "capaciteBloc": 30,
      "chambres": []
    }
  ]
}
```

Travail à faire

Partie 5 Services avancés

- **Service 11:** On désire annuler une réservation active d'un étudiant donné.

  → Créer le service adéquat et exposer le en respectant la signature suivante:

Reservation annulerReservation (long cinEtudiant) ;

NB: L'annulation s'effectue comme suit:

- Désaffecter la chambre associée.
- Supprimer la réservation.

Travail à faire

Partie 6 Spring Schedular

- Réaliser les trois services des trois slides suivants:

  → **Service 01:** Créer un service qui se déclenche **toutes les minutes** permettant d'afficher la liste des chambres du bloc en respectant la signature et le format de l'affichage suivant:

void listeChambresParBloc();

```
INFO 9884 --- [   scheduling-1] t.e.s.Services.Chambre.ChambreService    : Bloc => A ayant une capacité 30
INFO 9884 --- [   scheduling-1] t.e.s.Services.Chambre.ChambreService    : La liste des chambres pour ce bloc:
INFO 9884 --- [   scheduling-1] t.e.s.Services.Chambre.ChambreService    : NumChambre: 1 type: SIMPLE
INFO 9884 --- [   scheduling-1] t.e.s.Services.Chambre.ChambreService    : **********
INFO 9884 --- [   scheduling-1] t.e.s.Services.Chambre.ChambreService    : Bloc => B ayant une capacité 30
INFO 9884 --- [   scheduling-1] t.e.s.Services.Chambre.ChambreService    : Pas de chambre disponible dans ce bloc
INFO 9884 --- [   scheduling-1] t.e.s.Services.Chambre.ChambreService    : **********
INFO 9884 --- [   scheduling-1] t.e.s.Services.Chambre.ChambreService    : Bloc => Bloc C ayant une capacité 100
INFO 9884 --- [   scheduling-1] t.e.s.Services.Chambre.ChambreService    : Pas de chambre disponible dans ce bloc
INFO 9884 --- [   scheduling-1] t.e.s.Services.Chambre.ChambreService    : **********
```

Si vous rencontrez cette erreur: `failed to lazily initialize a collection of role:` **→ Solution Slide 32**

Travail à faire

Partie 6 Spring Schedular

- Réaliser les trois services des trois slides suivants:

  → **Service 02:** Créer un service qui se déclenche **toutes les 5 minutes** permettant d'afficher le nombre total des chambres ainsi que le pourcentage des chambres par type chambre en respectant la signature et le format de l'affichage suivant :

  void pourcentageChambreParTypeChambre();

```
INFO 19104 --- [   scheduling-1] t.e.s.Services.Chambre.ChambreService    : Nombre total des chambre: 3
INFO 19104 --- [   scheduling-1] t.e.s.Services.Chambre.ChambreService    : Le pourcentage des chambres pour le type SIMPLE est égale à 100.0
INFO 19104 --- [   scheduling-1] t.e.s.Services.Chambre.ChambreService    : Le pourcentage des chambres pour le type DOUBLE est égale à 0.0
INFO 19104 --- [   scheduling-1] t.e.s.Services.Chambre.ChambreService    : Le pourcentage des chambres pour le type TRIPLE est égale à 0.0
```

Travail à faire

Partie 6 Spring Schedular

- Réaliser les trois services des trois slides suivants:

  → **Service 03:** Créer un service qui se déclenche **toutes les 5 minutes** permettant d'afficher le nombre de places encore disponible pour chaque chambre pour l'année en cours en respectant la signature et le format de l'affichage suivants :

  void nbPlacesDisponibleParChambreAnneeEnCours();

```
INFO 22532 --- [   scheduling-1] t.e.s.Services.Chambre.ChambreService    : La chambre SIMPLE 1 est complete
INFO 22532 --- [   scheduling-1] t.e.s.Services.Chambre.ChambreService    : Le nombre de place disponible pour la chambre SIMPLE 6 est 1
INFO 22532 --- [   scheduling-1] t.e.s.Services.Chambre.ChambreService    : Le nombre de place disponible pour la chambre SIMPLE 7 est 1
```

Travail à faire

Partie 6 Spring Schedular

- Réaliser les trois services des trois slides suivants:

  → **Service 04:** Créer un service qui se déclenche **le 30/06 de chaque année** permettant d'annuler toutes les réservations actives de l'année universitaire courante en respectant la signature et le format de l'affichage suivants :

void annulerReservations();

```
t.e.s.S.Reservation.ReservationService    : La reservation 2023/2024-A-1-123456789 est annulée automatiquement
```

Erreur / Solution

Server response

Code	Details
Undocumented	Error: response status is 200

Probléme:

```
java.lang.StackOverflowError Create breakpoint : null
    at java.base/java.lang.ClassLoader.defineClass1(Native Method) ~[na:na]
    at java.base/java.lang.ClassLoader.defineClass(ClassLoader.java:1012) ~[na:na]
```

Explication: Cette erreur est généralement déclenchée par une récursion infinie. Les boucles ou les références circulaires dans les objets peuvent entraîner une sérialisation infinie lors de la conversion de l'objet en JSON, ce qui provoque une StackOverflowError lorsque vous essayez de le sérialiser.

Solutions:

1. Utiliser les objets **DTO** (Data Transfer Objects) pour sérialiser uniquement les informations nécessaires à l'extérieur de l'objet.
2. Utiliser l'annotation **@JsonIgnore** pour exclure des champs qui provoquent des références circulaires de la sérialisation.

Erreur / Solution

Probléme:

```
org.hibernate.LazyInitializationException Create breakpoint : failed to lazily initialize a collection of role: tn.esprit.spring.Entities.Entité X.Entité Y:
    at tn.esprit.spring.Services.ServiceClass.listeEvenementsParCategorie(ServiceClass.java:42) ~[classes/:na]
```

Explication: Cette erreur se produit généralement lorsque vous essayez d'accéder à une collection chargée de manière paresseuse (par exemple, une collection d'entités liées) en dehors d'une session Hibernate ou d'une transaction active.

Solutions:

1. Modifier la stratégie de chargement de votre collection

Slides suivants

Fetch

Explication: C'est un paramètre que vous pouvez utiliser avec Spring Data JPA pour spécifier comment les données associées à une entité doivent être chargées à partir de la base de données lorsqu'une requête est exécutée. Il est généralement utilisé pour gérer le chargement paresseux (lazy loading) et le chargement précoce (eager loading) des associations entre entités.

1. Le chargement **Lazy** signifie que les données associées ne sont chargées qu'au moment où vous accédez réellement à ces données. Par défaut, Spring Data JPA utilise le chargement **Lazy** pour les associations @OneToMany et @ManyToMany.

```
@Entity
public class A {
    @OneToMany(mappedBy = "a", fetch = FetchType.LAZY)
    private List<B> bList;
}
```

Dans cet exemple, la relation entre A et B est définie avec un chargement Lazy. Les objets de type "B" ne seront chargés que lorsque vous y accéderez explicitement.

Fetch

2. Le chargement **Eager** signifie que les données associées sont chargées en même temps que l'entité principale. Par défaut, Spring Data JPA utilise le chargement **Eager** pour les associations @OneToOne et @ManyToOne. Vous pouvez utiliser le paramètre "**fetch**" pour spécifier le chargement **Eager**.

```
@Entity
public class A {
    @OneToMany(mappedBy = "a", fetch = FetchType.EAGER)
    private List<B> bList;
}
```

Dans cet exemple, la relation entre A et B est définie avec un chargement Eager. Lorsque vous récupérez un objet de type "A", ses objets de type "B" associés seront également chargés automatiquement.

Si vous ne spécifiez pas explicitement le paramètre "**fetch**", Spring Data JPA utilise le chargement paresseux par défaut.

@OneTMany	LAZY
@ManyToMany	LAZY

@ManyToOne	EAGER
@OneToOne	EAGER

TP Projet « Foyer »

Si vous avez des questions, n'hésitez pas à nous contacter :

Département Informatique

UP ASI (Architectures des Systèmes d'Information)

Bureau E204