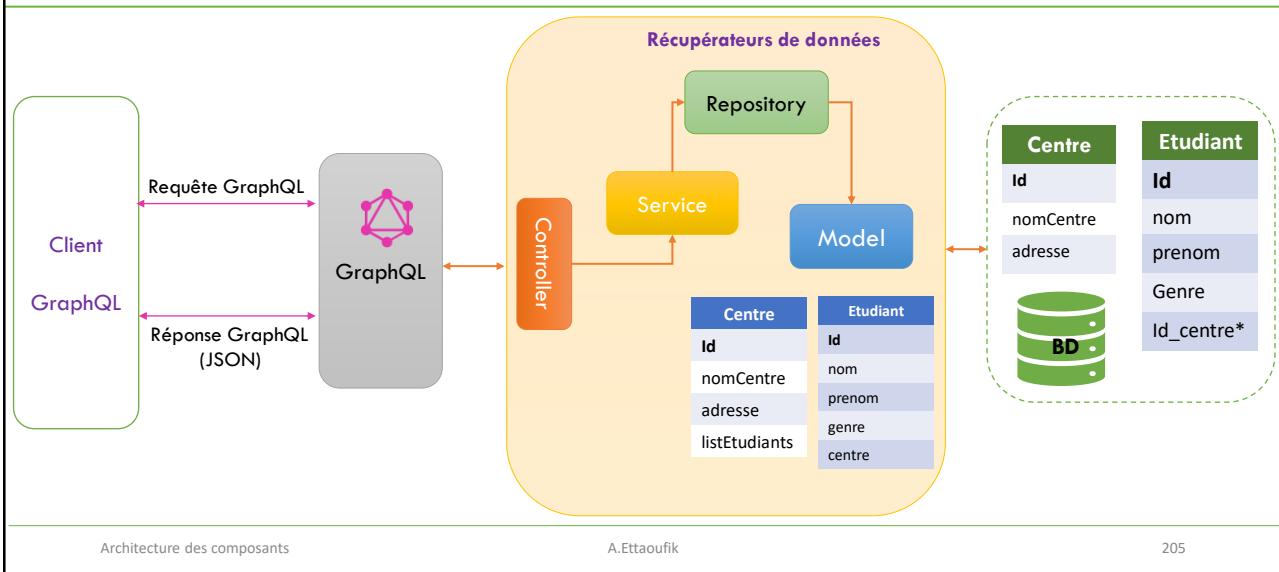


APPLICATION - SPRING DATA REST- GRAPHQL



APPLICATION - SPRING DATA REST- GRAPHQL

ÉTAPES À SUIVRE

- i. Création d'un projet Spring Boot y compris les dépendances
 - × **Spring Web**
 - × **H2 Database**
 - × **Lombok**
 - ii. Ajout des Entités JPA
 - iii. Création des Repository JPA
 - iv. Ajout d'une classe Controller
 - v. Création des objets de transfert de données (DTO)
 - vi. Définition du schéma GraphQL
 - vii. Définition de la source de données
 - viii. Teste
- × **Spring Data JPA**
 - × **Rest Repositories**
 - × **Spring for GraphQL**
 - × **WebSocket**

APPLICATION - SPRING DATA REST- GRAPHQL

Entités JPA

```
@Entity
@Data
@Builder
@AllArgsConstructor
@NoArgsConstructor
public class Centre {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    Long id;
    String nom;
    String adresse;
    @OneToMany(mappedBy = "centre", cascade = CascadeType.ALL)
    List<Etudiant> listEtudiants;
}
```

```
@Entity @Data
@AllArgsConstructor @NoArgsConstructor
@Builder @Table(name="etudiants")
public class Etudiant {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    Long id;
    @Column(name="nom_etudiant", nullable=false)
    String nom;
    @Column(name="prenom_etudiant")
    String prenom;
    @Enumerated(EnumType.STRING)
    Genre genre;
    @ManyToOne
    @NotNull
    @JoinColumn(name="centre_id")
    Centre centre;
}
```

APPLICATION - SPRING DATA REST- GRAPHQL

Création des objets de transfert de données (DTO)

```
public record EtudiantDTO (
    String nom,
    String prenom,
    Genre genre,
    Long centreId
) { }
```

NB : Ajouter la classe CentreDTO

APPLICATION - SPRING DATA REST- GRAPHQL

Ajouter la classe de mapping : EtudiantDTO->Etudiant

```
@Component
public class DtoToEtudiant {
    @Autowired
    CentreRepository centreRepository;
    public void toEtudiant(Etudiant et, EtudiantDTO dto) {
        Centre centre= centreRepository.findById(dto.centreId()).orElse(null);
        if (dto != null) {
            BeanUtils.copyProperties(etudiantDTO,etudiant);
            et.setCentre(centre);
        }
    }
}
```

APPLICATION - SPRING DATA REST- GRAPHQL

3- Repository JPA

```
public interface CentreRepository extends JpaRepository<Centre, Long> {}

public interface EtudiantRepository extends JpaRepository<Etudiant, Long> {}
```

APPLICATION - SPRING DATA REST- GRAPHQL

Services

```
@Service
public class EtudiantService {
    @Autowired
    DtoToEtudiant dtoToEtudiant;
    @Autowired
    EtudiantRepository etudiantRepository;

    private final Sinks.Many<Etudiant> sink = Sinks.many().multicast().onBackpressureBuffer();
    private final Sinks.Many<String> sinkSuppression = Sinks.many().multicast().onBackpressureBuffer();
    public List<Etudiant> getStudents() {
        return etudiantRepository.findAll();
    }
    public Etudiant getEtudiant(Long id) {
        return etudiantRepository.findById(id).orElse(null);
    }
    public Etudiant addEtudiant(EtudiantDTO etudiantDTO) {
        Etudiant etudiant=new Etudiant();
        dtoToEtudiant.toEtudiant(etudiant, etudiantDTO);
        etudiantRepository.save(etudiant);
        sink.tryEmitNext(etudiant);

        return etudiant;
    }
}
```

SERVICE (SUITE)

```
public Etudiant updateEtudiant(Long id, EtudiantDTO etudiantDTO){
    if(etudiantRepository.findById(id).isPresent()){
        Etudiant etudiant=etudiantRepository.findById(id).get();
        dtoToEtudiant.toEtudiant(etudiant,etudiantDTO);
        return etudiantRepository.save(etudiant);
    }
    return null;
}

public String deleteEtudiant(Long id){
    if(etudiantRepository.findById(id).isPresent()){
        Etudiant et=etudiantRepository.findById(id).get();
        etudiantRepository.deleteById(id);
        String msg=String.format("l'étudiant %s %s vient de quitter l'école !",et.getNom(), et.getPrenom());
        sinkSuppression.tryEmitNext(msg);
        return String.format("l'étudiant %s est bien supprimé !",id);
    }
    return String.format("l'étudiant %s n'existe pas !",id);
}
```

SERVICE (SUITE)

```
public Flux<Etudiant> getEtudiantAdded() {
    return sink.asFlux();
}

public Flux<String> etudiantMoved() {
    return sinkSuppression.asFlux();
}
```

APPLICATION - SPRING DATA REST- GRAPHQL

Services

```
@Service
public class CentreService {

    @Autowired
    CentreRepository centreRepository;

    public List<Etudiant> getCentres() {
        return centreRepository.findAll();
    }

    public Centre getCentre(Long id) {
        return centreRepository.findById(id).orElse(null);
    }
}
```

CONTROLLER

```
@Controller
public class EtudiantCentreController {
    @Autowired
    EtudiantService etudiantService;
    @Autowired
    CentreService centreService;
    @QueryMapping
    public List<Centre> getAllCentres(){
        return centreService.getCentres();
    }
    @QueryMapping
    public List<Etudiant> getAllEtudiants(){

        return etudiantService.getStudents();
    }
    @QueryMapping
    public Centre getCentreById(@Argument int id){
        return centreService.getCentre(id);
    }
}
```

Architecture des composants

A.ELLDUUIIK

415

CONTROLLER(SUITE)

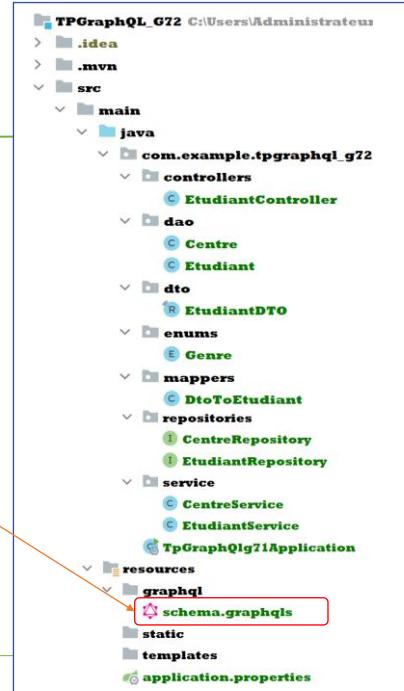
```
@QueryMapping
public Etudiant getEtudiantById(@Argument Long id){
    return etudiantService.getEtudiant(id);
}
@MutationMapping
public Etudiant addEtudiant(@Argument EtudiantDTO etudiantDTO) {
    return etudiantService.addEtudiant(etudiantDTO);
}
@MutationMapping
public String deleteEtudiant(@Argument Long id){
    return etudiantService.deleteEtudiant(id);
}
@MutationMapping
public Etudiant updateEtudiant(@Argument Long id,@Argument EtudiantDTO etudiantDTO) {
    return etudiantService.updateEtudiant(id,etudiantDTO);
}
@SubscriptionMapping
public Flux<Etudiant> etudiantAdded() {
    return etudiantService.getEtudiantAddedPublisher();
}
@SubscriptionMapping
public Flux<Etudiant> etudiantMoved() {
    return etudiantService.etudiantMoved();
```

APPLICATION - SPRING DATA REST- GRAPHQL

Ajout d'un fichier graphqls : schema.graphqls

Architecture des composants

A.Ettaoufik



APPLICATION - SPRING DATA REST- GRAPHQL

```
type Query{
    getAllEtudiants : [Etudiant]
    getEtudiantById(id:Float):Etudiant
    getAllCentres:[Centre]
    getCentreById(id:Float):Centre
}
type Mutation{
    addEtudiant(etudiantDTO : EtudiantDTO):Etudiant
    updateEtudiant(id:Float,etudiantDTO : EtudiantDTO):Etudiant
    deleteEtudiant(id:Float):String
}
type Subscription{
    etudiantAdded:Etudiant
    etudiantRemoved:String
}
enum Genre {
    Homme,
    Femme
}
```

Architecture des composants

A.Ettaoufik

```
type Etudiant{
    id:Float
    nom:String
    prenom:String
    genre:Genre
    centre:Centre
}
type Centre{
    id:Int
    nom: String
    adresse:String
    listEtudiants:[Etudiant]
}
input EtudiantDTO{
    nom:String
    prenom:String
    genre:String
    centreId:Float
}
```

218

APPLICATION - SPRING DATA REST- GRAPHQL

Configuration & dependencies

Application.properties

```
spring.h2.console.enabled=true
spring.datasource.username=12
spring.datasource.password=
spring.datasource.url=jdbc:h2:mem:centredb
spring.graphql.graphiql.enabled=true
spring.websocket.path=/graphql
```

APPLICATION - SPRING DATA REST- GRAPHQL

Ajout d'un jeu d'enregistrements

```
public class TpGraphQlApplication implements CommandLineRunner{
    @Autowired
    EtudiantRepository etudiantRepository;
    @Autowired
    CentreRepository centreRepository;
    public static void main(String[] args) {
        SpringApplication.run(TpGraphQlApplication.class, args);
    }
    @Override
    public void run(String... args) throws Exception {
        Centre centre1=Centre.builder()
            .nom("Maarif").adresse("Biranzarane").build();
        centreRepository.save(centre1);
        Centre centre2=Centre.builder()
            .nom("Oranges").adresse("Oulfa").build();
        centreRepository.save(centre2);
        Etudiant etl=Etudiant.builder()
            .nom("Adnani").prenom("Brahim").genre(Genre.Homme)
            .centre(centre1).build();
        etudiantRepository.save(etl);
    ...
}
```

TEST-QUERY

http://localhost:8080/graphiql?path=/graphql

```

query {
  getEtudiantById(id: 1) {
    nom
    prenom
    centre {
      nom
      adresse
    }
  }
}

query{
  findAllEtudiants{
    id
    nom
    prenom
    centre{
      nom
      adresse
    }
  }
}

query{
  getCentreById(id:1){
    nom
    listEtudiants{
      nom
      prenom
    }
  }
}

fragment fragmentEtu on Etudiant {
  nom,prenom,genre
}

query{
  findAllEtudiants{
    id
    ...fragmentEtu
    centre{
      nom
      adresse
    }
  }
}

```

Architecture des composants A.Ettaoufik

INSOMNIA POR GRAPHQL

POST ▼ http://localhost:8080/graphql

Send ▾

| | | | | | |
|--------|------|------|-------------|---------|---|
| Params | Body | Auth | Headers (4) | Scripts | D |
|--------|------|------|-------------|---------|---|

Preview ▾

```

1 ▼ {
2 ▼   "data": {
3 ▼     "getEtudiant": {
4       "nom": "Fadli",
5       "genre": "Homme"
6     }
7   }
8 }

```

Operations schema 🔑

```

1 ▼ query{
2 ▼   getEtudiant(id:3){
3     nom
4     genre
5   }
6 }
7

```

Architecture des composants A.Ettaoufik 222

TEST-MUTATION

The screenshot shows a GraphQL playground interface with three mutation tests:

```

1 mutation{
2   updateEtudiant(id:1, etudiant:{
3     nom:"UnNom"
4     prenom:"UnPrénom"
5     genre:"Homme"
6     centreId:2
7   })
8 {
9   nom
10  prenom
11  centre{
12    id
13    nom
14  }
15 }
}

```

```

mutation{
  updateEtudiant(id:1, etudiant:{
    nom:"UnNom"
    prenom:"UnPrénom"
    genre:"Homme"
    centreId:2
  })
}

```

```

mutation{
  deleteEtudiant(id:5)
}

```

```

mutation{
  deleteEtudiant(id:10)
}

```

On the right, the results are displayed:

```

1 mutation($n:String,$p:String,$g:String,$cId:Float){
2   addEtudiant(etudiant: {
3     nom:$n
4     prenom:$p
5     genre:$g
6     centreId:$cId
7   })
8   {
9     id
10    nom
11    centre{
12      id
13    }
14  }
15 }

```

Variables Headers

1 {"n": "unNom", "p": "unPrénom", "g": "Homme", "cId": 2 }

Architecture des composants

A.Ettaoufik

223

TEST-MUTATION (SUITE)

The screenshot shows a GraphQL playground interface with two mutation tests:

```

1 mutation($d:Int,$n:String,$p:String,$g:String,$i:Int){
2   updateEtudiant(id:$d,etudiant:{
3     nom:$n
4     prenom:$p
5     genre:$g
6     centreId:$i
7   })
8 {
9   nom
10  prenom
11  centre
12 }
}

```

```

1 mutation($et:EtudiantDTO){
2   updateEtudiant(id:3,etudiant:$et)
3 {
4   nom
5   prenom
6   centre
7 {
8   id
9   nom
10 }
11 }
}

```

On the right, the results are displayed:

```

1 mutation($d:Int,$n:String,$p:String,$g:String,$i:Int){
2   updateEtudiant(id:$d,etudiant:{
3     nom:"testNom3",
4     prenom:"testPrenom3",
5     genre:"Homme",
6     centreId:$i
7   })
8 {
9   nom
10  prenom
11  centre
12 }
}

```

```

1 mutation($et:EtudiantDTO){
2   updateEtudiant(id:3,etudiant:$et)
3 {
4   nom
5   prenom
6   centre
7 {
8   id
9   nom
10 }
11 }
}

```

Variables Headers

1 {"d":4, "n": "testNom3", "p": "testPrenom3", "g": "Homme", "i":2}

Variables Headers

1 {"et": {"nom": "testNom30", "prenom": "testPrenom30", "genre": "Homme", "centreId": 2}}

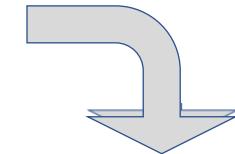
224

TEST-SUBSCRIPTION

```
1 mutation{  
2   addEtudiant(etudiant:{  
3     nom:"nom5"  
4     prenom:"prénom5"  
5     genre:"Homme"  
6     centreId:1  
7   }){  
8     id  
9     nom  
10    prenom  
11  }  
12 }
```

Client1

```
1 {  
2   "data": {  
3     "addEtudiant": {  
4       "id": 5,  
5       "nom": "nom5",  
6       "prenom": "prénom5"  
7     }  
8   }  
9 }
```



```
1 subscription{  
2   etudiantAdded{  
3     id  
4     nom  
5     prenom  
6     genre  
7     centre  
8     {  
9       id  
10      nom  
11      adresse  
12    }  
13  }  
14 }
```

Client2

```
{  
  "data": {  
    "etudiantAdded": {  
      "id": 5,  
      "nom": "nom5",  
      "prenom": "prénom5",  
      "genre": "Homme",  
      "centre": {  
        "id": 1,  
        "nom": "Maarif",  
        "adresse": "Biranzarane"  
      }  
    }  
  }  
}
```

Architecture des composants