Introduction et Préparation

Ce TP a pour but de manipuler une API GraphQL via l'interface Directus. Pour réaliser ces tests, nous avons utilisé le logiciel Postman. Toutes les requêtes ont été effectuées avec la méthode POST sur l'adresse suivante : http://localhost:8055/graphql

Lien GitHub du projet : https://github.com/lclerc561/DWM_ParadigmeBDD.git

I. Requêtes Query

1. **Liste des praticiens :** Nous récupérons l'id, le nom, le prénom, le téléphone et la ville de tous les praticiens.

```
query {
  Praticien {
    id
    nom
    prenom
    telephone
    ville
  }
}
```


```
{
  "data": {
    "Praticien": [
      {
        "id": "03f24d7b-efda-3fbe-ae53-0c0c0edef089",
        "nom": "Grenier",
        "prenom": "Josette",
        "telephone": "06 43 92 58 43",
        "ville": "Moreno"
      },
      {
        "id": "05644ac0-c4cb-36bf-a0cf-409075eb20c4",
        "nom": "Leduc",
        "prenom": "Aimée",
        "telephone": "+33 (0)4 85 19 99 00",
        "ville": "Carre-sur-Turpin"
      },
      {
        "id": "0768e78b-12c8-3694-b89b-d0a7451a8fb1",
        "nom": "Descamps",
        "prenom": "Alexandria",
        "telephone": "+33 (0)1 75 74 98 33",
        "ville": "Joubert"
      },
      {
        "id": "085c44e7-6245-3b10-9c1a-92f09524ef2a",
        "nom": "Gallet",
        "prenom": "Valérie",
```


2. Ajout de la spécialité : Grâce à la relation entre les tables, nous demandons le libellé de la spécialité imbriqué dans l'objet Praticien.

```
query {
  Praticien {
    id
    nom
    prenom
    ville
    specialite {
```

```
      libelle
    }
  }
}
```


```
{
  "data": {
    "Praticien": [
      {
        "id": "03f24d7b-efda-3fbe-ae53-0c0c0edef089",
        "nom": "Grenier",
        "prenom": "Josette",
        "ville": "Moreno",
        "specialite": {
          "libelle": "Imagerie médicale"
        }
      },
      {
        "id": "05644ac0-c4cb-36bf-a0cf-409075eb20c4",
        "nom": "Leduc",
        "prenom": "Aimée",
        "ville": "Carre-sur-Turpin",
        "specialite": {
          "libelle": "ophtalmologie"
        }
      },
      {
        "id": "0768e78b-12c8-3694-b89b-d0a7451a8fb1",
        "nom": "Descamps",
        "prenom": "Alexandria",
        "ville": "Joubert",
        "specialite": {
          "libelle": "ophtalmologie"
```


3. Filtrage par ville ("Paris") : Nous utilisons l'argument filter avec l'opérateur _eq (equal) sur le champ ville pour ne retourner que les praticiens parisiens.

```
query {
  Praticien(filter: { ville: { _eq: "Paris" } }) {
    nom
    prenom
    ville
  }
}
```


```
{
    "data": {
        "Praticien": [
            {
                "nom": "Gallet",
                "prenom": "Valérie",
                "ville": "Paris"
            },
            {
                "nom": "Pichon",
                "prenom": "Arnaude",
                "ville": "Paris"
            },
            {
                "nom": "Klein",
                "prenom": "Gabrielle",
                "ville": "Paris"
            },
            {
                "nom": "Paul",
                "prenom": "Marine",
                "ville": "Paris"
            },
            {
                "nom": "Guichard",
                "prenom": "Laurence",
                "ville": "Paris"
            },
```


4. Ajout de la structure d'appartenance : Nous récupérons les informations de la structure liée (nom et ville) en naviguant dans la relation structure

```
query {
  Praticien {
    nom
    structure {
      nom
      ville
    }
  }
}
```

```
{
    "data": {
        "Praticien": [
            {
                "nom": "Grenier",
                "structure": {
                    "nom": "Babinet Antoine",
                    "ville": "Moreno"
                }
            },
            {
                "nom": "Leduc",
                "structure": {
                    "nom": "Maison de Santé de Care-Sur_Turpin",
                    "ville": "Carre-sur-Turpin"
                }
            },
            {
                "nom": "Descamps",
                "structure": {
                    "nom": "Cabinet Toussaint Marechal",
                    "ville": "Joubert"
                }
            },
            {
                "nom": "Gallet",
                "structure": {
                    "nom": "Cabinet Bigot",
```


5. Filtrage sur les emails : Nous utilisons l'opérateur _contains sur le champ email pour sélectionner uniquement ceux contenant la chaîne ".fr".

```
query {
  Praticien(filter: { email: { _contains: ".fr" } }) {
    nom
    email
  }
}
```

```
{
    "data": {
        "Praticien": [
            {
                "nom": "Grenier",
                "email": "Josette.Grenier@wanadoo.fr"
            },
            {
                "nom": "Descamps",
                "email": "Alexandria.Descamps@club-internet.fr"
            },
            {
                "nom": "Pascal",
                "email": "Gilles.Pascal@sfr.fr"
            },
            {
                "nom": "Leleu",
                "email": "Isaac.Leleu@tele2.fr"
            },
            {
                "nom": "Didier",
                "email": "dith.Didier@club-internet.fr"
            },
            {
                "nom": "Garcia",
                "email": "Emmanuel.Garcia@sfr.fr"
            },
            {
```


6. Filtrage imbriqué (Ville de la structure) : Le filtre est appliqué sur la relation : on cherche les praticiens dont la *structure* a pour ville "Paris".

```
query {
  Praticien(filter: { structure: { ville: { _eq: "Paris" } } }) {
    nom
    structure {
      ville
    }
  }
}
```

```
{
    "data": {
        "Praticien": [
            {
                "nom": "Gallet",
                "structure": {
                    "ville": "Paris"
                }
            },
            {
                "nom": "Pichon",
                "structure": {
                    "ville": "Paris"
                }
            },
            {
                "nom": "Klein",
                "structure": {
                    "ville": "Paris"
                }
            },
            {
                "nom": "Paul",
                "structure": {
                    "ville": "Paris"
                }
            },
```


7. Utilisation des alias Pour récupérer deux listes distinctes dans la même réponse (Parisiens et habitants de Bourdon-les-Bains), nous nommons les résultats parisiens: et bourdons: via des alias.

```
query {
  parisiens: Praticien(filter: { ville: { _eq: "Paris" } }) {
    nom
    ville
  }
  bourdons: Praticien(filter: { ville: { _eq: "Bourdon-les-Bains" } }) {
    nom
    ville
  }
}
```

8. Utilisation des fragments : Pour éviter la répétition des champs (nom, prénom, email, ville) dans la requête précédente, nous avons défini un fragment infosBase réutilisé dans les deux parties de la query.

```
query {
  parisiens: Praticien(filter: { ville: { _eq: "Paris" } }) {
    ...infosBase
  }
  bourdons: Praticien(filter: { ville: { _eq: "Bourdon-les-Bains" } }) {
    ...infosBase
  }
}

fragment infosBase on Praticien {
  nom
  prenom
  email
  ville
}
```

```
{
    "data": {
        "parisiens": [
            {
                "nom": "Gallet",
                "prenom": "Valérie",
                "email": "Valrie.Gallet@dbmail.com",
                "ville": "Paris"
            },
            {
                "nom": "Pichon",
                "prenom": "Arnaude",
                "email": "Arnaude.Pichon@yahoo.fr",
                "ville": "Paris"
            },
            {
                "nom": "Klein",
                "prenom": "Gabrielle",
                "email": "Gabrielle.Klein@live.com",
                "ville": "Paris"
            },
            {
                "nom": "Paul",
                "prenom": "Marine",
                "email": "Marine.Paul@hotmail.fr",
                "ville": "Paris"
            },
            {
                "nom": "Goncalves",
                "prenom": "Noël",
                "email": "Nol.Goncalves@free.fr",
                "ville": "Paris"
            }
        ],
        "bourdons": [
            {
                "nom": "Marin",
                "prenom": "Isaac",
                "email": "Isaac.Marin@live.com",
                "ville": "Bourdon-les-Bains"
            },
            {
                "nom": "Gilbert",
                "prenom": "Adrien",
                "email": "Adrien.Gilbert@live.com",
                "ville": "Bourdon-les-Bains"
            },
```


9. Utilisation de variables : La requête est paramétrée avec $villeCherchee.

```
query GetPraticiensByVille($villeCherchee: String!) {
  Praticien(filter: { ville: { _eq: $villeCherchee } }) {
    nom
    ville
  }
}
```

La valeur "Nancy" est passée séparément dans les variables JSON, ce qui rend la requête dynamique.

```
{
  "villeCherchee": "Nancy"
}
```


```
{
    "data": {
        "Praticien": [
            {
                "nom": "Radio Plus",
                "ville": "Nancy"
            }
        ]
    }
}
```

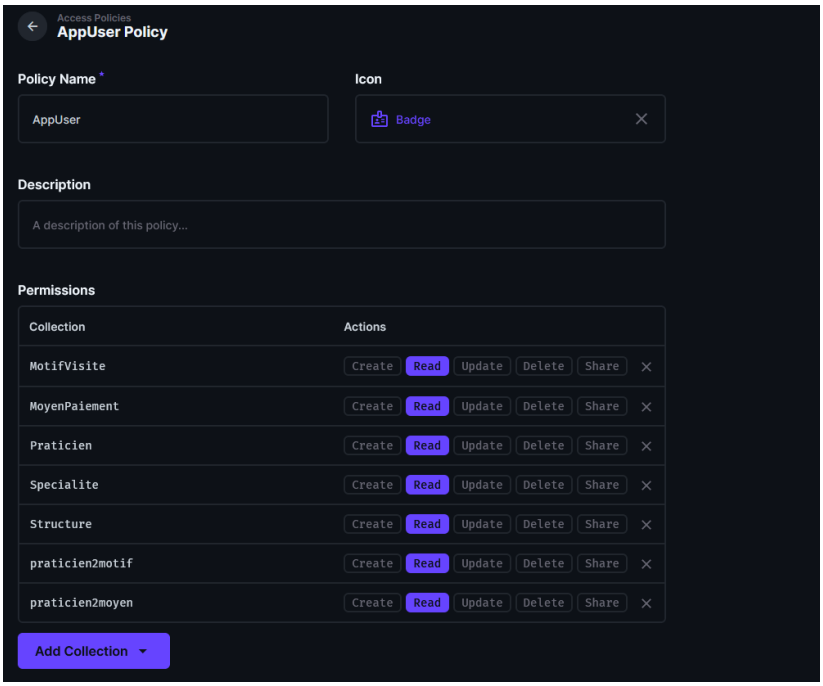

10. Liste inversée (Structures vers Praticiens) : Nous partons de l'entité Structure pour lister ses informations, puis nous récupérons la liste des praticiens rattachés à chaque structure avec leurs détails.

```
query {
  Structure {
    nom
    ville
    praticiens {
      nom
      prenom
      email
      specialite {
        libelle
      }
    }
  }
}
```


```
{
  "data": {
    "Structure": [
      {
        "nom": "Pichon Santé",
        "ville": "Diaz-sur-Boulanger",
        "praticiens": [
          {
            "nom": "Marchand",
            "prenom": "Suzanne",
            "email": "Suzanne.Marchand@sfr.fr",
            "specialite": {
              "libelle": "ophtalmologie"
            }
          },
          {
            "nom": "Martel",
            "prenom": "Denise",
            "email": "Denise.Martel@tele2.fr",
            "specialite": {
              "libelle": "Imagerie médicale"
            }
          },
          {
            "nom": "Roger",
            "prenom": "Philippine",
            "email": "Philippine.Roger@hotmail.fr",
            "specialite": {
```

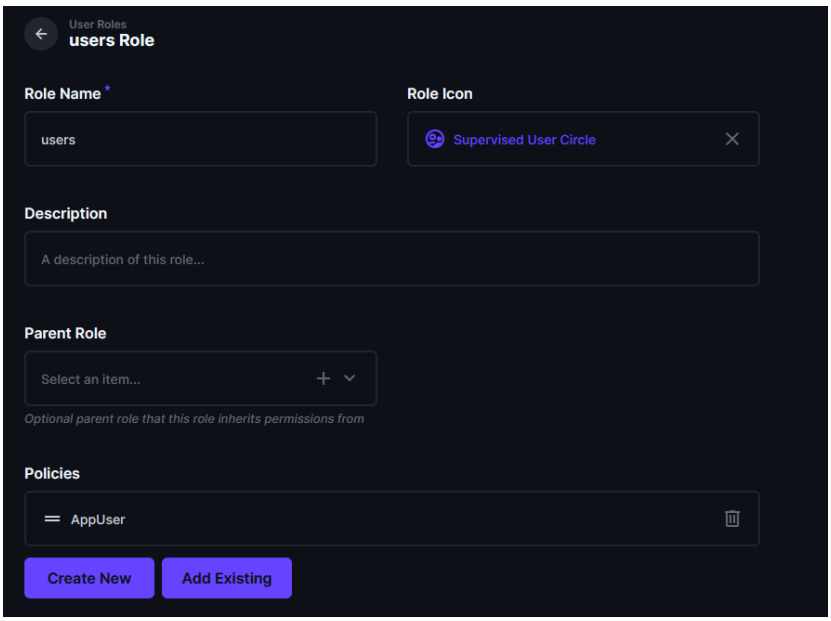

II. Autorisations dans Directus

Configuration Nous avons créé une Policy "AppUser" avec des droits de lecture et un rôle "users". Nous avons ensuite créé deux utilisateurs (un avec token statique, un pour JWT) et retiré les droits publics sur certaines collections. Sur directus : Settings > Access Policies

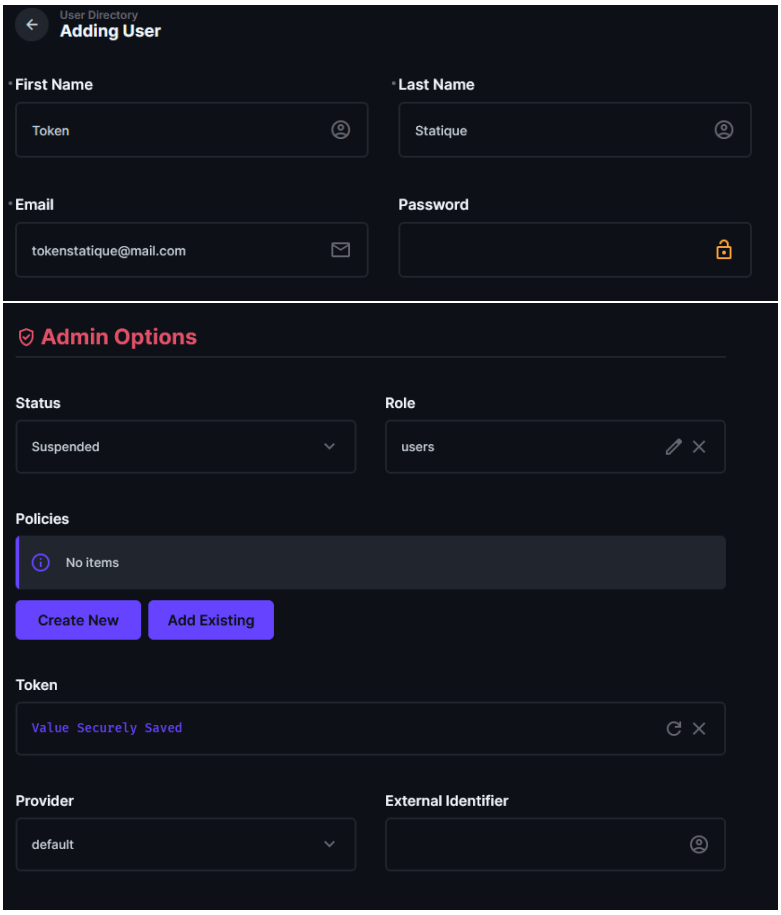


Création de rôles :

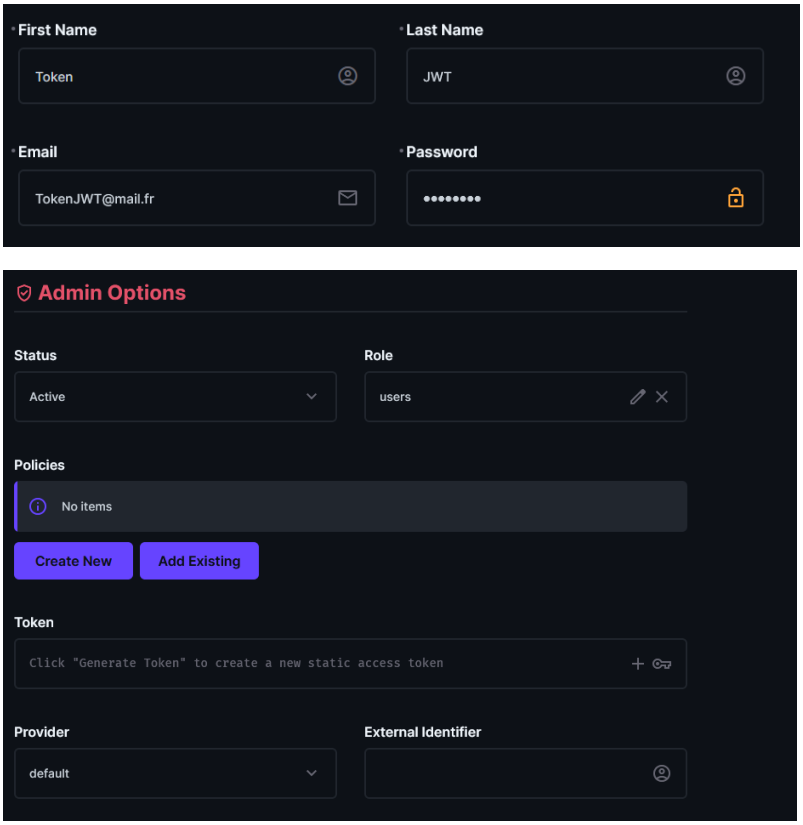
Sur directus : Settings > User Roles



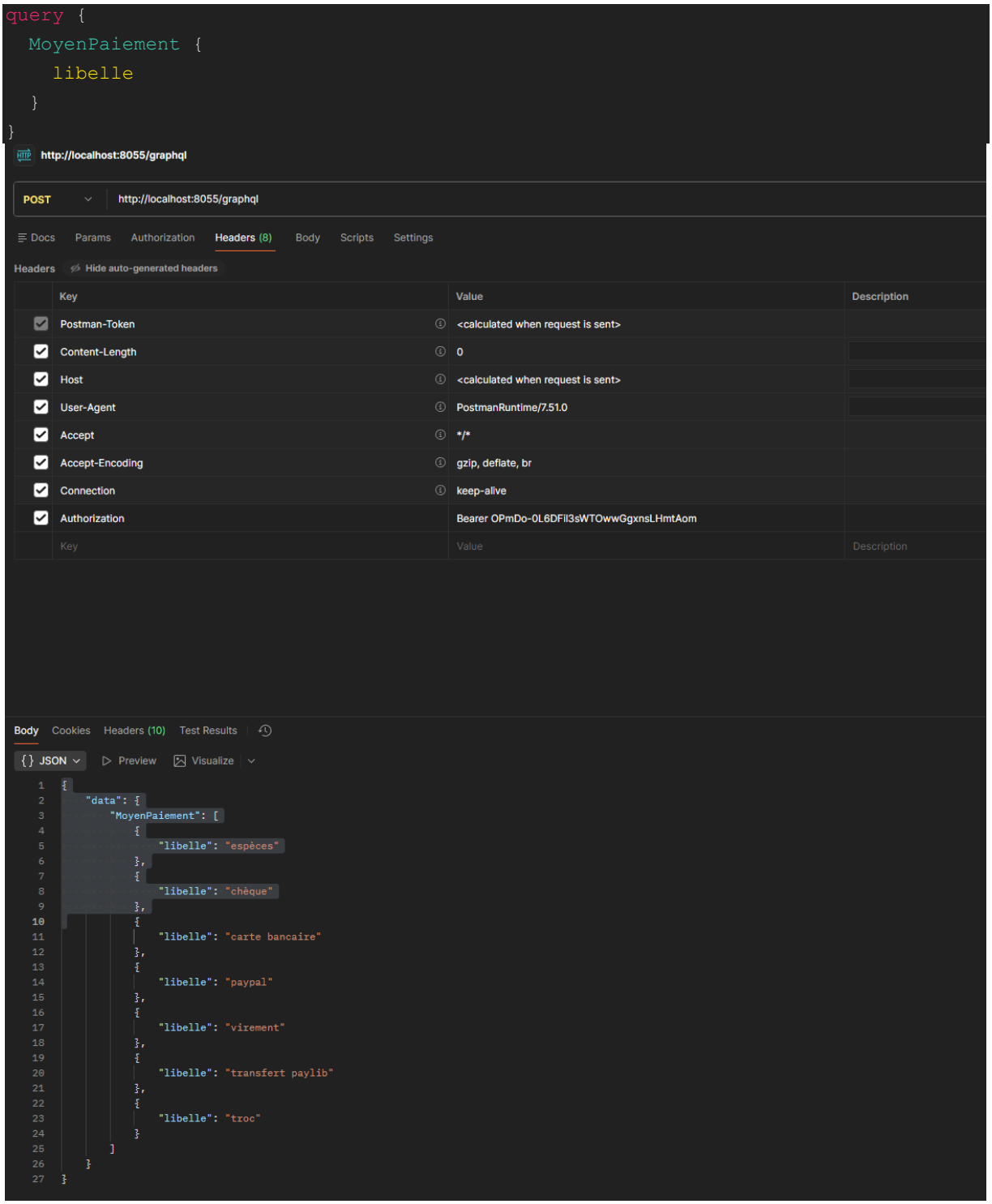
Ajout utilisateur token statique :



Token JWT :



Test avec Token Statique : L'accès à MoyenPaiement étant interdit au public, nous avons utilisé un Token statique dans le Header Authorization pour réussir la requête.



Test avec Token JWT : Nous avons d'abord effectué une mutation auth_login pour récupérer un access_token temporaire.

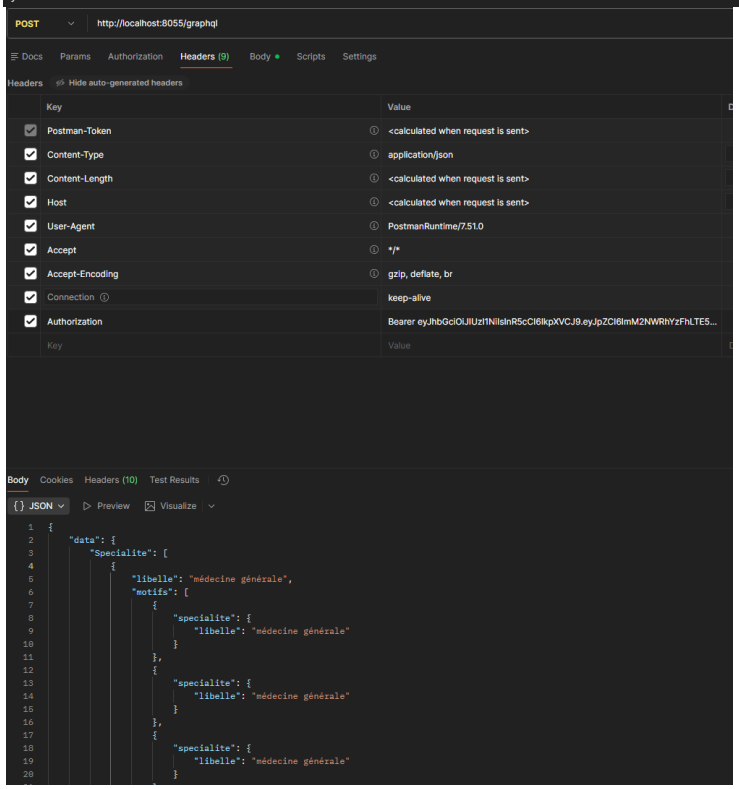
Requête POST à http://localhost:8055/graphql/system

```
mutation {
  auth_login(email: "TokenJWT@mail.fr", password: "TokenJWT") {
    access_token
    refresh_token
  }
}
```


```
{
  "data": {
    "auth_login": {
      "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
        eyJpZCI6ImM2NWRhYzFhLTE5YWQtNDZjMC1hMGE2LTFmYzFlZDk0YzU0ZCIsInJvbGUiOiJiMDBjMGMzNy1jMTJjLTQyYTgtYjdiMi0zNjk3ZDIxYzVmMGQiLCJhcHBfYWNjZXNzIjpmYWxzZSwiYWRtaW5fYWNjZXNzIjpmYWxzZSwiaWF0IjoxNzY4MzE1MDkzLCJleHAiOjE3NjgzMTU0OTMsImlzcyI6ImRpcmVjdHVzIn0.oh81noAzztAKgMp58YfuaJU37PkVgYLn__beIgAW4KY",
      "refresh_token": "GoWL-bbFxw2VydqBE_OjeFOlhsMs2ca0ue0z8n4GBleMs40QbPoE8nvyo-y14Kfp"
    }
  }
}
```


Ensuite, nous avons utilisé ce token pour interroger la liste des spécialités et leurs motifs de visite.

```
query {
  Specialite {
    libelle
    motifs {
      specialite {
        libelle
      }
    }
  }
}
```



III. Mutations GraphQL

1. Création de la spécialité : Nous utilisons create_Specialite_item pour ajouter "cardiologie".

```
mutation {
  create_Specialite_item(data: { libelle: "cardiologie" }) {
    id
    libelle
  }
}
```

Body Cookies Headers (10) Test Results

{} JSON Preview Visualize


```
{
    "data": {
        "create_Specialite_item": {
            "id": "6",
            "libelle": "cardiologie"
        }
    }
}
```


2. Création d'un praticien simple : Ajout d'un praticien avec ses coordonnées de base.

```
mutation {
  create_Praticien_item(data: {
    nom: "Dupont",
    prenom: "Jean",
    ville: "Paris",
    email: "jean.dupont@email.com",
    telephone: "0601020304"
  }) {
    id
    nom
  }
}
```


```
{
    "data": {
        "create_Praticien_item": {
            "id": "c6d8d67b-82bb-455f-b9fc-90791ed7fde2",
            "nom": "Dupont"
        }
    }
}
```


3. Mise à jour (Rattachement spécialité) Nous utilisons update_Praticien_item avec l'ID du praticien créé précédemment pour lui assigner l'ID de la spécialité cardiologie.

```
mutation {
  update_Praticien_item(id: "c6d8d67b-82bb-455f-b9fc-90791ed7fde2", data: {
    specialite: {
      id: 6
    }
  }) {
    id
    nom
    specialite {
      libelle
    }
  }
}
```

Body Cookies Headers (10) Test Results

{} JSON Preview Visualize

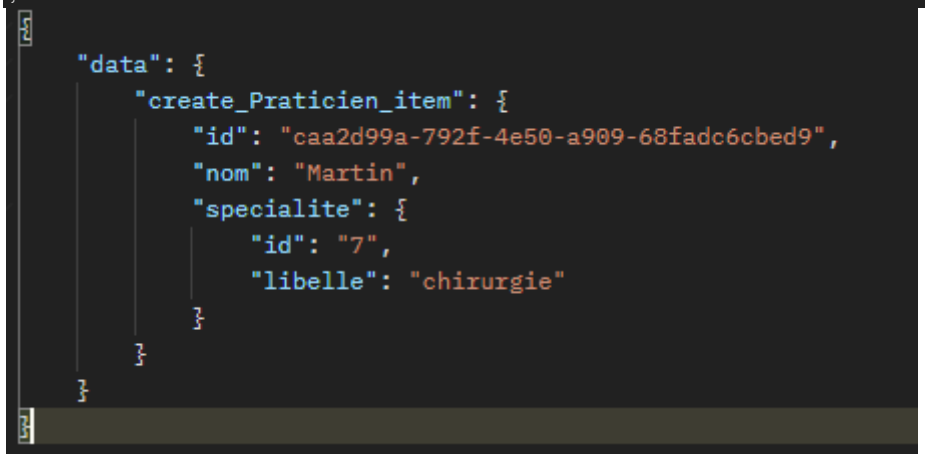

```
{
    "data": {
        "update_Praticien_item": {
            "id": "c6d8d67b-82bb-455f-b9fc-90791ed7fde2",
            "nom": "Dupont",
            "specialite": {
                "libelle": "cardiologie"
            }
        }
    }
}
```


4. Création avec rattachement immédiat Lors de la création du praticien "Durand", nous indiquons directement l'ID de la spécialité existante.

```
mutation {
  create_Praticien_item(data: {
    nom: "Durand",
    prenom: "Sophie",
    ville: "Lyon",
    email: "sophie.durand@hopital.fr",
    specialite: {
      id: 6
    }
  }) {
    id
    nom
    specialite { libelle }
  }
}
```

5. Création imbriquée (Praticien + Nouvelle spécialité) Nous créons le praticien "Martin" et, dans le même objet, nous définissons une nouvelle spécialité "chirurgie". Directus gère la création des deux entités et leur liaison.

```
mutation {
  create_Praticien_item(data: {
    nom: "Martin",
    prenom: "Lucas",
    ville: "Marseille",
    specialite: {
      libelle: "chirurgie"
    }
  }) {
    id
    nom
    specialite {
      id
      libelle
    }
  }
}
```



6. Ajout à la spécialité créée Nous créons le praticien "Petit" en le rattachant à l'ID de la spécialité "chirurgie" (ID 7) qui vient d'être générée.

```
mutation {
  create_Praticien_item(data: {
    nom: "Petit",
    prenom: "Julie",
    ville: "Nice",
    specialite: {
      id: 7
    }
  }) {
    id
    specialite { libelle }
  }
}
```

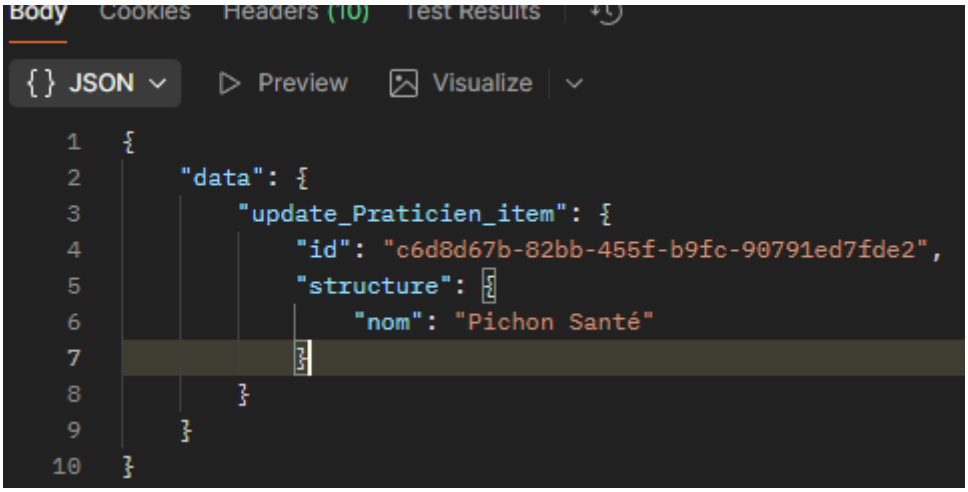
Body Cookies Headers (10) Test Results

{} JSON ▷ Preview Visualize


```
{
    "data": {
        "create_Praticien_item": {
            "id": "aeea55fb-b2fa-429c-8706-662d99244391",
            "specialite": {
                "libelle": "chirurgie"
            }
        }
    }
}
```


7. Rattachement à une structure Nous modifions le premier praticien pour lui assigner l'ID d'une structure existante ("Pichon Santé").

```
mutation {
  update_Praticien_item(id: "c6d8d67b-82bb-455f-b9fc-90791ed7fde2", data: {
    structure: {
      id: "255ecef6-14b9-3b5c-a40e-901665f4ed28"
    }
  }) {
    id
    structure { nom }
  }
}
```



8. Suppression : Nous utilisons delete_Praticien_items en passant un tableau contenant les IDs des deux derniers praticiens créés pour les effacer de la base.