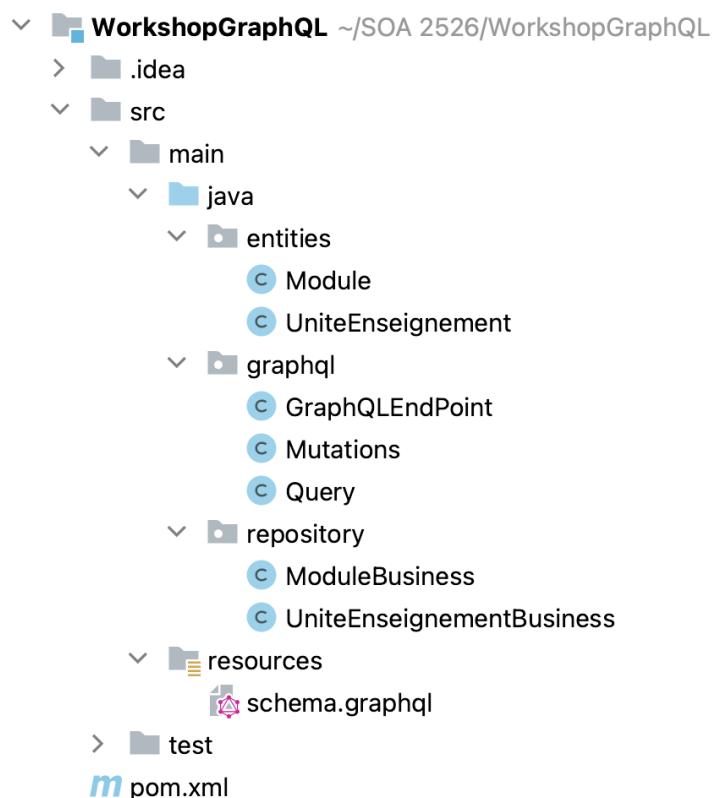


Configuration

Génération d'une application GraphQL

La mise en œuvre de l'application

Le projet doit avoir la structure **finale** suivante :



1. Ajouter les dépendances suivantes au fichier POM :

```
<dependencies>
```

```

<dependency>
<groupId>com.graphql-java</groupId>
<artifactId>graphql-java</artifactId>
<version>3.0.0</version>
</dependency>
<dependency>
<groupId>com.graphql-java</groupId>
<artifactId>graphql-java-tools</artifactId>
<version>3.2.0</version>
</dependency>
<dependency>
<groupId>com.graphql-java</groupId>
<artifactId>graphql-java-servlet</artifactId>
<version>4.0.0</version>
</dependency>
<dependency>
<groupId>javax.servlet</groupId>
<artifactId>javax.servlet-api</artifactId>
<version>3.0.1</version>
<scope>provided</scope> </dependency> </dependencies>

```

2. Créer la classe « **Query.java** » dans le package « **graphql** »

```

import com.coxautodev.graphql.tools.GraphQLRootResolver;

public class Query implements GraphQLRootResolver {
}

```

3. Créer le fichier « **schema.graphqls** » dans **src/main/ressources** :

```

type UniteEnseignement {
  code: Int!
  domaine: String!

```

```

    responsable: String!
    credits: Int!
    semestre: Int!
}
enum TypeModule {
    TRANSVERSAL
    PROFESSIONNEL
    RECHERCHE
}
type Module {
    matricule: String!
    nom: String!
    coefficient: Int!
    volumeHoraire: Int!
    type: TypeModule!
    uniteEnseignement: UniteEnseignement
}
type Query {
    getallmodules: [Module!]!
}
schema {
    query: Query
}

```

4. Créer la classe « **GraphQLEndpoint.java** » dans le package « **graphql** ».

```

package graphql;
import com.coxautodev.graphql.tools.SchemaParser;
import graphql.schema.GraphQLSchema;
import graphql.servlet.SimpleGraphQLServlet;
import repository.ModuleBusiness;

import javax.servlet.annotation.WebServlet;

@WebServlet(urlPatterns = "/graphql")
public class GraphQLEndPoint extends SimpleGraphQLServlet {

    public GraphQLEndPoint() {
        super(buildSchema());
    }

    private static GraphQLSchema buildSchema() {

```

```

        return SchemaParser.newParser()
            .file("schema.graphqls")
            .resolvers(new Query())
            .build()
            .makeExecutableSchema();
    }
}

```

NB :

@WebServlet est utilisée pour définir un composant Servlet dans une application Web. Cette annotation est spécifiée sur une classe et contient des métadonnées sur le servlet déclaré.

urlPatterns est le paramètre permettant de définir le chemin d'accès à la ressource.

5. Déployer le projet et tester l'url via **Postman** :

<http://localhost:port/nomProjet/graphql>

The screenshot shows the Postman interface with the following details:

- URL:** http://localhost:8083/WorkshopGraphQL_war_exploded/graphql
- Query Tab:** The "Query" tab is selected. A search bar contains "Search fields". Below it, a dropdown menu is open, showing "Query" with a checked checkbox labeled "getallmodules [Module!]!".
- Code Area:** The code area displays a GraphQL query:

```

query Getallmodules {
  getallmodules {
    matricule
    nom
    coefficient
    volumeHoraire
    type
  }
}

```
- Body Tab:** The "Body" tab is selected. It shows the JSON response from the API call:

```

{
  "data": {
    "getallmodules": [
      {
        "matricule": "M101",
        "nom": "Algorithmique",
        "coefficient": 3,
        "volumeHoraire": 30,
        "type": "PROFESSIONNEL"
      }
    ]
  }
}

```
- Status Bar:** Status: 200 OK Time: 13.66 ms Size: 479 B