Exercises — Database

version **#1,0**

Yet Another Kind of Assistants 2024 <assistants@tickets.assistants.epita.fr>

Copyright

This document is for internal use at EPITA (website) only.

Contents

*https://intra.forge.epita.fr

File Tree

```
database/
├── pom.xml
└── src/
    └── main/
        ├── java/
        │   └── fr/
        │       └── epita/
        │           └── assistants/
        │               └── data/
        │                   └── model/
        │                       ├── CourseModel.java   (to submit)
        │                       └── StudentModel.java   (to submit)
        └── resources/
            └── application.properties
```

1 Goal

This exercise will make you explore the `model` layer of Quarkus.

The goal of this exercise is to create models that would generate a simple database when run with the Quarkus framework. You will have to create your models so that they match the database diagram shown below.

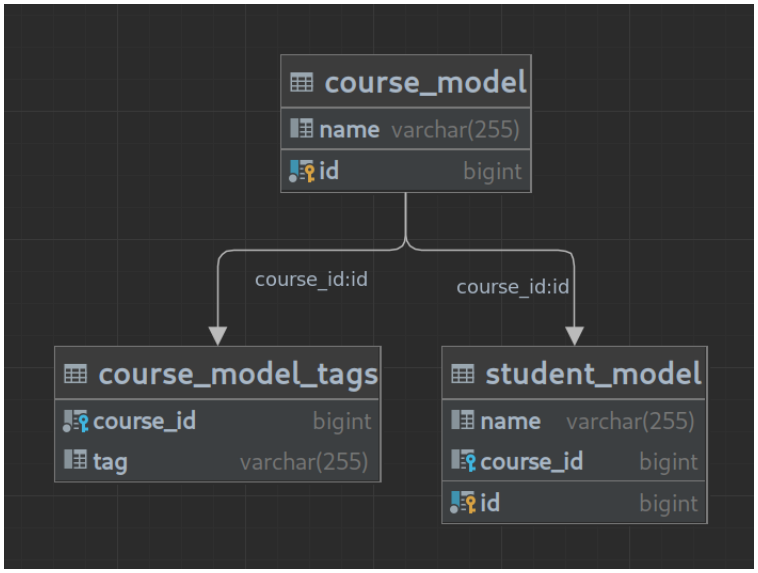
Tips

For projects with Quarkus, databases are never created manually and are instead autogenerated by the Hibernate ORM through the use of models.

Be careful!

The only models you have to implement are *course_model* and *student_model*. *course_model_tags* should be created automatically through the *@ElementCollection* annotation.

You do not need to write any database management or other complex behaviors. You do not even need to write any methods for this exercise.



Tips

In order to correctly visualise your database in IntellijIdea, you should open the *Database* tab in IntellijIdea, select your models and select *Diagrams -> Show diagram*

Be careful!

Be careful if you use your own computer, the *Database* tab is only visible if you are using the "Ultimate" version of IntellijIdea (default version on the PIE)

If the diagram view seems incorrect, use the *Diagrams -> Show diagram popup* option instead as the default option tends to not refresh correctly

To complete this exercise, the following annotations will help you, for the class:

- `@Entity`: Tells Quarkus that this class represents a model
- `@Table`: Lets you specify a name for the table containing a model

To complete this exercise, the following annotations will help you, for the attributes:

- `@Id`: Tells Quarkus that this field is the primary key of you model, you should use `@GeneratedValue` with it
- `@GeneratedValue`: Tells Quarkus to generate this value automatically according to a strategy specified in its arguments, you will need to use the **IDENTITY** strategy. If you want to understand the difference between the different values, you can read this.
- `@Column`: Lets you specify a name for the column containing a field
- `@JoinColumn`: Lets you specify which column Quarkus should use when making a *Join* request
- `@OneToMany`: Tells Quarkus that this field points to multiple objects from another model (You may use `@JoinColumn` with it)
- `@ManyToOne`: Tells Quarkus that multiple object of this class may point to a single object from another model
- `@ElementCollection`: Tells Quarkus that it should create a table with multiple elements pointing to this object (you may use its *joinColumns* argument)

Tips

You should use the *@OneToMany* and *@ManyToOne* to define the relationships between *course_model* and *student_model*.

You may find these guides on the @OneToMany and the @ManyToOne relationships useful.

2 How to start

2.1 Set-up the database

First, let us configure a PostgreSQL-specific environment variable:

```
42sh$ echo 'export PGDATA="$HOME/postgres_data"' >> ~/.bashrc
42sh$ echo 'export PGHOST="/tmp"' >> ~/.bashrc
42sh$ source ~/.bashrc
```

This first line adds a PGDATA environment variable to your .bashrc, containing the location of your choice for storing postgres data. The second specifies the host name of the machine on which the server will run. As the value begins with a slash, it will be used as the directory containing a socket on which postgres will listen. Then, let us initialize a new PostgreSQL database cluster. It will generate default databases and configurations in the $PGDATA directory.

```
42sh$ nix-shell -p postgresql
42sh$ initdb --locale "$LANG" -E UTF8
```

Be careful!

If you experience a permission denied, you may need to restore your rights using chmod 755 ~

2.2 Create a database

Your server is up and running! You have to set the `DB_USERNAME` variable to your login. PostgreSQL offers an interactive shell that connects to this server and behaves as a front-end for your operations on databases:

```
42sh$ export DB_USERNAME=<login>
42sh$ postgres -k "$PGHOST"
```

The previous command launches the database server, for the following ones, you need to enter a new shell in order to connect to your database.

```
42sh$ psql postgres
```

`psql` takes the name of the database you want to connect to. Here, we choose the default database, `postgres`. These commands will allow you to create and own a database named jws:

```
-- Give yourself all the rights
-- If you are not on the PIE, <login> should be your username
postgres=# ALTER ROLE "<login>" SUPERUSER;
-- Create a database named courses
postgres=# CREATE DATABASE courses OWNER <login>;
-- Connect to the courses database
postgres=# \c courses;
-- Create the courses schema
postgres=# CREATE SCHEMA courses;
-- Exit PostgreSQL
postgres=# \q
```

2.3 Given Files

In order to start the project in good conditions, a tarball is available on the intranet. It contains all the files needed for the proper functioning of the project.

You **should** decompress this tarball and open the directory with IntelliJ before reading the rest of the subject.

When the project is open in your IDE, you can type the following command to start the server.

```
42sh$ export DB_USERNAME=<login>
42sh$ mvn quarkus:dev
```

It will automatically create your tables and columns. You can observe them by connecting to your database, in a new shell:

```
42sh$ psql courses
```

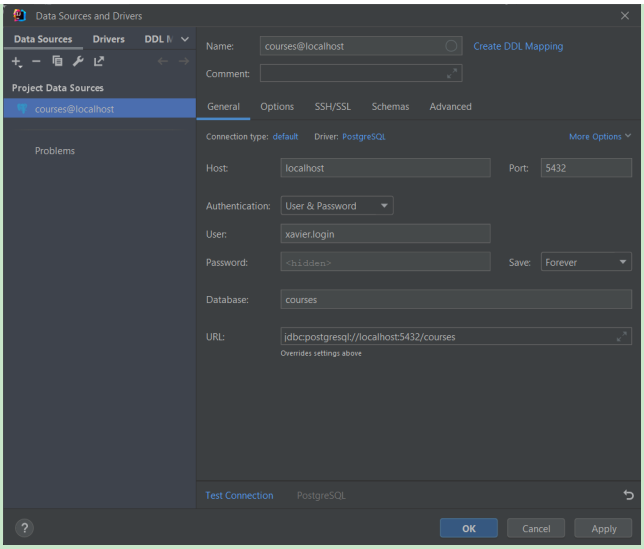
and doing these commands:

```
-- Show tables
courses=#$ SELECT tablename FROM pg_tables WHERE schemaname = 'courses';
-- tablename
-- ------------------
-- course_model
-- course_model_tags
-- student_model
-- (3 rows)

-- Exit PostgreSQL
courses=#$ \q
```

Tips

You can also see the content of your database using IntellijIdea. In order to do so, you must navigate to the *Database* tab, and then click on *new -> datasource -> postgresql*. And then, you should see the following screen:



Then, you must put your login in the *User* textfield and *courses* in the *Database* one.

Be careful!

During JWS you will need to put *jws* instead of *courses* for the *Database* textfield.

You mean it's working? For real this time?