Mise en œuvre de JDBC avec PostgreSQL

- Nom : Laeticia Mallat
- Matricule: 232951

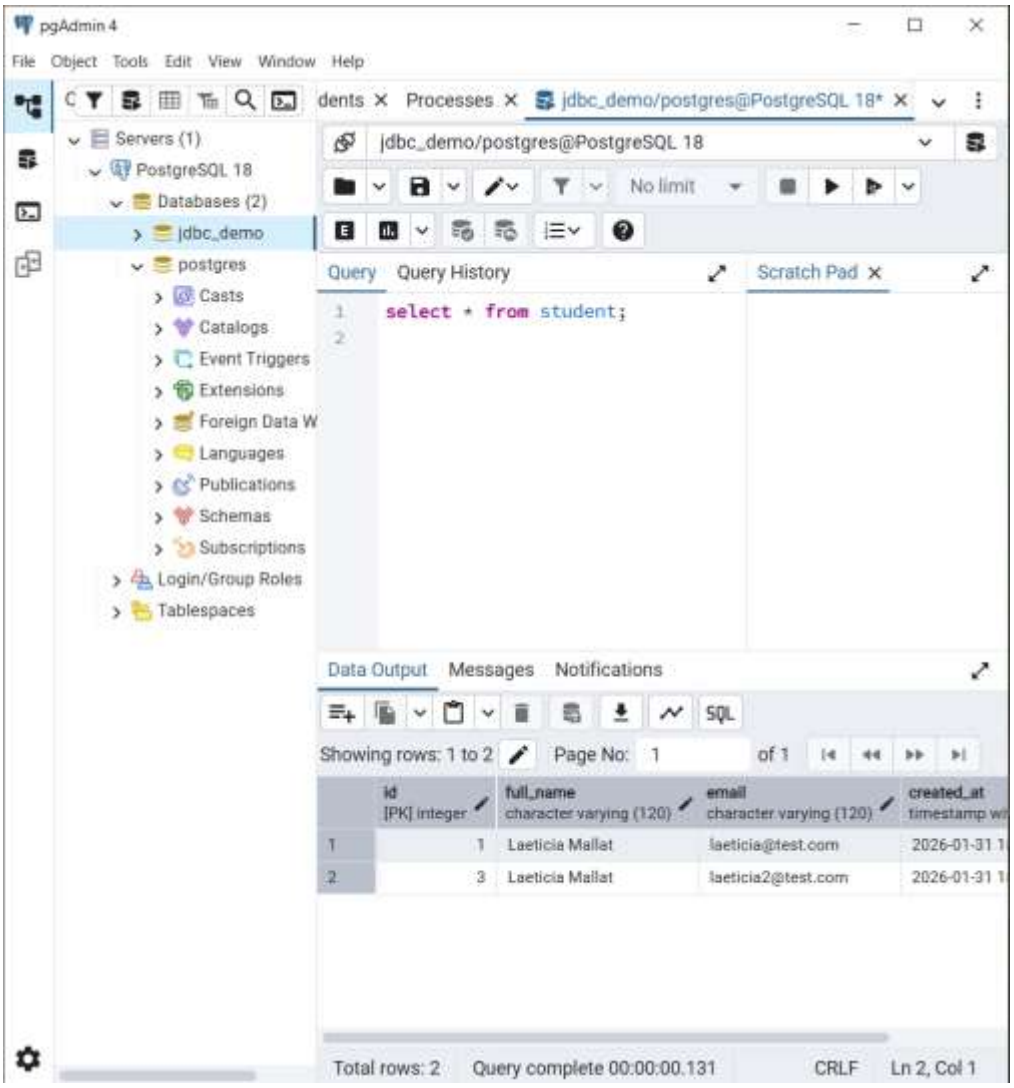
- Cours : Applications distribuée
- Année : 2025–2026
- Lien GitHub : https://github.com/Laeticia005/projects

Objectif du projet:

L'objectif de ce projet est de mettre en œuvre JDBC afin de permettre à une application Java d'accéder à une base de données PostgreSQL.

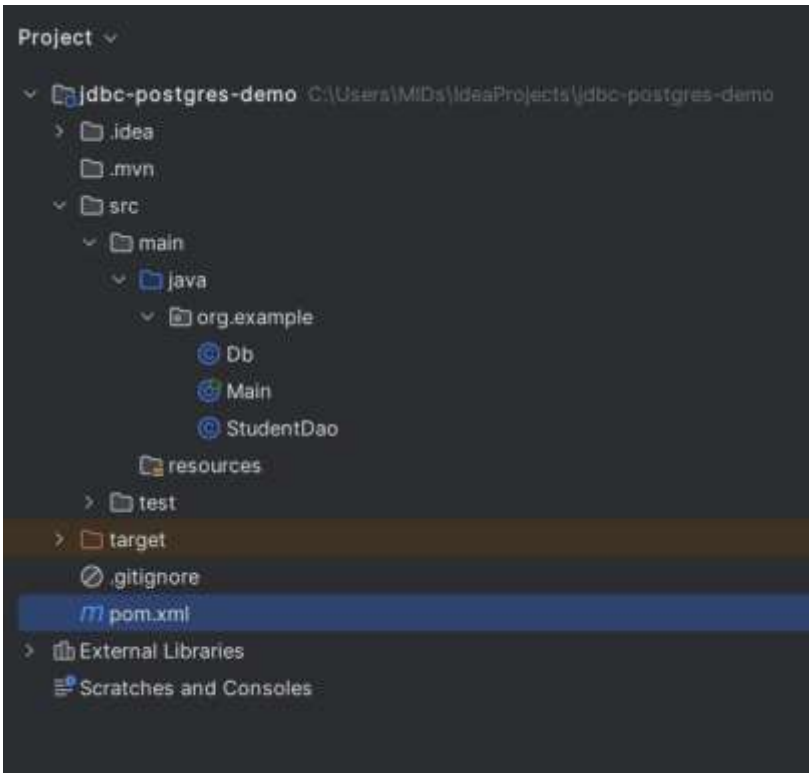
Environnement et outils utilises:

SGBD : PostgreSQL
Interface graphique : pgAdmin 4
JDK : Java 24
Gestionnaire de dépendances : Maven
Driver JDBC : PostgreSQL JDBC Driver 42.7.3
IDE : IntelliJ IDEA



Architecture du projet:

Le projet est structuré selon une architecture simple et claire.
La classe **Db.java** est responsable de l'établissement de la connexion à la base de données PostgreSQL via JDBC.
La classe **StudentDao.java** contient les méthodes permettant d'exécuter les opérations CRUD (INSERT et SELECT) à l'aide de PreparedStatement.
La classe **Main.java** représente l'application principale et appelle les méthodes du DAO pour interagir avec la base de données.



Main.java | Db.java | StudentDao.java | .gitignore | pom.xml (jdbc-postgres-demo)

```java
package org.example;

import java.sql.*;

public class StudentDao {

    public void insertStudent(Connection con,
                              String name,
                              String email) throws SQLException {

        String sql =
                "INSERT INTO student(full_name, email) VALUES (?, ?)";

        try (PreparedStatement ps = con.prepareStatement(sql)) {
            ps.setString(parameterIndex: 1, name);
            ps.setString(parameterIndex: 2, email);
            ps.executeUpdate();
        }
    }
    public void listStudents(Connection con) throws SQLException {
        String sql = "SELECT id, full_name, email FROM student";

        try (PreparedStatement ps = con.prepareStatement(sql);
             ResultSet rs = ps.executeQuery()) {

            while (rs.next()) {
                System.out.println(
                        rs.getInt(columnLabel: "id") + " | " +
                                rs.getString(columnLabel: "full_name") + " | " +
                                rs.getString(columnLabel: "email")
                );
            }
```

c-postgres-demo > src > main > java > org > example > StudentDao

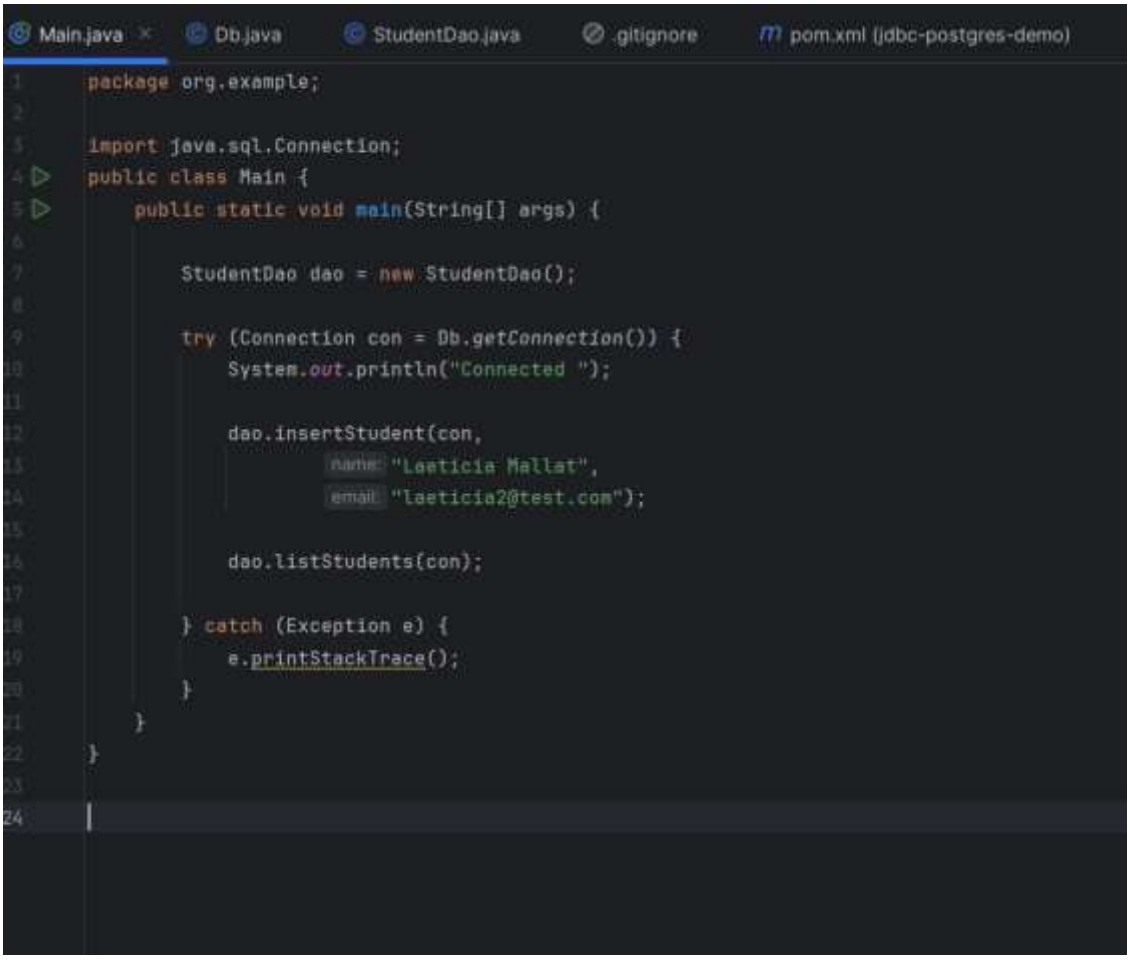
Main.java | Db.java | StudentDao.java | .gitignore | pom.xml (jdbc-postgres-demo)

```java
package org.example;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class Db {
    private static final String URL =
            "jdbc:postgresql://localhost:5432/jdbc_demo";
    private static final String USER = "postgres";
    private static final String PASS = "letiusj123";

    public static Connection getConnection() throws SQLException {
        return DriverManager.getConnection(URL, USER, PASS);
    }
    }
```


Main.java
Db.java
StudentDao.java
.gitignore
pom.xml (jdbc-postgres-demo)
package org.example;
import java.sql.Connection;
public class Main {
public static void main(String[] args) {
StudentDao dao = new StudentDao();
try (Connection con = Db.getConnection()) {
System.out.println("Connected ");
dao.insertStudent(con,
name: "Laeticia Mallat",
email: "laeticia2@test.com");
dao.listStudents(con);
} catch (Exception e) {
e.printStackTrace();
}
}
}

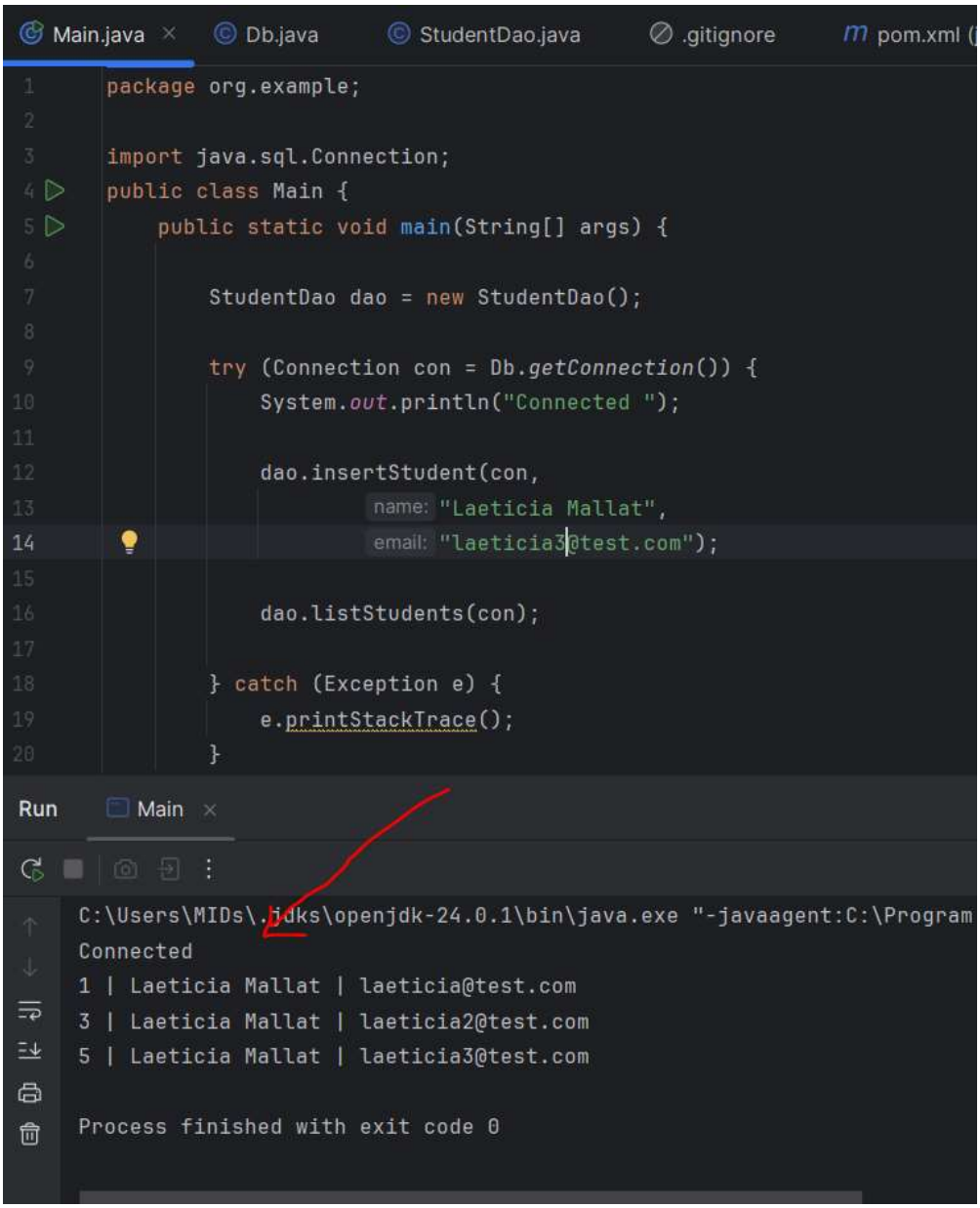
Dans le fichier pom.xml, j'ai ajouté la dépendance du driver JDBC PostgreSQL afin de permettre à l'application Java de communiquer avec la base de données PostgreSQL. Cette dépendance est gérée par Maven, qui télécharge automatiquement la bibliothèque nécessaire.
J'ai également ajouté le plugin exec-maven-plugin, qui permet de spécifier la classe principale de l'application et de faciliter son exécution via Maven.

```xml
<dependencies>
    <dependency>
        <groupId>org.postgresql</groupId>
        <artifactId>postgresql</artifactId>
        <version>42.7.3</version>
    </dependency>
</dependencies>

<build>
    <plugins>
        <plugin>
            <groupId>org.codehaus.mojo</groupId>
            <artifactId>exec-maven-plugin</artifactId>
            <version>3.3.0</version>
            <configuration>
                <mainClass>org.example.Main</mainClass>
            </configuration>
        </plugin>
    </plugins>
</build>
```

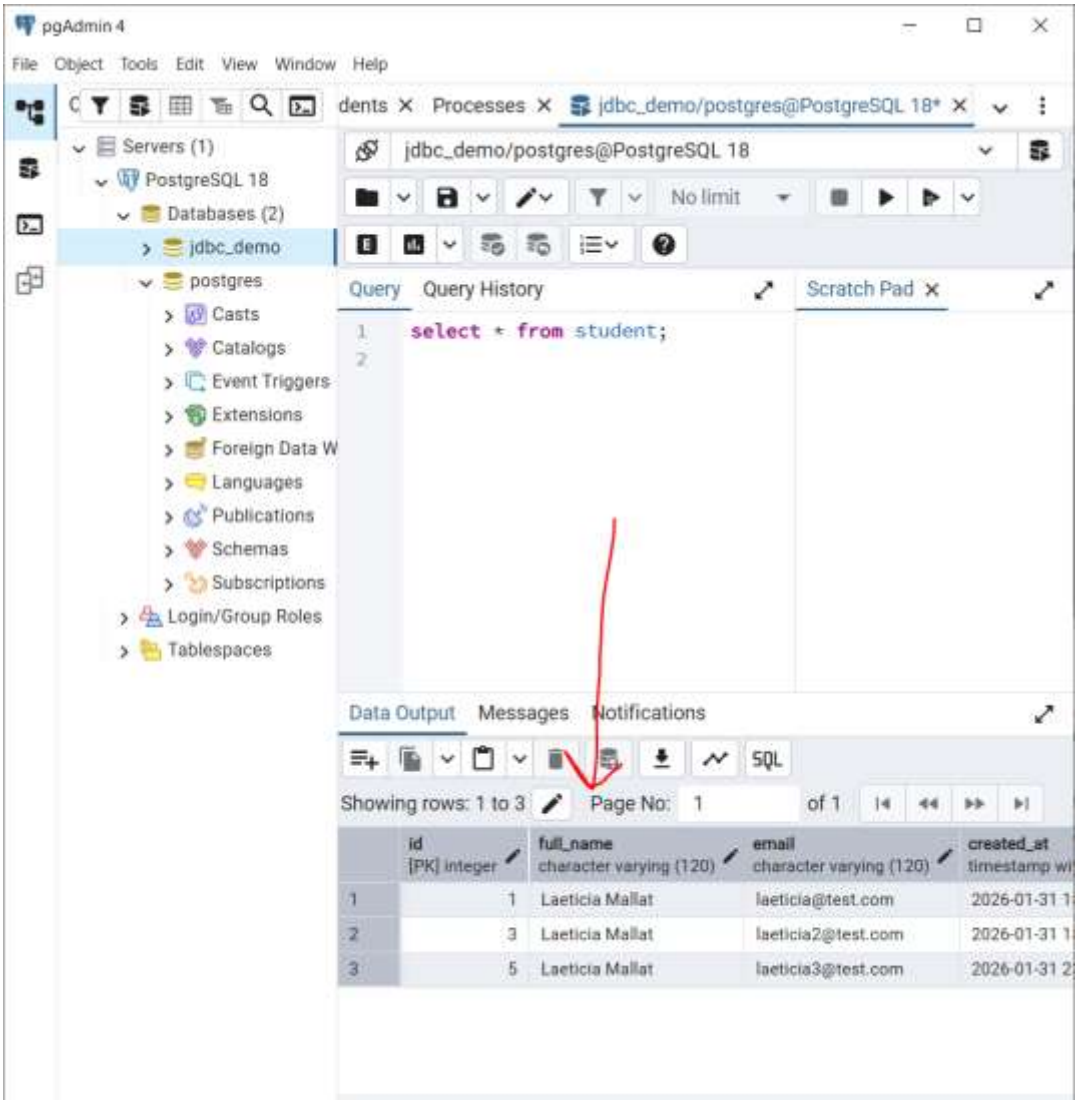
Mise en œuvre JDBC:

La connexion à PostgreSQL est réalisée à l'aide de l'API JDBC. L'URL JDBC (jdbc:postgresql://localhost:5432/jdbc_demo) permet d'indiquer l'emplacement du serveur et la base utilisée. Le driver JDBC PostgreSQL, ajouté via Maven, agit comme intermédiaire entre Java et PostgreSQL.
Les opérations CRUD sont exécutées à l'aide de PreparedStatement, ce qui permet de remplacer les paramètres ? par des valeurs via setString() et d'éviter les injections SQL. Les requêtes sont exécutées depuis Java et les résultats sont récupérés sous forme de ResultSet.



Résultats et conclusion:

Les tests ont montré que les opérations exécutées depuis l'application Java sont correctement persistées dans la base de données PostgreSQL. Les résultats peuvent être visualisés dans pgAdmin à l'aide de requêtes SELECT.
Ce projet a permis de comprendre concrètement le rôle de JDBC comme pont entre une application Java et une base de données relationnelle, ainsi que la mise en œuvre des opérations CRUD.


pgAdmin 4
File Object Tools Edit View Window Help
Servers (1)
PostgreSQL 18
Databases (2)
jdbc_demo
postgres
Casts
Catalogs
Event Triggers
Extensions
Foreign Data W
Languages
Publications
Schemas
Subscriptions
Login/Group Roles
Tablespaces
dents
Processes
jdbc_demo/postgres@PostgreSQL 18*
jdbc_demo/postgres@PostgreSQL 18
No limit
Query
Query History
Scratch Pad
1 select * from student;
2
Data Output
Messages
Notifications
SQL
Showing rows: 1 to 3
Page No: 1
of 1
id
[PK] integer
full_name
character varying (120)
email
character varying (120)
created_at
timestamp wi
1 1 Laeticia Mallat laeticia@test.com 2026-01-31 1
2 3 Laeticia Mallat laeticia2@test.com 2026-01-31 1
3 5 Laeticia Mallat laeticia3@test.com 2026-01-31 2