Spring Boot avec PostgreSQL

Pour mettre en place notre application nous allons utiliser toujours la même application de ToDoList avec H2 pour l'utiliser avec postgreSQL.

On importe les dépendances de postgreSQL dans le fichier pom.xml

```xml
<dependency>
    <groupId>org.postgresql</groupId>
    <artifactId>postgresql</artifactId>
    <scope>runtime</scope>
</dependency>
```

On paramètre le fichier application.properties

```
spring.application.name=demo

#spring.datasource.url=jdbc:h2:file:C:/Users/PC/test1;DB_CLOSE_ON_EXIT=FALSE
#spring.datasource.driverClassName=org.h2.Driver
#spring.datasource.username=sa
#spring.datasource.password=
#spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
#spring.jpa.hibernate.ddl-auto=update
#spring.h2.console.enabled=true
#spring.h2.console.path=/h2-console
#spring.main.allow-circular-references=true

spring.datasource.url=jdbc:postgresql://localhost:5432/test1
spring.datasource.username=postgres
spring.datasource.password=yannis
spring.datasource.driver-class-name=org.postgresql.Driver
# JPA properties
spring.jpa.database-platform=org.hibernate.dialect.PostgreSQLDialect
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
```

On crée notre model

```java
package crud.example.demo.model;

import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.ResponseStatus;

import jakarta.persistence.*;

@Entity
@Table(name = "todolist")
public class ToDo {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String titre;
    private String description;
    private boolean status;

    @ResponseStatus(HttpStatus.NOT_FOUND)
    public class ResourceNotFoundException extends RuntimeException {
        public ResourceNotFoundException(String message) {
            super(message);
        }
    }

    public ToDo() {}
```

```java
    public ToDo(String titre) {
        this.titre = titre;
        this.description = description;
        this.status = false;
    }

    public Long getId() { return id; }
    public String getTitre() { return titre; }
    public void setTitre(String titre) { this.titre = titre; }
    public String getDescription() { return description; }
    public void setDescription() { this.description = description; }
    public boolean isStatus() { return status; }
    public void setStatus(boolean status) { this.status = status; }
}
```

Le controller

```java
package crud.example.demo.controller;

import crud.example.demo.model.ToDo;
import crud.example.demo.service.ToDoService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Pageable;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.*;

//import java.util.HashMap;
import java.util.List;
//import java.util.Optional;

@RestController
@RequestMapping("/lister")
@CrossOrigin(origins = "*")
public class ToDoController {
    @Autowired
    private ToDoService toDoService;

    @ResponseStatus(HttpStatus.NOT_FOUND)
    public class ResourceNotFoundException extends RuntimeException {
        public ResourceNotFoundException(String message) {
            super(message);
        }
    }
```

```java
    @GetMapping
    public List<ToDo> getAllTasks(Pageable pageable) {
        return toDoService.getAllToDos(pageable);
    }

    @GetMapping("/{id}")
    public ResponseEntity getTaskById(@PathVariable Long id) {
        return ResponseEntity.ok(toDoService.getToDoById(id));
    }

    @PostMapping
    public ResponseEntity<ToDo> createTask(@RequestBody @Validated ToDo toDo) {
        ToDo createdToDo = toDoService.createToDo(toDo);
        return new ResponseEntity<>(createdToDo, HttpStatus.CREATED);
    }

    @PutMapping("/{id}")
    public ResponseEntity<ToDo> updateTask(@PathVariable Long id, @RequestBody ToDo toDo) {
        ToDo updatedToDo = toDoService.updateToDo(id, toDo);
        return ResponseEntity.ok(updatedToDo);
    }

    @PatchMapping("/{id}")
    public ResponseEntity<ToDo> updateTache(@PathVariable Long id, @Validated ToDo toDo){
        ToDo updateTache = toDoService.updateToDo(id, toDo);
        return ResponseEntity.ok(updateTache);
    }
```

```java
    @DeleteMapping("/{id}")
    public void deleteTask(@PathVariable Long id) {
        toDoService.deleteToDo(id);
    }

}
```

Le service

```java
package crud.example.demo.service;

import crud.example.demo.model.ToDo;
import crud.example.demo.repository.ToDoRepository;
import org.springframework.stereotype.Service;
import org.springframework.web.bind.annotation.ResponseStatus;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.dao.DuplicateKeyException;
import org.springframework.data.domain.Pageable;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;

import java.util.List;

@Service
public class ToDoService {
    @Autowired
    private ToDoRepository toDoRepository;

    @ResponseStatus(HttpStatus.NOT_FOUND)
    public class ResourceNotFoundException extends RuntimeException {
        public ResourceNotFoundException(String message) {
            super(message);
        }
    }

    public List<ToDo> getAllToDos(Pageable pageable) {
        return toDoRepository.findAll();
    }
```

```java
    public ResponseEntity getToDoById(Long id){
        return ResponseEntity.ok(toDoRepository.findById(id));
    }

    public ToDo createToDo(ToDo toDo) {
        if (toDoRepository.existsByTitre(toDo.getTitre())) {
            throw new DuplicateKeyException(msg:"La tâche avec ce titre existe déjà");
        }
        return toDoRepository.save(toDo);
    }

    public ToDo updateToDo(Long id, ToDo updateToDo){
        return toDoRepository.findById(id).map(toDo -> {
            toDo.setTitre(updateToDo.getTitre());
            toDo.setDescription();
            toDo.setStatus(updateToDo.isStatus());
            return toDoRepository.save(toDo);
        }).orElseThrow(() -> new ResourceNotFoundException("La tâche n'a pas été trouvée avec l'id" + id));
    }

    public void deleteToDo(Long id) {
        if (!toDoRepository.existsById(id)) {
            throw new ResourceNotFoundException("La tâche n'a pas été trouvée avec l'id" + id);
        }
        toDoRepository.deleteById(id);
    }

}
```

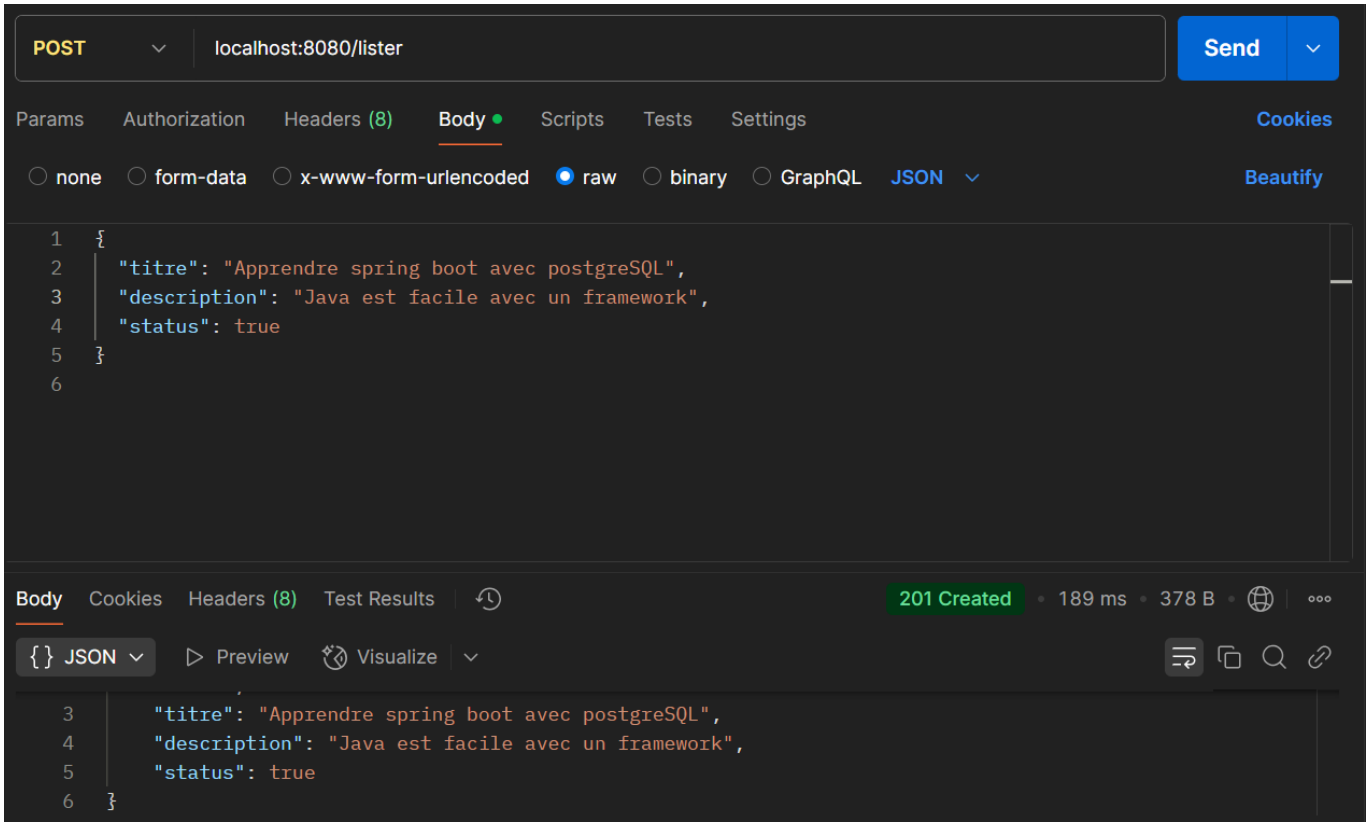
Le respository

```java
package crud.example.demo.repository;

import crud.example.demo.model.ToDo;
import jakarta.websocket.Decoder.Text;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

public interface ToDoRepository extends JpaRepository<ToDo, Long>{
    boolean existsByTitre(String titre);
}
```

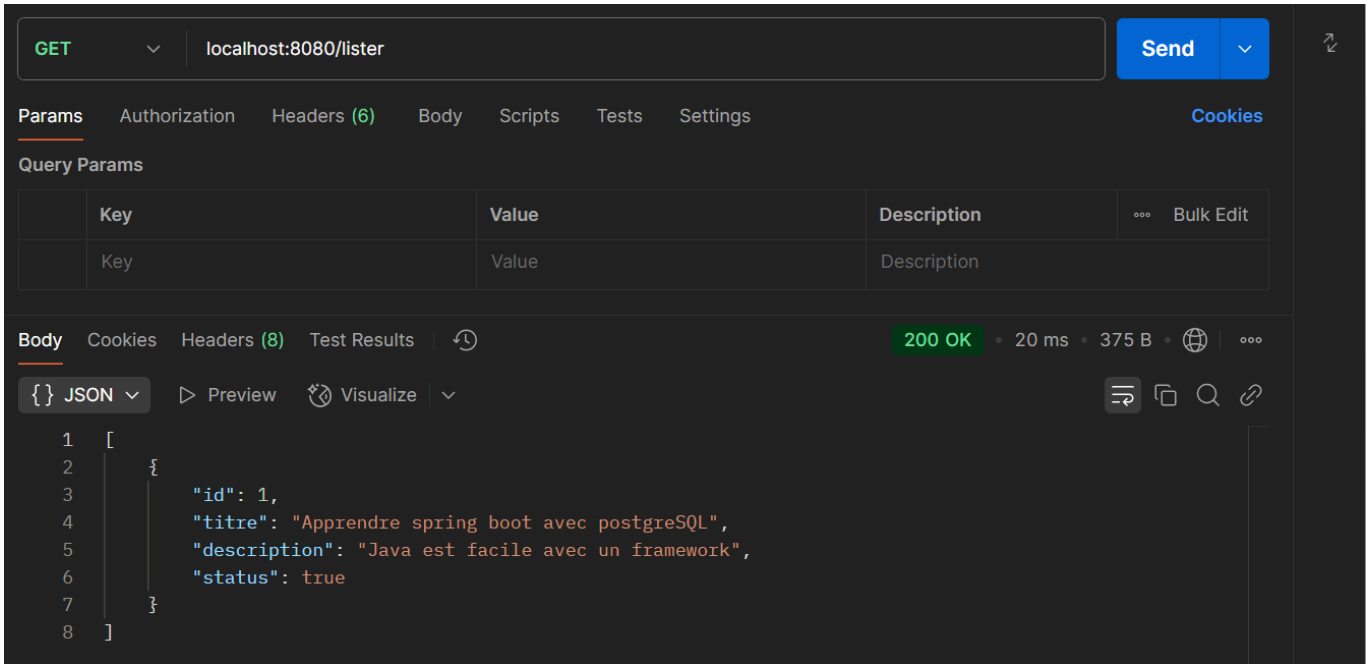
On démarre l'application

```
PS C:\Users\PC\Downloads\demo\demo> mvn spring-boot:run
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------< crud.example:demo >-------------------------
[INFO] Building demo 0.0.1-SNAPSHOT
```

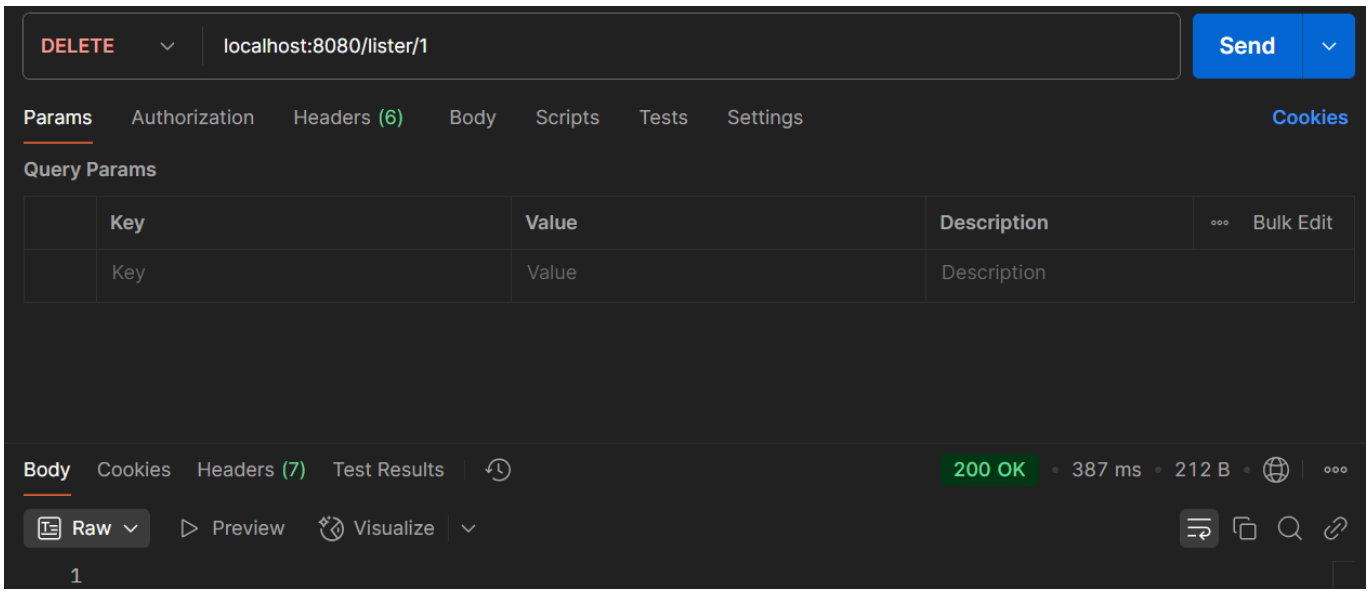
On ajoute un enregistrement avec la méthode POST



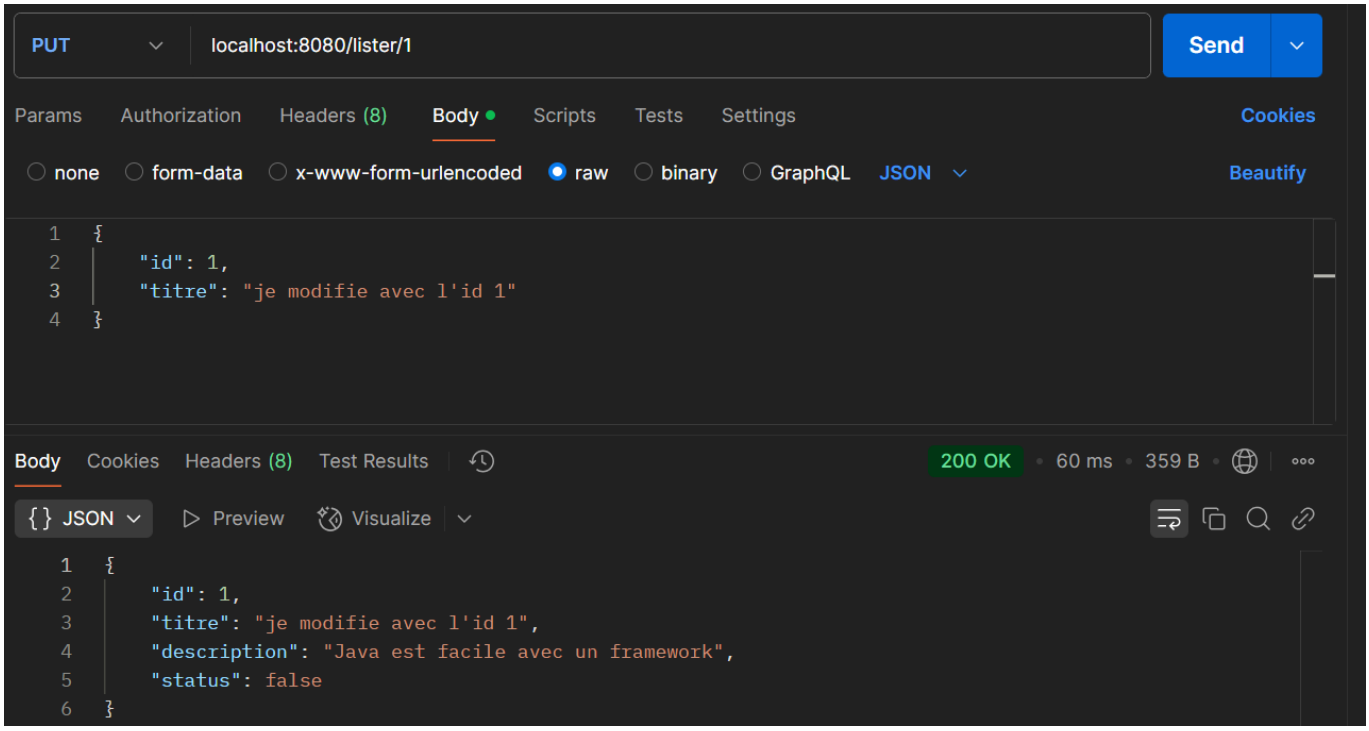
On affiche avec la méthode GET



On supprime avec la méthode DELETE



On Modifie avec la méthode PUT



On peut toutefois vérifier si la base de données est créée dans postgreSQL