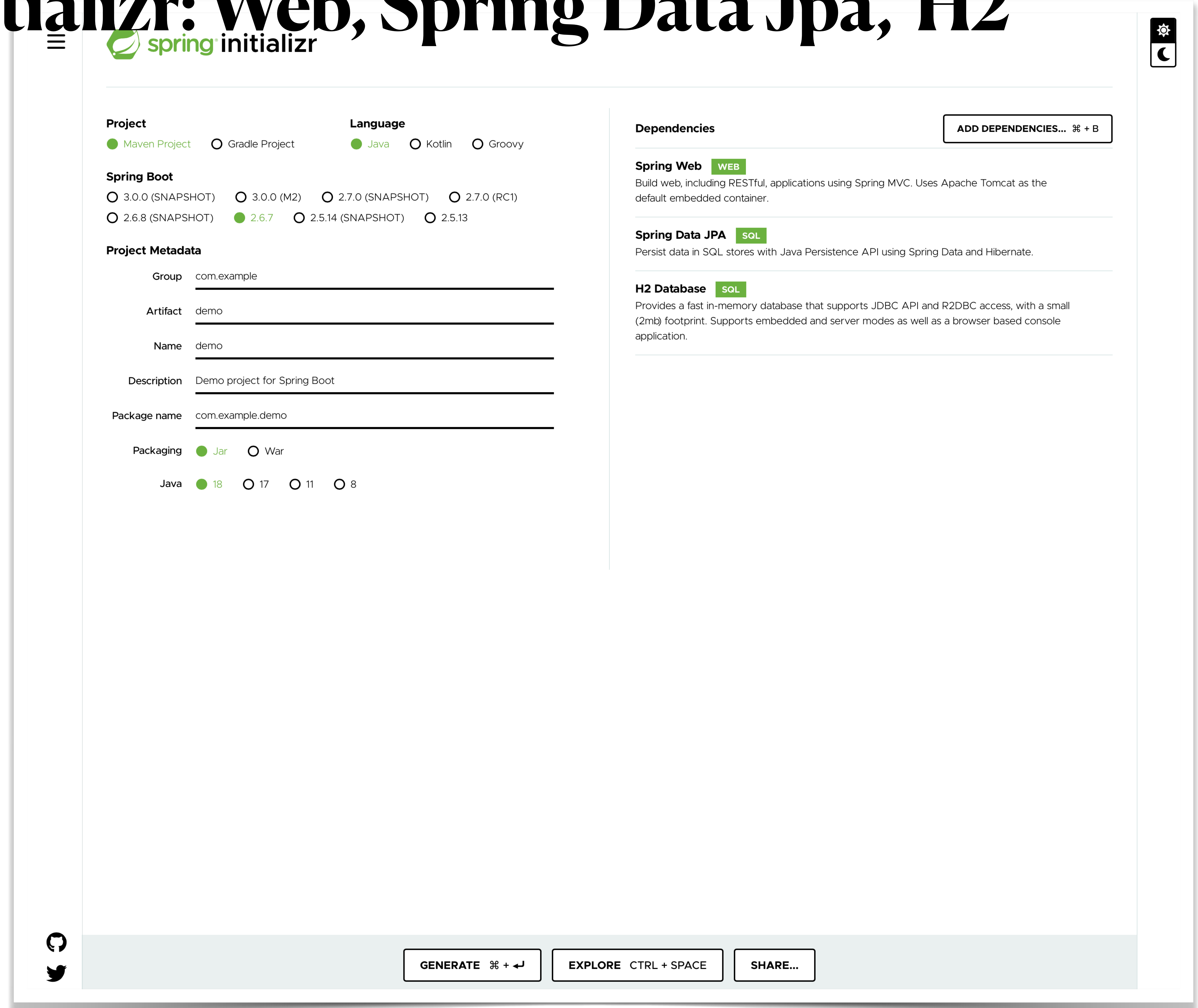
@ManyToMany

Spring Boot Recipe

1. Spring Initializr (Web, Spring Data JPA, H2): generate & import
2. Add swagger-ui dependency
3. Create model: Student and Lab
4. Create application layers (empty classes/interfaces with superpowers)
5. Add Hibernate superpowers to the model classes **(@ManyToMany)**
6. **data.sql**
7. Endpoints call repository
8.**nameGeneratesQuery**, **@Query**

Spring Initializr: Web, Spring Data Jpa, H2



Project
[✓ Maven Project] Gradle Project

Language
[✓ Java] Kotlin Groovy

Spring Boot
3.0.0 (SNAPSHOT) 3.0.0 (M2) 2.7.0 (SNAPSHOT) 2.7.0 (RC1)
2.6.8 (SNAPSHOT) [✓ 2.6.7] 2.5.14 (SNAPSHOT) 2.5.13

Project Metadata

Group	com.example
Artifact	demo
Name	demo
Description	Demo project for Spring Boot
Package name	com.example.demo
Packaging	[✓ Jar] War
Java	[✓ 18] 17 11 8

Dependencies

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Spring Data JPA SQL
Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

H2 Database SQL
Provides a fast in-memory database that supports JDBC API and R2DBC access, with a small (2mb) footprint. Supports embedded and server modes as well as a browser based console application.

Swagger

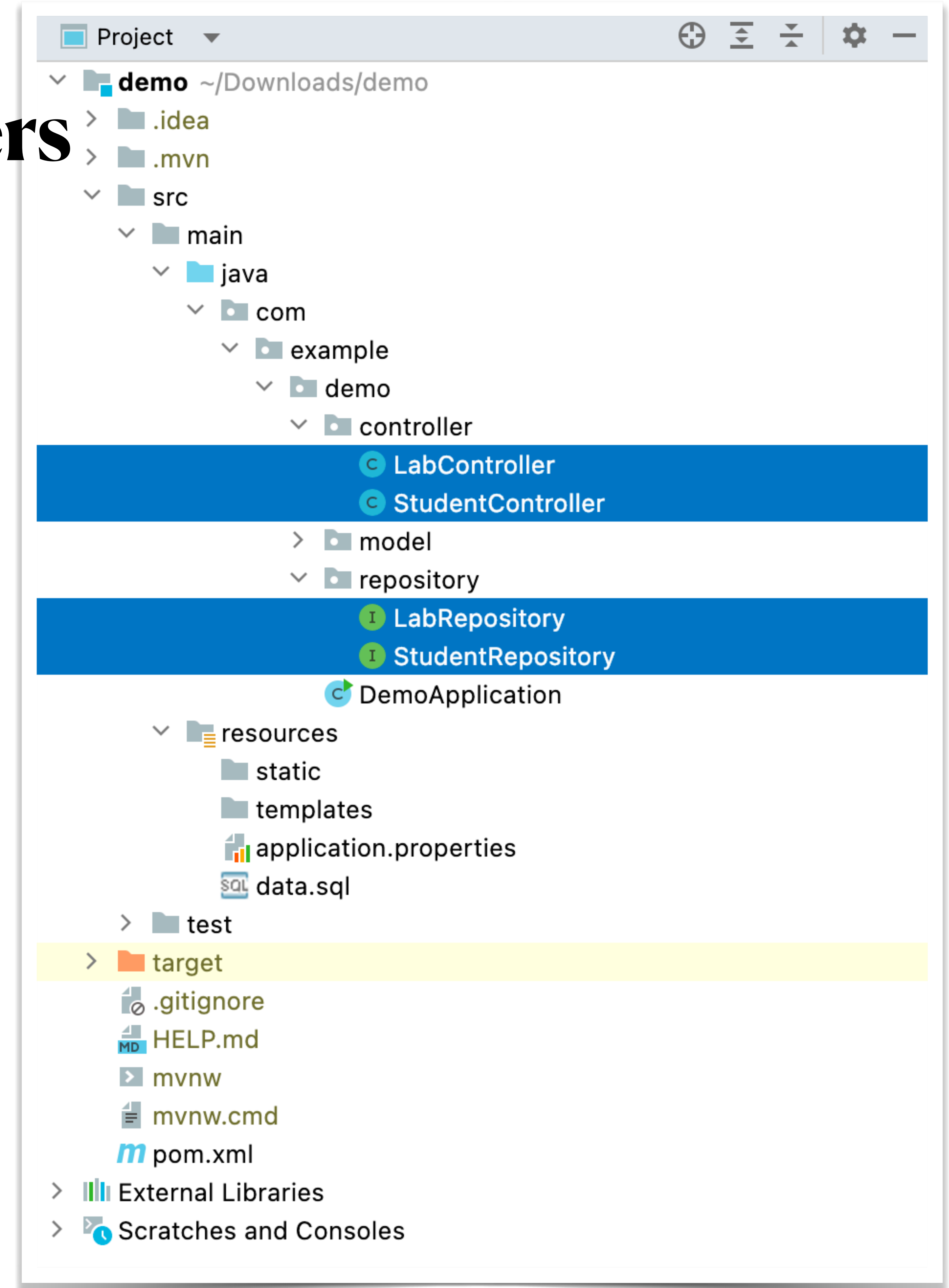
```xml
<dependency>
  <groupId>org.springdoc</groupId>
  <artifactId>springdoc-openapi-ui</artifactId>
  <version>1.6.8</version>
</dependency>
```

The model

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

import java.util.Set;

public class Lab {

    private Long id;

    private String title;
    private String description;
    private Set<Student> students;

    public Lab() {
    }

    public Lab(Long id, String title, String description, Set<Student> students) {
        this.id = id;
        this.title = title;
        this.description = description;
        this.students = students;
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public Set<Student> getStudents() {
        return students;
    }

    public void setStudents(Set<Student> students) {
        this.students = students;
    }
}
```

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

import java.util.Set;

public class Student {

    private Long id;

    private String firstName;
    private String lastName;

    private Set<Lab> labs;

    public Student() {
    }

    public Student(Long id, String firstName, String lastName, Set<Lab> labs) {
        this.id = id;
        this.firstName = firstName;
        this.lastName = lastName;
        this.labs = labs;
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public Set<Lab> getLabs() {
        return labs;
    }

    public void setLabs(Set<Lab> labs) {
        this.labs = labs;
    }
}
```

Application layers



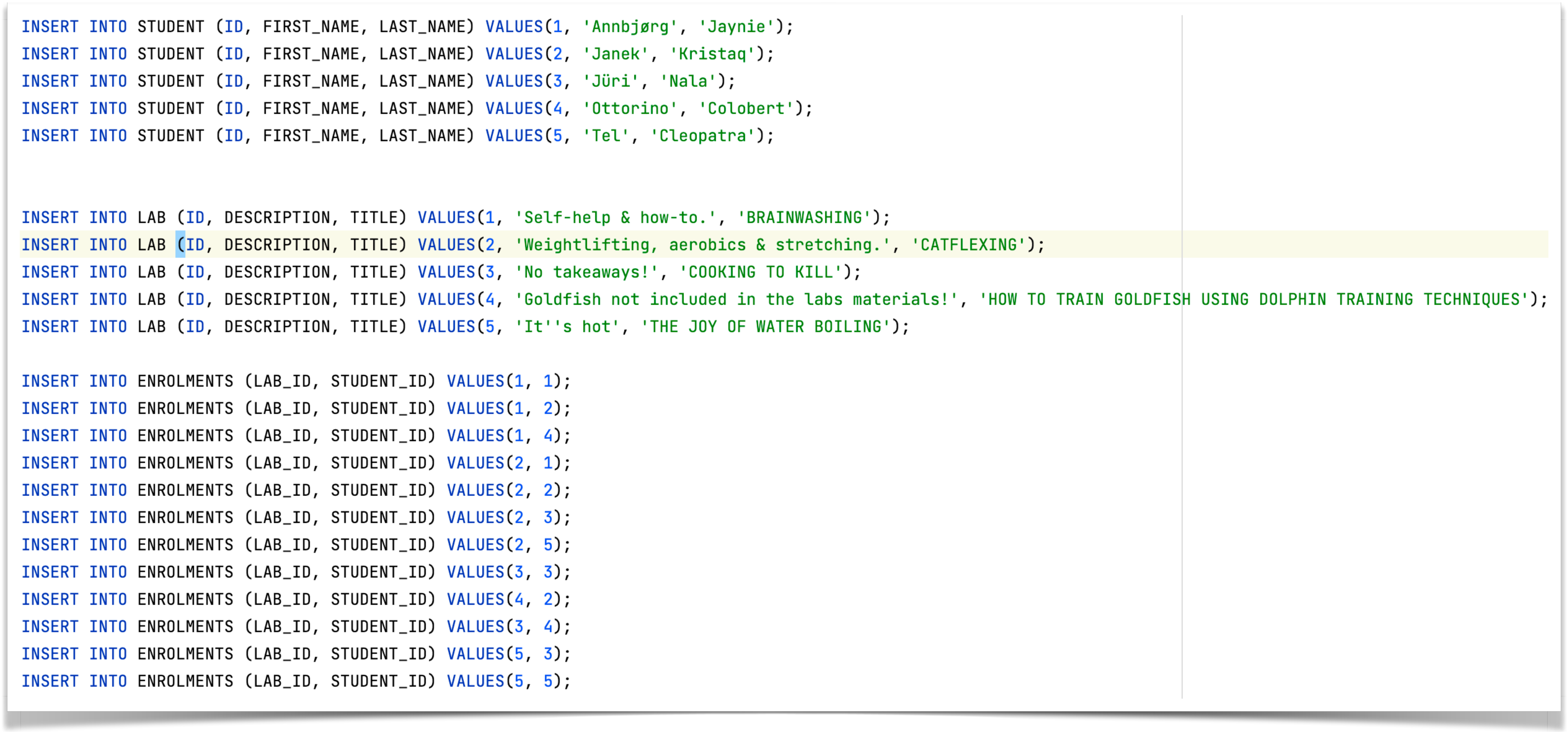
```
Project
demo ~/Downloads/demo
  .idea
  .mvn
  src
    main
      java
        com
          example
            demo
              controller
                LabController
                StudentController
              model
              repository
                LabRepository
                StudentRepository
              DemoApplication
      resources
        static
        templates
        application.properties
        data.sql
    test
  target
  .gitignore
  HELP.md
  mvnw
  mvnw.cmd
  pom.xml
External Libraries
Scratches and Consoles
```

@ManyToMany

```java
@Entity
public class Lab {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String title;
    private String description;
    // CascadeType.ALL is the easiest to use in the code
    // but may cause us some troubles during runtime - why?
    @ManyToMany(cascade = CascadeType.ALL)
    @JoinTable(name = "enrolments",
     joinColumns = @JoinColumn(name = "lab_id"),
     inverseJoinColumns = @JoinColumn(name = "student_id"))
    @JsonIgnoreProperties(value = {"labs"})
    private Set<Student> students;

    // constructors /getters / setters

}
```

```java
@Entity
public class Student {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String firstName;
    private String lastName;

    @ManyToMany(mappedBy = "students")
    @JsonIgnoreProperties(value = {"students"})
    private Set<Lab> labs;

      // constructors /getters / setters

}
```

data.sql



```sql
INSERT INTO STUDENT (ID, FIRST_NAME, LAST_NAME) VALUES(1, 'Annbjørg', 'Jaynie');
INSERT INTO STUDENT (ID, FIRST_NAME, LAST_NAME) VALUES(2, 'Janek', 'Kristaq');
INSERT INTO STUDENT (ID, FIRST_NAME, LAST_NAME) VALUES(3, 'Jüri', 'Nala');
INSERT INTO STUDENT (ID, FIRST_NAME, LAST_NAME) VALUES(4, 'Ottorino', 'Colobert');
INSERT INTO STUDENT (ID, FIRST_NAME, LAST_NAME) VALUES(5, 'Tel', 'Cleopatra');


INSERT INTO LAB (ID, DESCRIPTION, TITLE) VALUES(1, 'Self-help & how-to.', 'BRAINWASHING');
INSERT INTO LAB (ID, DESCRIPTION, TITLE) VALUES(2, 'Weightlifting, aerobics & stretching.', 'CATFLEXING');
INSERT INTO LAB (ID, DESCRIPTION, TITLE) VALUES(3, 'No takeaways!', 'COOKING TO KILL');
INSERT INTO LAB (ID, DESCRIPTION, TITLE) VALUES(4, 'Goldfish not included in the labs materials!', 'HOW TO TRAIN GOLDFISH USING DOLPHIN TRAINING TECHNIQUES');
INSERT INTO LAB (ID, DESCRIPTION, TITLE) VALUES(5, 'It''s hot', 'THE JOY OF WATER BOILING');

INSERT INTO ENROLMENTS (LAB_ID, STUDENT_ID) VALUES(1, 1);
INSERT INTO ENROLMENTS (LAB_ID, STUDENT_ID) VALUES(1, 2);
INSERT INTO ENROLMENTS (LAB_ID, STUDENT_ID) VALUES(1, 4);
INSERT INTO ENROLMENTS (LAB_ID, STUDENT_ID) VALUES(2, 1);
INSERT INTO ENROLMENTS (LAB_ID, STUDENT_ID) VALUES(2, 2);
INSERT INTO ENROLMENTS (LAB_ID, STUDENT_ID) VALUES(2, 3);
INSERT INTO ENROLMENTS (LAB_ID, STUDENT_ID) VALUES(2, 5);
INSERT INTO ENROLMENTS (LAB_ID, STUDENT_ID) VALUES(3, 3);
INSERT INTO ENROLMENTS (LAB_ID, STUDENT_ID) VALUES(4, 2);
INSERT INTO ENROLMENTS (LAB_ID, STUDENT_ID) VALUES(3, 4);
INSERT INTO ENROLMENTS (LAB_ID, STUDENT_ID) VALUES(5, 3);
INSERT INTO ENROLMENTS (LAB_ID, STUDENT_ID) VALUES(5, 5);
```

The endpoints

```java
@RestController
public class LabController {

   // The right way to inject dependencies
    private final LabRepository labRepository;

    public LabController(LabRepository labRepository) {
        this.labRepository = labRepository;
    }

    @GetMapping("/labs")
    public ResponseEntity<List<Lab>> getAllLabs() {
        List<Lab> labs = labRepository.findAll();
        return ResponseEntity
                .ok()
                .body(labs);
    }

    @PostMapping("/labs")
    public ResponseEntity<Lab> createLab(@RequestBody Lab lab) {
        Lab result = labRepository.save(lab);
        return ResponseEntity
                .ok()
                .body(result);
    }

}
```

```java
@RestController
public class StudentController {

    private final StudentRepository studentRepository;

    public StudentController(StudentRepository studentRepository) {
        this.studentRepository = studentRepository;
    }

    @GetMapping("/students")
    public ResponseEntity<List<Student>> getAllLabs() {

        students = studentRepository.findAll();

        return ResponseEntity
                .ok()
                .body(students);
    }

    @PostMapping("/students")
    public ResponseEntity<Student> createLab(@RequestBody Student student) {
        Student result = studentRepository.save(student);
        return ResponseEntity
                .ok()
                .body(result);
    }
}
```

nameGeneratesQuery

```java
public interface LabRepository extends JpaRepository<Lab, Long> {
    /*
        1. The name is not trivial!
        2. Any potential issues with the return type?
     */
    Lab findByTitle(String title);
}
```

```java
@GetMapping("/labs/{title}")
public ResponseEntity<Lab> getByTitle(@PathVariable String title) {
    Lab lab = labRepository.findByTitle(title);
    return ResponseEntity
            .ok()
            .body(lab);
}
```

@Query

```java
public interface StudentRepository extends JpaRepository<Student, Long> {
    // The method name does not impact the query
    @Query(value = "SELECT * FROM STUDENT ORDER BY LAST_NAME", nativeQuery = true)
    List<Student> findAllOrdered();
}
```

```java
@GetMapping("/students")
public ResponseEntity<List<Student>> getAllLabs(@RequestParam(required = false, defaultValue = "false")
                                                                                    boolean orderedByLastName) {
    List<Student> students;
    if(orderedByLastName){
        students = studentRepository.findAllOrdered();
    } else {
        students = studentRepository.findAll();
    }
    return ResponseEntity
            .ok()
            .body(students);
}
```