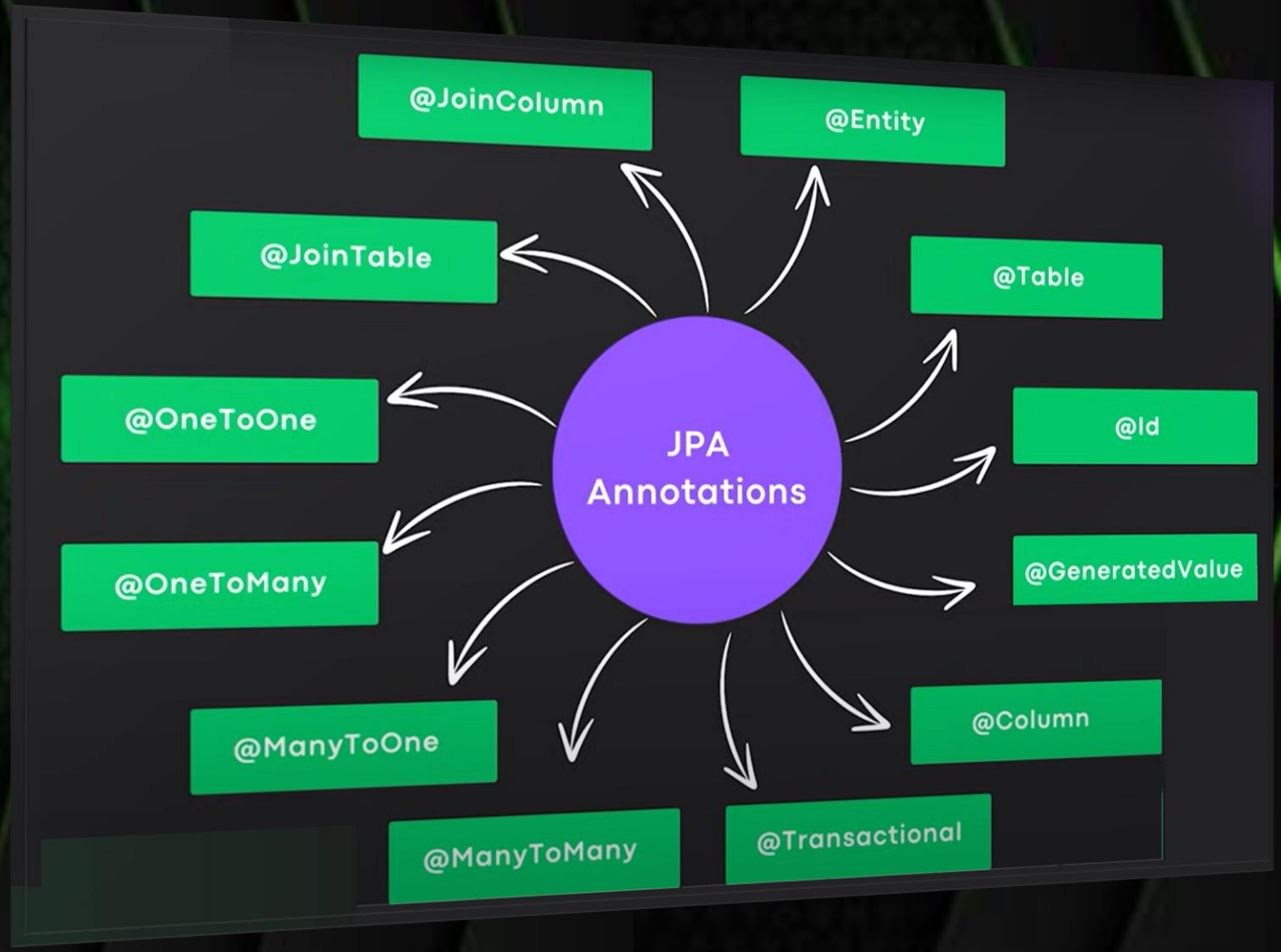
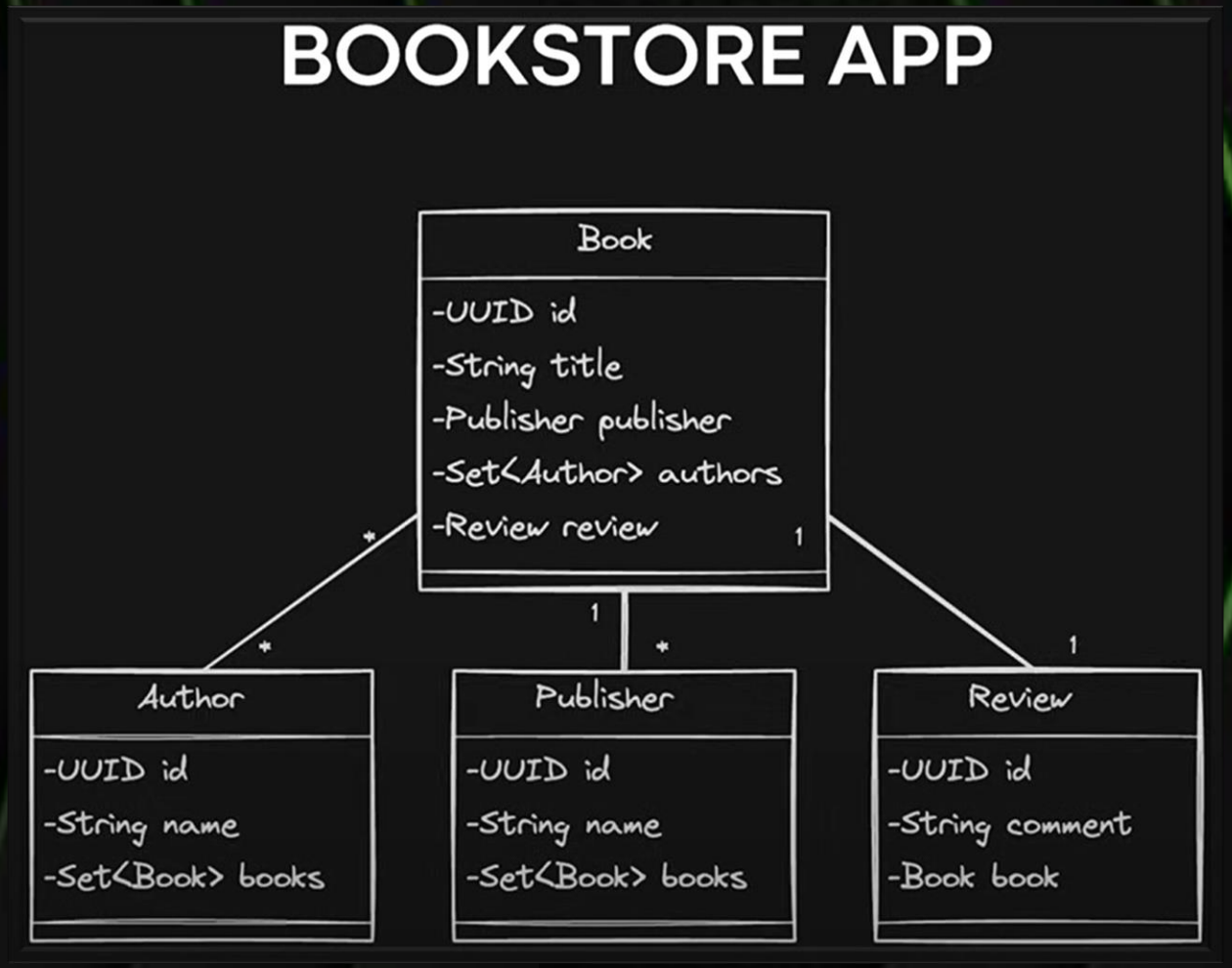
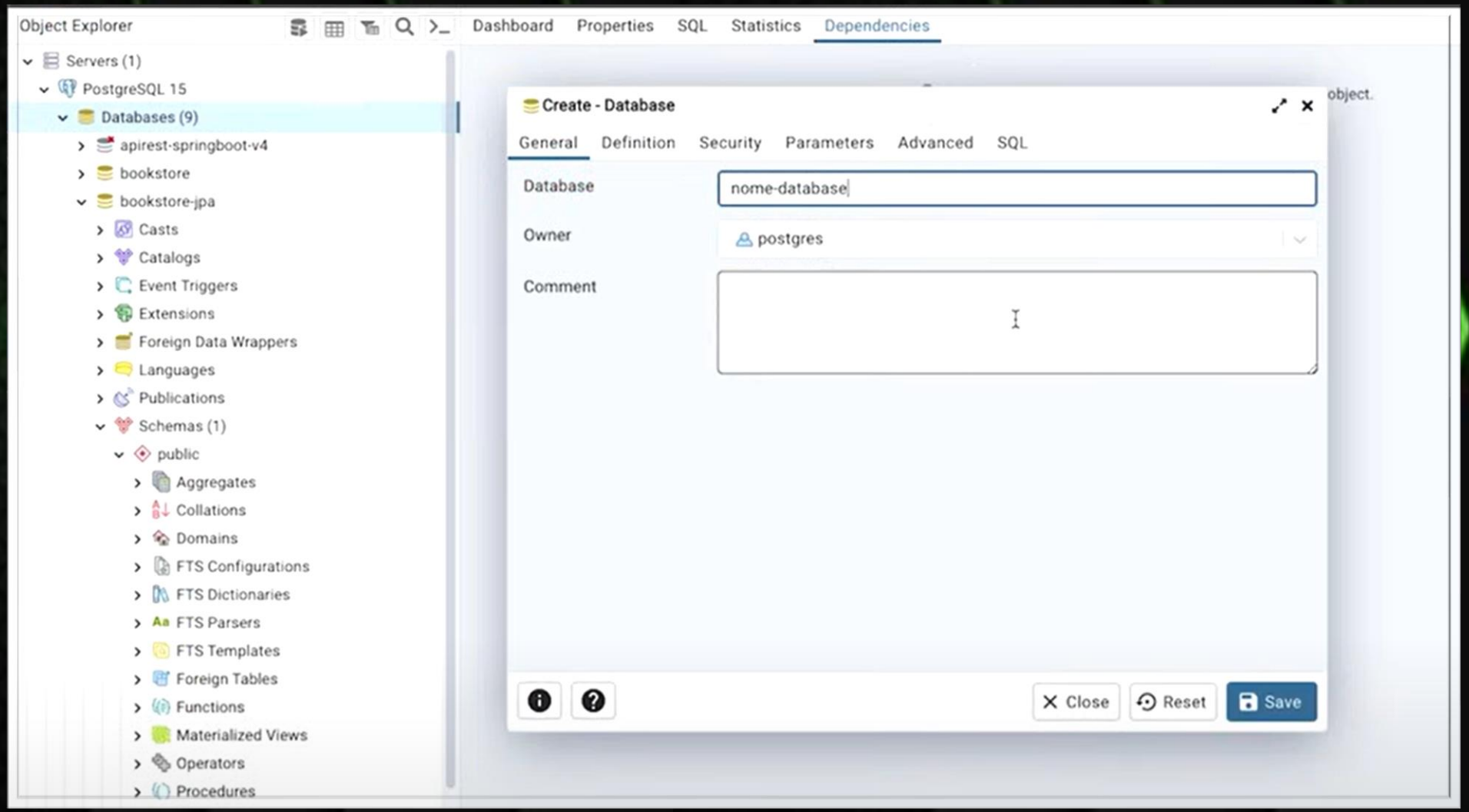
Ciclo de Vida do Projeto Hibernate - Annotations



Projeto de Trabalho




Object Explorer
Dashboard
Properties
SQL
Statistics
Dependencies
Servers (1)
PostgreSQL 15
Databases (9)
apirest-springboot-v4
bookstore
bookstore-jpa
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas (1)
public
Aggregates
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Create - Database
General
Definition
Security
Parameters
Advanced
SQL
Database
nome-database
Owner
postgres
Comment
Close
Reset
Save

Command Prompt - psql -U postgres

```
Microsoft Windows [Version 10.0.19045.4780]
(c) Microsoft Corporation. All rights reserved.

C:\Users\aluno>cd C:\VIVO2024\SITES\spring-data-jpa-main\spring-data-jpa-main

C:\VIVO2024\SITES\spring-data-jpa-main\spring-data-jpa-main>psql -U postgres
Senha para o usuário postgres:
psql (16.4)
ADVERTÊNCIA: A página de código da console (850) difere da página de código do Windows (1252)
             os caracteres de 8 bits podem não funcionar corretamente. Veja a página de
             referência do psql "Notes for Windows users" para obter detalhes.
Digite "help" para obter ajuda.

postgres=# create database "teste";
CREATE DATABASE
postgres=# chcp 1252
postgres-#
postgres-#
```

pom.xml > { } Grammars > https://maven.apache.org/xsd/maven-4.0.0.xsd > Cache > file:///C:/Users/aluno/.lemminx/cache/https/maven

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>3.3.0</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.bookstore</groupId>
    <artifactId>jpa</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>jpa</name>
    <description>Demo project for Spring Boot</description>
    <properties>
        <java.version>17</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
```

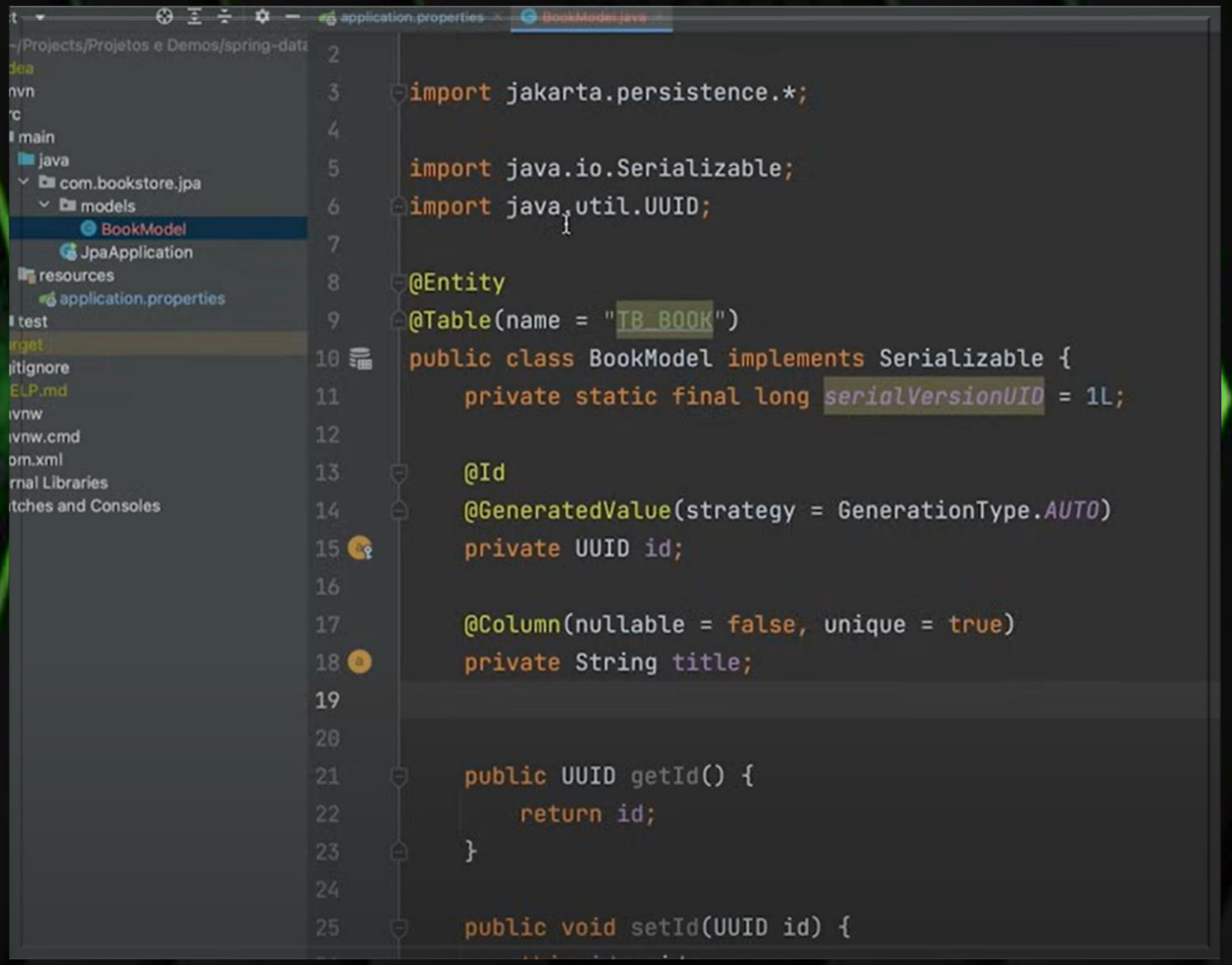

```xml
        <dependency>
            <groupId>org.postgresql</groupId>
            <artifactId>postgresql</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
</project>
```

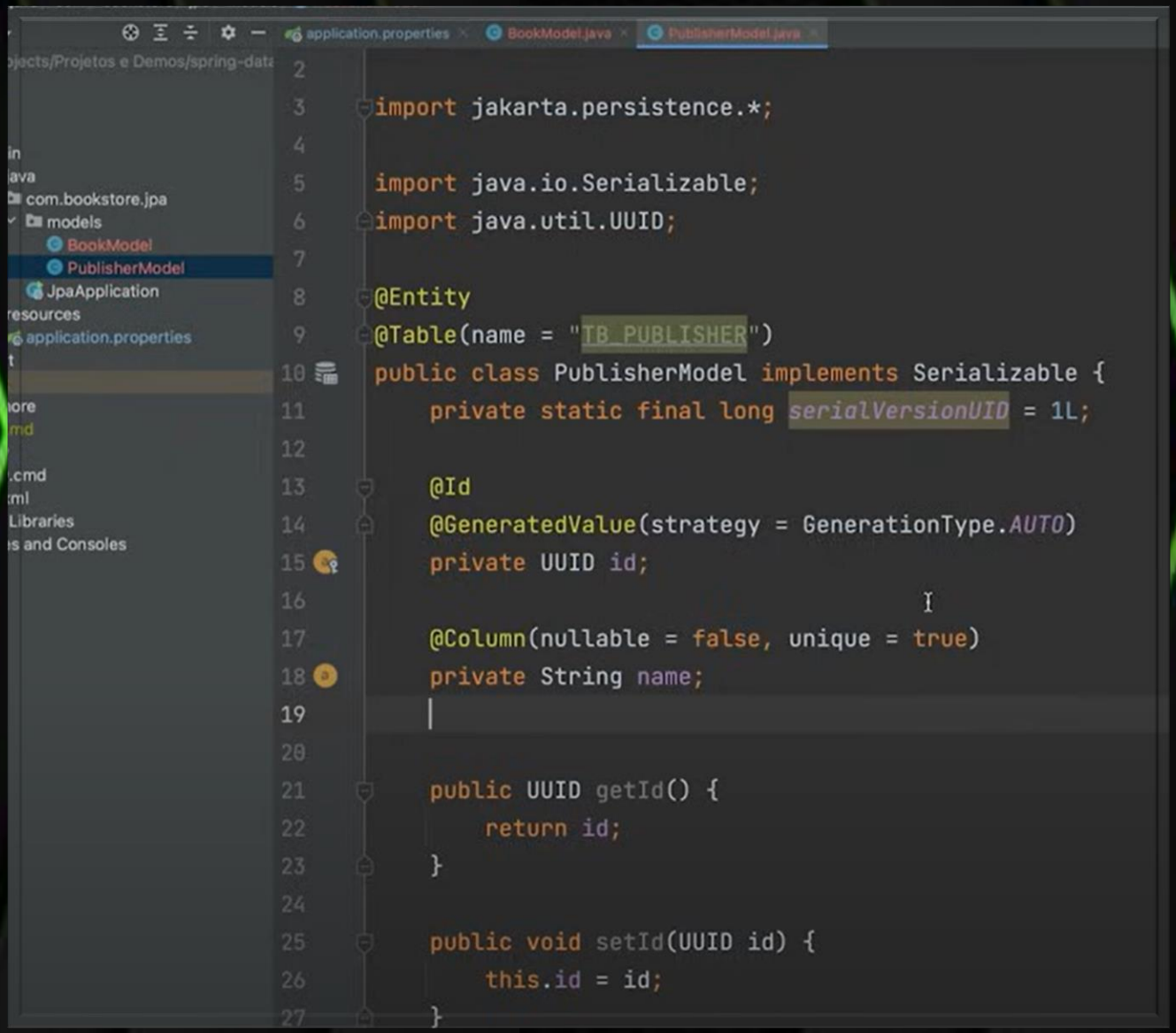
JPA
Java Persistence API
HIBERNATE

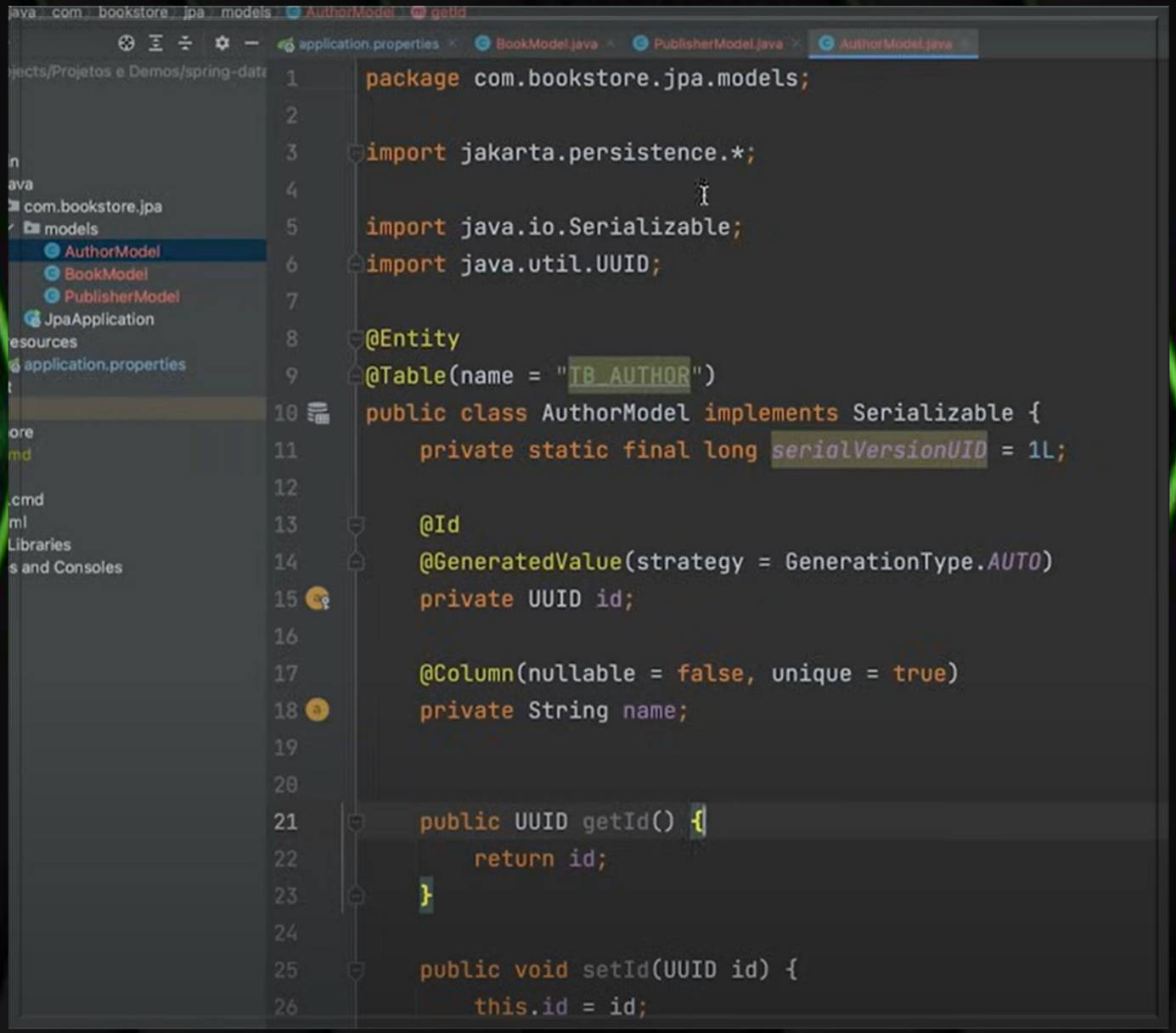
application.properties

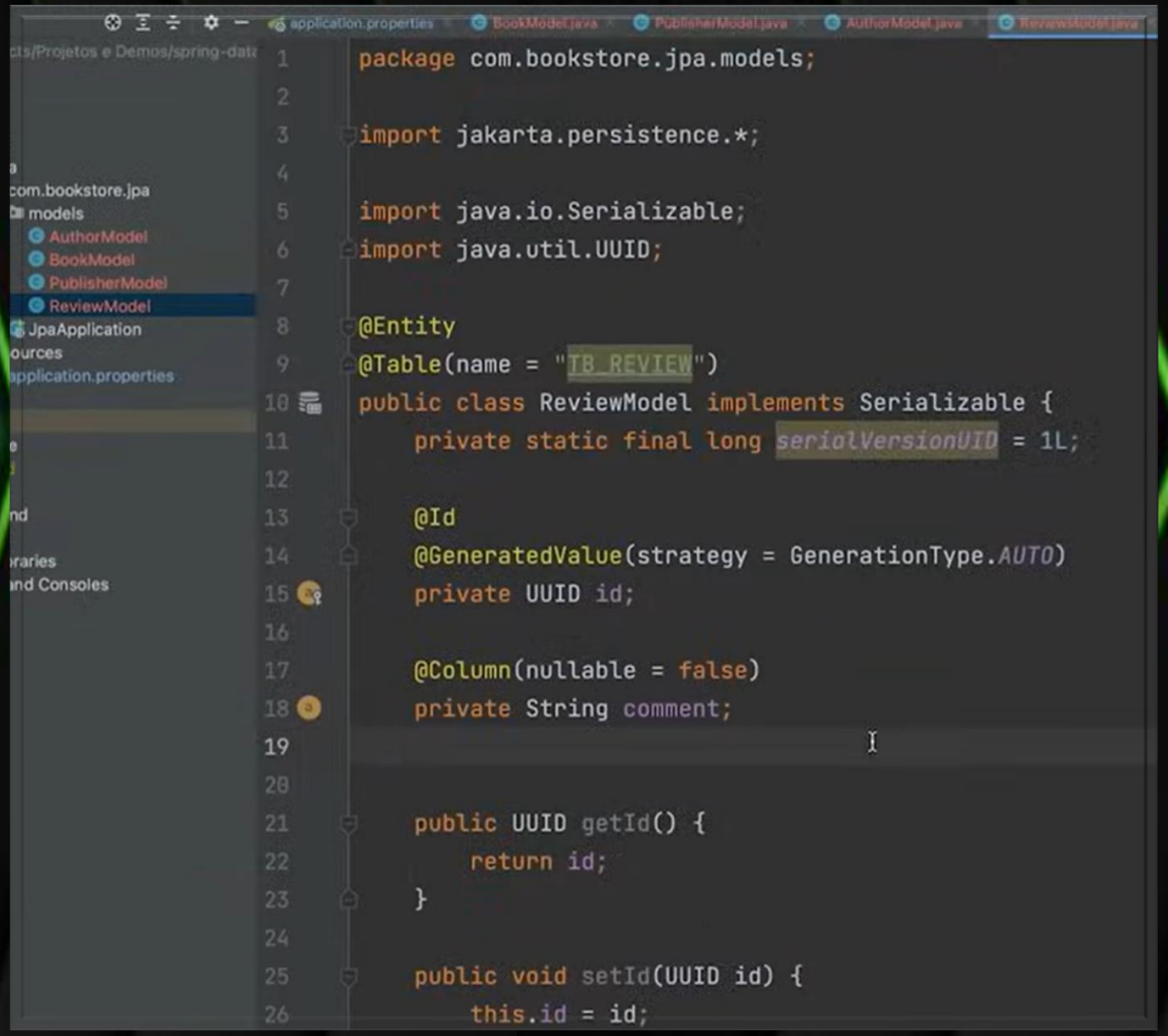
```
spring.application.name=jpa

spring.datasource.url= jdbc:postgresql://localhost:5432/bookstore-jpa
spring.datasource.username=postgres
spring.datasource.password=banco123
spring.jpa.hibernate.ddl-auto=update

spring.jpa.properties.hibernate.jdbc.lob.non_contextual_creation=true
spring.jpa.show-sql=true
```

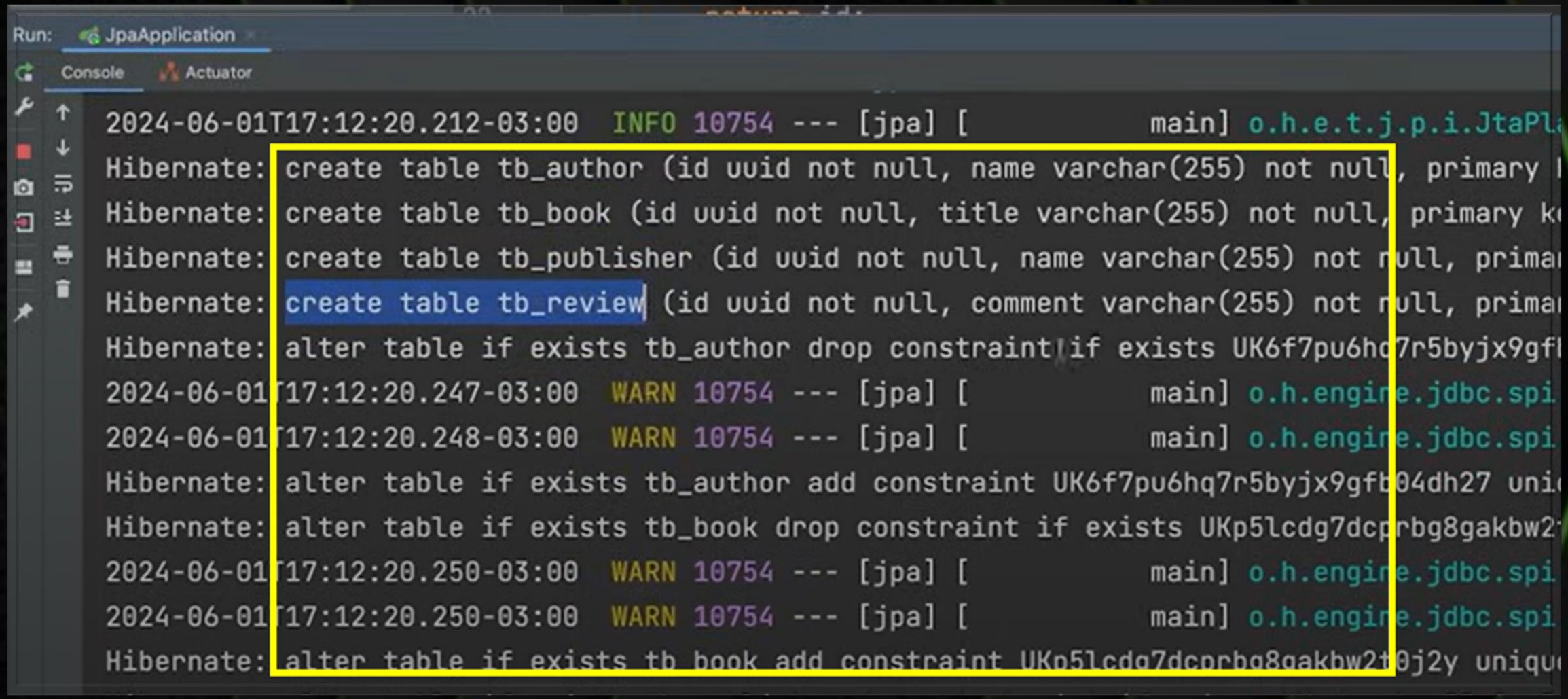
JPA
Java Persistence API
HIBERNATE

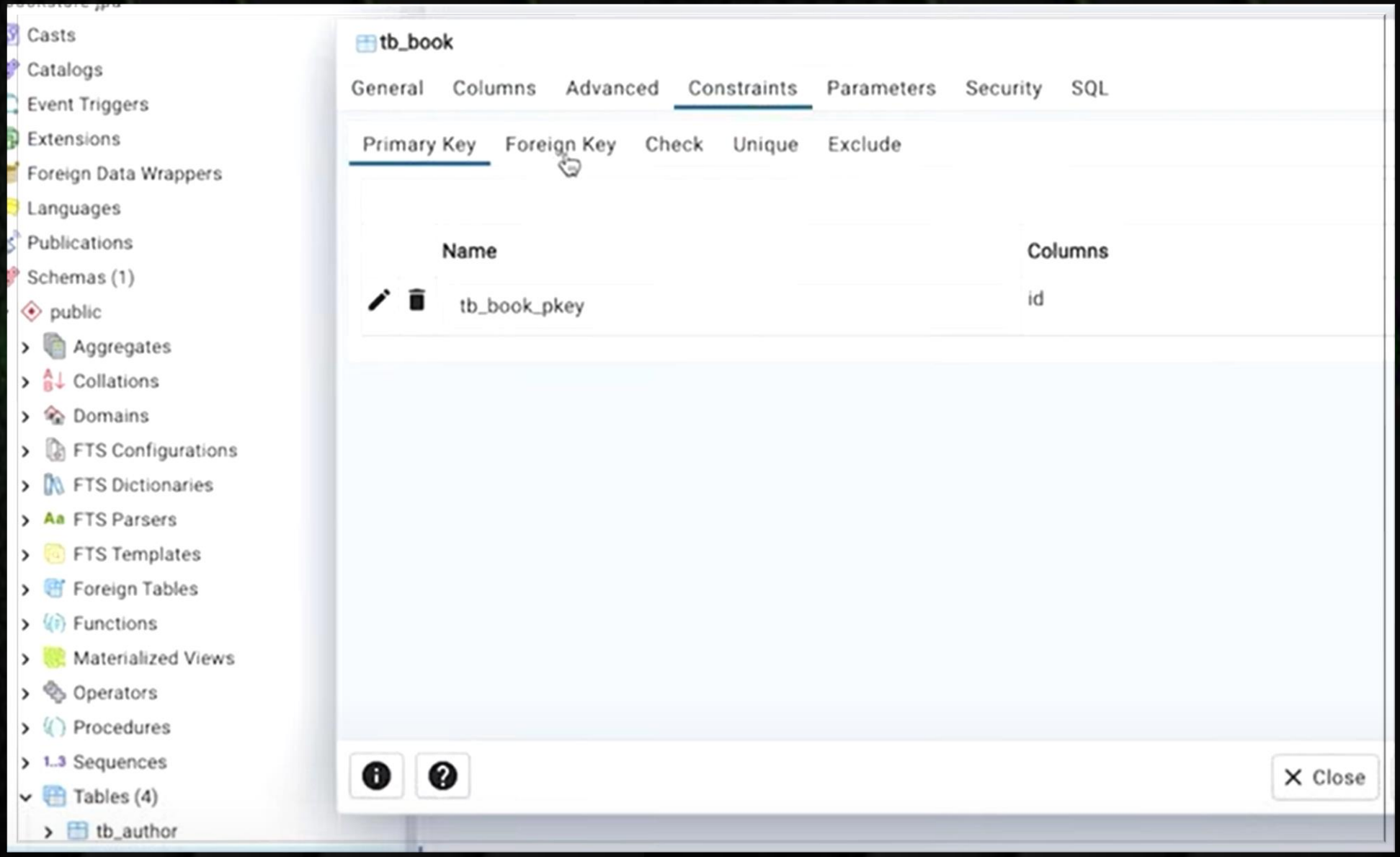

application.properties
BookModel.java
~/Projects/Projetos e Demos/spring-data
main
java
com.bookstore.jpa
models
BookModel
JpaApplication
resources
application.properties
test
.gitignore
HELP.md
mvnw
mvnw.cmd
pom.xml
External Libraries
Scratches and Consoles
import jakarta.persistence.*;
import java.io.Serializable;
import java.util.UUID;
@Entity
@Table(name = "TB_BOOK")
public class BookModel implements Serializable {
private static final long serialVersionUID = 1L;
@Id
@GeneratedValue(strategy = GenerationType.AUTO)
private UUID id;
@Column(nullable = false, unique = true)
private String title;
public UUID getId() {
return id;
}
public void setId(UUID id) {


application.properties
BookModel.java
PublisherModel.java
com.bookstore.jpa
models
BookModel
PublisherModel
JpaApplication
resources
application.properties
Libraries
and Consoles
import jakarta.persistence.*;
import java.io.Serializable;
import java.util.UUID;
@Entity
@Table(name = "TB_PUBLISHER")
public class PublisherModel implements Serializable {
private static final long serialVersionUID = 1L;
@Id
@GeneratedValue(strategy = GenerationType.AUTO)
private UUID id;
@Column(nullable = false, unique = true)
private String name;
public UUID getId() {
return id;
}
public void setId(UUID id) {
this.id = id;
}

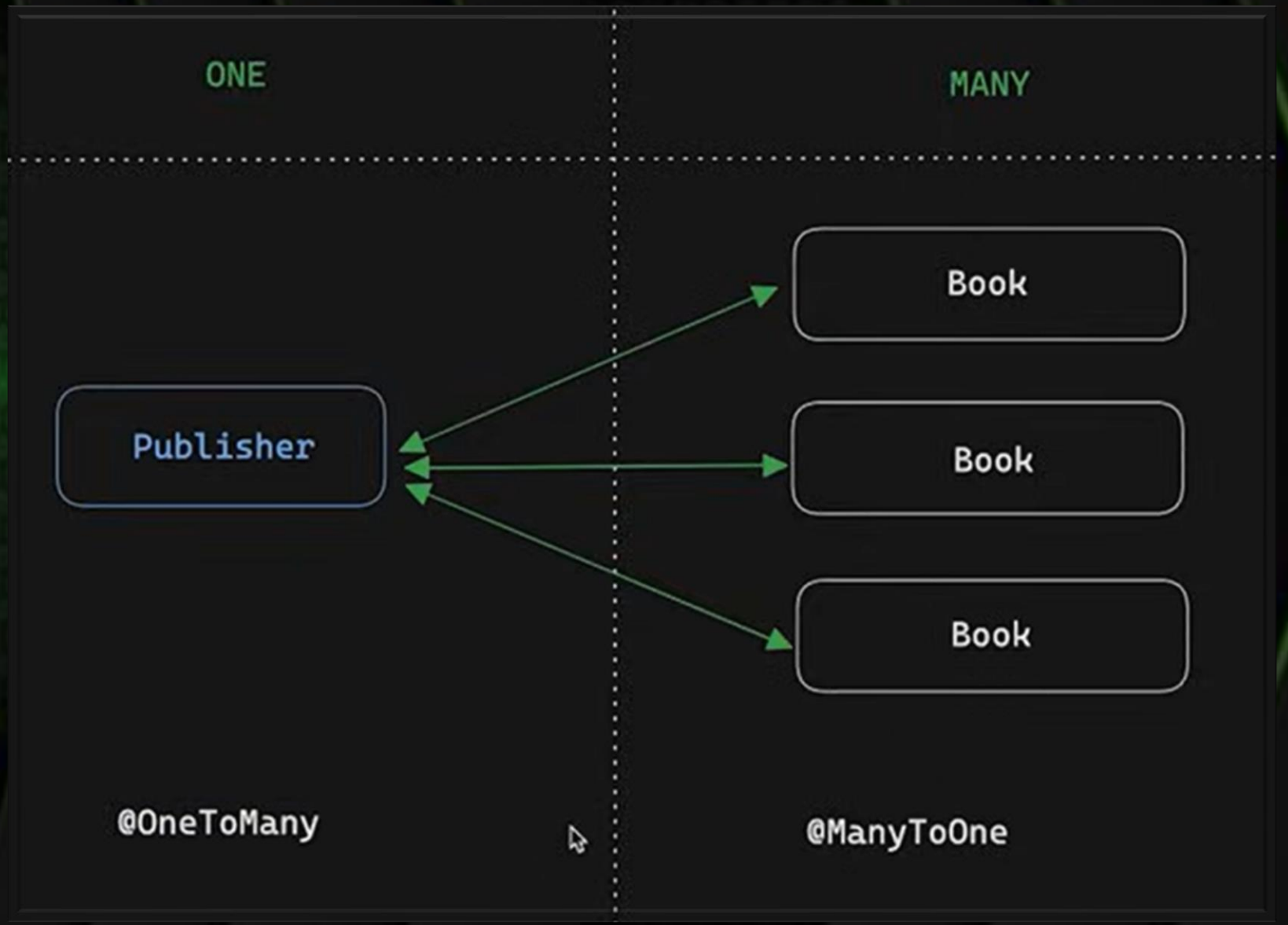

java com bookstore jpa models AuthorModel getId
application.properties
BookModel.java
PublisherModel.java
AuthorModel.java
com.bookstore.jpa
models
AuthorModel
BookModel
PublisherModel
JpaApplication
application.properties
Libraries
and Consoles
package com.bookstore.jpa.models;
import jakarta.persistence.*;
import java.io.Serializable;
import java.util.UUID;
@Entity
@Table(name = "TB_AUTHOR")
public class AuthorModel implements Serializable {
private static final long serialVersionUID = 1L;
@Id
@GeneratedValue(strategy = GenerationType.AUTO)
private UUID id;
@Column(nullable = false, unique = true)
private String name;
public UUID getId() {
return id;
}
public void setId(UUID id) {
this.id = id;


application.properties
BookModel.java
PublisherModel.java
AuthorModel.java
ReviewModel.java
com.bookstore.jpa
models
AuthorModel
BookModel
PublisherModel
ReviewModel
JpaApplication
application.properties
package com.bookstore.jpa.models;
import jakarta.persistence.*;
import java.io.Serializable;
import java.util.UUID;
@Entity
@Table(name = "TB_REVIEW")
public class ReviewModel implements Serializable {
private static final long serialVersionUID = 1L;
@Id
@GeneratedValue(strategy = GenerationType.AUTO)
private UUID id;
@Column(nullable = false)
private String comment;
public UUID getId() {
return id;
}
public void setId(UUID id) {
this.id = id;

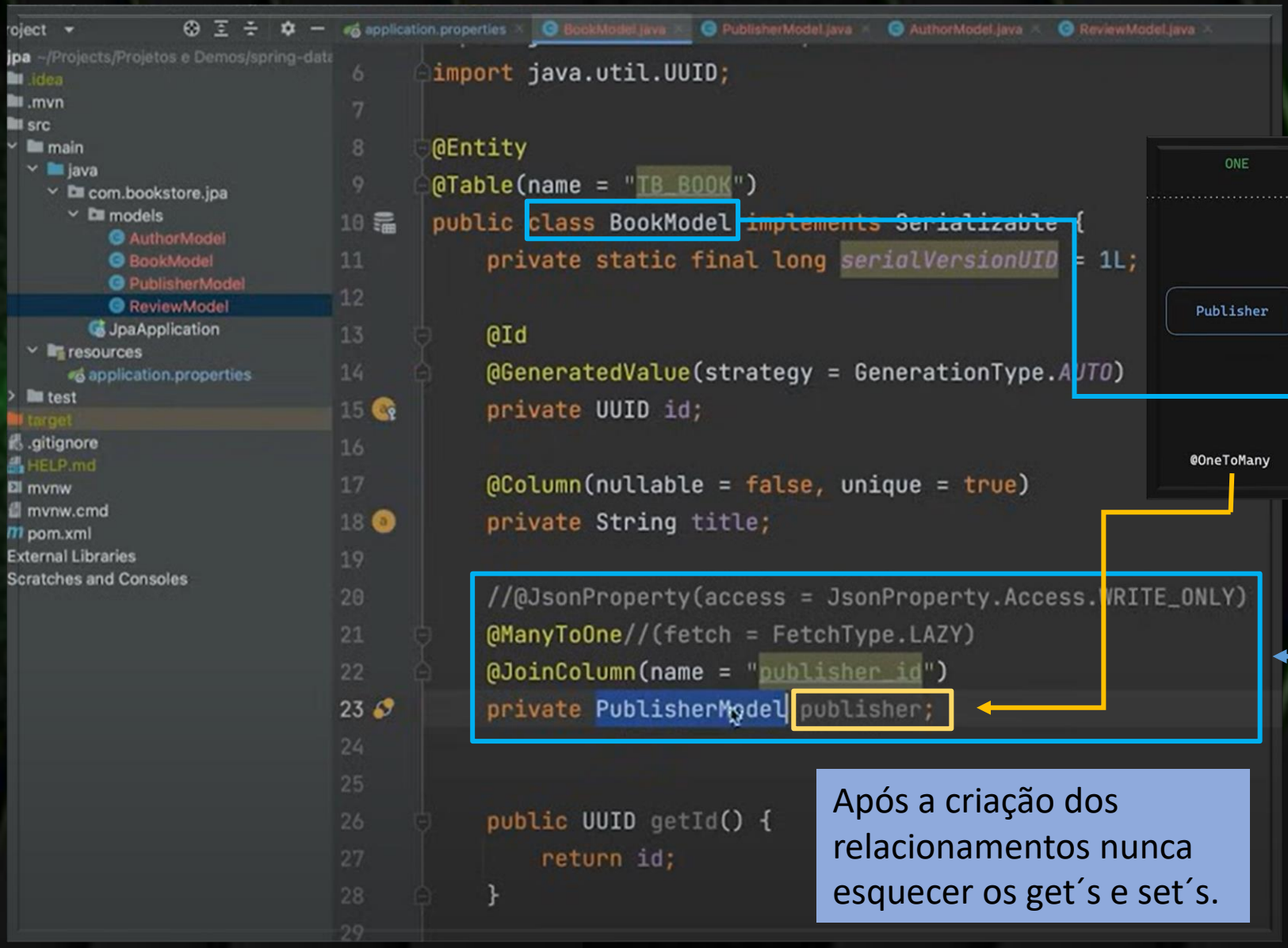

Run: JpaApplication
Console
Actuator
2024-06-01T17:12:20.212-03:00 INFO 10754 --- [jpa] [main] o.h.e.t.j.p.i.JtaPl
Hibernate: create table tb_author (id uuid not null, name varchar(255) not null, primary
Hibernate: create table tb_book (id uuid not null, title varchar(255) not null, primary k
Hibernate: create table tb_publisher (id uuid not null, name varchar(255) not null, prima
Hibernate: create table tb_review (id uuid not null, comment varchar(255) not null, prima
Hibernate: alter table if exists tb_author drop constraint if exists UK6f7pu6hq7r5byjx9gf
2024-06-01T17:12:20.247-03:00 WARN 10754 --- [jpa] [main] o.h.engine.jdbc.spi
2024-06-01T17:12:20.248-03:00 WARN 10754 --- [jpa] [main] o.h.engine.jdbc.spi
Hibernate: alter table if exists tb_author add constraint UK6f7pu6hq7r5byjx9gfb04dh27 uni
Hibernate: alter table if exists tb_book drop constraint if exists UKp5lcdg7dcprbg8gakbw2
2024-06-01T17:12:20.250-03:00 WARN 10754 --- [jpa] [main] o.h.engine.jdbc.spi
2024-06-01T17:12:20.250-03:00 WARN 10754 --- [jpa] [main] o.h.engine.jdbc.spi
Hibernate: alter table if exists tb_book add constraint UKp5lcdg7dcprbg8gakbw2t0j2y uniqu

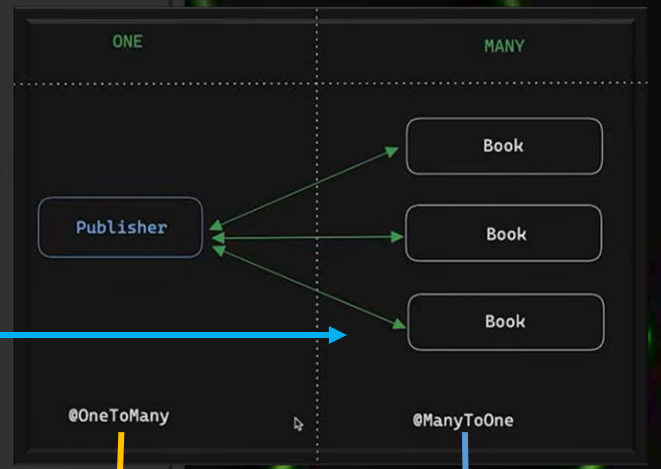

Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas (1)
public
Aggregates
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (4)
tb_author
tb_book
General
Columns
Advanced
Constraints
Parameters
Security
SQL
Primary Key
Foreign Key
Check
Unique
Exclude
Name
Columns
tb_book_pkey
id
Close

@OneToMany



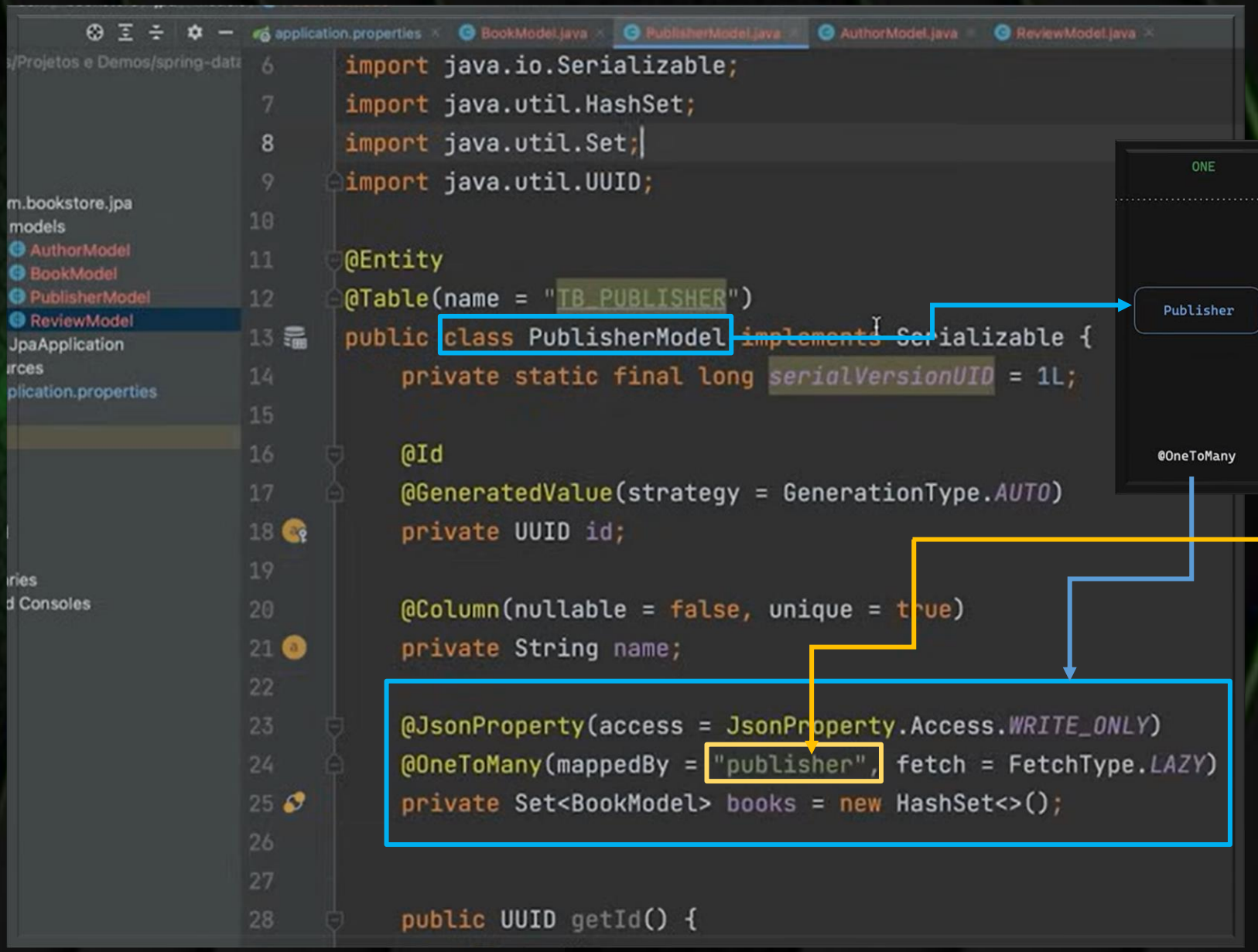
@ManyToOne

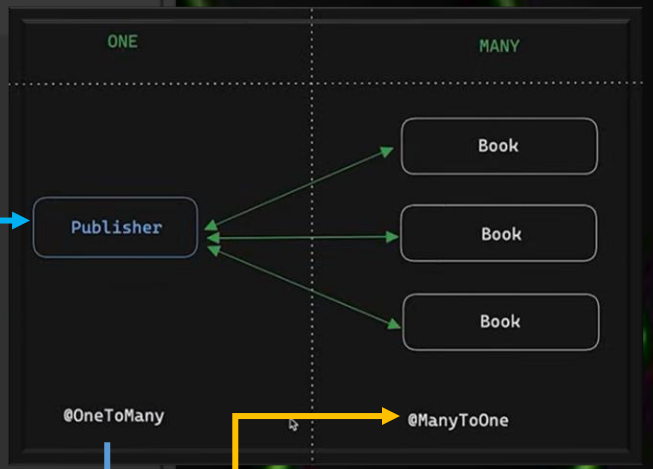




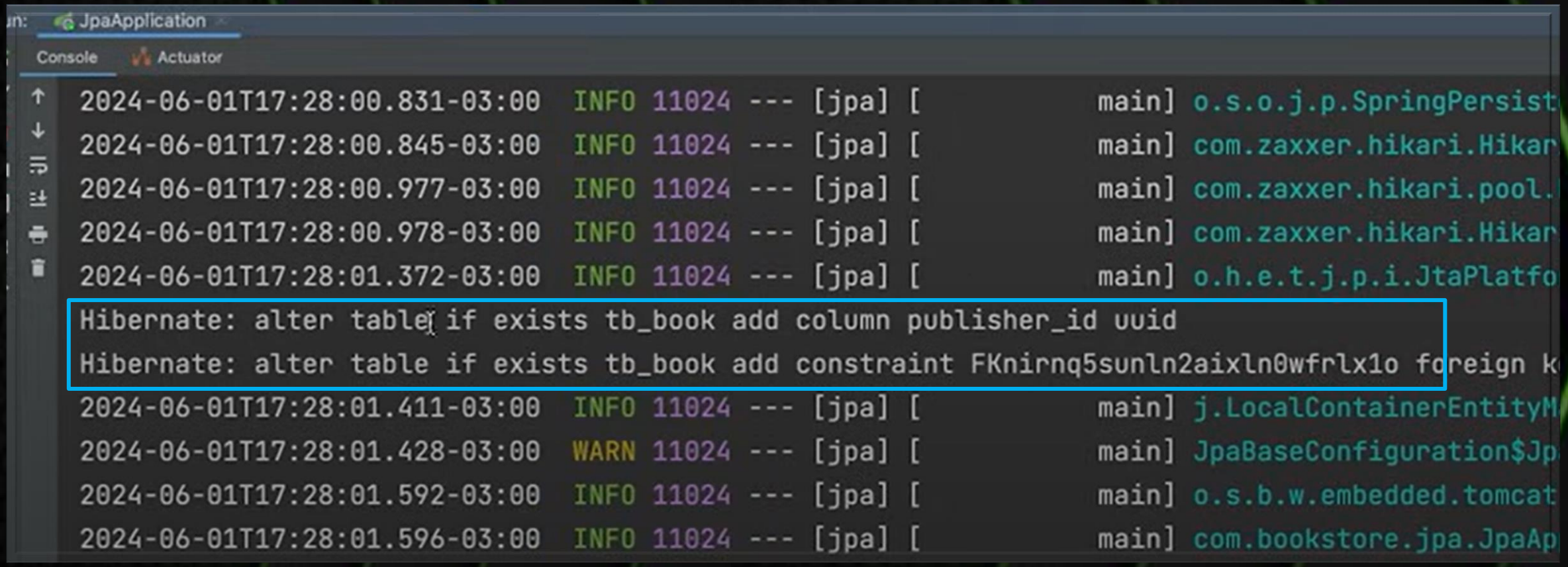
Após a criação dos relacionamentos nunca esquecer os get´s e set´s.

@OneToMany

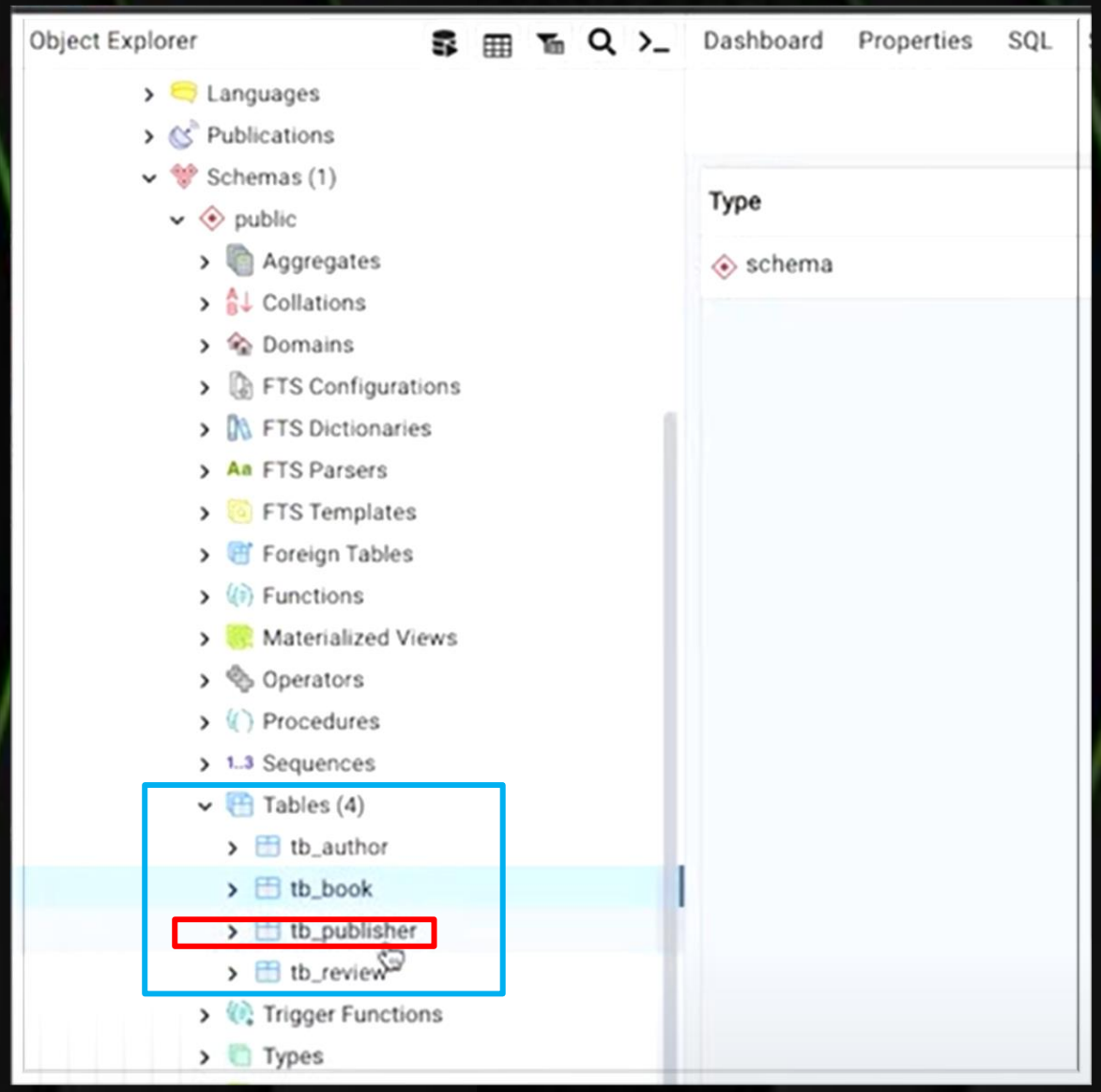

application.properties
BookModel.java
PublisherModel.java
AuthorModel.java
ReviewModel.java
/Projetos e Demos/spring-data
m.bookstore.jpa
models
AuthorModel
BookModel
PublisherModel
ReviewModel
JpaApplication
rces
plication.properties
ries
d Consoles
6 import java.io.Serializable;
7 import java.util.HashSet;
8 import java.util.Set;
9 import java.util.UUID;
10
11 @Entity
12 @Table(name = "TB_PUBLISHER")
13 public class PublisherModel implements Serializable {
14 private static final long serialVersionUID = 1L;
15
16 @Id
17 @GeneratedValue(strategy = GenerationType.AUTO)
18 private UUID id;
19
20 @Column(nullable = false, unique = true)
21 private String name;
22
23 @JsonProperty(access = JsonProperty.Access.WRITE_ONLY)
24 @OneToMany(mappedBy = "publisher", fetch = FetchType.LAZY)
25 private Set<BookModel> books = new HashSet<>();
26
27
28 public UUID getId() {


ONE
MANY
Publisher
Book
Book
Book
@OneToMany
@ManyToOne

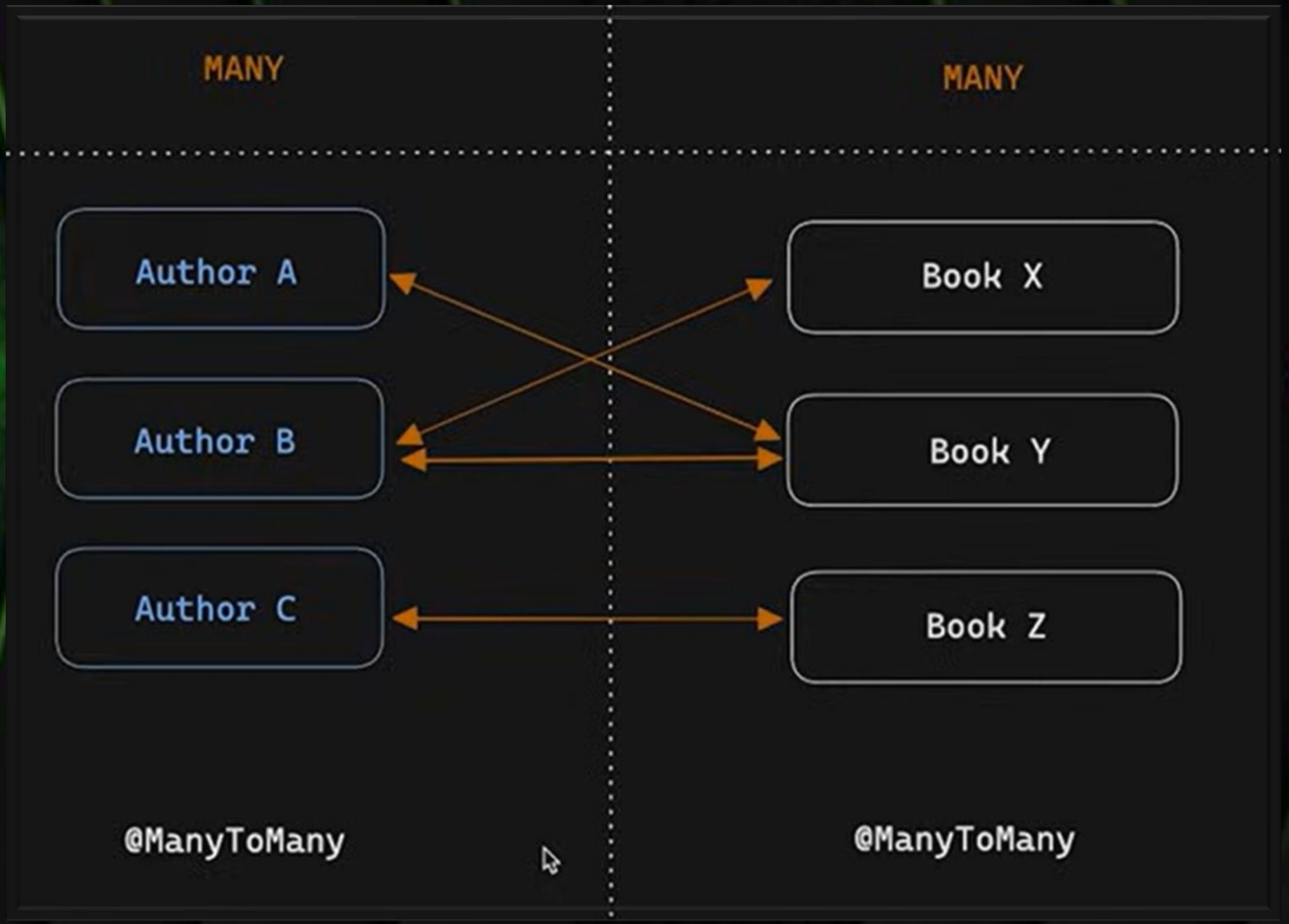
Spring Run


JpaApplication
Console
Actuator
2024-06-01T17:28:00.831-03:00 INFO 11024 --- [jpa] [main] o.s.o.j.p.SpringPersist
2024-06-01T17:28:00.845-03:00 INFO 11024 --- [jpa] [main] com.zaxxer.hikari.Hikar
2024-06-01T17:28:00.977-03:00 INFO 11024 --- [jpa] [main] com.zaxxer.hikari.pool.
2024-06-01T17:28:00.978-03:00 INFO 11024 --- [jpa] [main] com.zaxxer.hikari.Hikar
2024-06-01T17:28:01.372-03:00 INFO 11024 --- [jpa] [main] o.h.e.t.j.p.i.JtaPlatfo
Hibernate: alter table if exists tb_book add column publisher_id uuid
Hibernate: alter table if exists tb_book add constraint FKnirnq5sunln2aixln0wfrlx1o foreign k
2024-06-01T17:28:01.411-03:00 INFO 11024 --- [jpa] [main] j.LocalContainerEntityM
2024-06-01T17:28:01.428-03:00 WARN 11024 --- [jpa] [main] JpaBaseConfiguration$Jp
2024-06-01T17:28:01.592-03:00 INFO 11024 --- [jpa] [main] o.s.b.w.embedded.tomcat
2024-06-01T17:28:01.596-03:00 INFO 11024 --- [jpa] [main] com.bookstore.jpa.JpaAp

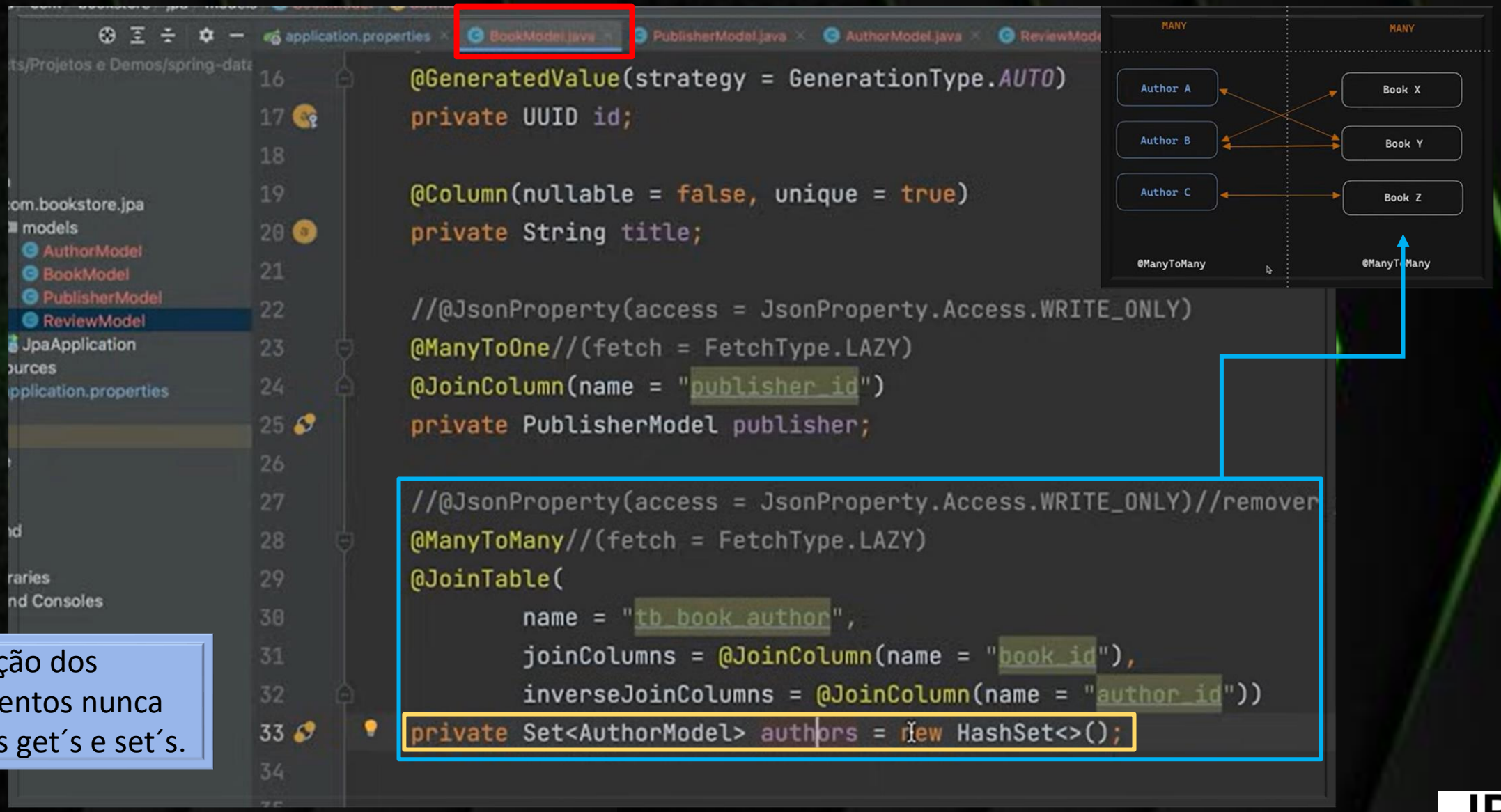
BD


Object Explorer
Dashboard
Properties
SQL
Languages
Publications
Schemas (1)
public
Aggregates
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (4)
tb_author
tb_book
tb_publisher
tb_review
Trigger Functions
Types
Type
schema

@ManyToMany

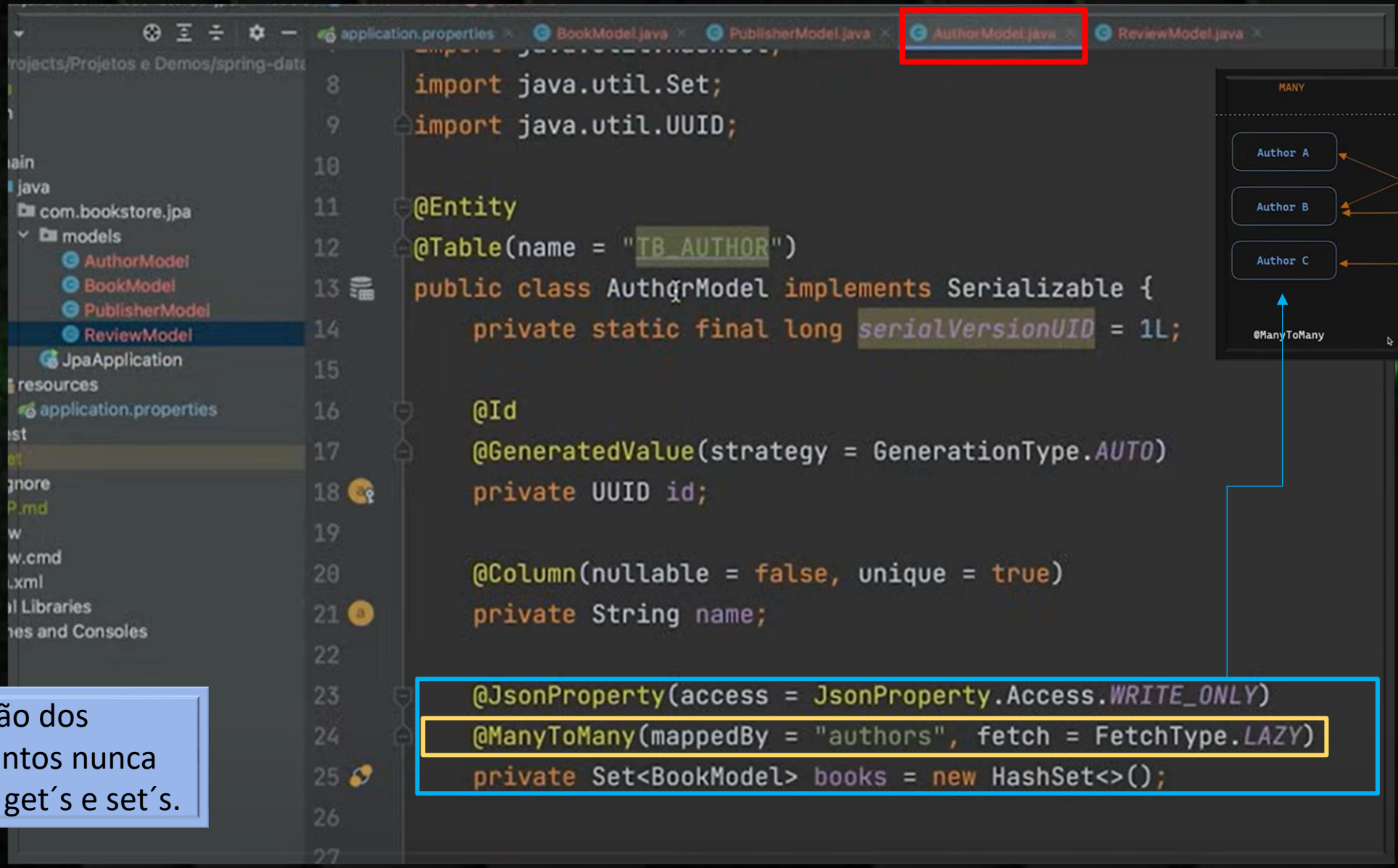

MANY
MANY
Author A
Author B
Author C
Book X
Book Y
Book Z
@ManyToMany
@ManyToMany

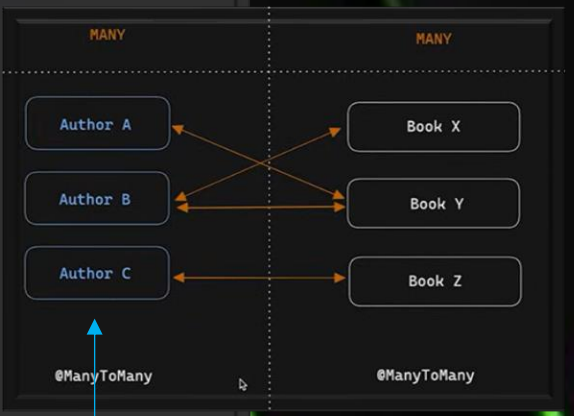
@ManyToMany



Após a criação dos relacionamentos nunca esquecer os get´s e set´s.

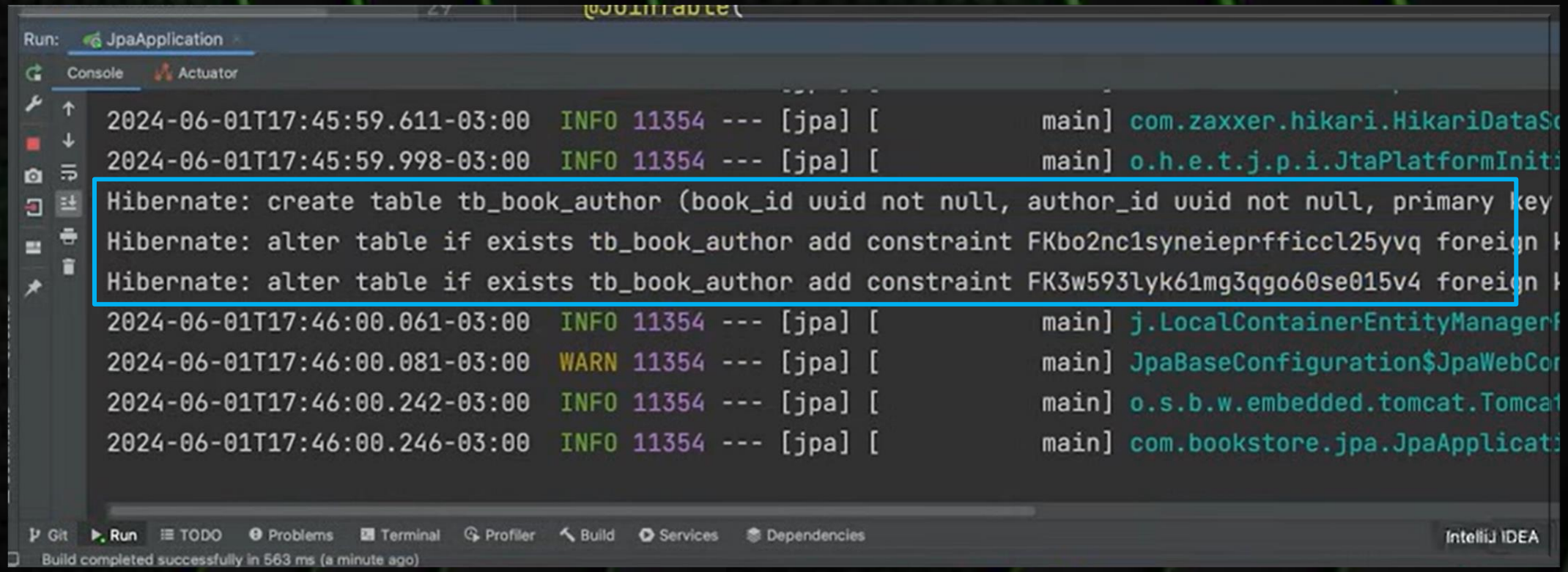
@ManyToMany

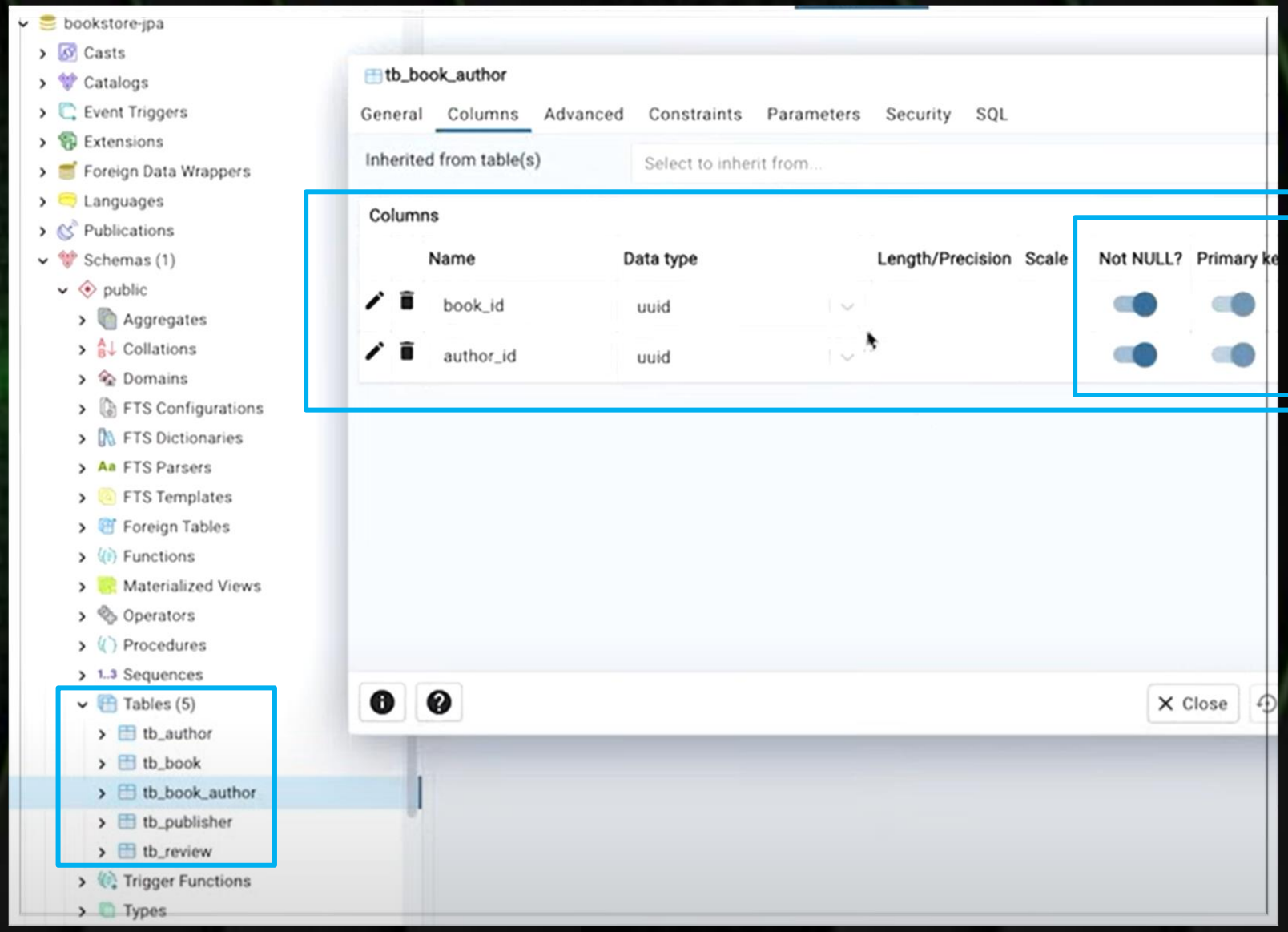


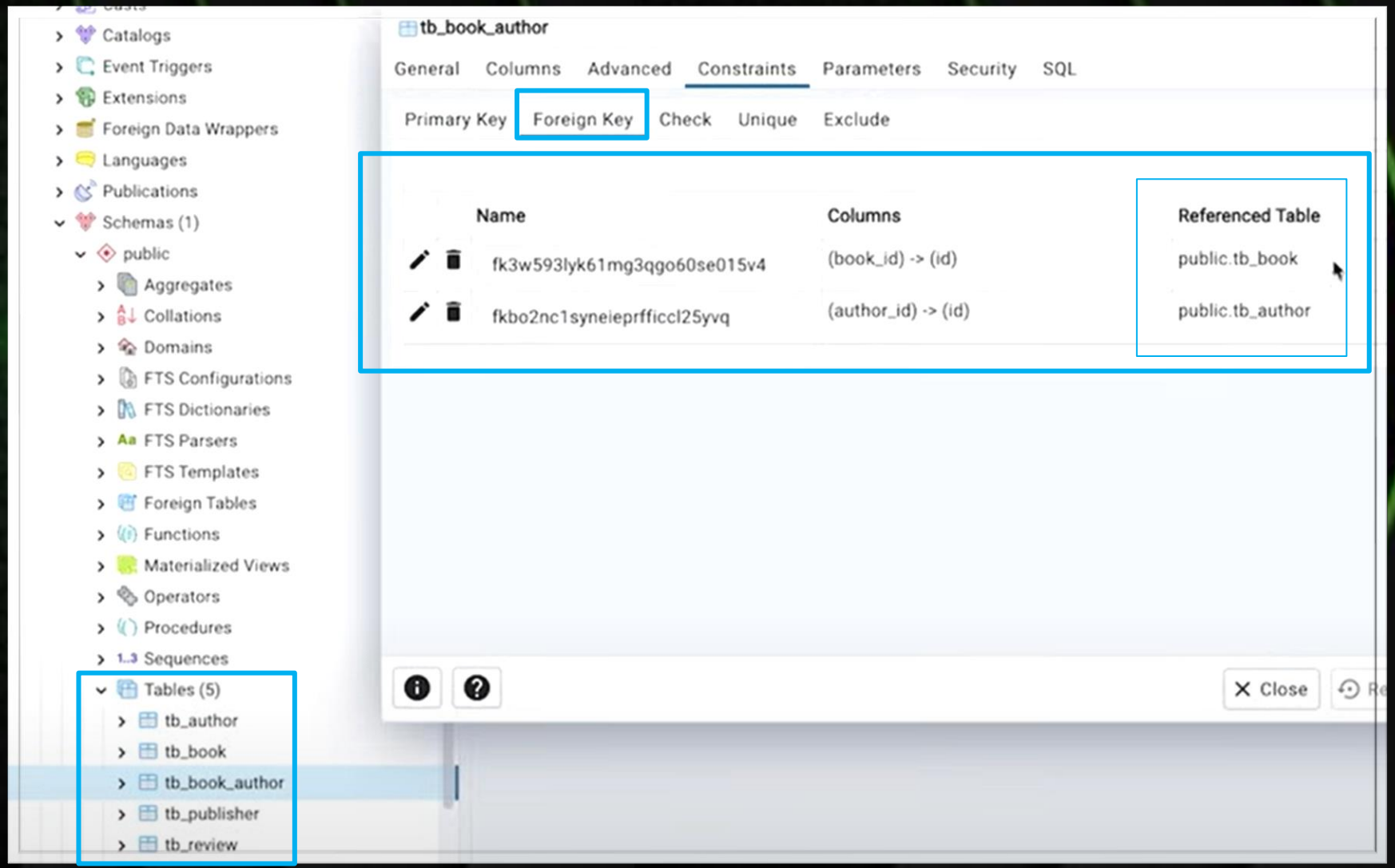


Após a criação dos relacionamentos nunca esquecer os get´s e set´s.

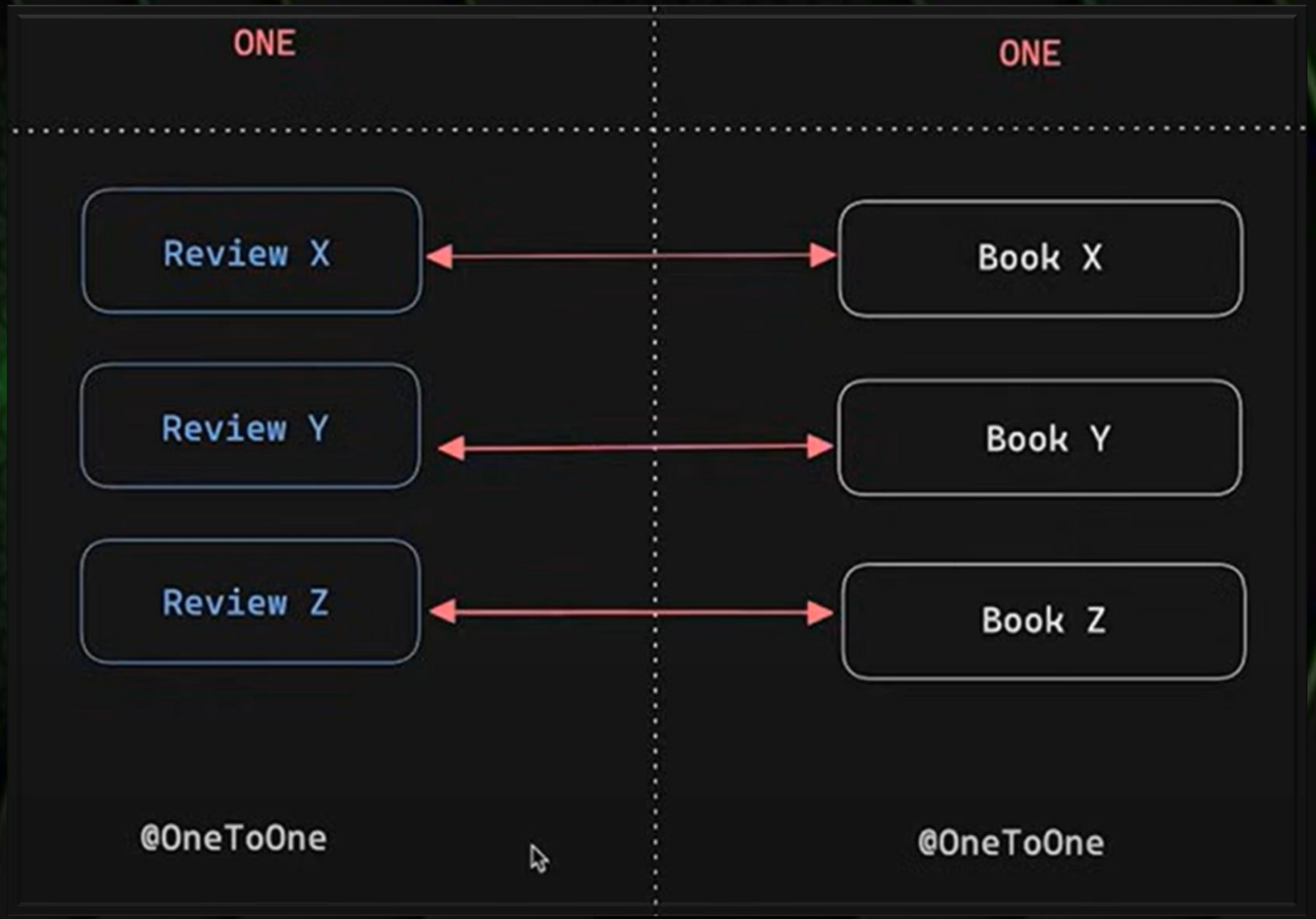
Spring Run


Run: JpaApplication
Console
Actuator
2024-06-01T17:45:59.611-03:00 INFO 11354 --- [jpa] [main] com.zaxxer.hikari.HikariDataS
2024-06-01T17:45:59.998-03:00 INFO 11354 --- [jpa] [main] o.h.e.t.j.p.i.JtaPlatformIniti
Hibernate: create table tb_book_author (book_id uuid not null, author_id uuid not null, primary key
Hibernate: alter table if exists tb_book_author add constraint FKbo2nc1syneieprfficcl25yvq foreign
Hibernate: alter table if exists tb_book_author add constraint FK3w593lyk61mg3qgo60se015v4 foreign
2024-06-01T17:46:00.061-03:00 INFO 11354 --- [jpa] [main] j.LocalContainerEntityManagerF
2024-06-01T17:46:00.081-03:00 WARN 11354 --- [jpa] [main] JpaBaseConfiguration$JpaWebCon
2024-06-01T17:46:00.242-03:00 INFO 11354 --- [jpa] [main] o.s.b.w.embedded.tomcat.Tomcat
2024-06-01T17:46:00.246-03:00 INFO 11354 --- [jpa] [main] com.bookstore.jpa.JpaApplicati
Git
Run
TODO
Problems
Terminal
Profiler
Build
Services
Dependencies
IntelliJ IDEA
Build completed successfully in 563 ms (a minute ago)

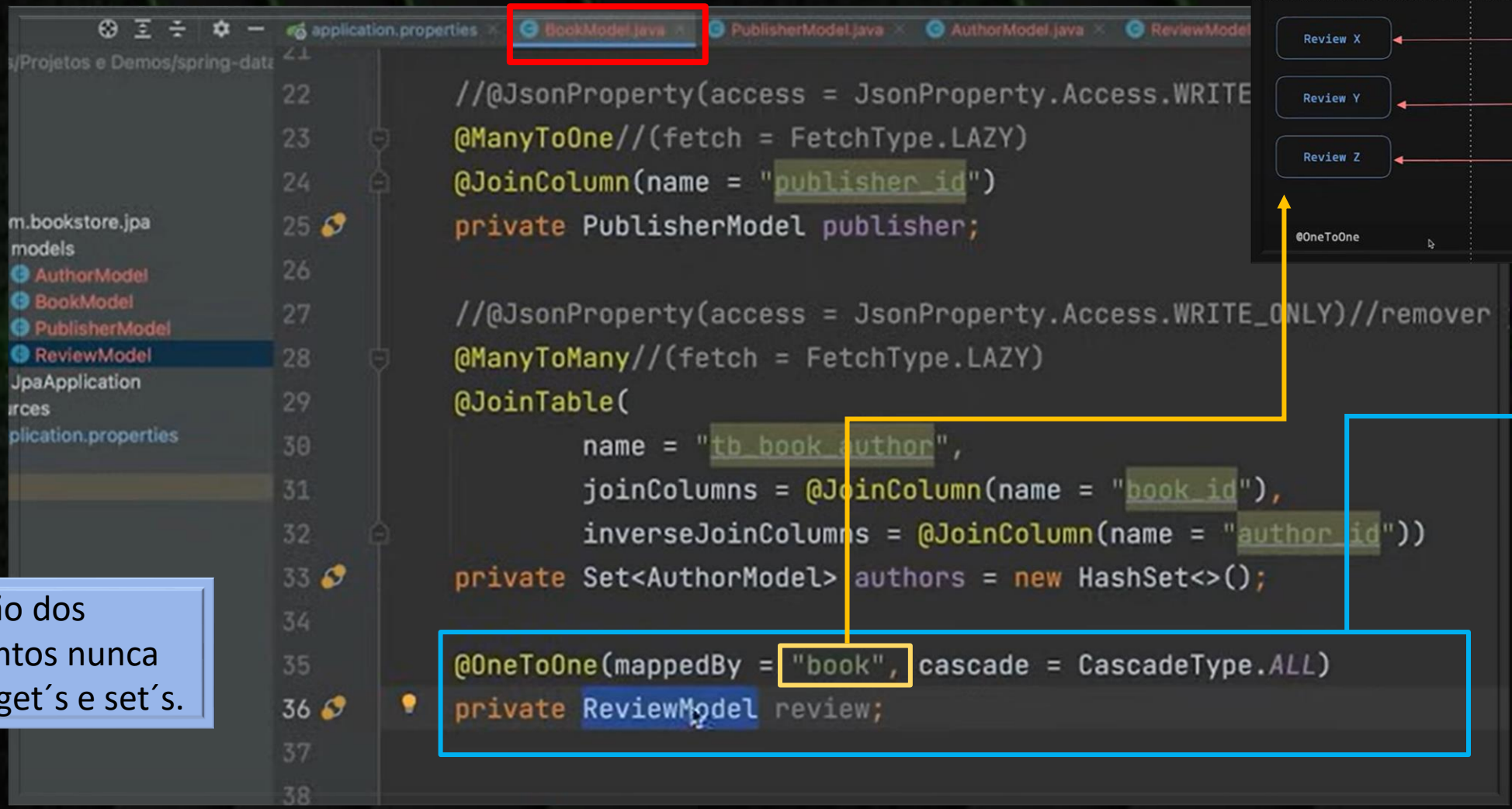

bookstore-jpa
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas (1)
public
Aggregates
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (5)
tb_author
tb_book
tb_book_author
tb_publisher
tb_review
Trigger Functions
Types
tb_book_author
General
Columns
Advanced
Constraints
Parameters
Security
SQL
Inherited from table(s)
Select to inherit from...
Columns
Name
Data type
Length/Precision
Scale
Not NULL?
Primary ke
book_id
uuid
author_id
uuid
Close

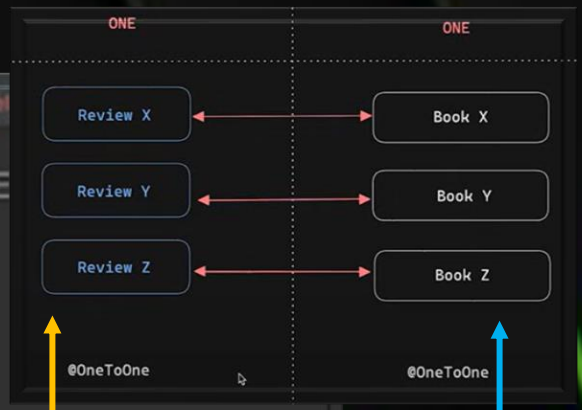

Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas (1)
public
Aggregates
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (5)
tb_author
tb_book
tb_book_author
tb_publisher
tb_review
tb_book_author
General
Columns
Advanced
Constraints
Parameters
Security
SQL
Primary Key
Foreign Key
Check
Unique
Exclude
Name
Columns
Referenced Table
fk3w593lyk61mg3qgo60se015v4
(book_id) -> (id)
public.tb_book
fkbo2nc1syneieprfficcl25yvq
(author_id) -> (id)
public.tb_author
Close

@OneToOne


ONE
ONE
Review X
Book X
Review Y
Book Y
Review Z
Book Z
@OneToOne
@OneToOne

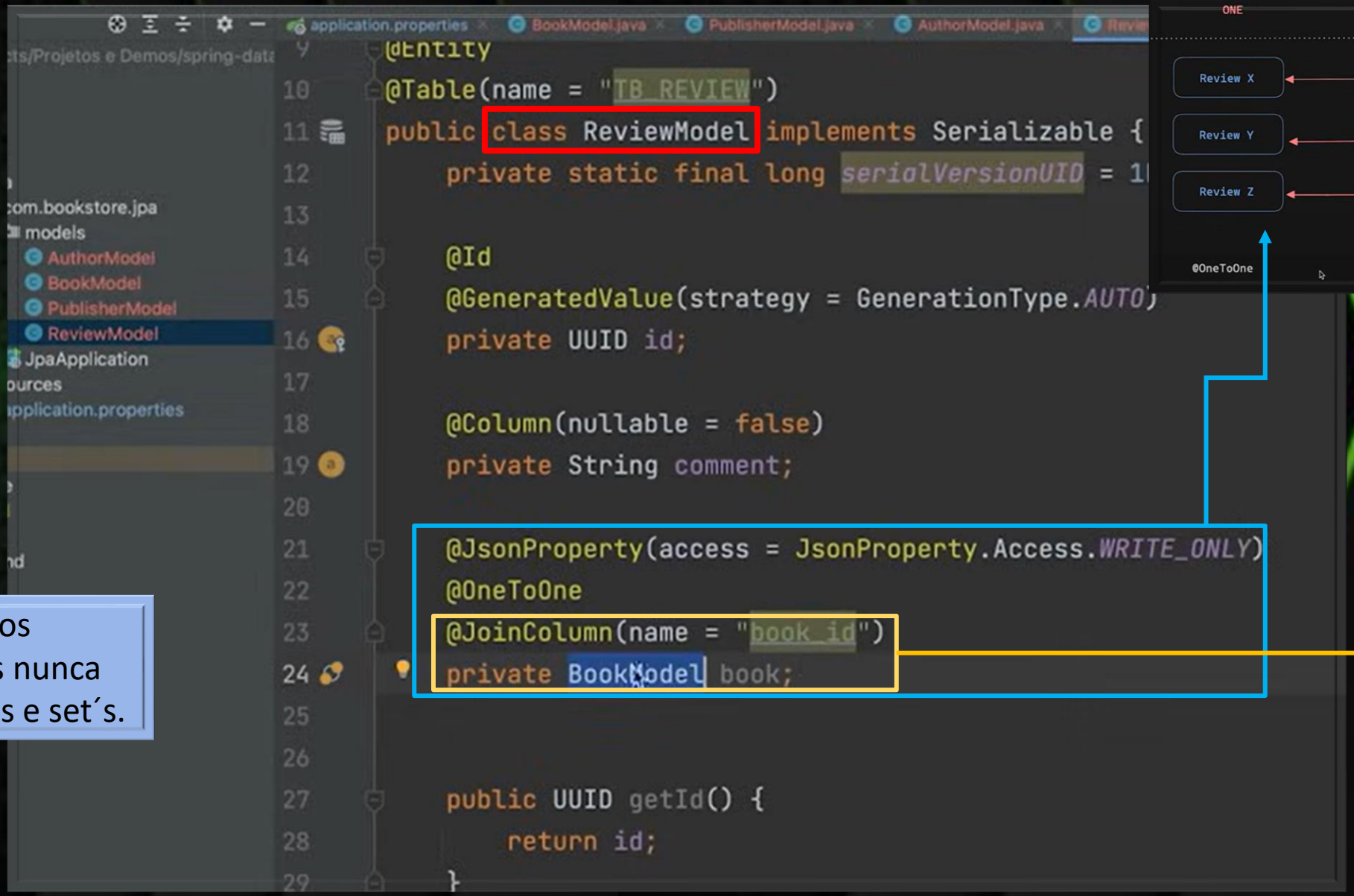
@OneToOne

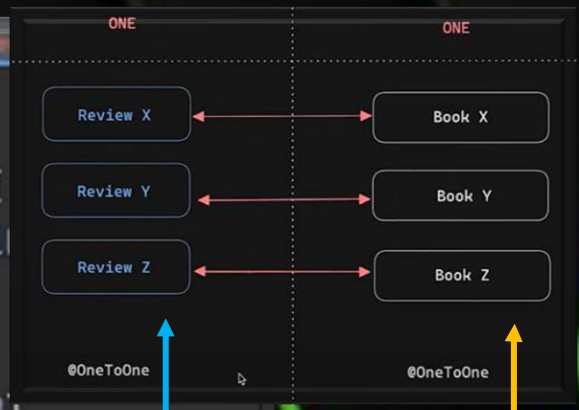




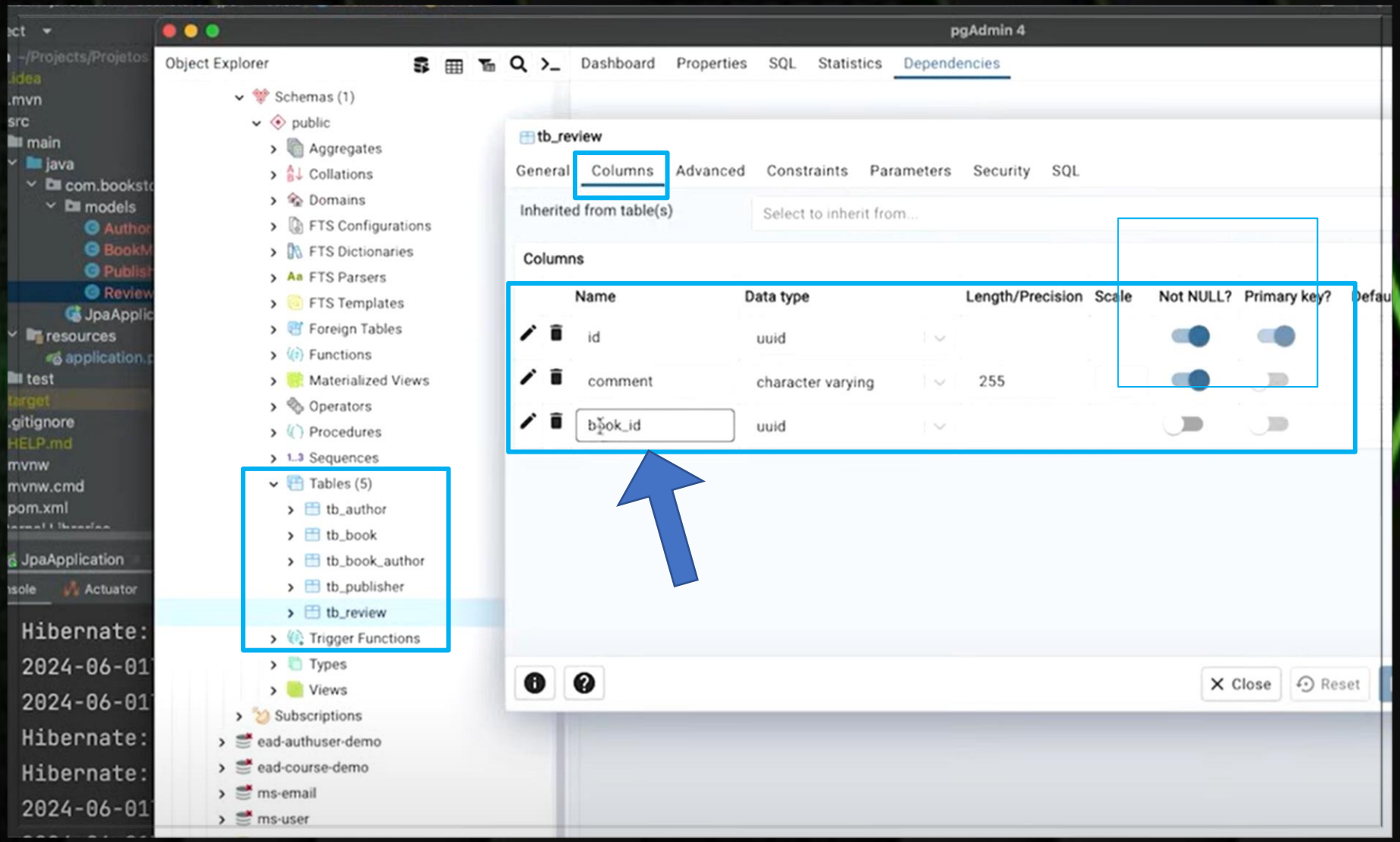
Após a criação dos relacionamentos nunca esquecer os get´s e set´s.

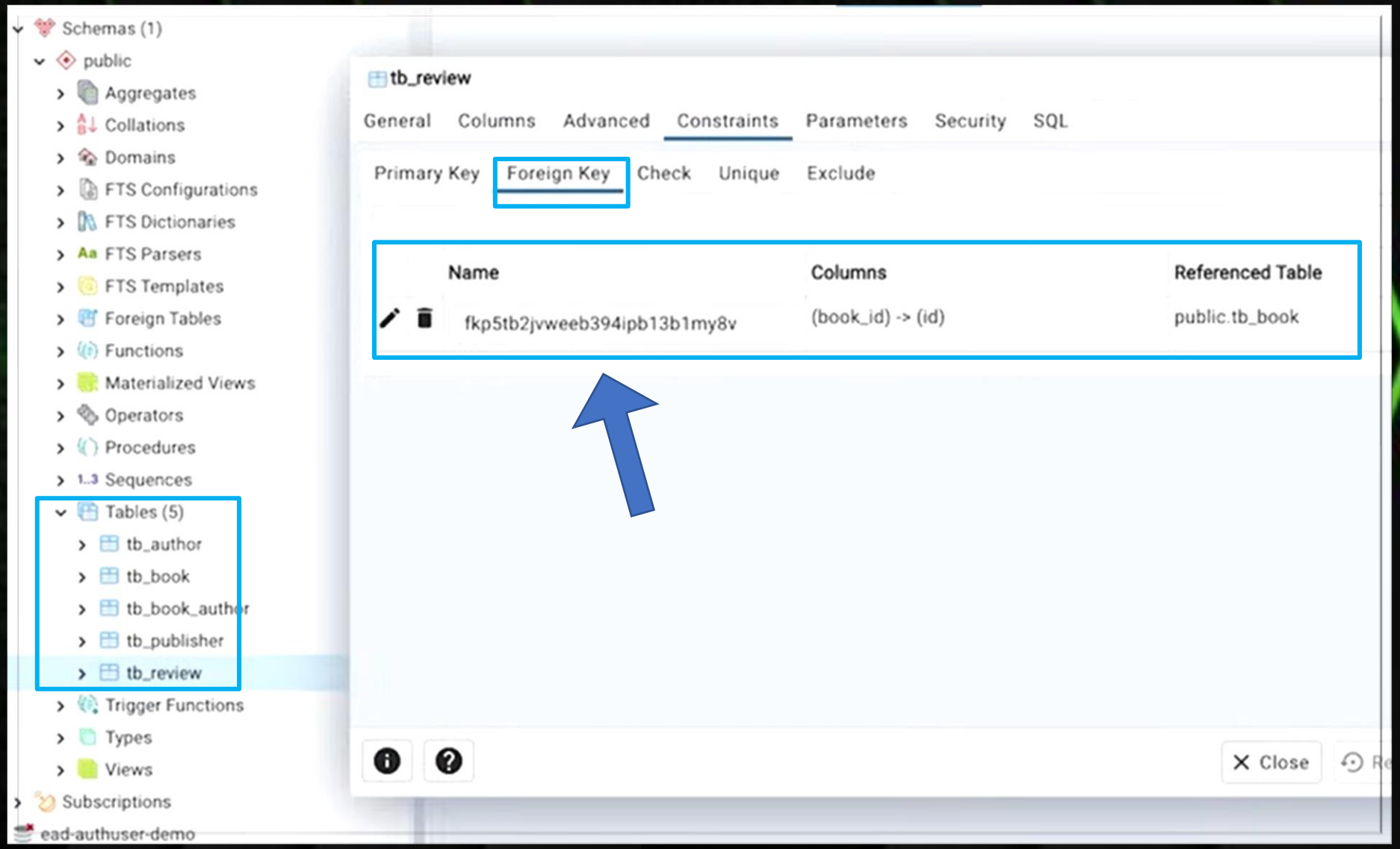
@OneToOne

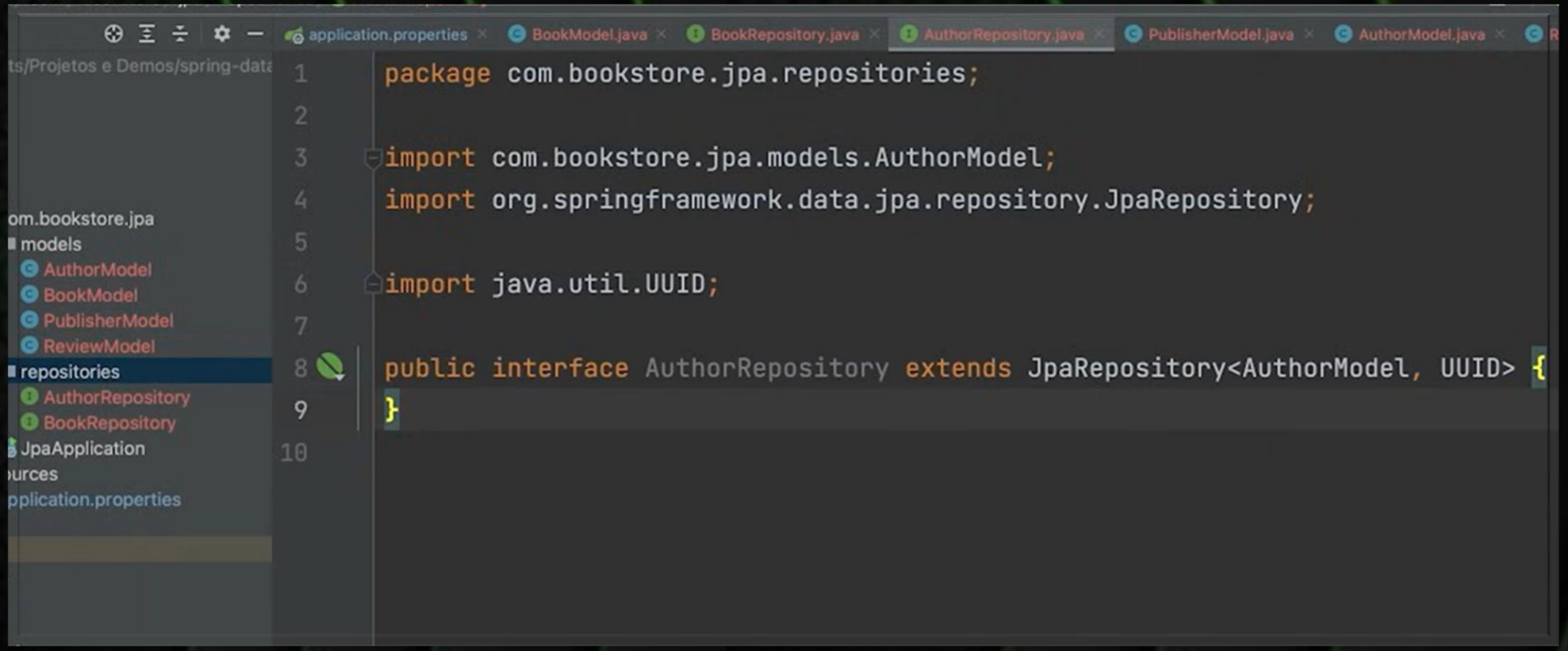


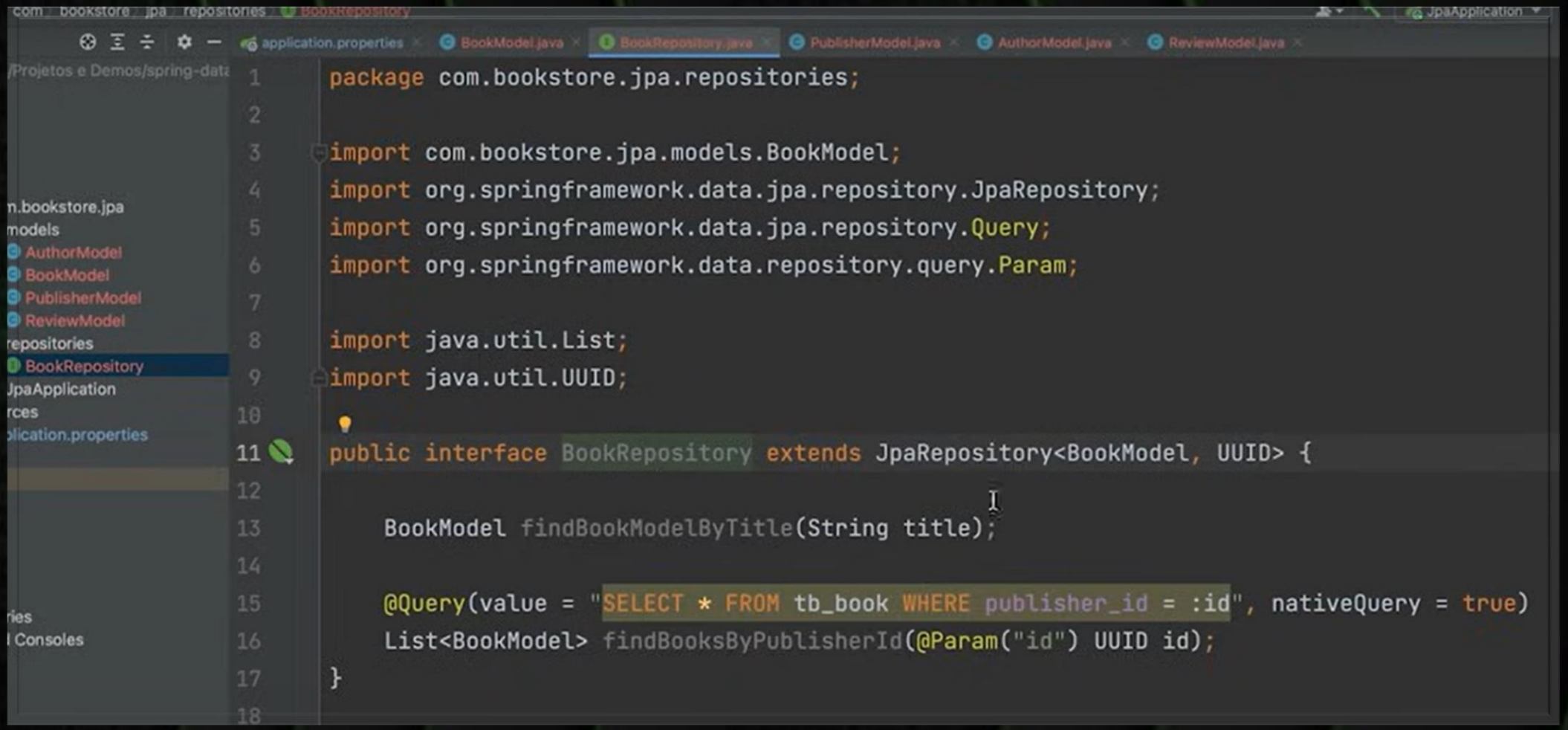


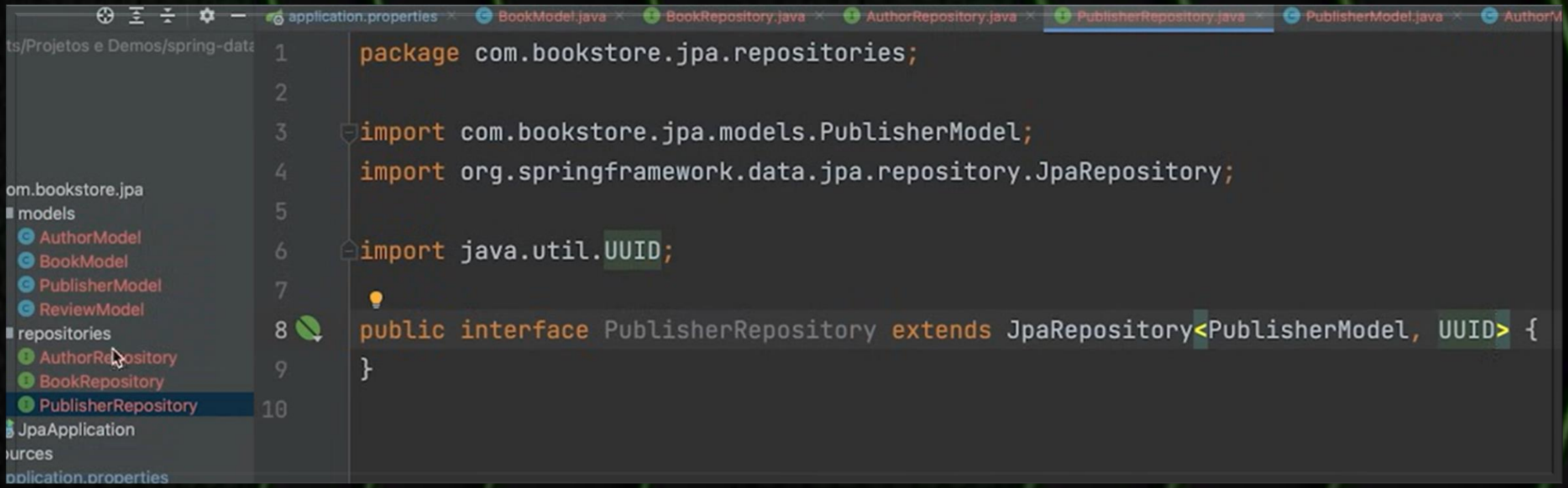
Após a criação dos relacionamentos nunca esquecer os get´s e set´s.

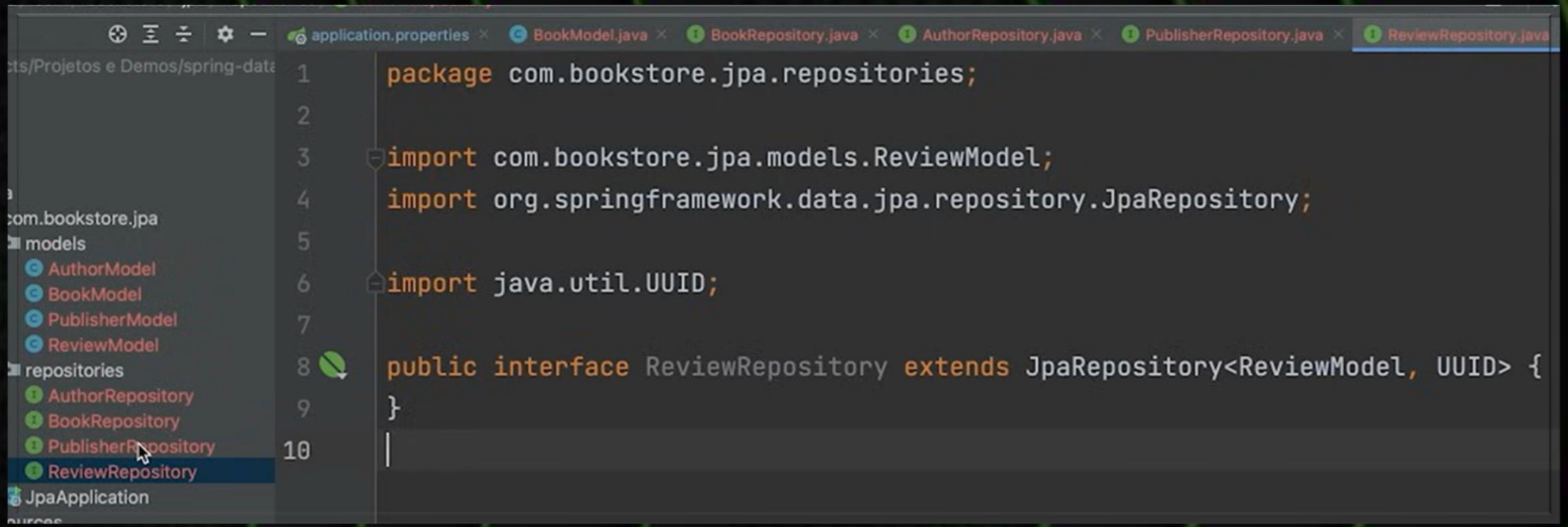

pgAdmin 4
Object Explorer
Dashboard
Properties
SQL
Statistics
Dependencies
Schemas (1)
public
Aggregates
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (5)
tb_author
tb_book
tb_book_author
tb_publisher
tb_review
Trigger Functions
Types
Views
Subscriptions
ead-authuser-demo
ead-course-demo
ms-email
ms-user
tb_review
General
Columns
Advanced
Constraints
Parameters
Security
SQL
Inherited from table(s)
Select to inherit from...
Columns
Name
Data type
Length/Precision
Scale
Not NULL?
Primary key?
id
uuid
comment
character varying
255
book_id
uuid
Close
Reset
JpaApplication
Actuator
Hibernate:
2024-06-01
Hibernate:
Hibernate:
2024-06-01

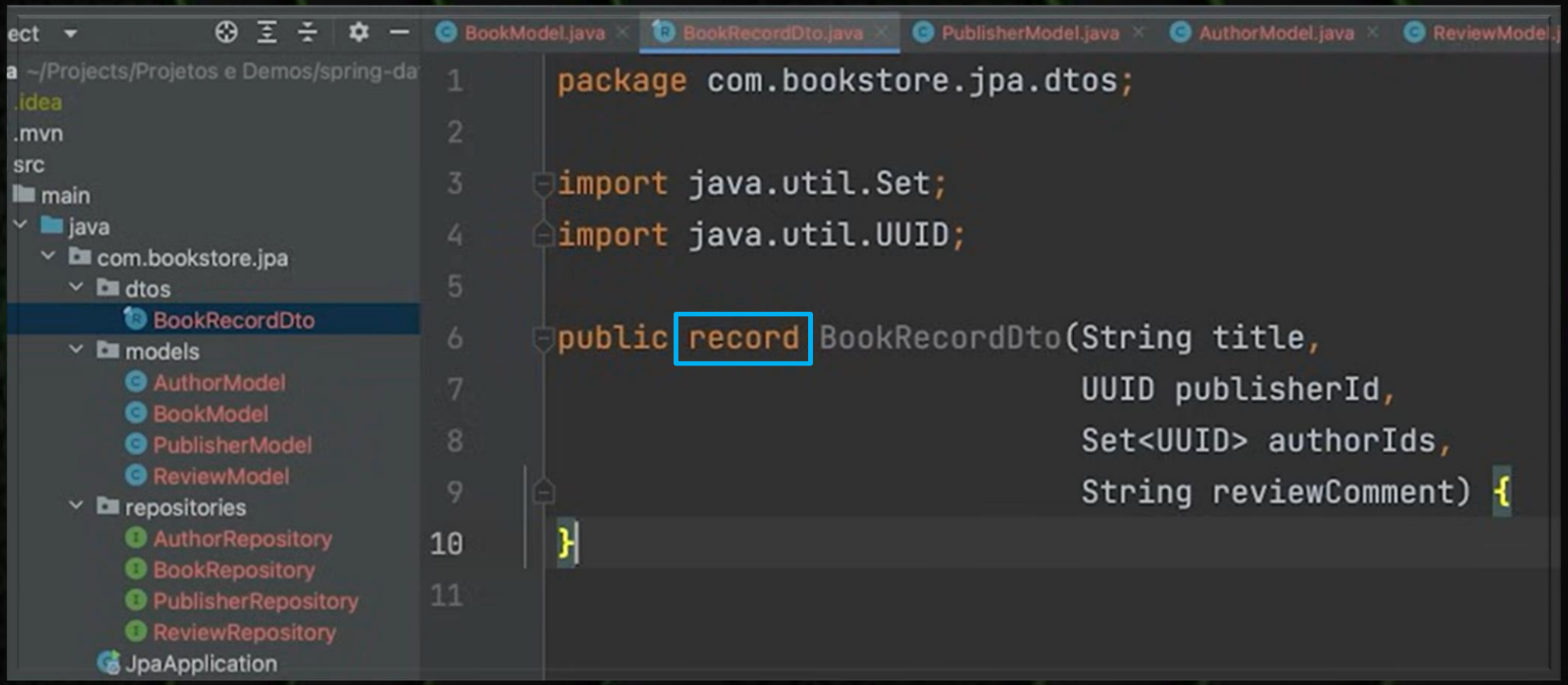

Schemas (1)
public
Aggregates
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (5)
tb_author
tb_book
tb_book_author
tb_publisher
tb_review
Trigger Functions
Types
Views
Subscriptions
ead-authuser-demo
tb_review
General
Columns
Advanced
Constraints
Parameters
Security
SQL
Primary Key
Foreign Key
Check
Unique
Exclude
Name
Columns
Referenced Table
fkp5tb2jvweeb394ipb13b1my8v
(book_id) -> (id)
public.tb_book
Close

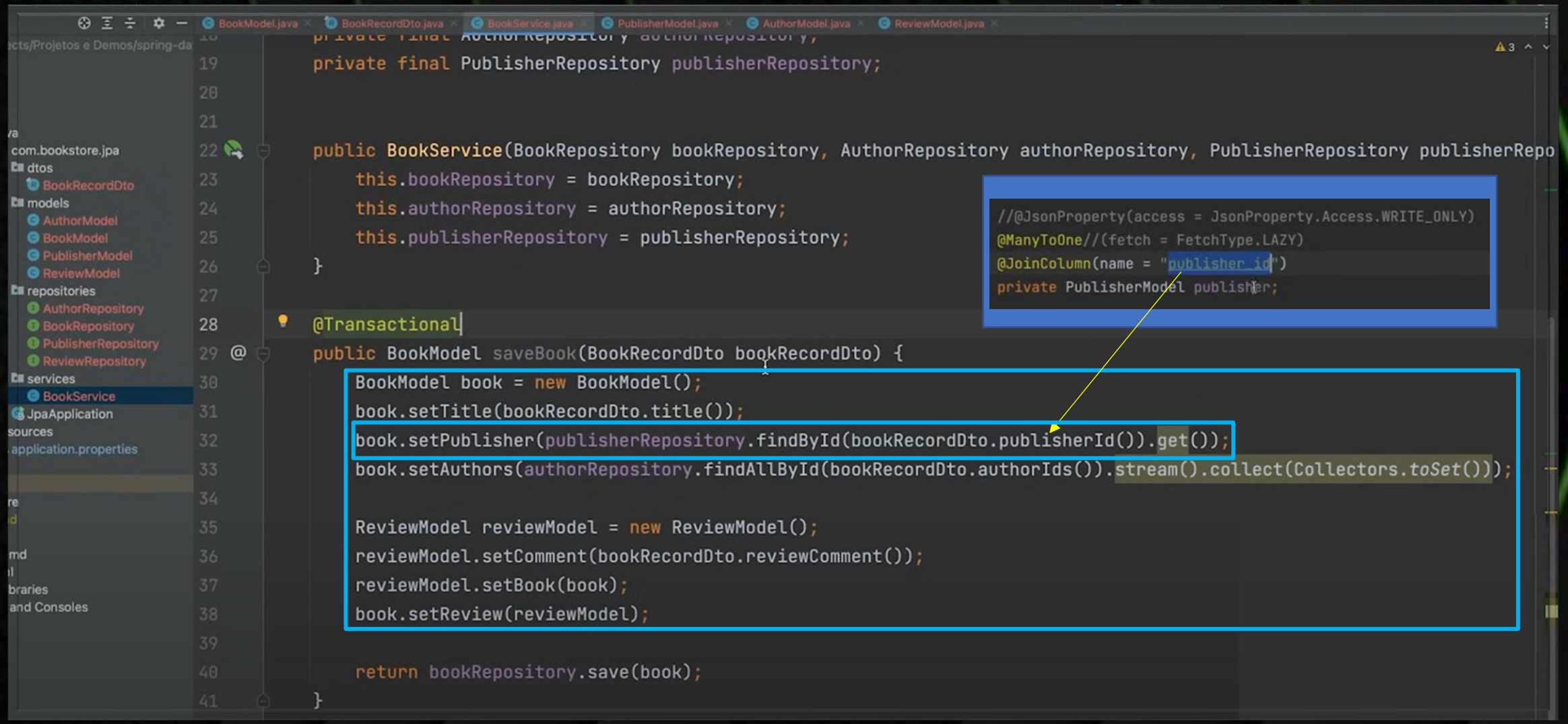

application.properties
BookModel.java
BookRepository.java
AuthorRepository.java
PublisherModel.java
AuthorModel.java
ts/Projetos e Demos/spring-data
om.bookstore.jpa
models
AuthorModel
BookModel
PublisherModel
ReviewModel
repositories
AuthorRepository
BookRepository
JpaApplication
urces
pplication.properties
1 package com.bookstore.jpa.repositories;
2
3 import com.bookstore.jpa.models.AuthorModel;
4 import org.springframework.data.jpa.repository.JpaRepository;
5
6 import java.util.UUID;
7
8 public interface AuthorRepository extends JpaRepository<AuthorModel, UUID> {
9 }
10

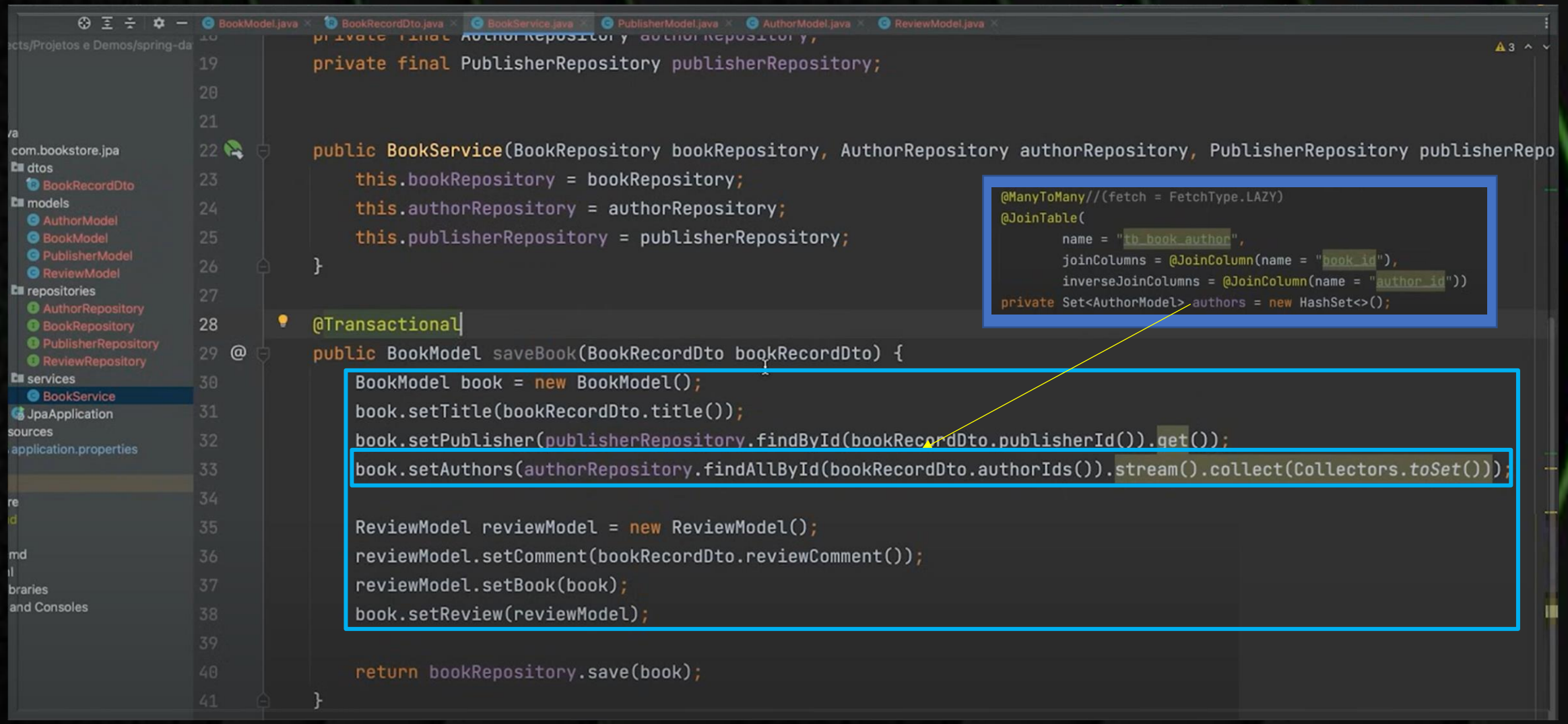

com bookstore jpa repositories BookRepository
JpaApplication
application.properties
BookModel.java
BookRepository.java
PublisherModel.java
AuthorModel.java
ReviewModel.java
/Projetos e Demos/spring-data
n.bookstore.jpa
models
AuthorModel
BookModel
PublisherModel
ReviewModel
repositories
BookRepository
JpaApplication
rces
lication.properties
ries
Consoles
1 package com.bookstore.jpa.repositories;
2
3 import com.bookstore.jpa.models.BookModel;
4 import org.springframework.data.jpa.repository.JpaRepository;
5 import org.springframework.data.jpa.repository.Query;
6 import org.springframework.data.repository.query.Param;
7
8 import java.util.List;
9 import java.util.UUID;
10
11 public interface BookRepository extends JpaRepository<BookModel, UUID> {
12
13 BookModel findBookModelByTitle(String title);
14
15 @Query(value = "SELECT * FROM tb_book WHERE publisher_id = :id", nativeQuery = true)
16 List<BookModel> findBooksByPublisherId(@Param("id") UUID id);
17 }
18

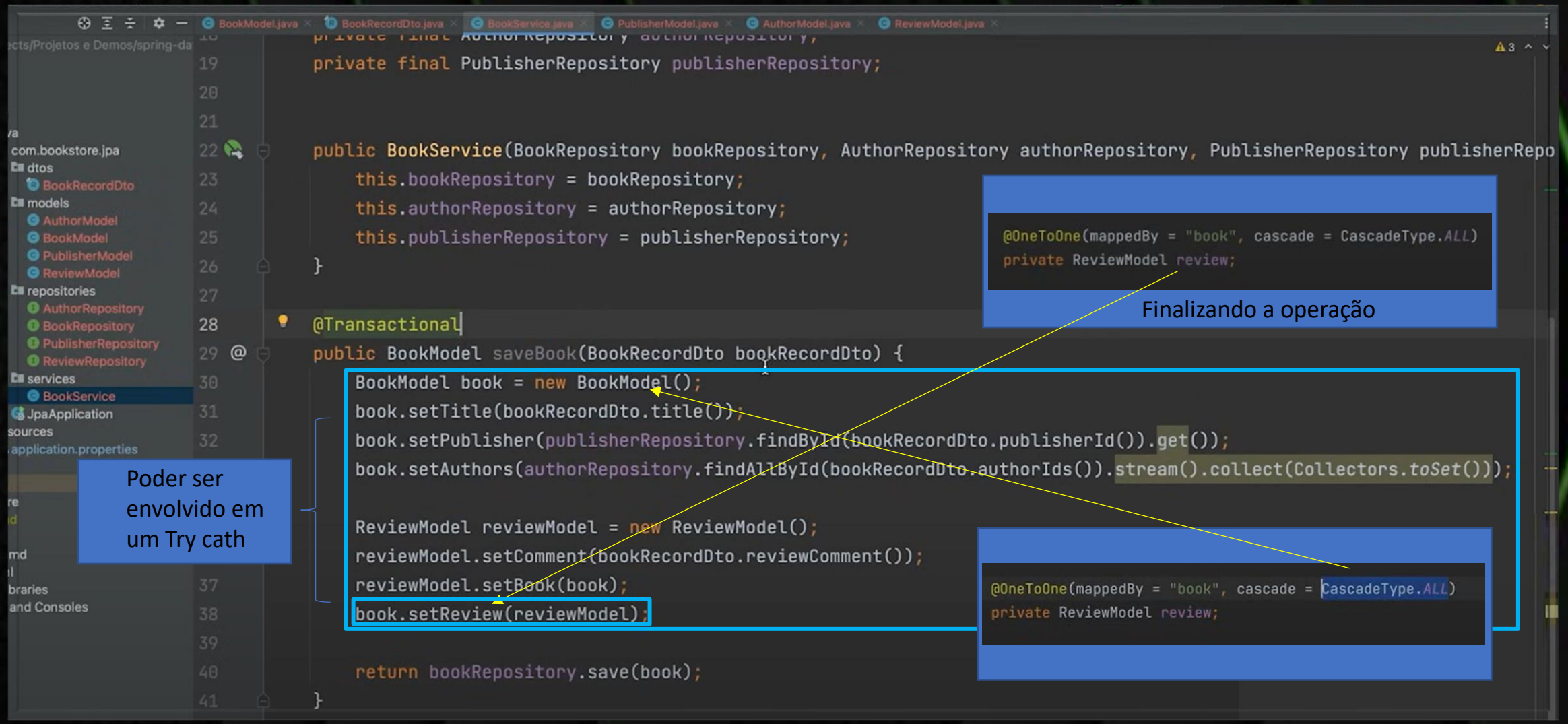

application.properties
BookModel.java
BookRepository.java
AuthorRepository.java
PublisherRepository.java
PublisherModel.java
ts/Projetos e Demos/spring-data
om.bookstore.jpa
models
AuthorModel
BookModel
PublisherModel
ReviewModel
repositories
BookRepository
PublisherRepository
JpaApplication
urces
pplication.properties
1 package com.bookstore.jpa.repositories;
2
3 import com.bookstore.jpa.models.PublisherModel;
4 import org.springframework.data.jpa.repository.JpaRepository;
5
6 import java.util.UUID;
7
8 public interface PublisherRepository extends JpaRepository<PublisherModel, UUID> {
9 }
10

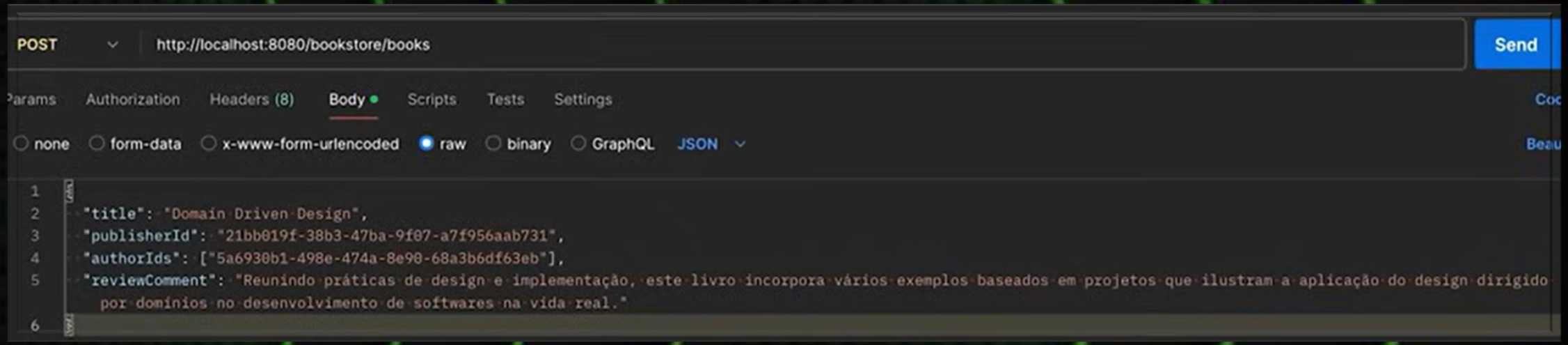

application.properties
BookModel.java
BookRepository.java
AuthorRepository.java
PublisherRepository.java
ReviewRepository.java
com.bookstore.jpa
models
AuthorModel
BookModel
PublisherModel
ReviewModel
repositories
AuthorRepository
BookRepository
PublisherRepository
ReviewRepository
JpaApplication
package com.bookstore.jpa.repositories;
import com.bookstore.jpa.models.ReviewModel;
import org.springframework.data.jpa.repository.JpaRepository;
import java.util.UUID;
public interface ReviewRepository extends JpaRepository<ReviewModel, UUID> {
}


BookModel.java
BookRecordDto.java
PublisherModel.java
AuthorModel.java
~/Projects/Projetos e Demos/spring-da
.idea
.mvn
src
main
java
com.bookstore.jpa
dtos
BookRecordDto
models
AuthorModel
BookModel
PublisherModel
ReviewModel
repositories
AuthorRepository
BookRepository
PublisherRepository
ReviewRepository
JpaApplication
1 package com.bookstore.jpa.dtos;
2
3 import java.util.Set;
4 import java.util.UUID;
5
6 public record BookRecordDto(String title,
7 UUID publisherId,
8 Set<UUID> authorIds,
9 String reviewComment) {
10 }
11


BookModel.java
BookRecordDto.java
BookService.java
PublisherModel.java
AuthorModel.java
ReviewModel.java
com.bookstore.jpa
dtos
BookRecordDto
models
AuthorModel
BookModel
PublisherModel
ReviewModel
repositories
AuthorRepository
BookRepository
PublisherRepository
ReviewRepository
services
BookService
JpaApplication
application.properties
private final PublisherRepository publisherRepository;
public BookService(BookRepository bookRepository, AuthorRepository authorRepository, PublisherRepository publisherRepo
this.bookRepository = bookRepository;
this.authorRepository = authorRepository;
this.publisherRepository = publisherRepository;
@Transactional
public BookModel saveBook(BookRecordDto bookRecordDto) {
BookModel book = new BookModel();
book.setTitle(bookRecordDto.title());
book.setPublisher(publisherRepository.findById(bookRecordDto.publisherId()).get());
book.setAuthors(authorRepository.findAllById(bookRecordDto.authorIds()).stream().collect(Collectors.toSet()));
ReviewModel reviewModel = new ReviewModel();
reviewModel.setComment(bookRecordDto.reviewComment());
reviewModel.setBook(book);
book.setReview(reviewModel);
return bookRepository.save(book);
//@JsonProperty(access = JsonProperty.Access.WRITE_ONLY)
@ManyToOne//(fetch = FetchType.LAZY)
@JoinColumn(name = "publisher_id")
private PublisherModel publisher;

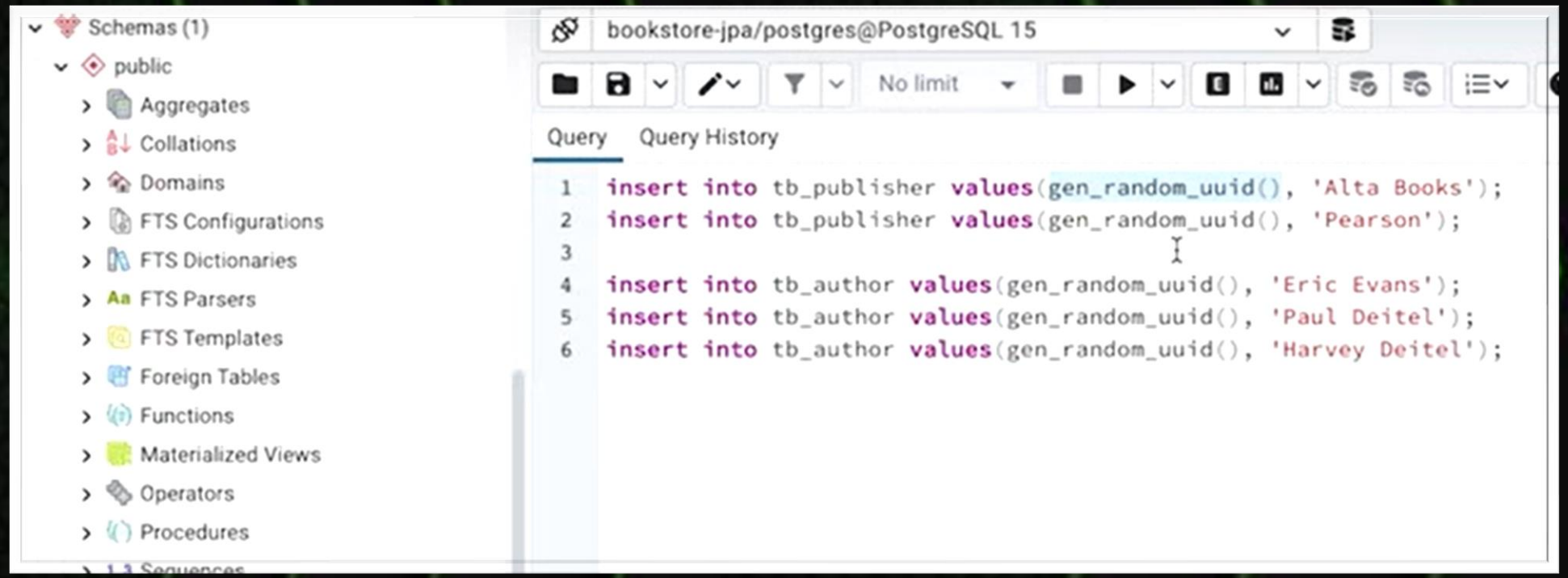

BookModel.java
BookRecordDto.java
BookService.java
PublisherModel.java
AuthorModel.java
ReviewModel.java
com.bookstore.jpa
dtos
BookRecordDto
models
AuthorModel
BookModel
PublisherModel
ReviewModel
repositories
AuthorRepository
BookRepository
PublisherRepository
ReviewRepository
services
BookService
JpaApplication
application.properties
private final PublisherRepository publisherRepository;
public BookService(BookRepository bookRepository, AuthorRepository authorRepository, PublisherRepository publisherRepo
this.bookRepository = bookRepository;
this.authorRepository = authorRepository;
this.publisherRepository = publisherRepository;
@Transactional
public BookModel saveBook(BookRecordDto bookRecordDto) {
BookModel book = new BookModel();
book.setTitle(bookRecordDto.title());
book.setPublisher(publisherRepository.findById(bookRecordDto.publisherId()).get());
book.setAuthors(authorRepository.findAllById(bookRecordDto.authorIds()).stream().collect(Collectors.toSet()));
ReviewModel reviewModel = new ReviewModel();
reviewModel.setComment(bookRecordDto.reviewComment());
reviewModel.setBook(book);
book.setReview(reviewModel);
return bookRepository.save(book);
}
@ManyToMany//(fetch = FetchType.LAZY)
@JoinTable(
name = "tb_book_author",
joinColumns = @JoinColumn(name = "book_id"),
inverseJoinColumns = @JoinColumn(name = "author_id"))
private Set<AuthorModel> authors = new HashSet<>();


BookModel.java
BookRecordDto.java
BookService.java
PublisherModel.java
AuthorModel.java
ReviewModel.java
com.bookstore.jpa
dtos
BookRecordDto
models
AuthorModel
BookModel
PublisherModel
ReviewModel
repositories
AuthorRepository
BookRepository
PublisherRepository
ReviewRepository
services
BookService
JpaApplication
sources
application.properties
private final PublisherRepository publisherRepository;
public BookService(BookRepository bookRepository, AuthorRepository authorRepository, PublisherRepository publisherRepo
this.bookRepository = bookRepository;
this.authorRepository = authorRepository;
this.publisherRepository = publisherRepository;
}
@Transactional
public BookModel saveBook(BookRecordDto bookRecordDto) {
BookModel book = new BookModel();
book.setTitle(bookRecordDto.title());
book.setPublisher(publisherRepository.findById(bookRecordDto.publisherId()).get());
book.setAuthors(authorRepository.findAllById(bookRecordDto.authorIds()).stream().collect(Collectors.toSet()));
ReviewModel reviewModel = new ReviewModel();
reviewModel.setComment(bookRecordDto.reviewComment());
reviewModel.setBook(book);
book.setReview(reviewModel);
return bookRepository.save(book);
}
@OneToOne(mappedBy = "book", cascade = CascadeType.ALL)
private ReviewModel review;
Finalizando a operação
Poder ser envolvido em um Try cath
@OneToOne(mappedBy = "book", cascade = CascadeType.ALL)
private ReviewModel review;

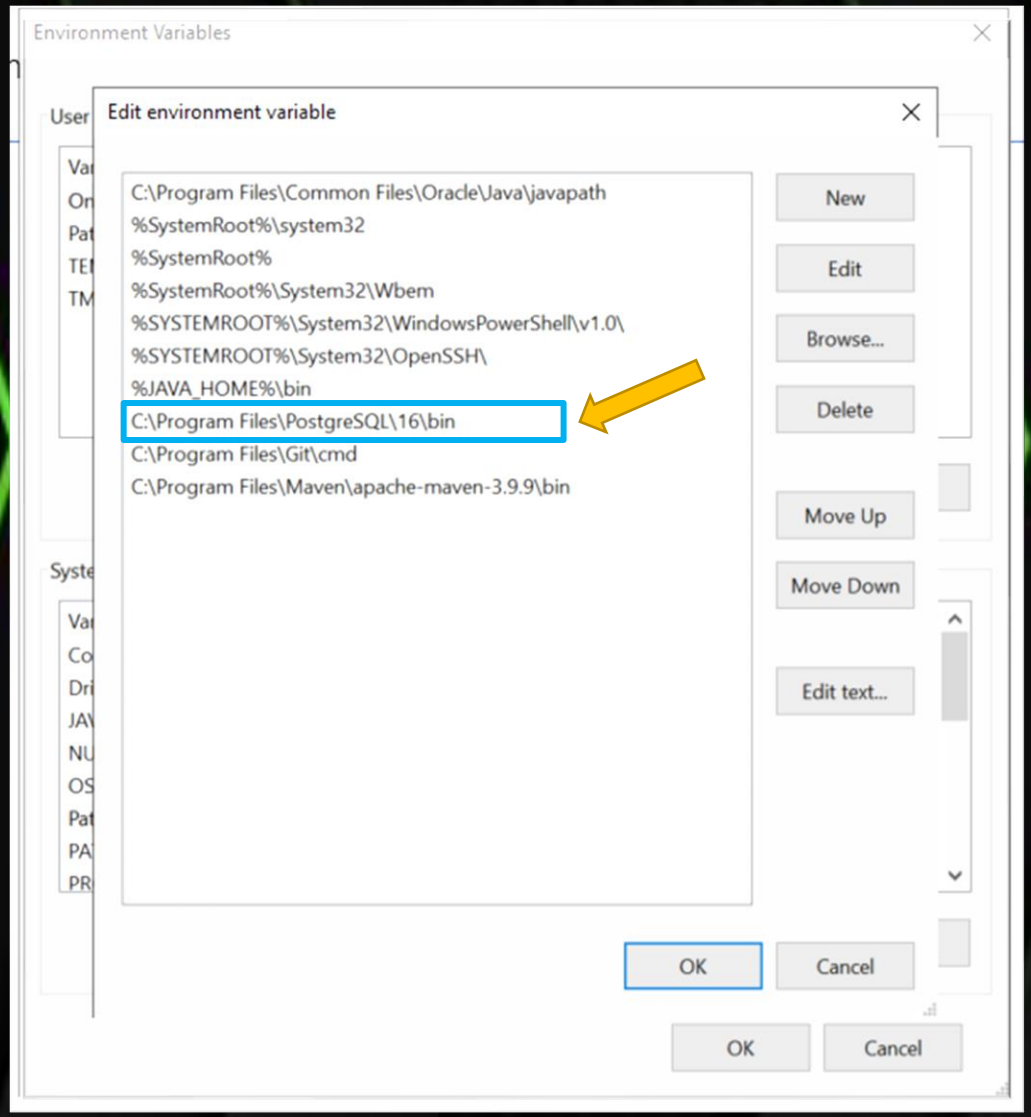

POST
http://localhost:8080/bookstore/books
Send
Params
Authorization
Headers (8)
Body
Scripts
Tests
Settings
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
{
"title": "Domain Driven Design",
"publisherId": "21bb019f-38b3-47ba-9f07-a7f956aab731",
"authorIds": ["5a6930b1-498e-474a-8e90-68a3b6df63eb"],
"reviewComment": "Reunindo práticas de design e implementação, este livro incorpora vários exemplos baseados em projetos que ilustram a aplicação do design dirigido por domínios no desenvolvimento de softwares na vida real."
}


```java
@RestController
@RequestMapping("/bookstore/books")
public class BookController {

    private final BookService bookService;

    public BookController(BookService bookService) {
        this.bookService = bookService;
    }

    @PostMapping
    public ResponseEntity<BookModel> saveBook(@RequestBody BookRecordDto bookRecordDto) {
        return ResponseEntity.status(HttpStatus.CREATED).body(bookService.saveBook(bookRecordDto));
    }
}
```

PgAdmin

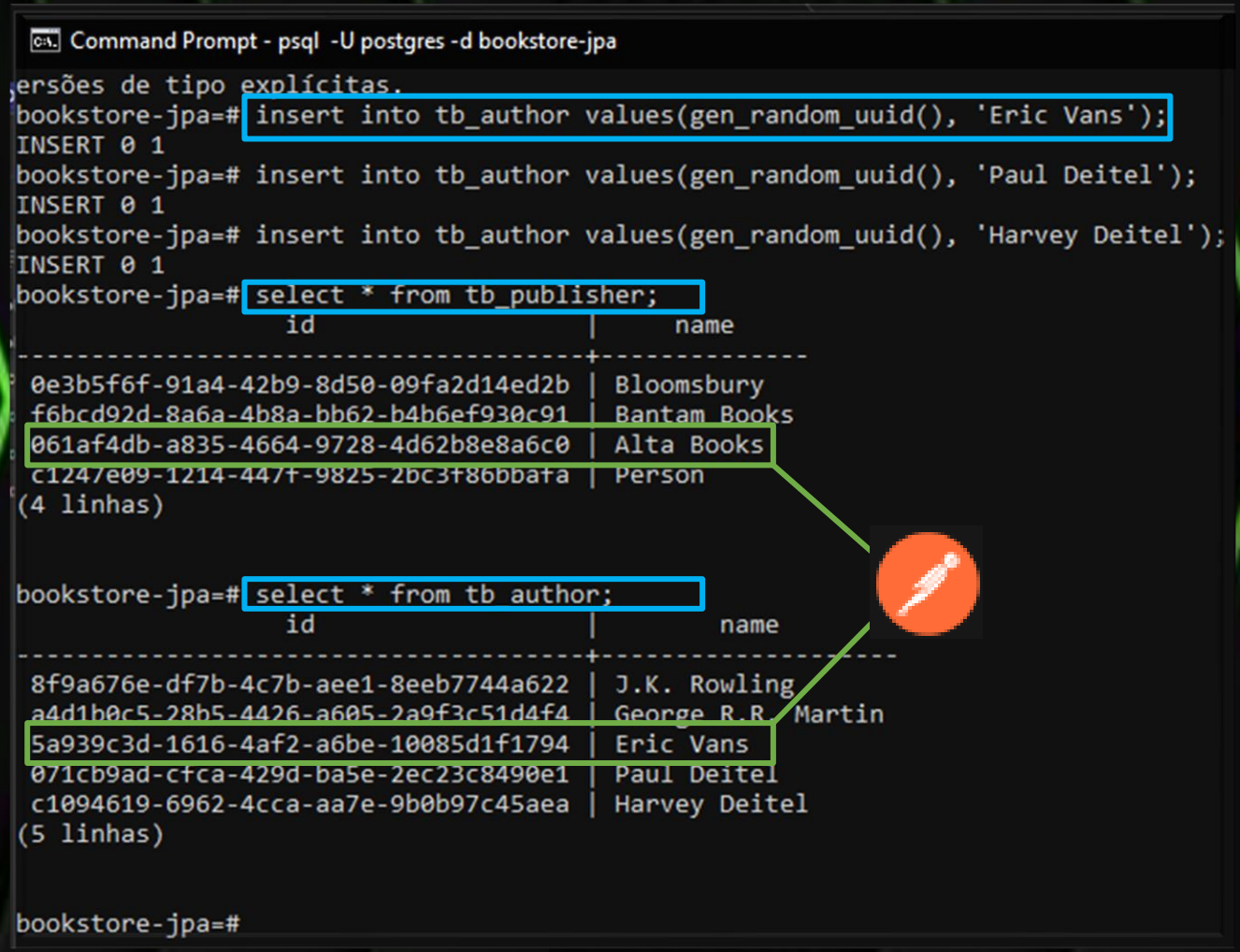

Schemas (1)
public
Aggregates
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
bookstore-jpa/postgres@PostgreSQL 15
No limit
Query
Query History
1 insert into tb_publisher values(gen_random_uuid(), 'Alta Books');
2 insert into tb_publisher values(gen_random_uuid(), 'Pearson');
3
4 insert into tb_author values(gen_random_uuid(), 'Eric Evans');
5 insert into tb_author values(gen_random_uuid(), 'Paul Deitel');
6 insert into tb_author values(gen_random_uuid(), 'Harvey Deitel');

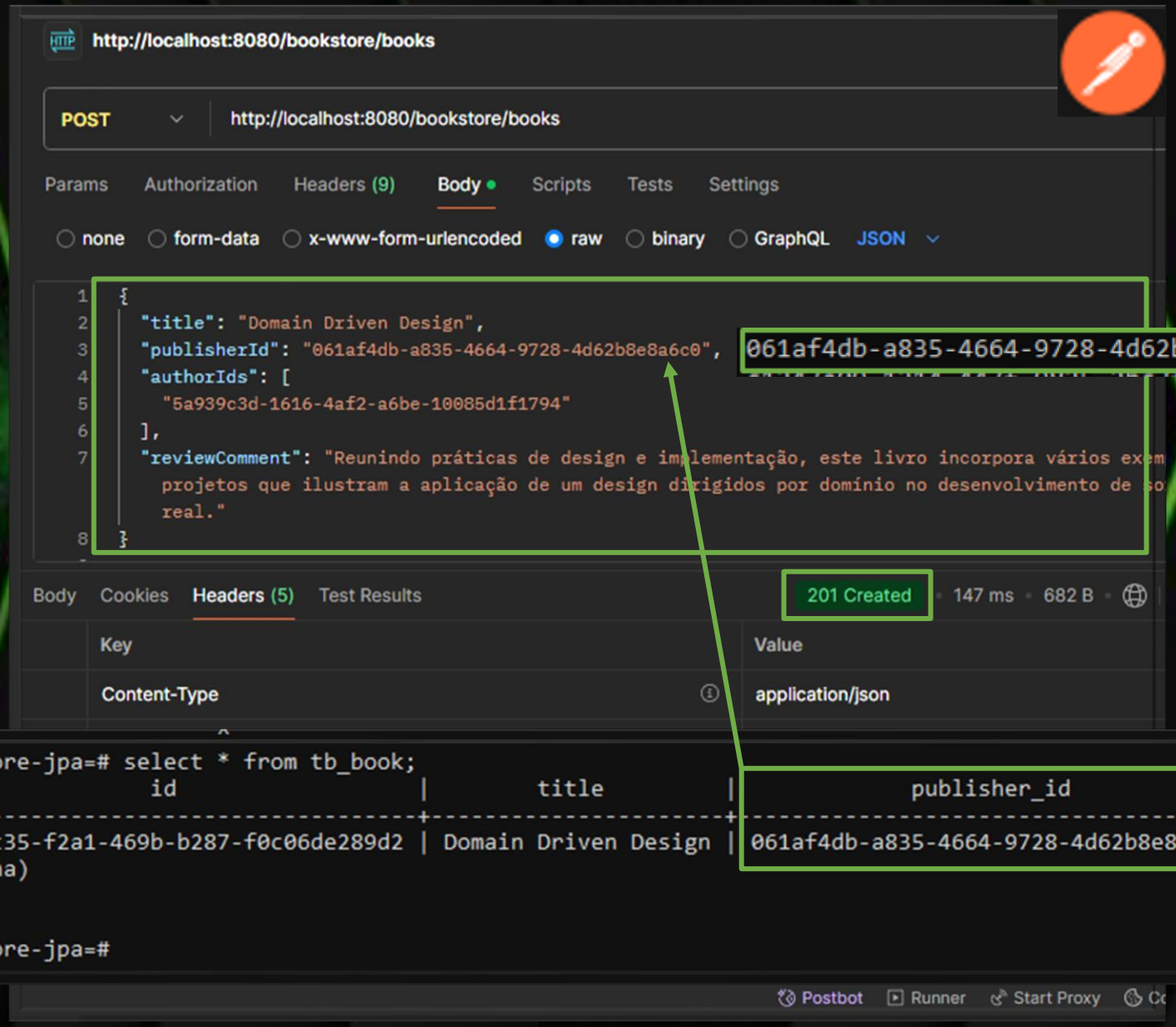
Configurando psql command line

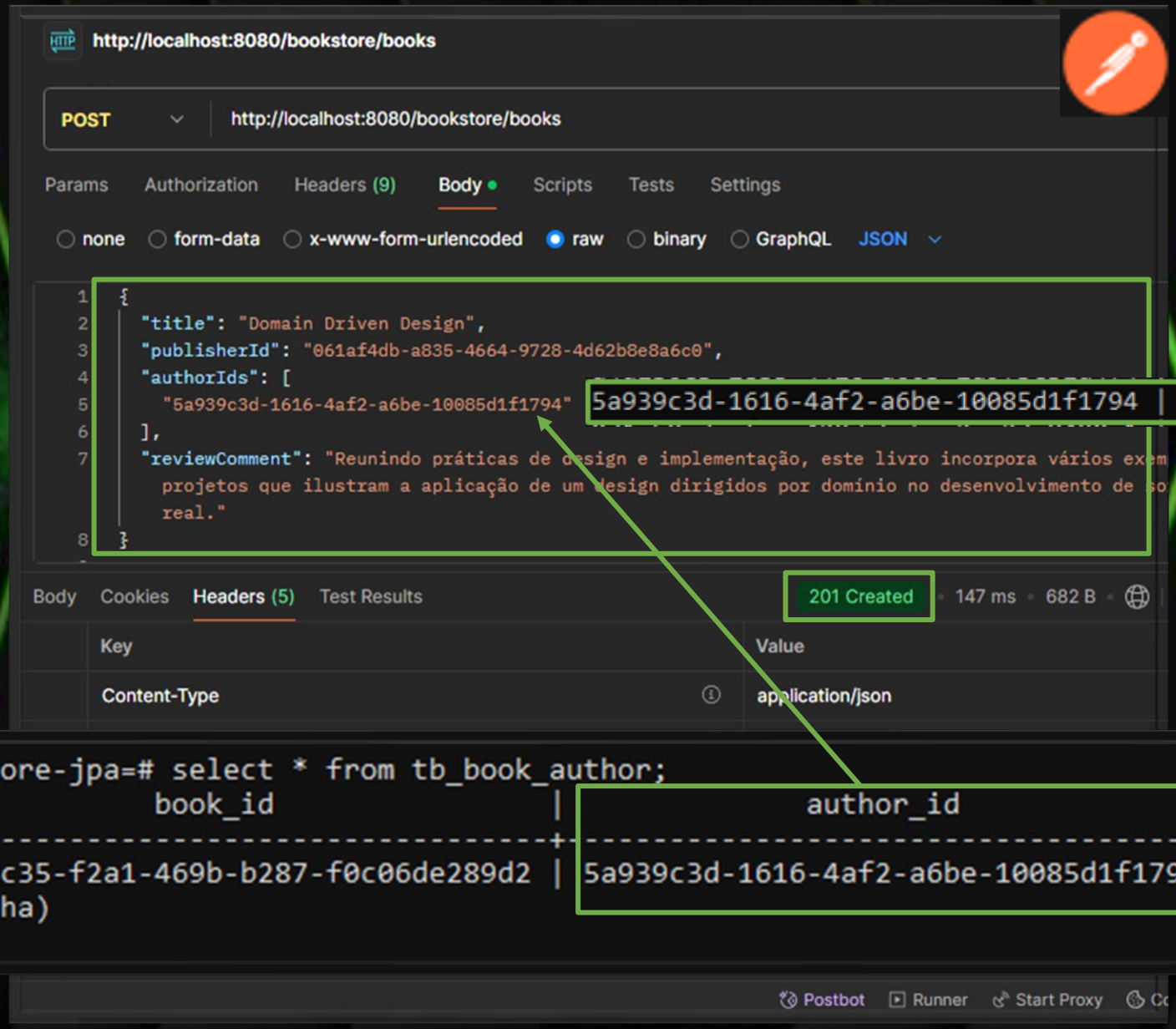


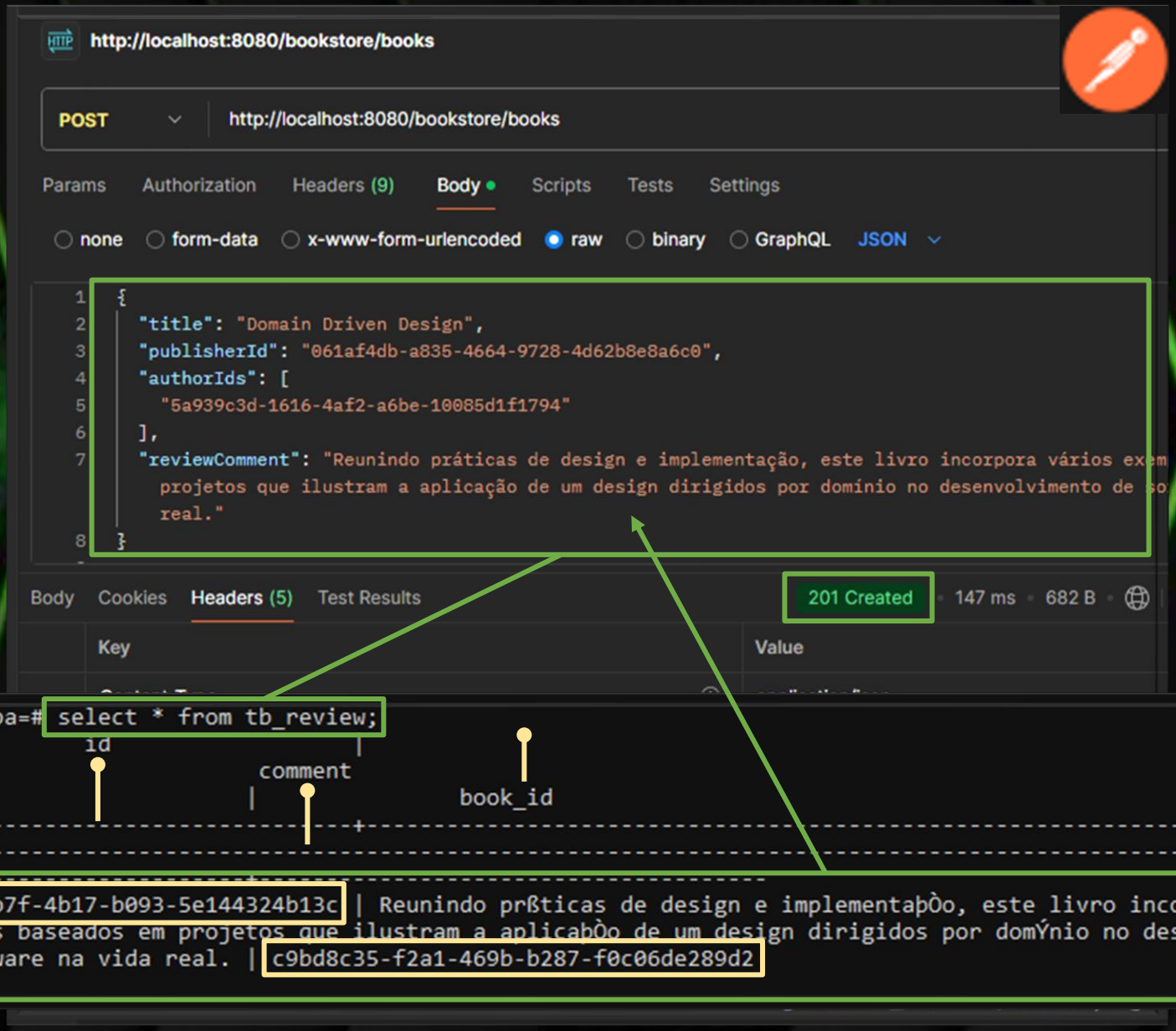
psql –U postgres –d bookstore-jpa
Password: *******

```
bookstore-jpa=# insert into tb_publisher values(gen_random_uuid(), 'Alta Books');
INSERT 0 1
bookstore-jpa=# insert into tb_publisher values(gen_random_uuid(), 'Person');
INSERT 0 1
bookstore-jpa=# insert into tb_author values(gen_random_uuid(), 'Eric Vans');
INSERT 0 1
bookstore-jpa=# insert into tb_author values(gen_random_uuid(), 'Paul Deitel');
INSERT 0 1
bookstore-jpa=# insert into tb_author values(gen_random_uuid(), 'Harvey Deitel');
INSERT 0 1
bookstore-jpa=#
```


Command Prompt - psql -U postgres -d bookstore-jpa
ersões de tipo explícitas.
bookstore-jpa=# insert into tb_author values(gen_random_uuid(), 'Eric Vans');
INSERT 0 1
bookstore-jpa=# insert into tb_author values(gen_random_uuid(), 'Paul Deitel');
INSERT 0 1
bookstore-jpa=# insert into tb_author values(gen_random_uuid(), 'Harvey Deitel');
INSERT 0 1
bookstore-jpa=# select * from tb_publisher;
id | name
0e3b5f6f-91a4-42b9-8d50-09fa2d14ed2b | Bloomsbury
f6bcd92d-8a6a-4b8a-bb62-b4b6ef930c91 | Bantam Books
061af4db-a835-4664-9728-4d62b8e8a6c0 | Alta Books
c1247e09-1214-447f-9825-2bc3f86bbafa | Person
(4 linhas)
bookstore-jpa=# select * from tb_author;
id | name
8f9a676e-df7b-4c7b-aee1-8eeb7744a622 | J.K. Rowling
a4d1b0c5-28b5-4426-a605-2a9f3c51d4f4 | George R.R. Martin
5a939c3d-1616-4af2-a6be-10085d1f1794 | Eric Vans
071cb9ad-cfca-429d-ba5e-2ec23c8490e1 | Paul Deitel
c1094619-6962-4cca-aa7e-9b0b97c45aea | Harvey Deitel
(5 linhas)
bookstore-jpa=#


http://localhost:8080/bookstore/books
POST
http://localhost:8080/bookstore/books
Params
Authorization
Headers (9)
Body
Scripts
Tests
Settings
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
{
"title": "Domain Driven Design",
"publisherId": "061af4db-a835-4664-9728-4d62b8e8a6c0",
"authorIds": [
"5a939c3d-1616-4af2-a6be-10085d1f1794"
],
"reviewComment": "Reunindo práticas de design e implementação, este livro incorpora vários exem
projetos que ilustram a aplicação de um design dirigidos por domínio no desenvolvimento de so
real."
}
061af4db-a835-4664-9728-4d62b8e8a6c0 | Alta Books
Body
Cookies
Headers (5)
Test Results
201 Created
147 ms
682 B
Key
Value
Content-Type
application/json
bookstore-jpa=# select * from tb_book;
id | title | publisher_id
c9bd8c35-f2a1-469b-b287-f0c06de289d2 | Domain Driven Design | 061af4db-a835-4664-9728-4d62b8e8a6c0
(1 linha)
bookstore-jpa=#
Postbot
Runner
Start Proxy

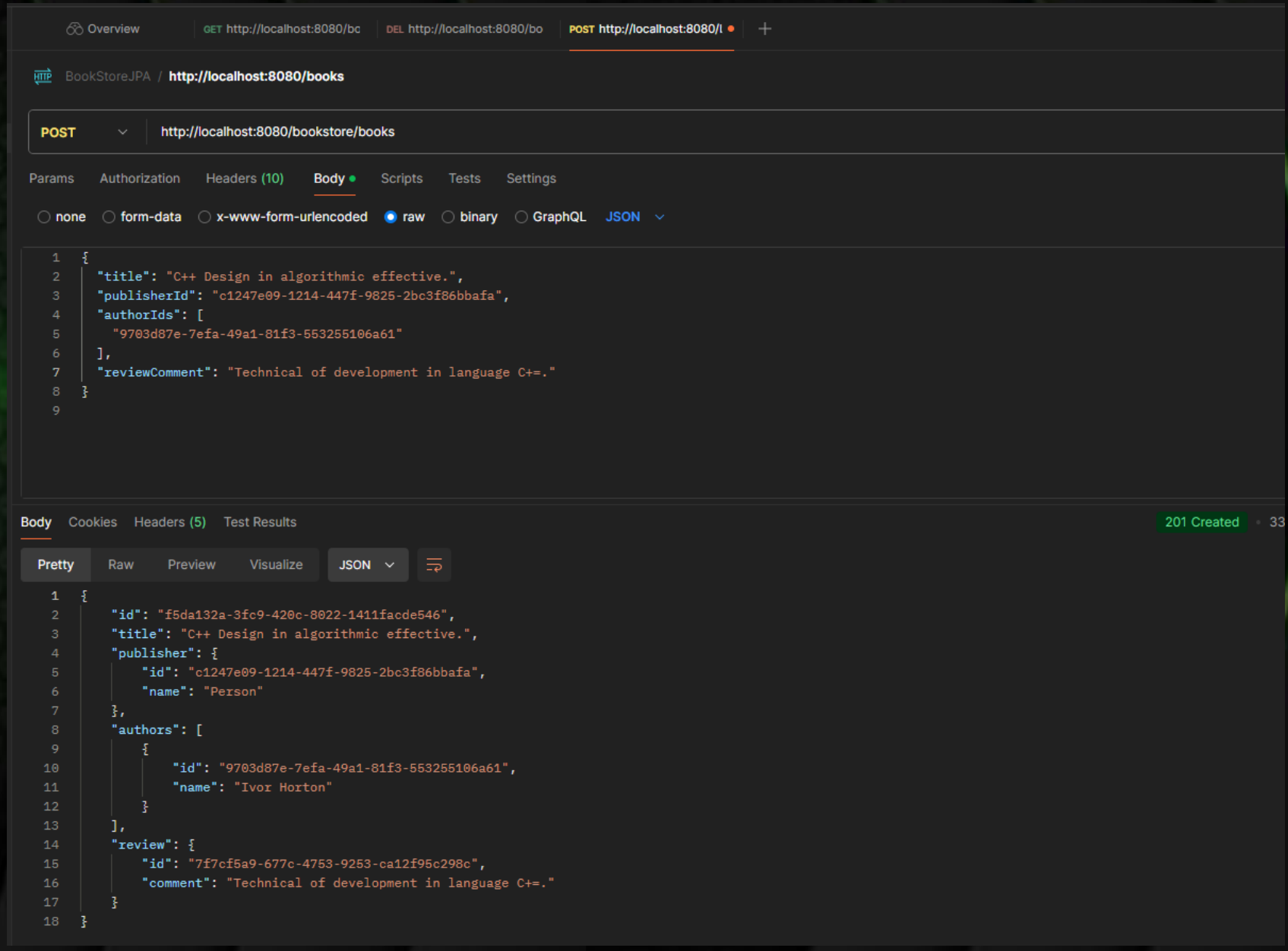

http://localhost:8080/bookstore/books
POST
http://localhost:8080/bookstore/books
Params
Authorization
Headers (9)
Body
Scripts
Tests
Settings
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
{
"title": "Domain Driven Design",
"publisherId": "061af4db-a835-4664-9728-4d62b8e8a6c0",
"authorIds": [
"5a939c3d-1616-4af2-a6be-10085d1f1794"
],
"reviewComment": "Reunindo práticas de design e implementação, este livro incorpora vários exem
projetos que ilustram a aplicação de um design dirigidos por domínio no desenvolvimento de so
real."
}
5a939c3d-1616-4af2-a6be-10085d1f1794 | Eric Vans
Body
Cookies
Headers (5)
Test Results
201 Created
147 ms
682 B
Key
Value
Content-Type
application/json
bookstore-jpa=# select * from tb_book_author;
book_id | author_id
c9bd8c35-f2a1-469b-b287-f0c06de289d2 | 5a939c3d-1616-4af2-a6be-10085d1f1794
(1 linha)
Postbot
Runner
Start Proxy


http://localhost:8080/bookstore/books
POST
http://localhost:8080/bookstore/books
Params
Authorization
Headers (9)
Body
Scripts
Tests
Settings
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
{
"title": "Domain Driven Design",
"publisherId": "061af4db-a835-4664-9728-4d62b8e8a6c0",
"authorIds": [
"5a939c3d-1616-4af2-a6be-10085d1f1794"
],
"reviewComment": "Reunindo práticas de design e implementação, este livro incorpora vários exem
projetos que ilustram a aplicação de um design dirigidos por domínio no desenvolvimento de so
real."
}
Body
Cookies
Headers (5)
Test Results
201 Created
147 ms
682 B
Key
Value
bookstore-jpa=# select * from tb_review;
id
comment
book_id
916461ee-7b7f-4b17-b093-5e144324b13c | Reunindo prßticas de design e implementaþÒo, este livro incorpora vßr
ios exemplos baseados em projetos que ilustram a aplicabÒo de um design dirigidos por domÝnio no desenvolvime
nto de software na vida real. | c9bd8c35-f2a1-469b-b287-f0c06de289d2
(1 linha)

Particularities

ote that when commenting
ut lines 24, 25, 29 and 30,
the outputs do not inhibit
elationship dependencies...

```
23
24      //@JsonProperty(access = JsonProperty.Access.WRITE_ONLY)
25      @ManyToOne //(fetch = FetchType.EAGER)
26      @JoinColumn(name = "publisher_id")
27      private PublisherModel publisher;
28
29      //@JsonProperty(access = JsonProperty.Access.WRITE_ONLY)
30      @ManyToMany //(fetch = FetchType.EAGER)
31      @JoinTable(
32              name = "tb_book_author",
33              joinColumns = @JoinColumn(name = "book_id"),
34              inverseJoinColumns = @JoinColumn(name = "author_id"))
35      private Set<AuthorModel> authors = new HashSet<>();
```


Overview
GET http://localhost:8080/bc
DEL http://localhost:8080/bo
POST http://localhost:8080/l
BookStoreJPA / http://localhost:8080/books
POST
http://localhost:8080/bookstore/books
Params
Authorization
Headers (10)
Body
Scripts
Tests
Settings
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
{
"title": "C++ Design in algorithmic effective.",
"publisherId": "c1247e09-1214-447f-9825-2bc3f86bbafa",
"authorIds": [
"9703d87e-7efa-49a1-81f3-553255106a61"
],
"reviewComment": "Technical of development in language C+=."
}
Body
Cookies
Headers (5)
Test Results
201 Created
Pretty
Raw
Preview
Visualize
JSON
{
"id": "f5da132a-3fc9-420c-8022-1411facde546",
"title": "C++ Design in algorithmic effective.",
"publisher": {
"id": "c1247e09-1214-447f-9825-2bc3f86bbafa",
"name": "Person"
},
"authors": [
{
"id": "9703d87e-7efa-49a1-81f3-553255106a61",
"name": "Ivor Horton"
}
],
"review": {
"id": "7f7cf5a9-677c-4753-9253-ca12f95c298c",
"comment": "Technical of development in language C+=."
}
}

Particularities

ote that when commenting lines 24, 25, 29 and 30, the outputs inhibit relationship dependencies...

```
23
24      @JsonProperty(access = JsonProperty.Access.WRITE_ONLY)
25      @ManyToOne (fetch = FetchType.EAGER)
26      @JoinColumn(name = "publisher_id")
27      private PublisherModel publisher;
28
29      @JsonProperty(access = JsonProperty.Access.WRITE_ONLY)
30      @ManyToMany (fetch = FetchType.EAGER)
31      @JoinTable(
32              name = "tb_book_author",
33              joinColumns = @JoinColumn(name = "book_id"),
34              inverseJoinColumns = @JoinColumn(name = "author_id"))
35      private Set<AuthorModel> authors = new HashSet<>();
```

BookStoreJPA / **http://localhost:8080/books**

POST http://localhost:8080/bookstore/books

Params Authorization Headers (10) Body Scripts Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
{
  "title": "Don Quixote.",
  "publisherId": "0d0febf0-2e61-40cb-8213-78095f6761c7",
  "authorIds": [
    "7382d257-3d0a-4757-9cac-9194bab4682f",
    "3ee5416c-c900-43bd-8489-6542911d5b6e"
  ],
  "reviewComment": "Fyodor Dostoyevsky. A founding work of modern Western literature, Cervantes' masterpiece has been translated into more than 60 languages."
}
```

Body Cookies Headers (5) Test Results 201 Created • 264 ms • 4

Pretty Raw Preview Visualize JSON

```
{
    "id": "9f268d56-1b75-4e76-baf6-af3c484c6b19",
    "title": "Don Quixote.",
    "review": {
        "id": "942e7e9e-483c-43b4-bc5e-35c9b063d835",
        "comment": "Fyodor Dostoyevsky. A founding work of modern Western literature, Cervantes' masterpiece has been translated into more than 60 languages."
    }
}
```

Entrando com dados..

```
bookstore-jpa=# insert into tb_publisher values(gen_random_uuid(), 'Microsoft’); <ENTER>
INSERT 0 1

bookstore-jpa=# insert into tb_author values(gen_random_uuid(), 'Ivor Horton’); <ENTER>
INSERT 0 1
bookstore-jpa=#


bookstore-jpa=# select * from tb_publisher; <ENTER>

                id                |     name
--------------------------------------+-------------
 0e3b5f6f-91a4-42b9-8d50-09fa2d14ed2b | Bloomsbury
 f6bcd92d-8a6a-4b8a-bb62-b4b6ef930c91 | Bantam Books
 061af4db-a835-4664-9728-4d62b8e8a6c0 | Alta Books
 c1247e09-1214-447f-9825-2bc3f86bbafa | Person
                                      | Microsoft
(5 linhas)
```


```
bookstore-jpa=# select * from tb_author; <ENTER>

                  id                  |       name
--------------------------------------+-------------------
 8f9a676e-df7b-4c7b-aee1-8eeb7744a622 | J.K. Rowling
 a4d1b0c5-28b5-4426-a605-2a9f3c51d4f4 | George R.R. Martin
 5a939c3d-1616-4af2-a6be-10085d1f1794 | Eric Vans
 071cb9ad-cfca-429d-ba5e-2ec23c8490e1 | Paul Deitel
 c1094619-6962-4cca-aa7e-9b0b97c45aea | Harvey Deitel
 9703d87e-7efa-49a1-81f3-553255106a61 | Ivor Horton
(6 linhas)
```

POST http://localhost:8080/bookstore/books Send

Params Authorization Headers (9) Body Scripts Tests Settings Cookies

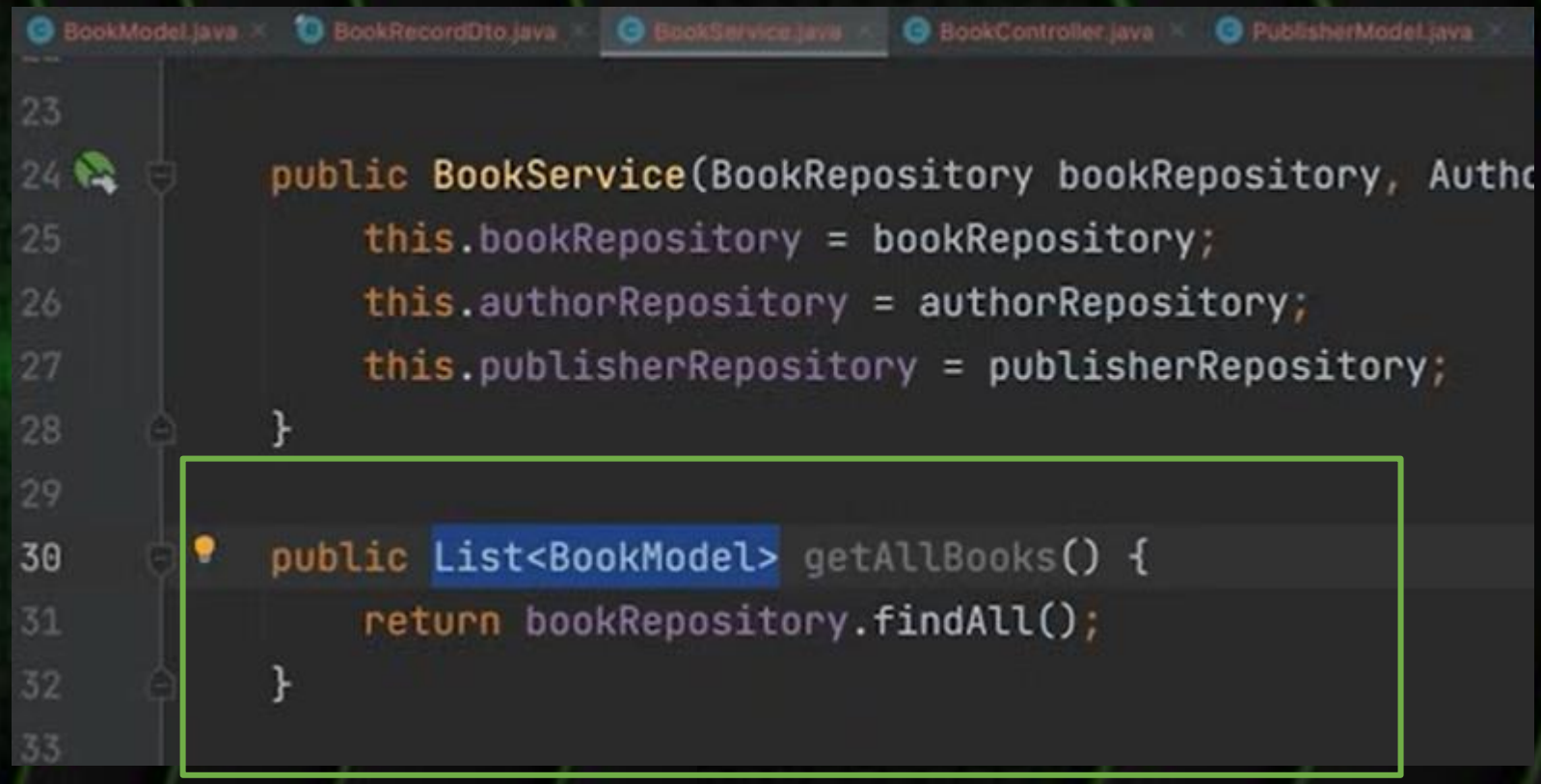
none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
{
  "title": "Introduction C++ Development System.",
  "publisherId": "826eb1ff-06da-4fdc-b339-b1d7764b4f37",
  "authorIds": [
    "9703d87e-7efa-49a1-81f3-553255106a61"
  ],
  "reviewComment": "I began to study C a while back. I think it's really important to get the fundamentals and approaches right and King's book just isn't good IMO."
}
```

Body Cookies Headers (5) Test Results 201 Created 52 ms 629 B Save Response

Pretty Raw Preview Visualize JSON

```
{
    "id": "0bd9cd25-7a6e-49c1-9849-29b46384d95d",
    "title": "Introduction C++ Development System.",
    "publisher": {
        "id": "826eb1ff-06da-4fdc-b339-b1d7764b4f37",
        "name": "Microsoft"
    },
    "authors": [
        {
            "id": "9703d87e-7efa-49a1-81f3-553255106a61",
            "name": "Ivor Horton"
        }
    ],
    "review": {
        "id": "f21b8483-2619-4f9f-81f0-5a52e0673ca7",
        "comment": "I began to study C a while back. I think it's really important to get the fundamentals and approaches right and King's book just isn't good IMO."
    }
}
```


BookModel.java
BookRecordDto.java
BookService.java
BookController.java
PublisherModel.java
23
24 public BookService(BookRepository bookRepository, Autho
25 this.bookRepository = bookRepository;
26 this.authorRepository = authorRepository;
27 this.publisherRepository = publisherRepository;
28 }
29
30 public List<BookModel> getAllBooks() {
31 return bookRepository.findAll();
32 }
33


```java
public class BookController {

    private final BookService bookService;

    public BookController(BookService bookService) {
        this.bookService = bookService;
    }

    @GetMapping
    public ResponseEntity<List<BookModel>> getAllBooks() {
        return ResponseEntity.status(HttpStatus.OK).body(bookService.getAllBooks());
    }
```

```
23
24      //@JsonProperty(access = JsonProperty.Access.WRITE_ONLY)
25      @ManyToOne //(fetch = FetchType.LAZY)
26      @JoinColumn(name = "publisher_id")
27      private PublisherModel publisher;
28
29      //@JsonProperty(access = JsonProperty.Access.WRITE_ONLY)
30      @ManyToMany //(fetch = FetchType.LAZY)
31      @JoinTable(
32              name = "tb_book_author",
33              joinColumns = @JoinColumn(name = "book_id"),
34              inverseJoinColumns = @JoinColumn(name = "author_id")
35      private Set<AuthorModel> authors = new HashSet<>();
```

Clean package - Lazy

Overview | POST http://localhost:8080/l | GET Get data | No environment

REST API basics: CRUD, test & variable / **Get data** | Save | Share

GET | http://localhost:8080/bookstore/books | Send

Params | Authorization | Headers (7) | Body | Scripts | Tests | Settings | Cookies

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

Body | Cookies | Headers (5) | Test Results (1/1) | 200 OK | 206 ms | 829 B | Save Response

Pretty | Raw | Preview | Visualize | JSON


```
[
    {
        "id": "c9bd8c35-f2a1-469b-b287-f0c06de289d2",
        "title": "Domain Driven Design",
        "review": {
            "id": "916461ee-7b7f-4b17-b093-5e144324b13c",
            "comment": "Reunindo práticas de design e implementação, este livro incorpora vários exemplos baseados em projetos que ilustram a aplicação de um design dirigidos por domínio no desenvolvimento de software na vida real."
        }
    },
    {
        "id": "0bd9cd25-7a6e-49c1-9849-29b46384d95d",
        "title": "Introduction C++ Development System.",
        "review": {
            "id": "f21b8483-2619-4f9f-81f0-5a52e0673ca7",
            "comment": "I began to study C a while back. I think it's really important to get the fundamentals and approaches right and King's book just isn't good IMO."
        }
    }
]
```


@Delete Service

```java
@Transactional
public void deleteBook(UUID id){
    bookRepository.deleteById(id);
}
```

@Delete controller

```
32
33      @DeleteMapping("/{id}")
34      public ResponseEntity<String> deleteBook(@PathVariable UUID id) {
35          bookService.deleteBook(id);
36          return ResponseEntity.status(HttpStatus.OK).body("Book deleted successfully.");
37      }
38  }
39
```

Before delete method

BookStoreJPA / **http://localhost:8080/bookstore/books**

Save | Share

GET | http://localhost:8080/bookstore/books | Send

Params | Authorization | Headers (7) | Body | Scripts | Tests | Settings | Cookies

none | form-data | x-www-form-urlencoded | raw | binary | GraphQL | JSON | Beautify

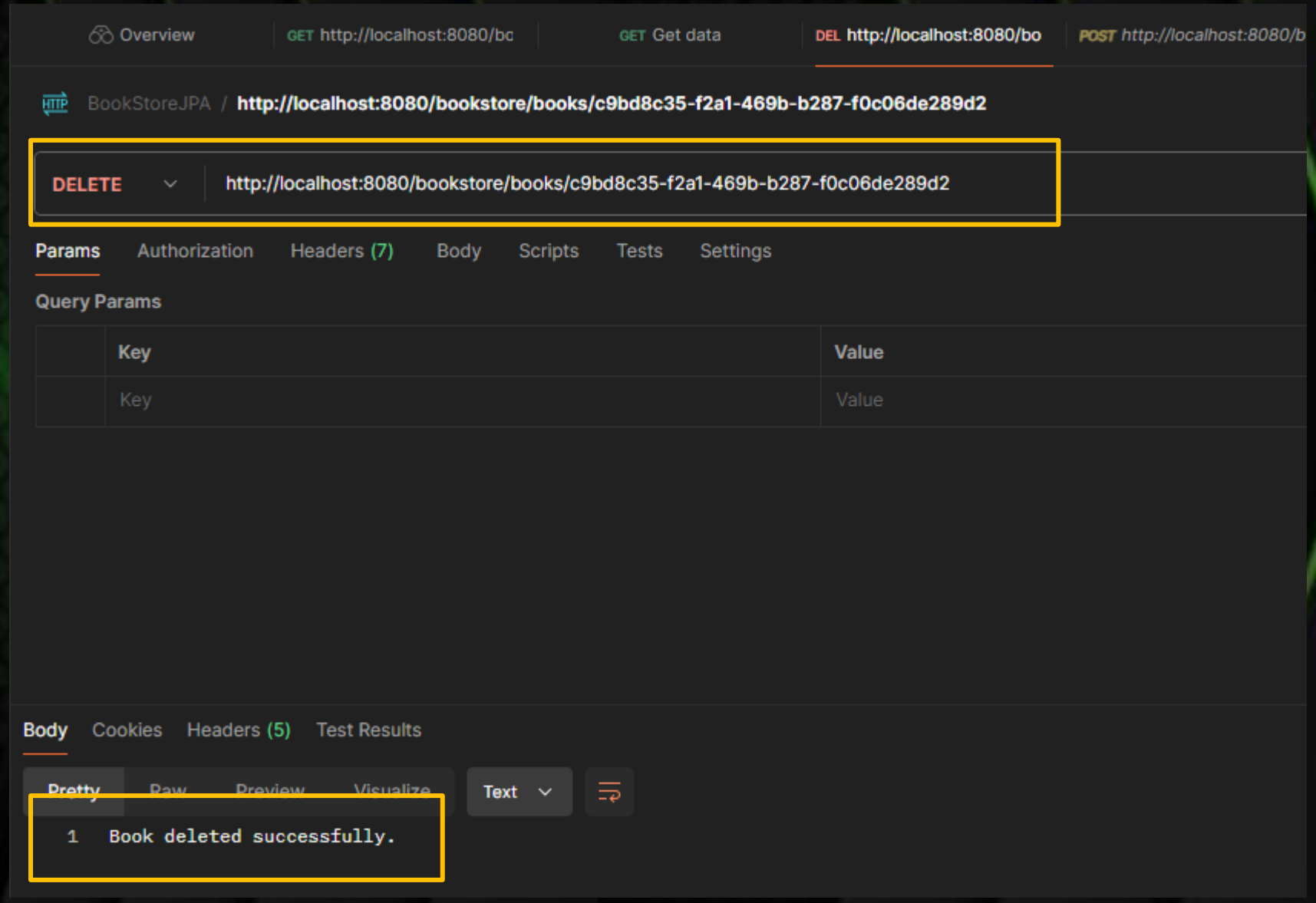
Body | Cookies | Headers (5) | Test Results | 200 OK | 11 ms | 829 B | Save Response

Pretty | Raw | Preview | Visualize | JSON


```
[
    {
        "id": "c9bd8c35-f2a1-469b-b287-f0c06de289d2",
        "title": "Domain Driven Design",
        "review": {
            "id": "916461ee-7b7f-4b17-b093-5e144324b13c",
            "comment": "Reunindo práticas de design e implementação, este livro incorpora vários exemplos baseados em projetos que ilustram a aplicação de um design dirigidos por domínio no desenvolvimento de software na vida real."
        }
    },
    {
        "id": "0bd9cd25-7a6e-49c1-9849-29b46384d95d",
        "title": "Introduction C++ Development System.",
        "review": {
            "id": "f21b8483-2619-4f9f-81f0-5a52e0673ca7",
            "comment": "I began to study C a while back. I think it's really important to get the fundamentals and approaches right and King's book just isn't good IMO."
        }
    }
]
```

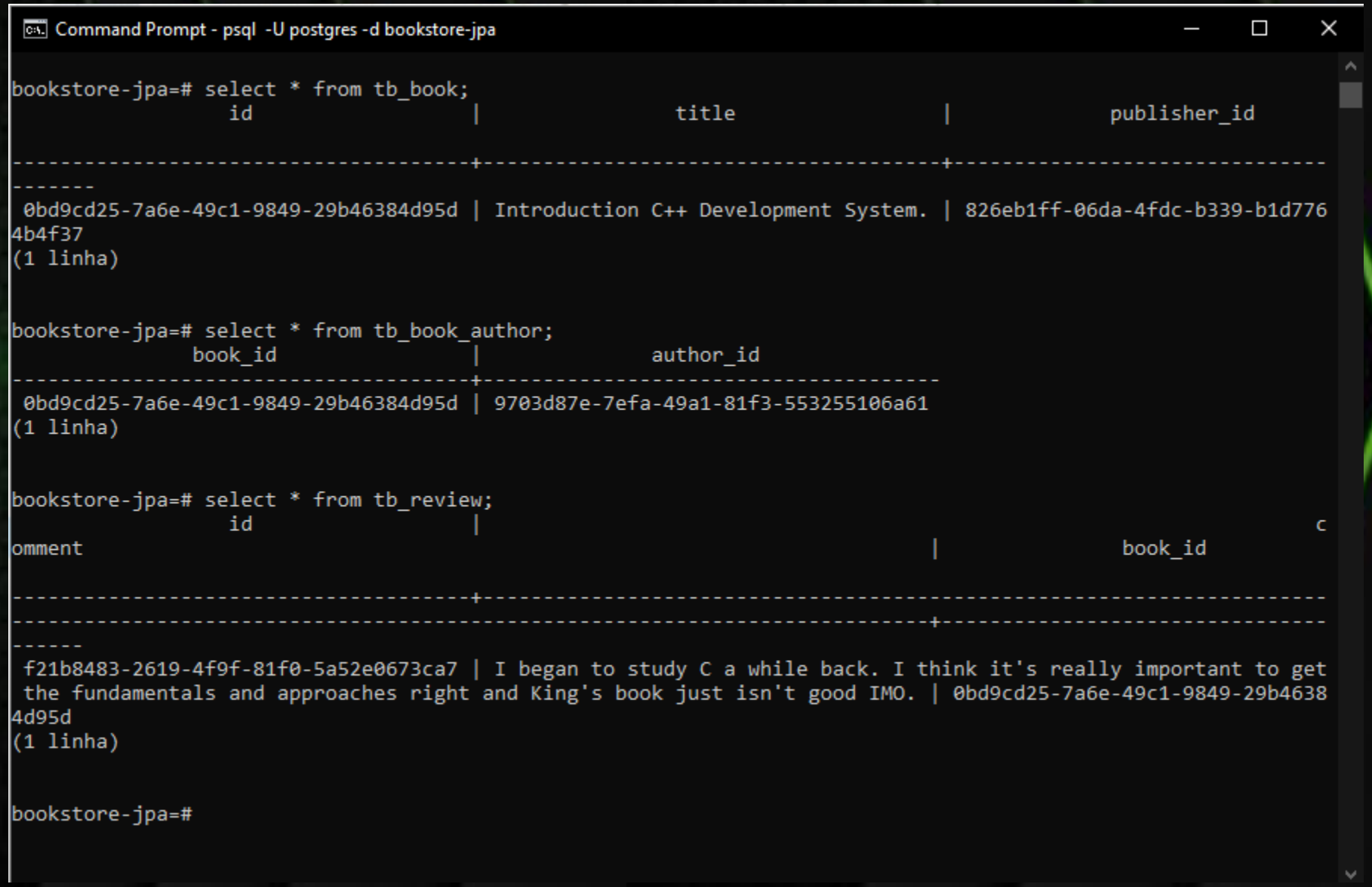

JPA
Java Persistence API
HIBERNATE

@Delete Verb


Overview
GET http://localhost:8080/bc
GET Get data
DEL http://localhost:8080/bo
POST http://localhost:8080/b
BookStoreJPA / http://localhost:8080/bookstore/books/c9bd8c35-f2a1-469b-b287-f0c06de289d2
DELETE
http://localhost:8080/bookstore/books/c9bd8c35-f2a1-469b-b287-f0c06de289d2
Params
Authorization
Headers (7)
Body
Scripts
Tests
Settings
Query Params
Key
Value
Key
Value
Body
Cookies
Headers (5)
Test Results
Pretty
Raw
Preview
Visualize
Text
1 Book deleted successfully.

Deleting dependency

```
bookstore-jpa=# select * from tb_publisher;
                  id                  |     name
--------------------------------------+--------------
 0e3b5f6f-91a4-42b9-8d50-09fa2d14ed2b | Bloomsbury
 f6bcd92d-8a6a-4b8a-bb62-b4b6ef930c91 | Bantam Books
 061af4db-a835-4664-9728-4d62b8e8a6c0 | Alta Books
 c1247e09-1214-447f-9825-2bc3f86bbafa | Person
 826eb1ff-06da-4fdc-b339-b1d7764b4f37 | Microsoft
(5 linhas)


bookstore-jpa=# select * from tb_author;
                  id                  |        name
--------------------------------------+--------------------
 8f9a676e-df7b-4c7b-aee1-8eeb7744a622 | J.K. Rowling
 a4d1b0c5-28b5-4426-a605-2a9f3c51d4f4 | George R.R. Martin
 5a939c3d-1616-4af2-a6be-10085d1f1794 | Eric Vans
 071cb9ad-cfca-429d-ba5e-2ec23c8490e1 | Paul Deitel
 c1094619-6962-4cca-aa7e-9b0b97c45aea | Harvey Deitel
 9703d87e-7efa-49a1-81f3-553255106a61 | Ivor Horton
(6 linhas)
```


Command Prompt - psql -U postgres -d bookstore-jpa
bookstore-jpa=# select * from tb_book;
id | title | publisher_id
0bd9cd25-7a6e-49c1-9849-29b46384d95d | Introduction C++ Development System. | 826eb1ff-06da-4fdc-b339-b1d7764b4f37
(1 linha)
bookstore-jpa=# select * from tb_book_author;
book_id | author_id
0bd9cd25-7a6e-49c1-9849-29b46384d95d | 9703d87e-7efa-49a1-81f3-553255106a61
(1 linha)
bookstore-jpa=# select * from tb_review;
id | comment | book_id
f21b8483-2619-4f9f-81f0-5a52e0673ca7 | I began to study C a while back. I think it's really important to get the fundamentals and approaches right and King's book just isn't good IMO. | 0bd9cd25-7a6e-49c1-9849-29b46384d95d
(1 linha)
bookstore-jpa=#

After @Delete