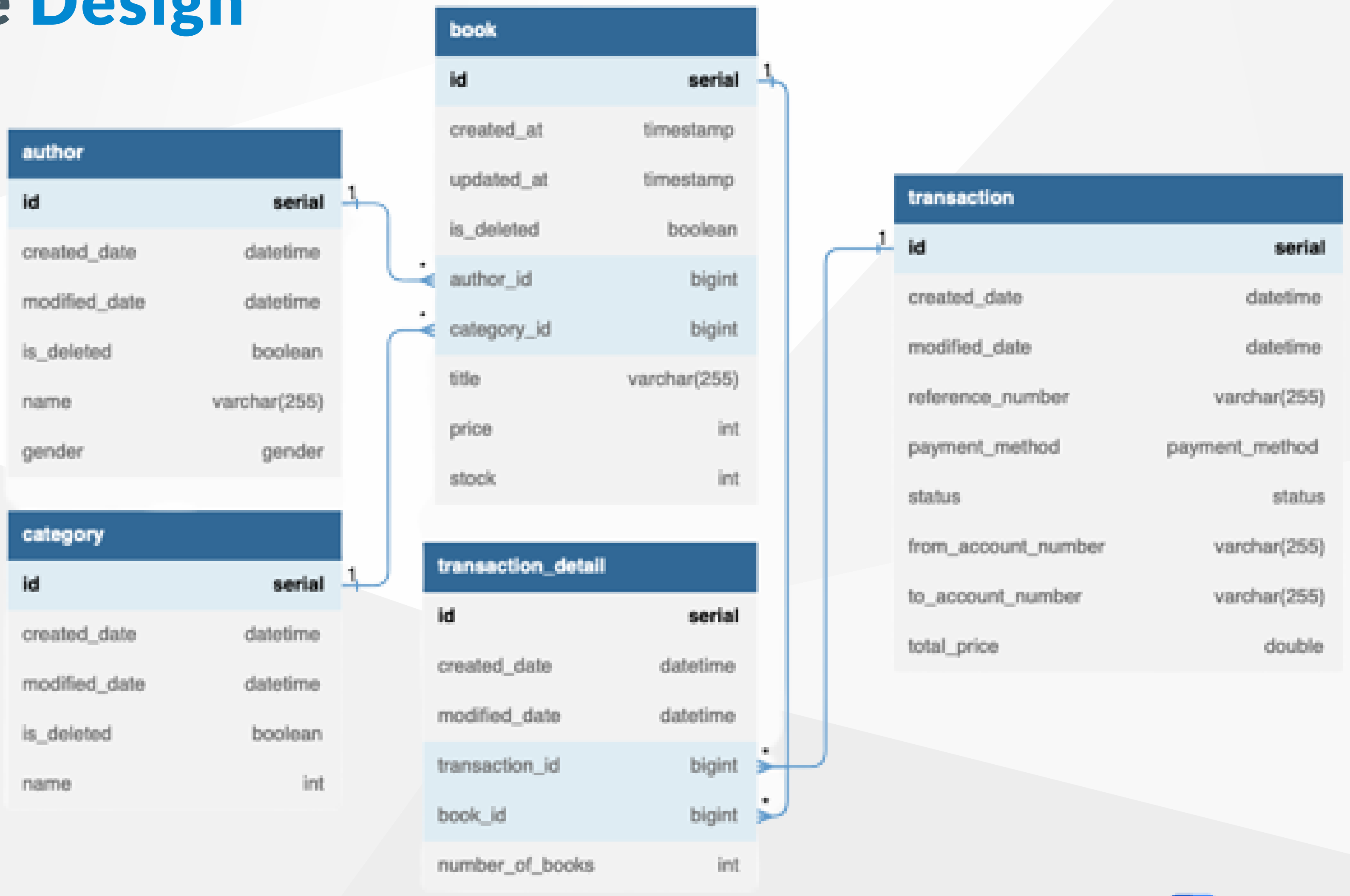
Spring Boot JPA Relationship

Rawlabs Academy

Database **Design**



author	
id	**serial**
created_date	datetime
modified_date	datetime
is_deleted	boolean
name	varchar(255)
gender	gender

category	
id	**serial**
created_date	datetime
modified_date	datetime
is_deleted	boolean
name	int

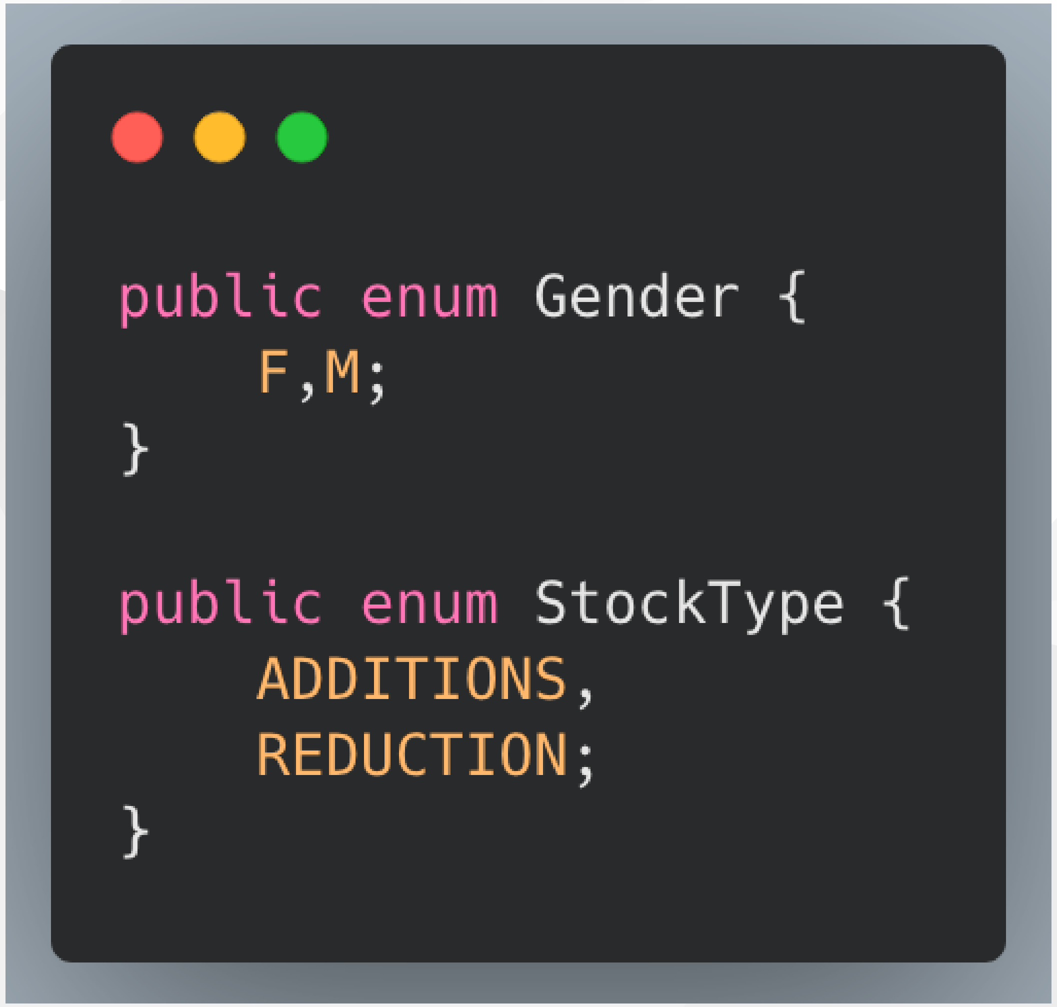
book	
id	**serial**
created_at	timestamp
updated_at	timestamp
is_deleted	boolean
author_id	bigint
category_id	bigint
title	varchar(255)
price	int
stock	int

transaction_detail	
id	**serial**
created_date	datetime
modified_date	datetime
transaction_id	bigint
book_id	bigint
number_of_books	int

transaction	
id	**serial**
created_date	datetime
modified_date	datetime
reference_number	varchar(255)
payment_method	payment_method
status	status
from_account_number	varchar(255)
to_account_number	varchar(255)
total_price	double

dbdiagram.io

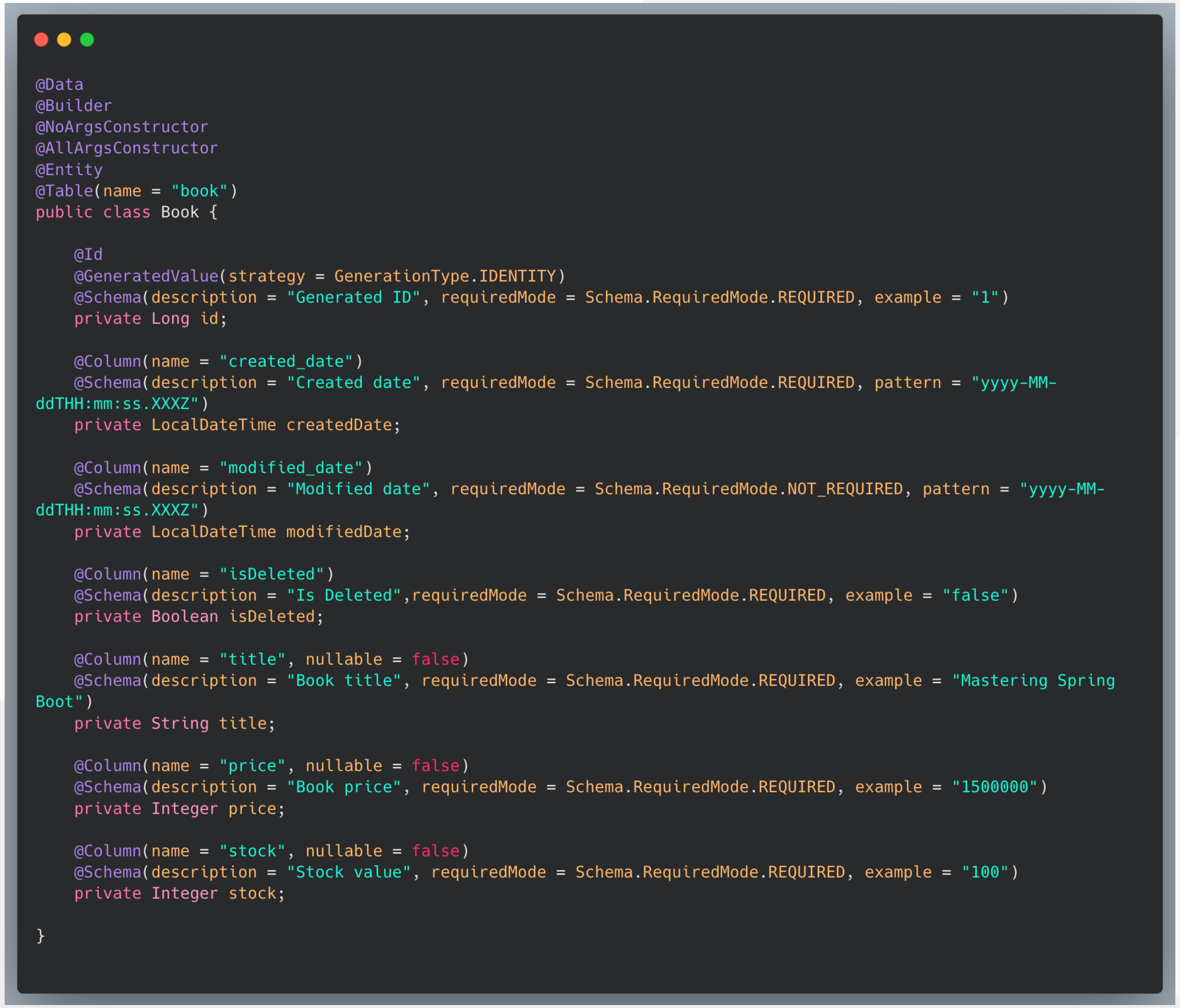
Constant Enum



```
public enum Gender {
    F,M;
}

public enum StockType {
    ADDITIONS,
    REDUCTION;
}
```

Book DAO

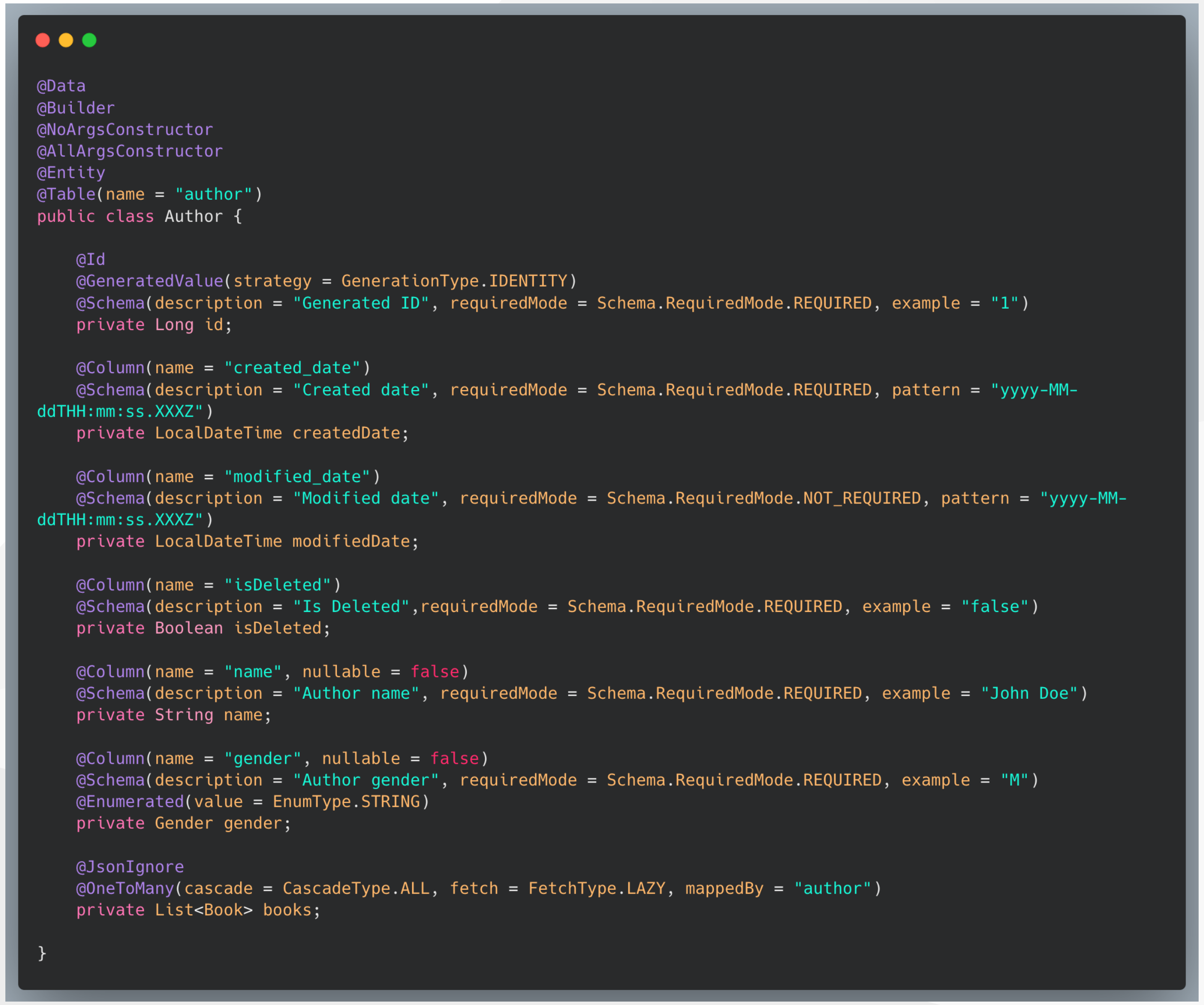
In the previous material, we have created a `Book` model and then adjust it as in the example



```java
@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
@Entity
@Table(name = "book")
public class Book {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Schema(description = "Generated ID", requiredMode = Schema.RequiredMode.REQUIRED, example = "1")
    private Long id;

    @Column(name = "created_date")
    @Schema(description = "Created date", requiredMode = Schema.RequiredMode.REQUIRED, pattern = "yyyy-MM-
ddTHH:mm:ss.XXXZ")
    private LocalDateTime createdDate;

    @Column(name = "modified_date")
    @Schema(description = "Modified date", requiredMode = Schema.RequiredMode.NOT_REQUIRED, pattern = "yyyy-MM-
ddTHH:mm:ss.XXXZ")
    private LocalDateTime modifiedDate;

    @Column(name = "isDeleted")
    @Schema(description = "Is Deleted",requiredMode = Schema.RequiredMode.REQUIRED, example = "false")
    private Boolean isDeleted;

    @Column(name = "title", nullable = false)
    @Schema(description = "Book title", requiredMode = Schema.RequiredMode.REQUIRED, example = "Mastering Spring
Boot")
    private String title;

    @Column(name = "price", nullable = false)
    @Schema(description = "Book price", requiredMode = Schema.RequiredMode.REQUIRED, example = "1500000")
    private Integer price;

    @Column(name = "stock", nullable = false)
    @Schema(description = "Stock value", requiredMode = Schema.RequiredMode.REQUIRED, example = "100")
    private Integer stock;

}
```

Author DAO

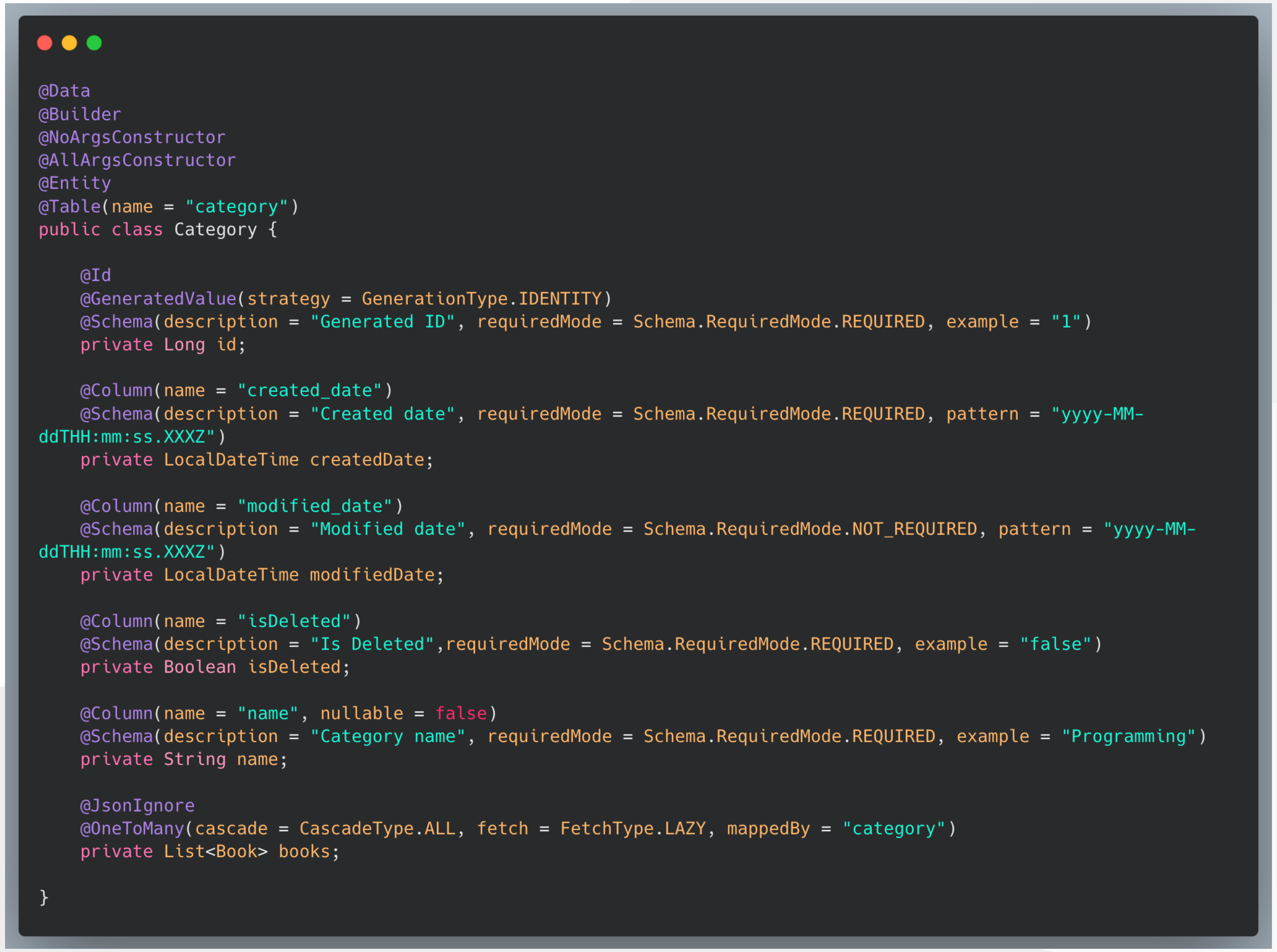
- `@JsonIgnore` used for ignore field from json response
- `@OneToMany` based on DB design that mean the **Author** can have **many books**.
- `cascade` When we perform some action on the target entity, the same action will be applied to the associated entity
- `fetchType` how to fetch the data, `LAZY` is fetch **when needed** and `EAGER` fetch **immediatelly**



```
@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
@Entity
@Table(name = "author")
public class Author {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Schema(description = "Generated ID", requiredMode = Schema.RequiredMode.REQUIRED, example = "1")
    private Long id;

    @Column(name = "created_date")
    @Schema(description = "Created date", requiredMode = Schema.RequiredMode.REQUIRED, pattern = "yyyy-MM-
ddTHH:mm:ss.XXXZ")
    private LocalDateTime createdDate;

    @Column(name = "modified_date")
    @Schema(description = "Modified date", requiredMode = Schema.RequiredMode.NOT_REQUIRED, pattern = "yyyy-MM-
ddTHH:mm:ss.XXXZ")
    private LocalDateTime modifiedDate;

    @Column(name = "isDeleted")
    @Schema(description = "Is Deleted",requiredMode = Schema.RequiredMode.REQUIRED, example = "false")
    private Boolean isDeleted;

    @Column(name = "name", nullable = false)
    @Schema(description = "Author name", requiredMode = Schema.RequiredMode.REQUIRED, example = "John Doe")
    private String name;

    @Column(name = "gender", nullable = false)
    @Schema(description = "Author gender", requiredMode = Schema.RequiredMode.REQUIRED, example = "M")
    @Enumerated(value = EnumType.STRING)
    private Gender gender;

    @JsonIgnore
    @OneToMany(cascade = CascadeType.ALL, fetch = FetchType.LAZY, mappedBy = "author")
    private List<Book> books;

}
```

Category DAO

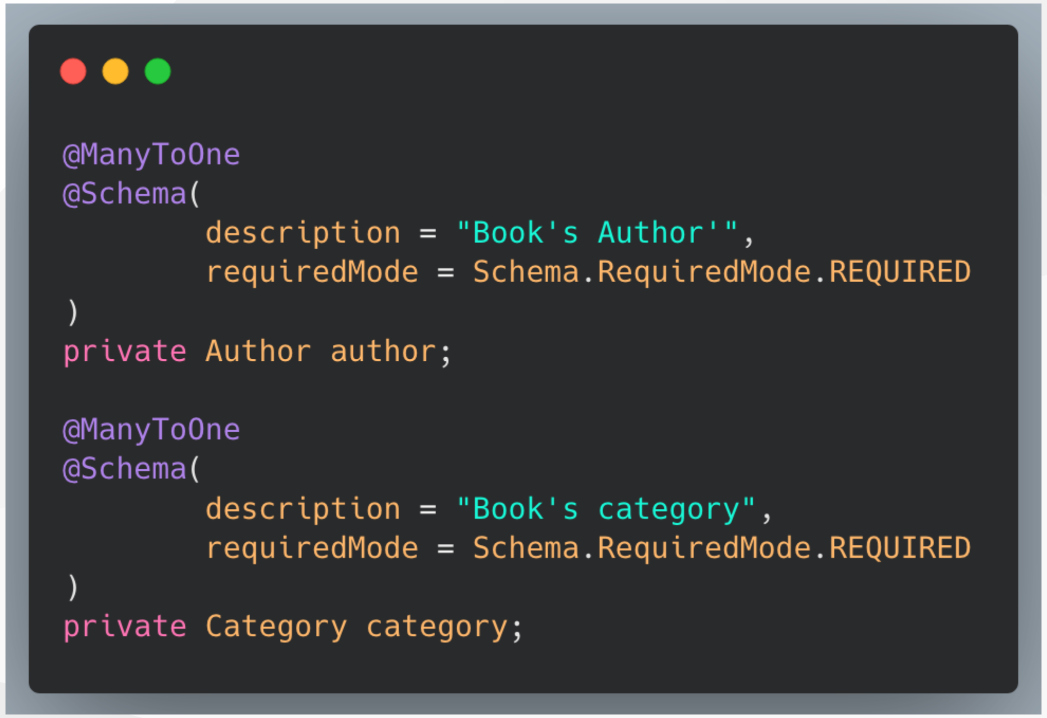
- @OneToMany based on DB design that mean the **Category** can have **many books**.
- fetchType how to fetch the data, LAZY is fetch **when needed** and EAGER fetch **immediatelly**
- mappedBy to be used for mapping attribute on the Book **DAO**



```
@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
@Entity
@Table(name = "category")
public class Category {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Schema(description = "Generated ID", requiredMode = Schema.RequiredMode.REQUIRED, example = "1")
    private Long id;

    @Column(name = "created_date")
    @Schema(description = "Created date", requiredMode = Schema.RequiredMode.REQUIRED, pattern = "yyyy-MM-
ddTHH:mm:ss.XXXZ")
    private LocalDateTime createdDate;

    @Column(name = "modified_date")
    @Schema(description = "Modified date", requiredMode = Schema.RequiredMode.NOT_REQUIRED, pattern = "yyyy-MM-
ddTHH:mm:ss.XXXZ")
    private LocalDateTime modifiedDate;

    @Column(name = "isDeleted")
    @Schema(description = "Is Deleted",requiredMode = Schema.RequiredMode.REQUIRED, example = "false")
    private Boolean isDeleted;

    @Column(name = "name", nullable = false)
    @Schema(description = "Category name", requiredMode = Schema.RequiredMode.REQUIRED, example = "Programming")
    private String name;

    @JsonIgnore
    @OneToMany(cascade = CascadeType.ALL, fetch = FetchType.LAZY, mappedBy = "category")
    private List<Book> books;

}
```

Update the Book DAO

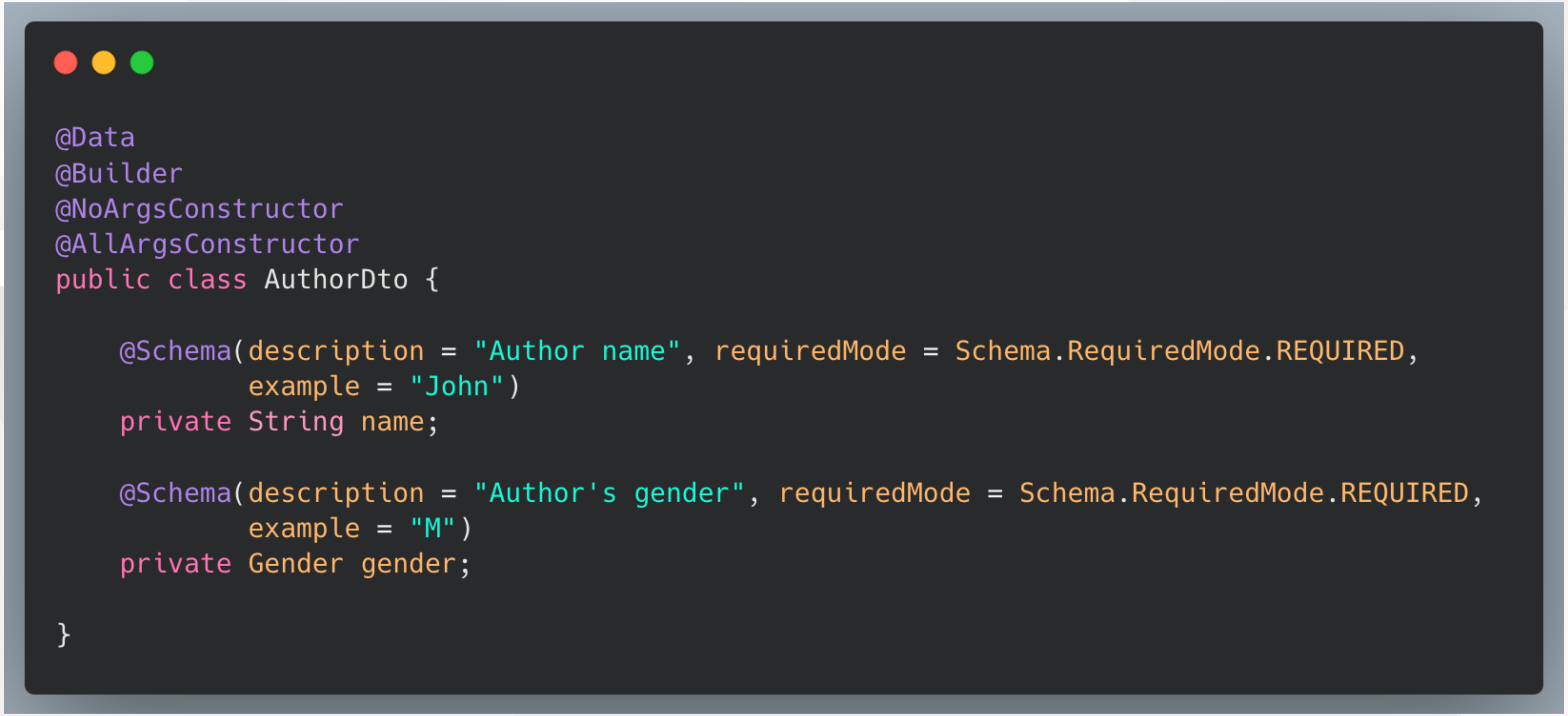
- `@ManyToOne` based on DB design that mean `Book` DAO have foreign key `author` and `category` then be mapped on each associated entity
- Foreign key will be generated automatically from `author` into `author_id` and `category` into `category_id` on database.



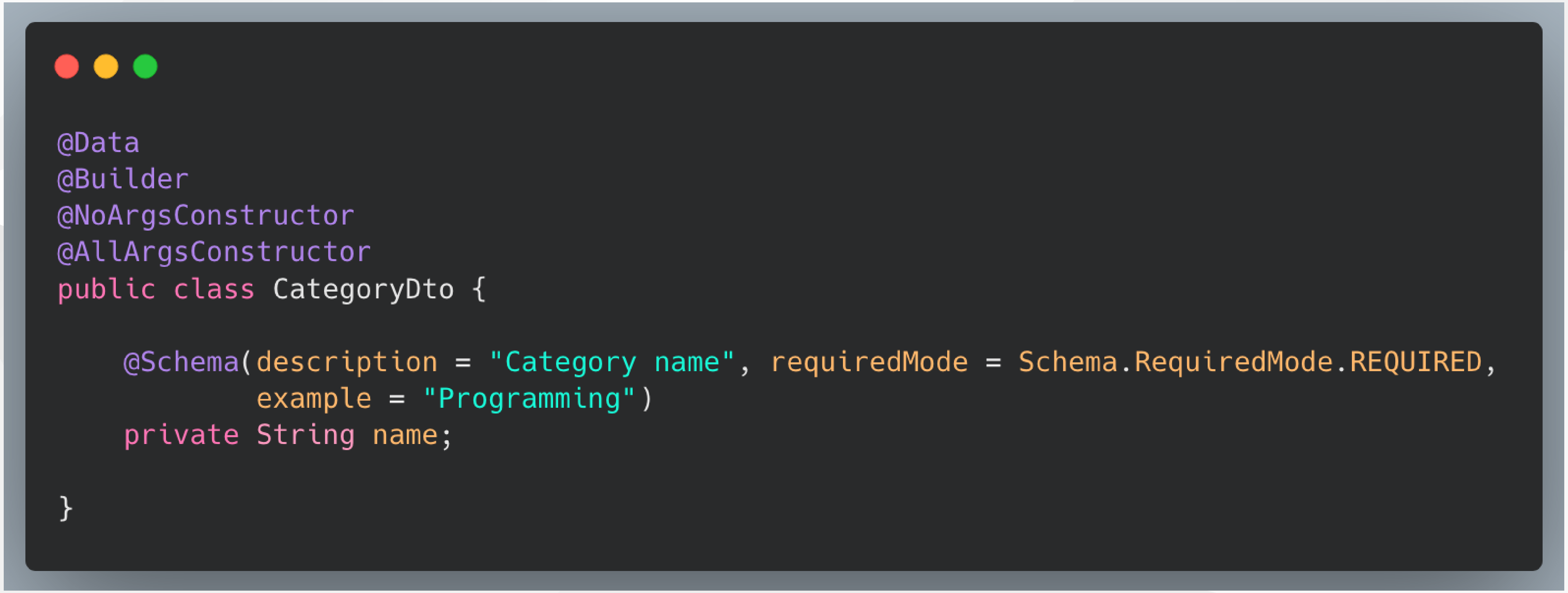
```java
@ManyToOne
@Schema(
        description = "Book's Author'",
        requiredMode = Schema.RequiredMode.REQUIRED
)
private Author author;

@ManyToOne
@Schema(
        description = "Book's category",
        requiredMode = Schema.RequiredMode.REQUIRED
)
private Category category;
```

Data Transfer Object

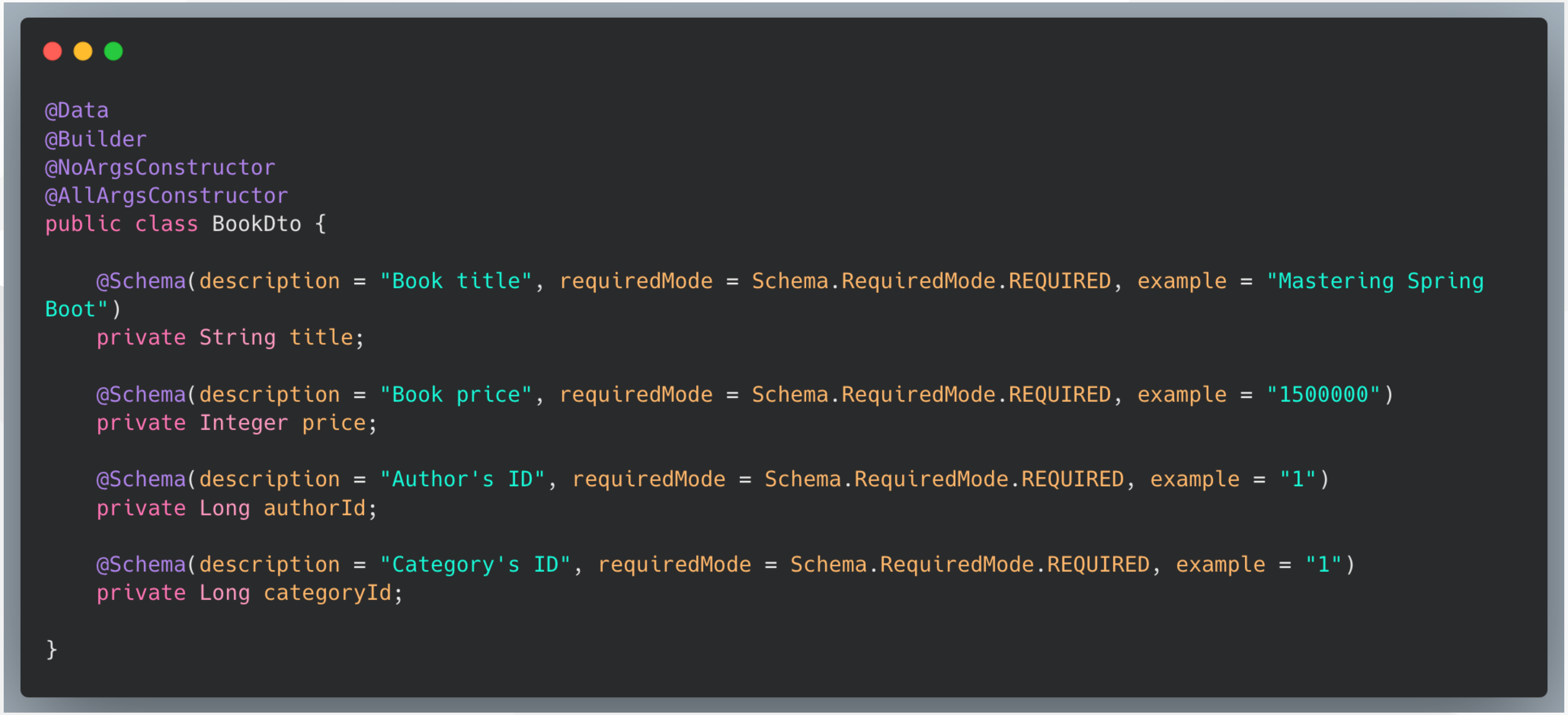
(DTO)



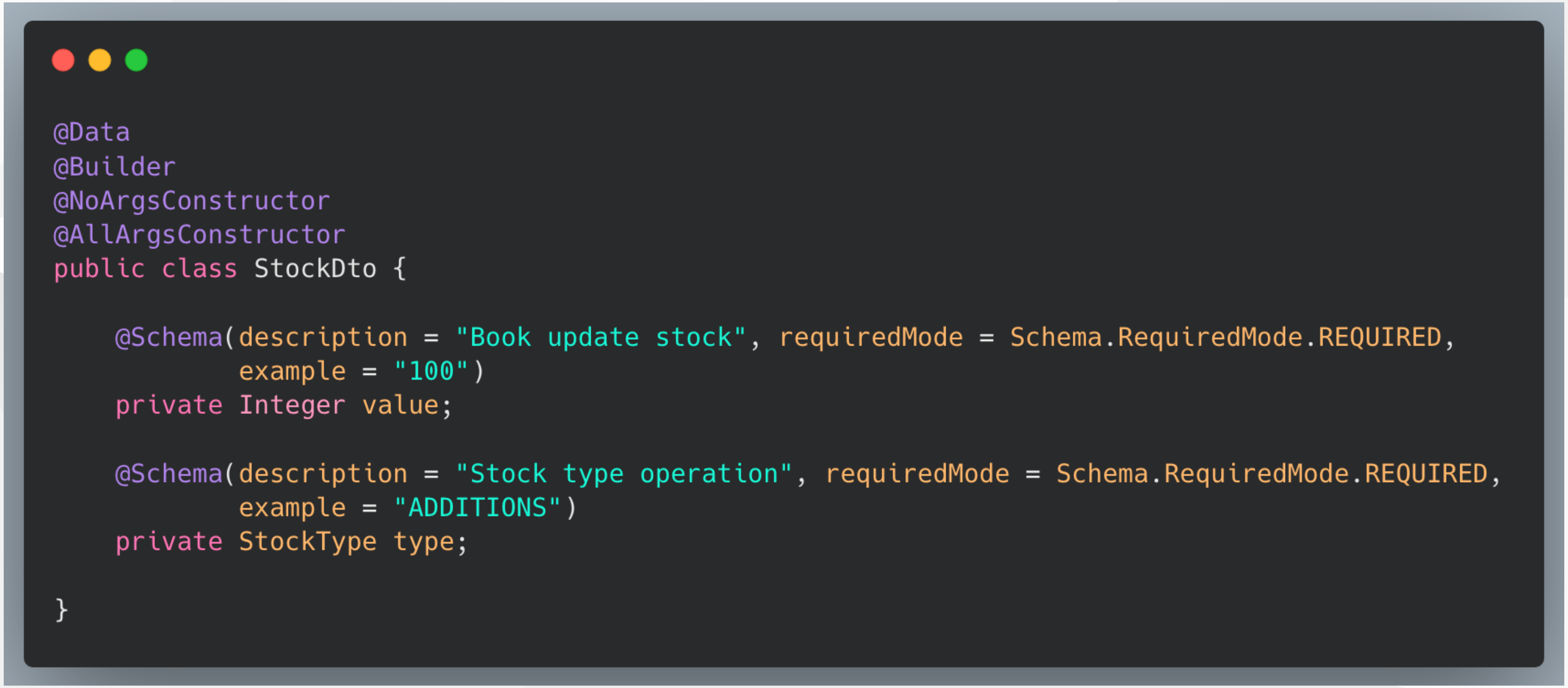
```java
@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class AuthorDto {

    @Schema(description = "Author name", requiredMode = Schema.RequiredMode.REQUIRED,
            example = "John")
    private String name;

    @Schema(description = "Author's gender", requiredMode = Schema.RequiredMode.REQUIRED,
            example = "M")
    private Gender gender;

}
```



```java
@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class CategoryDto {

    @Schema(description = "Category name", requiredMode = Schema.RequiredMode.REQUIRED,
            example = "Programming")
    private String name;

}
```

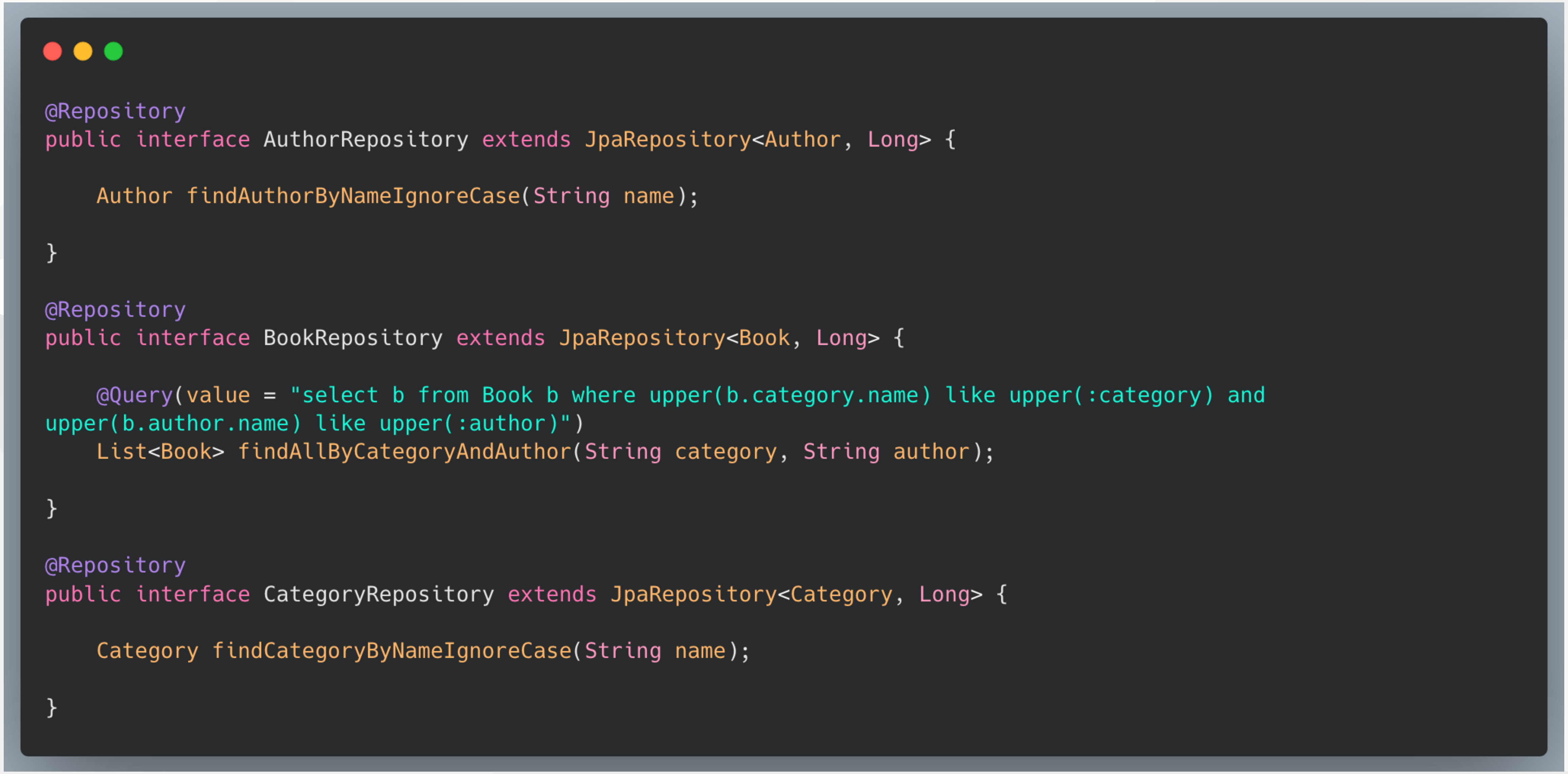


```java
@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class BookDto {

    @Schema(description = "Book title", requiredMode = Schema.RequiredMode.REQUIRED, example = "Mastering Spring
Boot")
    private String title;

    @Schema(description = "Book price", requiredMode = Schema.RequiredMode.REQUIRED, example = "1500000")
    private Integer price;

    @Schema(description = "Author's ID", requiredMode = Schema.RequiredMode.REQUIRED, example = "1")
    private Long authorId;

    @Schema(description = "Category's ID", requiredMode = Schema.RequiredMode.REQUIRED, example = "1")
    private Long categoryId;

}
```


```java
@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class StockDto {

    @Schema(description = "Book update stock", requiredMode = Schema.RequiredMode.REQUIRED,
            example = "100")
    private Integer value;

    @Schema(description = "Stock type operation", requiredMode = Schema.RequiredMode.REQUIRED,
            example = "ADDITIONS")
    private StockType type;

}
```

Repository

(JPA Repository)



```java
@Repository
public interface AuthorRepository extends JpaRepository<Author, Long> {

    Author findAuthorByNameIgnoreCase(String name);

}

@Repository
public interface BookRepository extends JpaRepository<Book, Long> {

    @Query(value = "select b from Book b where upper(b.category.name) like upper(:category) and
upper(b.author.name) like upper(:author)")
    List<Book> findAllByCategoryAndAuthor(String category, String author);

}

@Repository
public interface CategoryRepository extends JpaRepository<Category, Long> {

    Category findCategoryByNameIgnoreCase(String name);

}
```

`BookRepository` Explanation