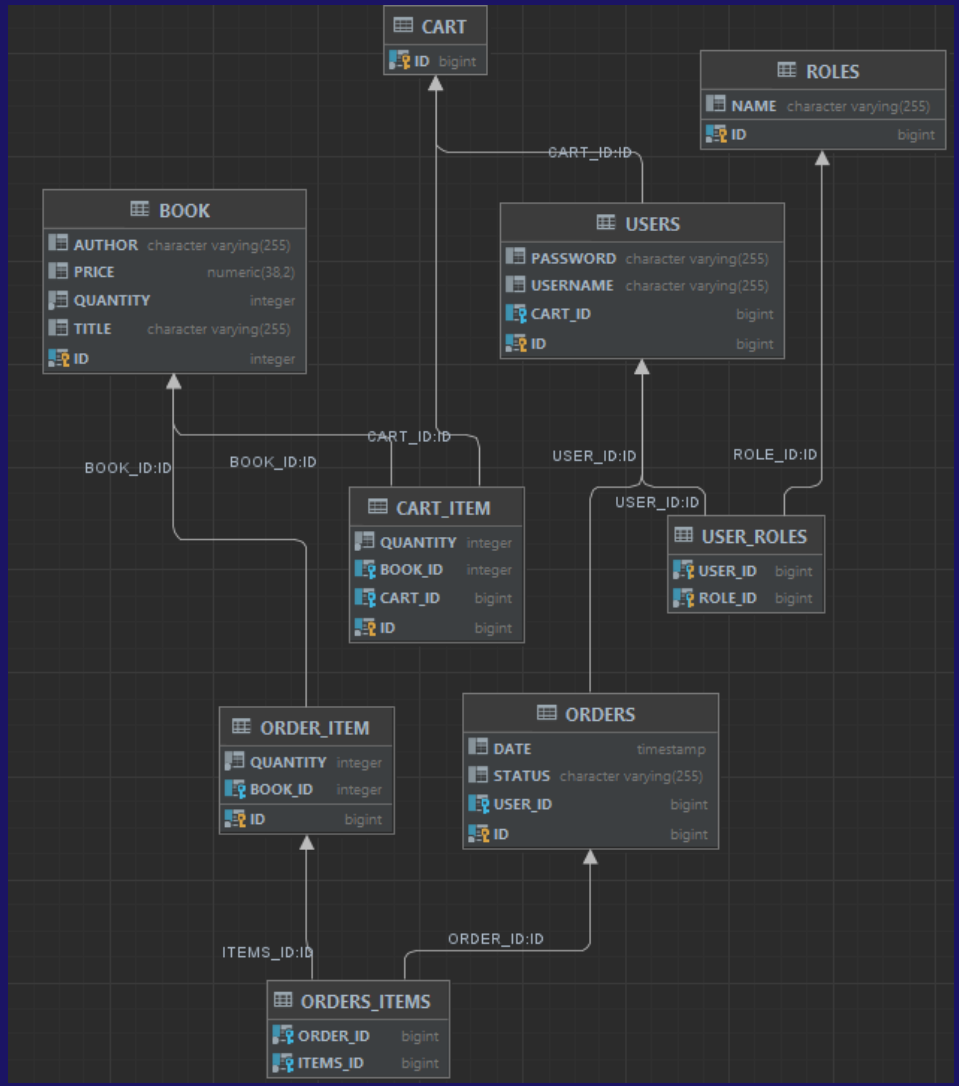
Bookstore

5

Zamówienia

Zamówienia powinny umożliwić użytkownikowi złożenie zamówienia produktów z koszyka (a później dokonania opłaty za produkty) .

W przeciwieństwie do koszyka, każdy User może posiadać wiele zamówień.



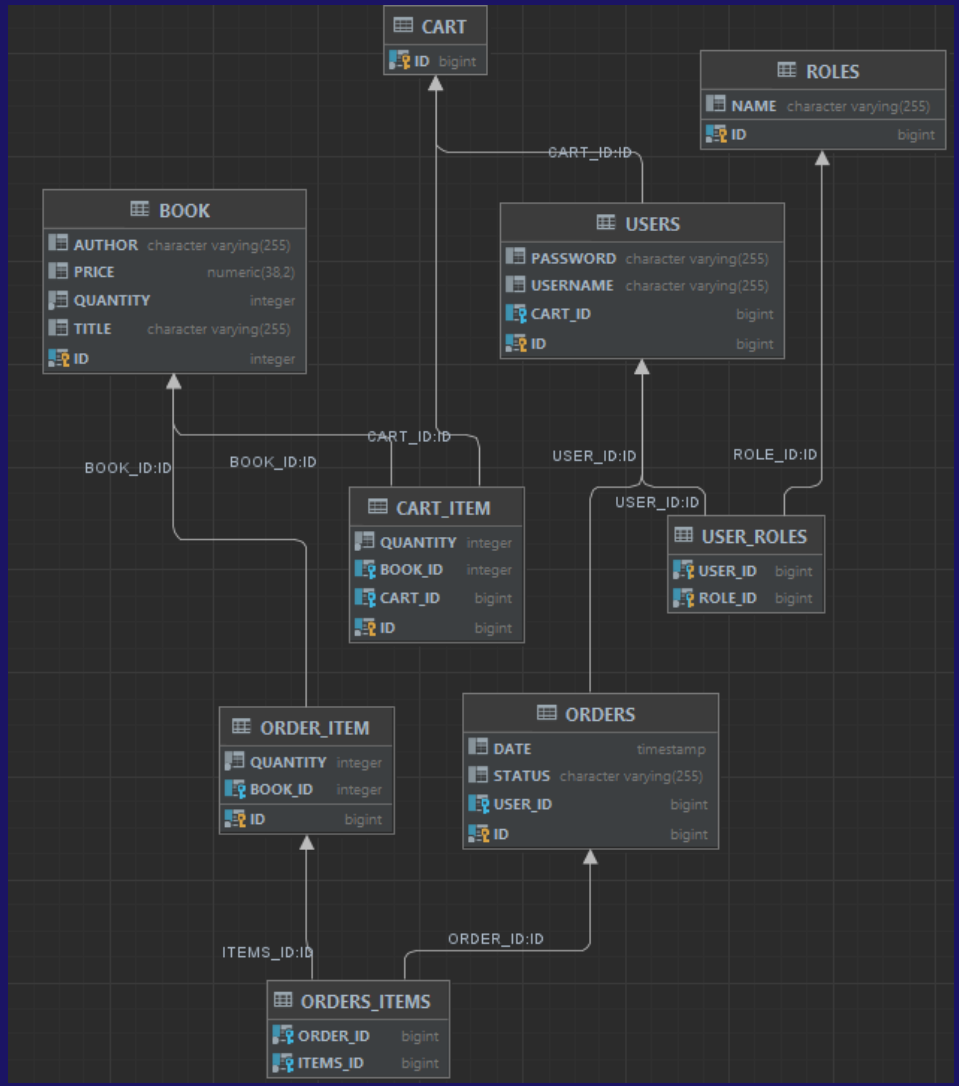
Zamówienia

Zamówienie – Order będzie składało się z daty, statusu, id usera oraz własnego id.

Relacja z userem:

```
@ManyToOne
@JoinColumn(name = "user_id")
private User user;
```

```
@OneToMany(mappedBy = "user", cascade =
CascadeType.ALL, orphanRemoval = true)
private List<Order> orders = new ArrayList<>();
```

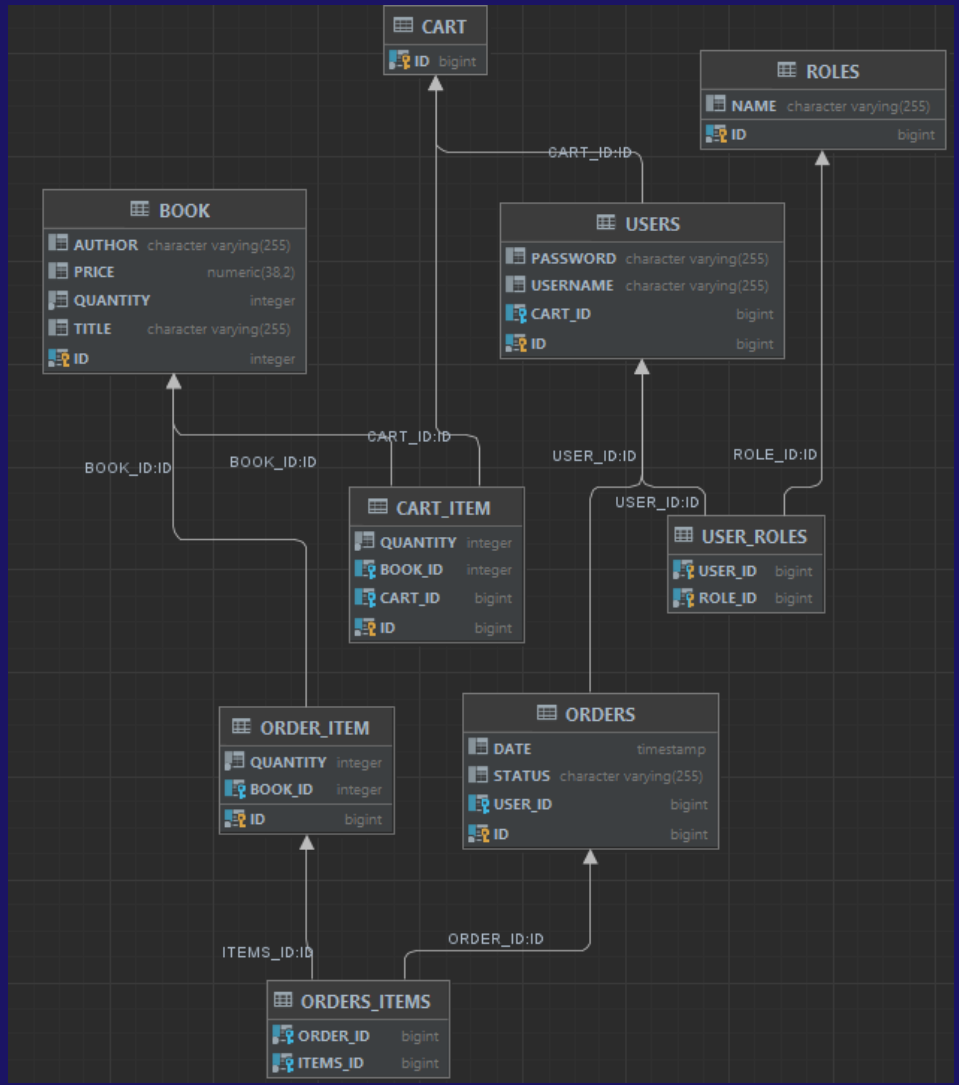


Zamówienia

Podobnie jak koszyk, zamówienie będzie posiadało pozycje (order_item), jedno zamówienie może mieć wiele pozycji (order_item).

Order_item reprezentuje książkę i jej ilość w zamówieniu.
Jest to relacja:

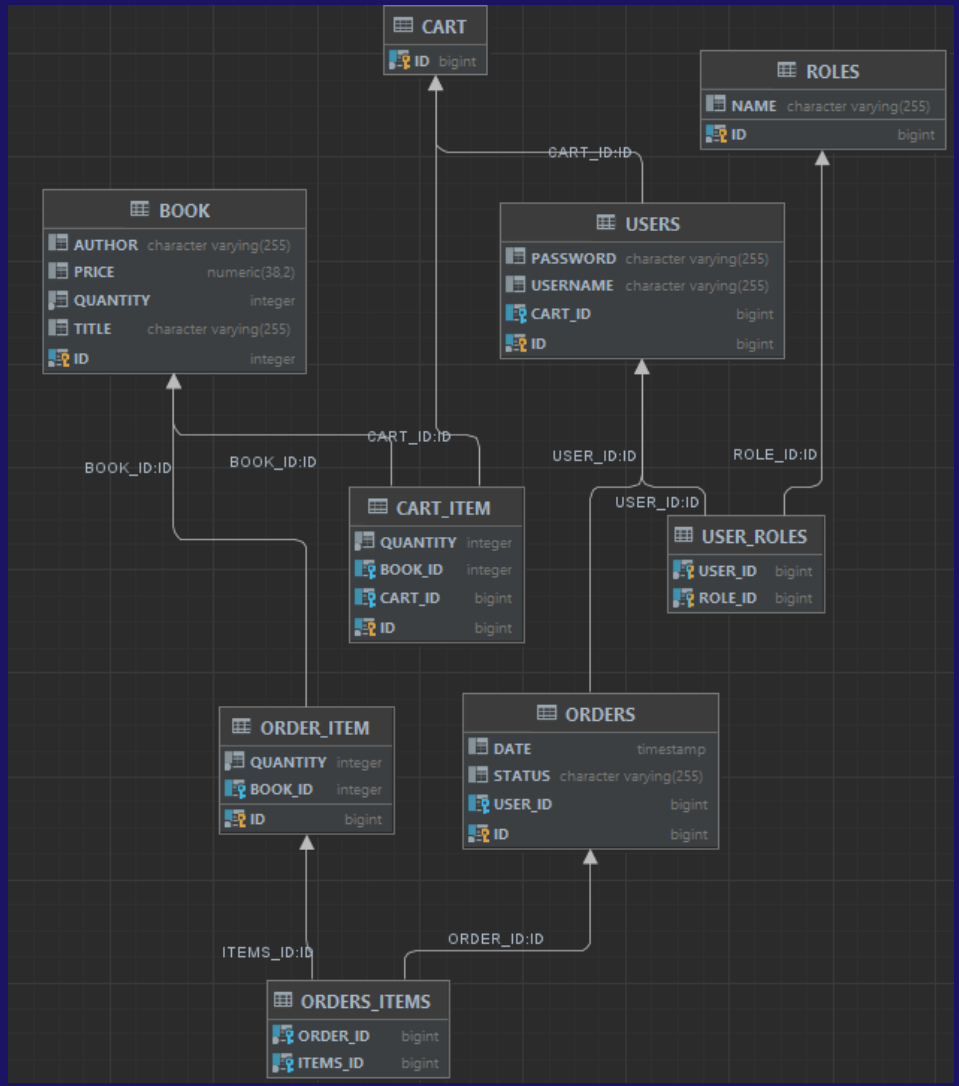
```
@OneToMany(cascade = CascadeType.ALL,
orphanRemoval = true)
private List<OrderItem> items = new
ArrayList<>();
```



Zamówienia

Statusy zamówienia mogą wyglądać następująco:

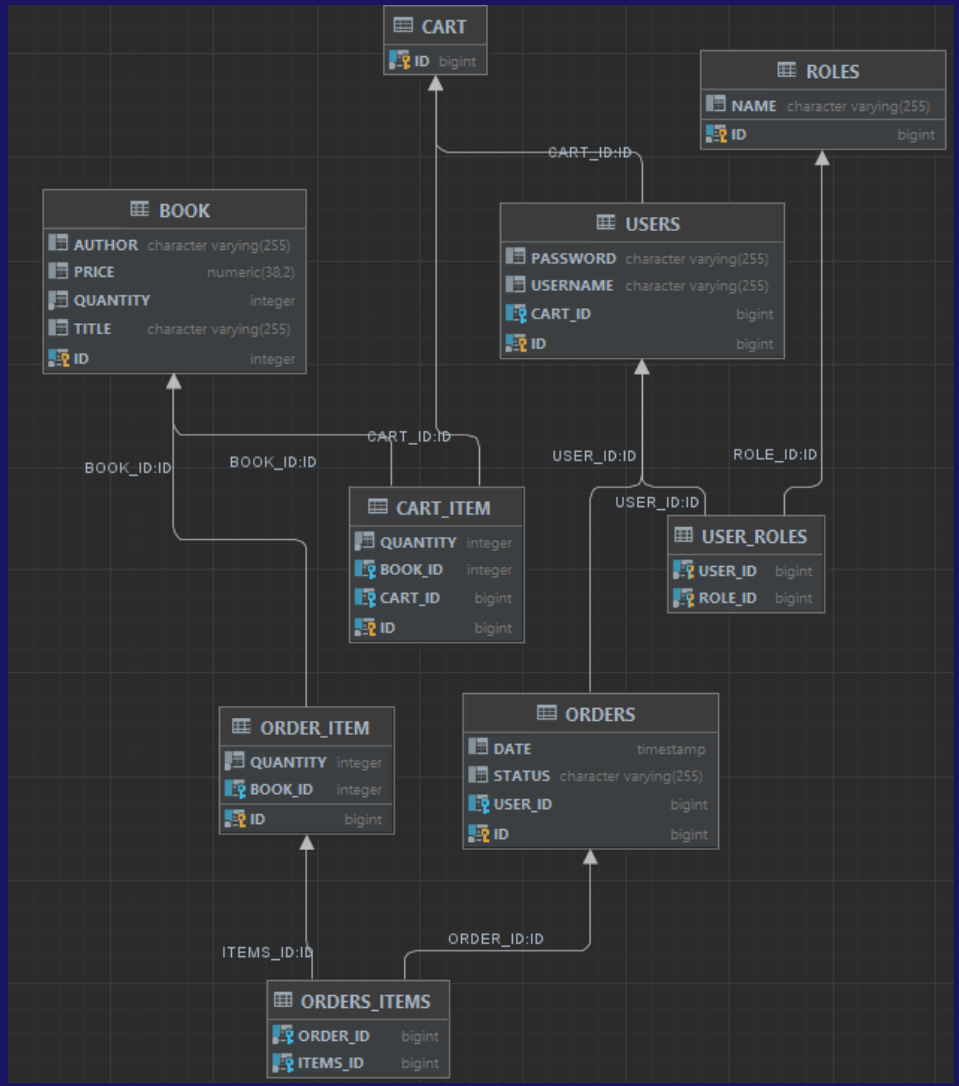
```
public enum OrderStatus {
    SUBMITTED,
    PAID,
    SHIPPED,
    COMPLETED
}
```



Zamówienia

Podstawowe repozytorium:

```
public interface OrderRepository extends
JpaRepository<Order, Long> {
}
```



Zamówienia

Metody w serwisie:

```
@Transactional
public Order submitOrder() {
    User user = userService.getCurrentUser();
    Cart cart = user.getCart();

    Order order = new Order();
    order.setDate(new Date());
    order.setStatus(OrderStatus.SUBMITTED);
    order.setUser(user);

    for (CartItem cartItem : cart.getItems()) {
        OrderItem orderItem = new OrderItem();
        orderItem.setBook(cartItem.getBook());

orderItem.setQuantity(cartItem.getQuantity());
        order.getItems().add(orderItem);
    }

    cart.getItems().clear();
    cartService.saveCart(cart);

    return orderRepository.save(order);
}
```

```
@Transactional
public Order getOrder(Long orderId) {
    return orderRepository.findById(orderId).orElseThrow(()
 -> new RuntimeException("Order not found"));
}

@Transactional
public void saveOrder(Order order) {
    orderRepository.save(order);
}
```

Zamówienia

Kontroler:

```
@Controller
@RequestMapping("/order")
public class OrderController {
    @Autowired
    private OrderService orderService;

    @PostMapping("/submit")
    public String submitOrder() {
        Order order = orderService.submitOrder();
        return "redirect:/order/" +
order.getId();
    }

    @GetMapping("/{orderId}")
    public String getOrder(@PathVariable Long
orderId, Model model) {
        Order order =
orderService.getOrder(orderId);
        model.addAttribute("order", order);
        return "order";
    }
}
```

Zamówienia

Dane na stronie internetowej zamówienia:

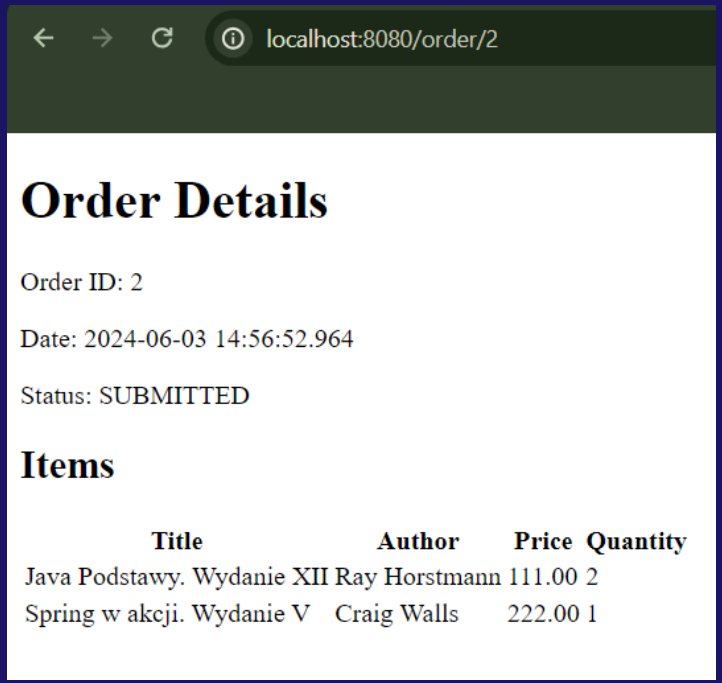
```
<h1>Order Details</h1>
<p>Order ID: <span
th:text="${order.id}"></span></p>
<p>Date: <span
th:text="${order.date}"></span></p>
<p>Status: <span
th:text="${order.status}"></span></p>
<h2>Items</h2>
<table>
    <thead>
    <tr>
        <th>Title</th>
        <th>Author</th>
        <th>Price</th>
        <th>Quantity</th>
    </tr>
    </thead>
    <tbody>
    <tr th:each="item : ${order.items}">
        <td th:text="${item.book.title}"></td>
        <td th:text="${item.book.author}"></td>
        <td th:text="${item.book.price}"></td>
        <td th:text="${item.quantity}"></td>
    </tr>
    </tbody>
</table>
```

Formularz w koszyku:

```
<form th:action="@{/order/submit}" method="post">
    <button type="submit">Submit Order</button>
</form>
```

Zamówienia


localhost:8080/cart
Koszyk
Tytuł Autor Cena Ilość Usuń
Java Podstawy. Wydanie XII Ray Horstmann 111.00 2 usuń
Spring w akcji. Wydanie V Craig Walls 222.00 1 usuń
Zamow


localhost:8080/order/2
Order Details
Order ID: 2
Date: 2024-06-03 14:56:52.964
Status: SUBMITTED
Items
Title Author Price Quantity
Java Podstawy. Wydanie XII Ray Horstmann 111.00 2
Spring w akcji. Wydanie V Craig Walls 222.00 1

Zamówienia

TODO:
-klasy Order(tabela nie może się tak nazywać), OrderItem, Enum dla statusu,OrderRepository, OrderService,

-Strona zamówienia

-zmiany: klasa User, strona koszyka,

- wyrzucanie błędu na stronie, gdy podczas składania zamówienia jest większa ilość książek niż dostępna w sklepie.

Następnie:
-dodanie funkcjonalności dla admina:
Wyświetlenie wszystkich zamówień, edycja zamówień (zmiana statusu)

-dodanie funkcjonalności dla usera: strona z historią wszystkich jego zamówień.