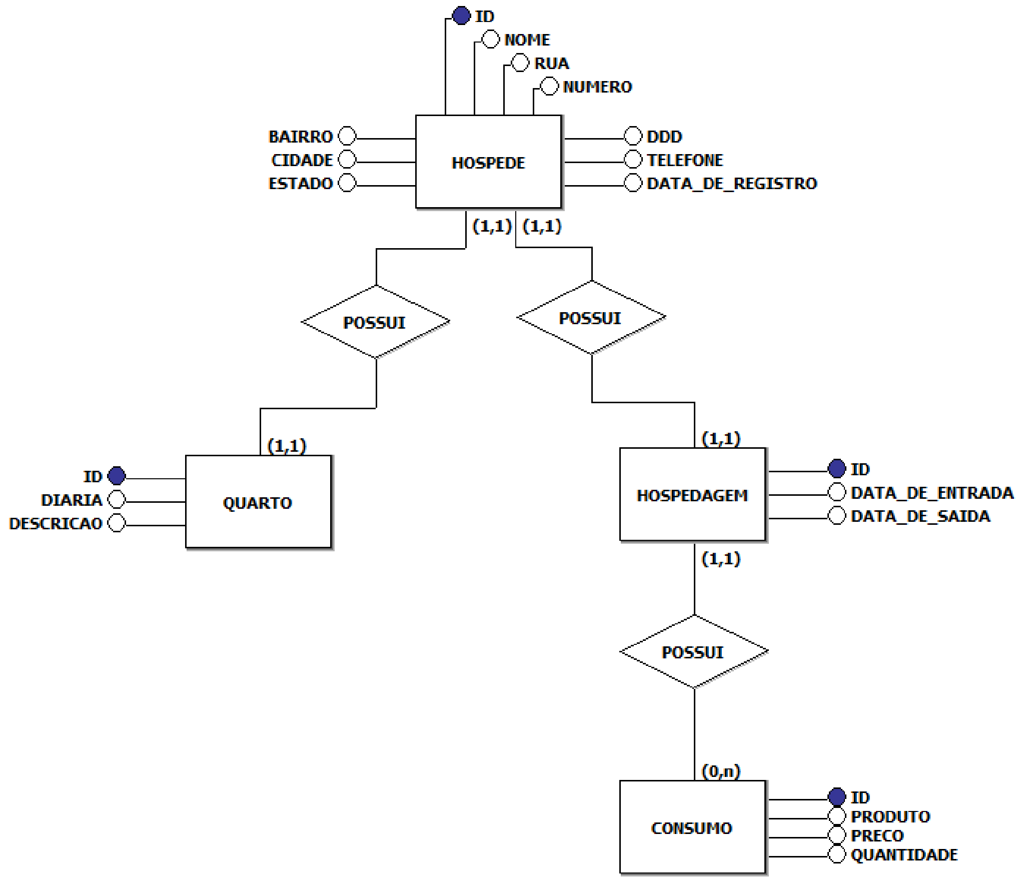
Gerenciamento de Hotéis

Leonardo Carmona

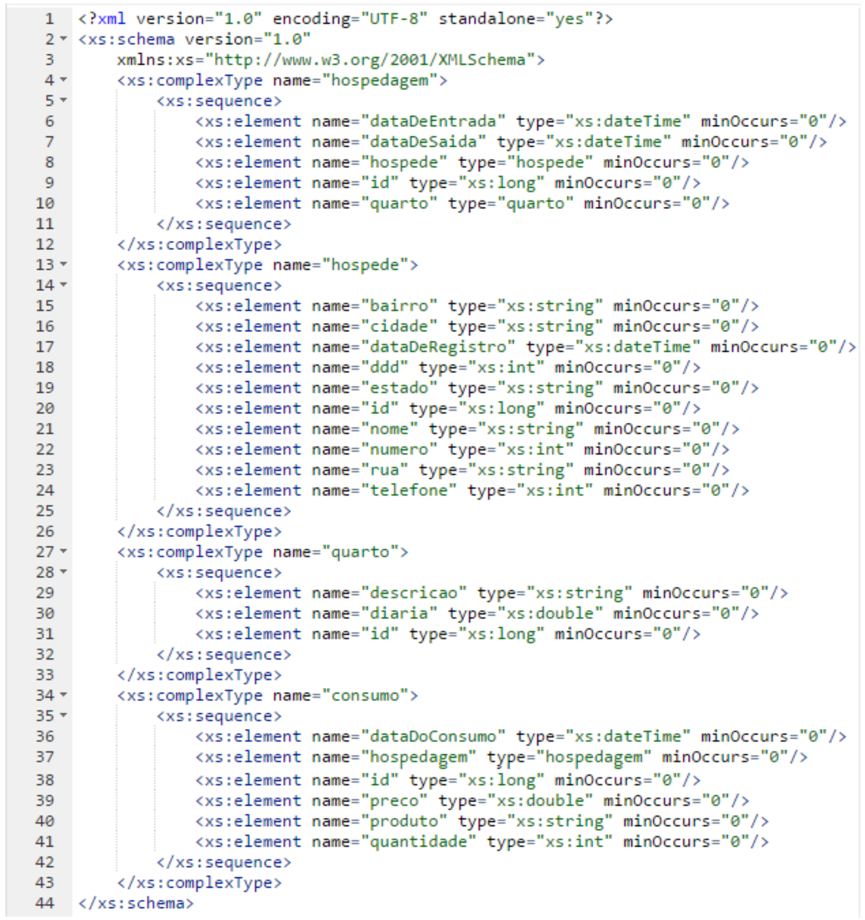
Apresentação

O objetivo do projeto do projeto é criar um mapeamento com hibernate para um *gerenciador de hotéis*

Diagrama do Modelo Conceitual (ER)

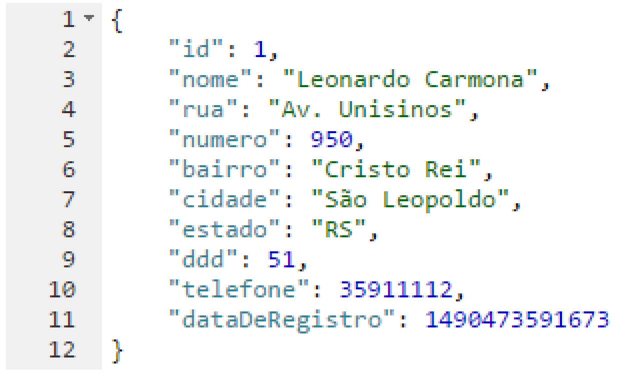


XSD

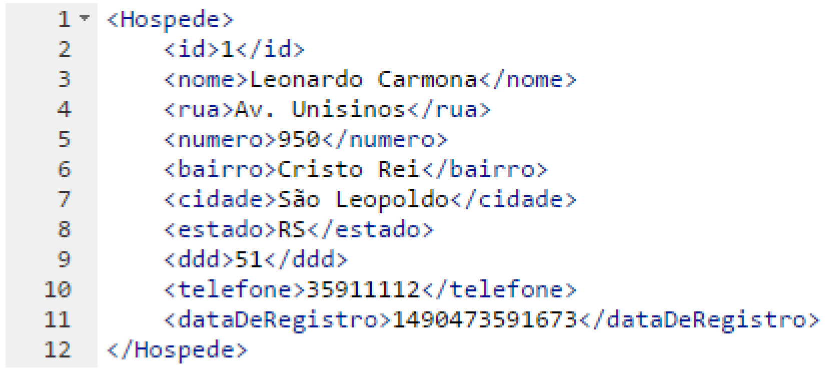


```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema version="1.0"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:complexType name="hospedagem">
        <xs:sequence>
            <xs:element name="dataDeEntrada" type="xs:dateTime" minOccurs="0"/>
            <xs:element name="dataDeSaida" type="xs:dateTime" minOccurs="0"/>
            <xs:element name="hospede" type="hospede" minOccurs="0"/>
            <xs:element name="id" type="xs:long" minOccurs="0"/>
            <xs:element name="quarto" type="quarto" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="hospede">
        <xs:sequence>
            <xs:element name="bairro" type="xs:string" minOccurs="0"/>
            <xs:element name="cidade" type="xs:string" minOccurs="0"/>
            <xs:element name="dataDeRegistro" type="xs:dateTime" minOccurs="0"/>
            <xs:element name="ddd" type="xs:int" minOccurs="0"/>
            <xs:element name="estado" type="xs:string" minOccurs="0"/>
            <xs:element name="id" type="xs:long" minOccurs="0"/>
            <xs:element name="nome" type="xs:string" minOccurs="0"/>
            <xs:element name="numero" type="xs:int" minOccurs="0"/>
            <xs:element name="rua" type="xs:string" minOccurs="0"/>
            <xs:element name="telefone" type="xs:int" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="quarto">
        <xs:sequence>
            <xs:element name="descricao" type="xs:string" minOccurs="0"/>
            <xs:element name="diaria" type="xs:double" minOccurs="0"/>
            <xs:element name="id" type="xs:long" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="consumo">
        <xs:sequence>
            <xs:element name="dataDoConsumo" type="xs:dateTime" minOccurs="0"/>
            <xs:element name="hospedagem" type="hospedagem" minOccurs="0"/>
            <xs:element name="id" type="xs:long" minOccurs="0"/>
            <xs:element name="preco" type="xs:double" minOccurs="0"/>
            <xs:element name="produto" type="xs:string" minOccurs="0"/>
            <xs:element name="quantidade" type="xs:int" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

JSON / XML (Hospede)

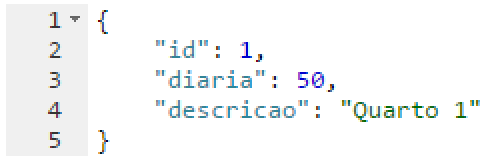


```json
{
    "id": 1,
    "nome": "Leonardo Carmona",
    "rua": "Av. Unisinos",
    "numero": 950,
    "bairro": "Cristo Rei",
    "cidade": "São Leopoldo",
    "estado": "RS",
    "ddd": 51,
    "telefone": 35911112,
    "dataDeRegistro": 1490473591673
}
```

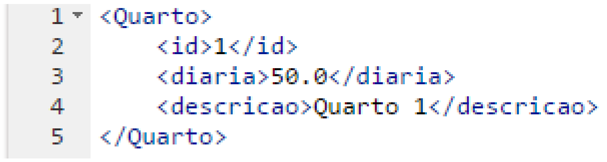


```xml
<Hospede>
    <id>1</id>
    <nome>Leonardo Carmona</nome>
    <rua>Av. Unisinos</rua>
    <numero>950</numero>
    <bairro>Cristo Rei</bairro>
    <cidade>São Leopoldo</cidade>
    <estado>RS</estado>
    <ddd>51</ddd>
    <telefone>35911112</telefone>
    <dataDeRegistro>1490473591673</dataDeRegistro>
</Hospede>
```

JSON / XML (Quarto)

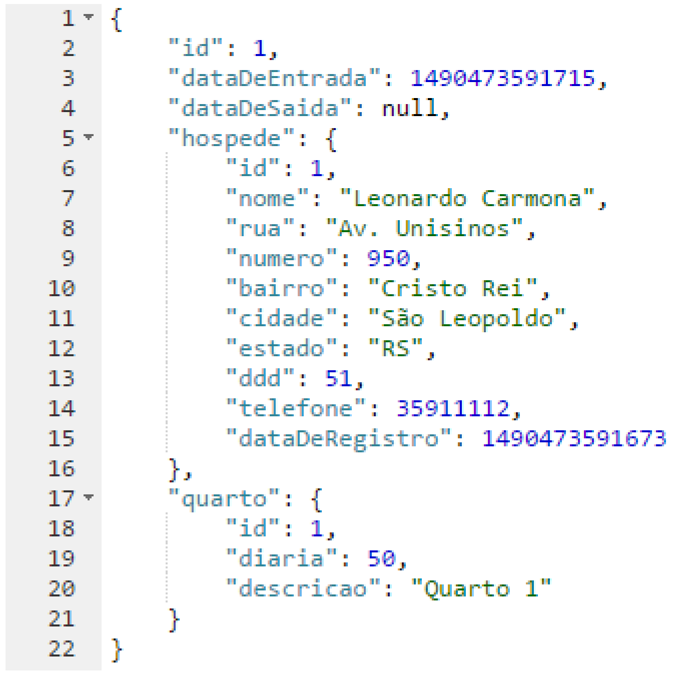


```
{
    "id": 1,
    "diaria": 50,
    "descricao": "Quarto 1"
}
```

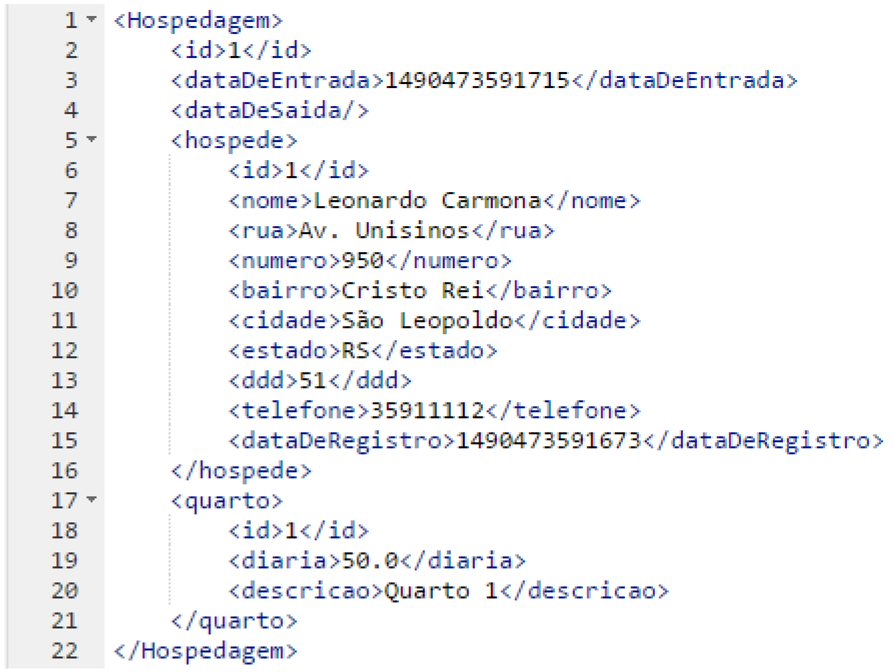


```
<Quarto>
    <id>1</id>
    <diaria>50.0</diaria>
    <descricao>Quarto 1</descricao>
</Quarto>
```

JSON / XML (Hospedagem)

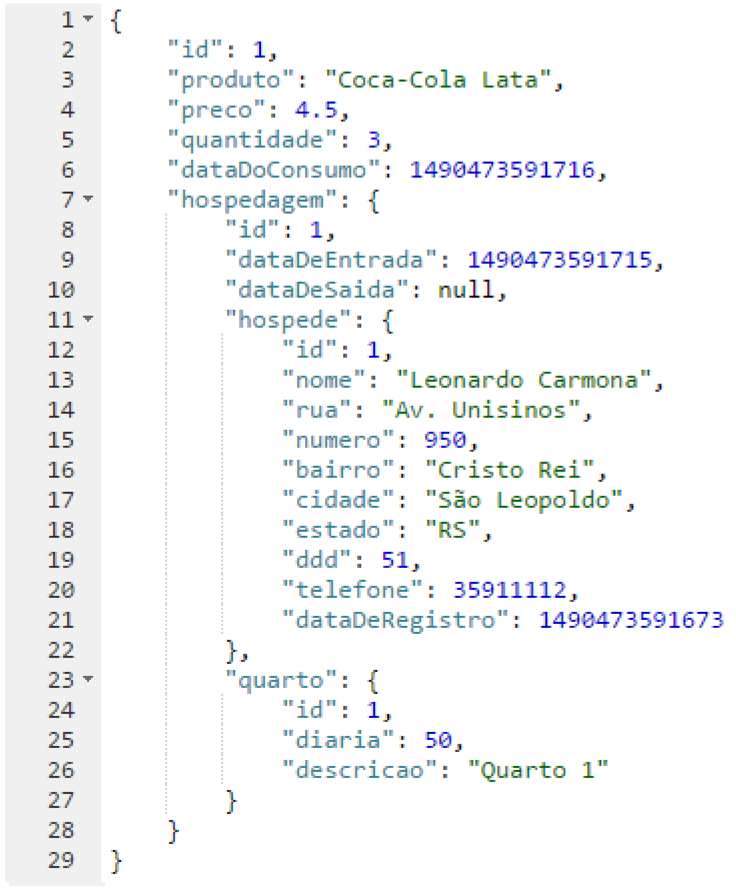


```json
{
    "id": 1,
    "dataDeEntrada": 1490473591715,
    "dataDeSaida": null,
    "hospede": {
        "id": 1,
        "nome": "Leonardo Carmona",
        "rua": "Av. Unisinos",
        "numero": 950,
        "bairro": "Cristo Rei",
        "cidade": "São Leopoldo",
        "estado": "RS",
        "ddd": 51,
        "telefone": 35911112,
        "dataDeRegistro": 1490473591673
    },
    "quarto": {
        "id": 1,
        "diaria": 50,
        "descricao": "Quarto 1"
    }
}
```

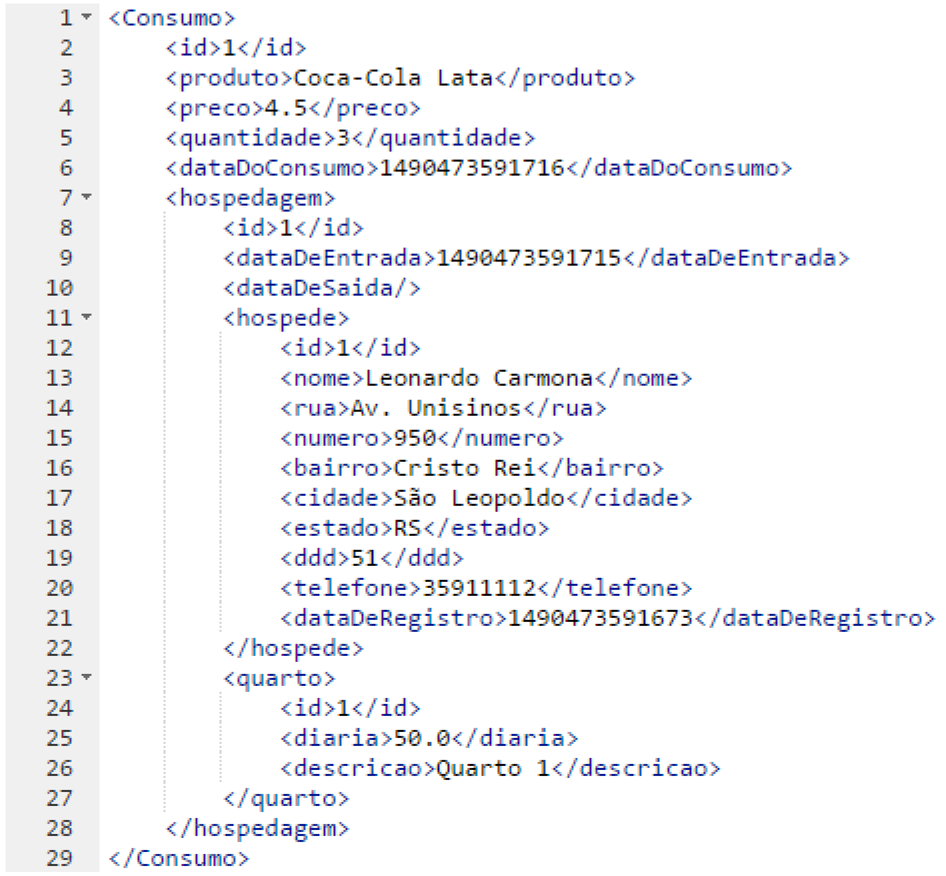


```xml
<Hospedagem>
    <id>1</id>
    <dataDeEntrada>1490473591715</dataDeEntrada>
    <dataDeSaida/>
    <hospede>
        <id>1</id>
        <nome>Leonardo Carmona</nome>
        <rua>Av. Unisinos</rua>
        <numero>950</numero>
        <bairro>Cristo Rei</bairro>
        <cidade>São Leopoldo</cidade>
        <estado>RS</estado>
        <ddd>51</ddd>
        <telefone>35911112</telefone>
        <dataDeRegistro>1490473591673</dataDeRegistro>
    </hospede>
    <quarto>
        <id>1</id>
        <diaria>50.0</diaria>
        <descricao>Quarto 1</descricao>
    </quarto>
</Hospedagem>
```

JSON / XML (Consumo)

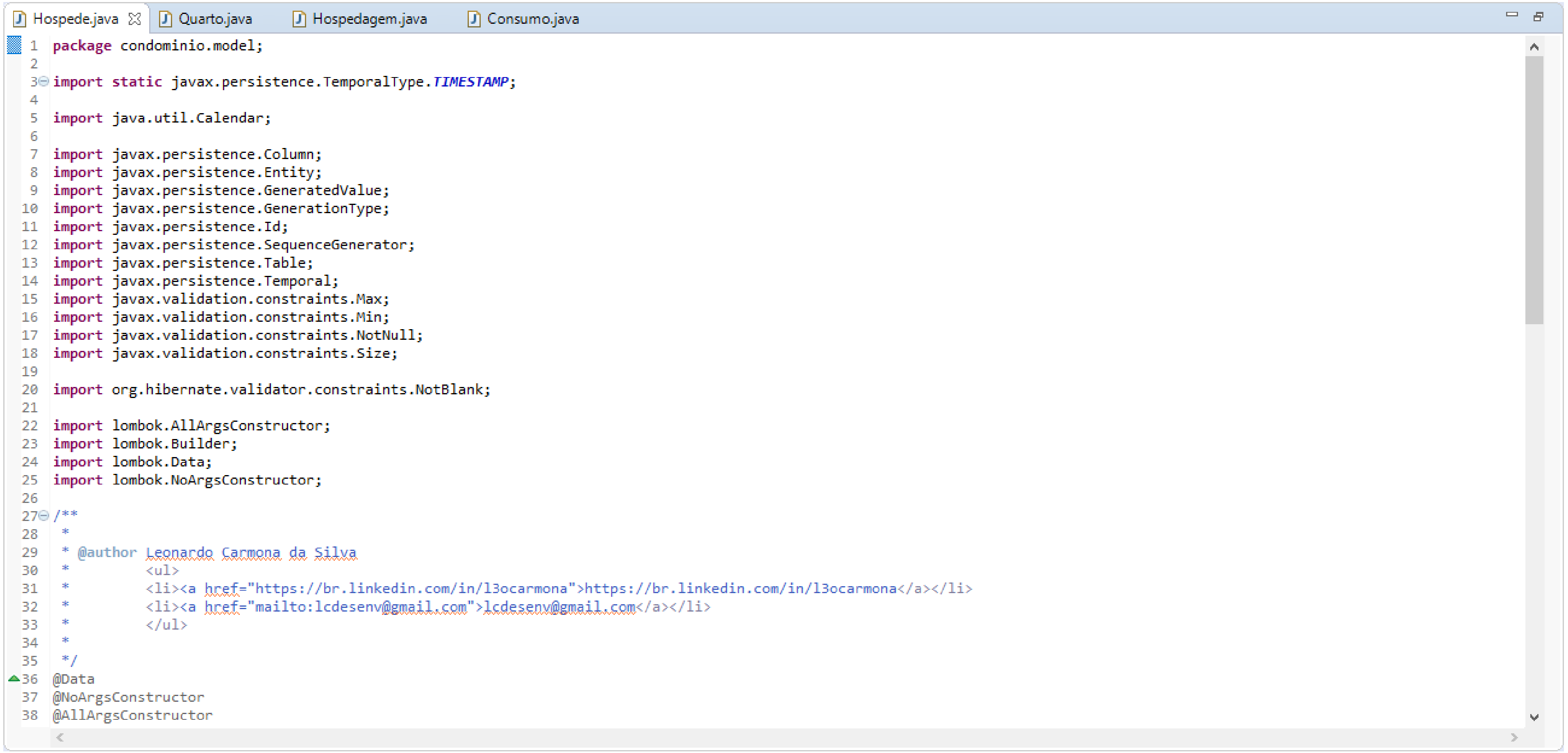


```json
{
    "id": 1,
    "produto": "Coca-Cola Lata",
    "preco": 4.5,
    "quantidade": 3,
    "dataDoConsumo": 1490473591716,
    "hospedagem": {
        "id": 1,
        "dataDeEntrada": 1490473591715,
        "dataDeSaida": null,
        "hospede": {
            "id": 1,
            "nome": "Leonardo Carmona",
            "rua": "Av. Unisinos",
            "numero": 950,
            "bairro": "Cristo Rei",
            "cidade": "São Leopoldo",
            "estado": "RS",
            "ddd": 51,
            "telefone": 35911112,
            "dataDeRegistro": 1490473591673
        },
        "quarto": {
            "id": 1,
            "diaria": 50,
            "descricao": "Quarto 1"
        }
    }
}
```



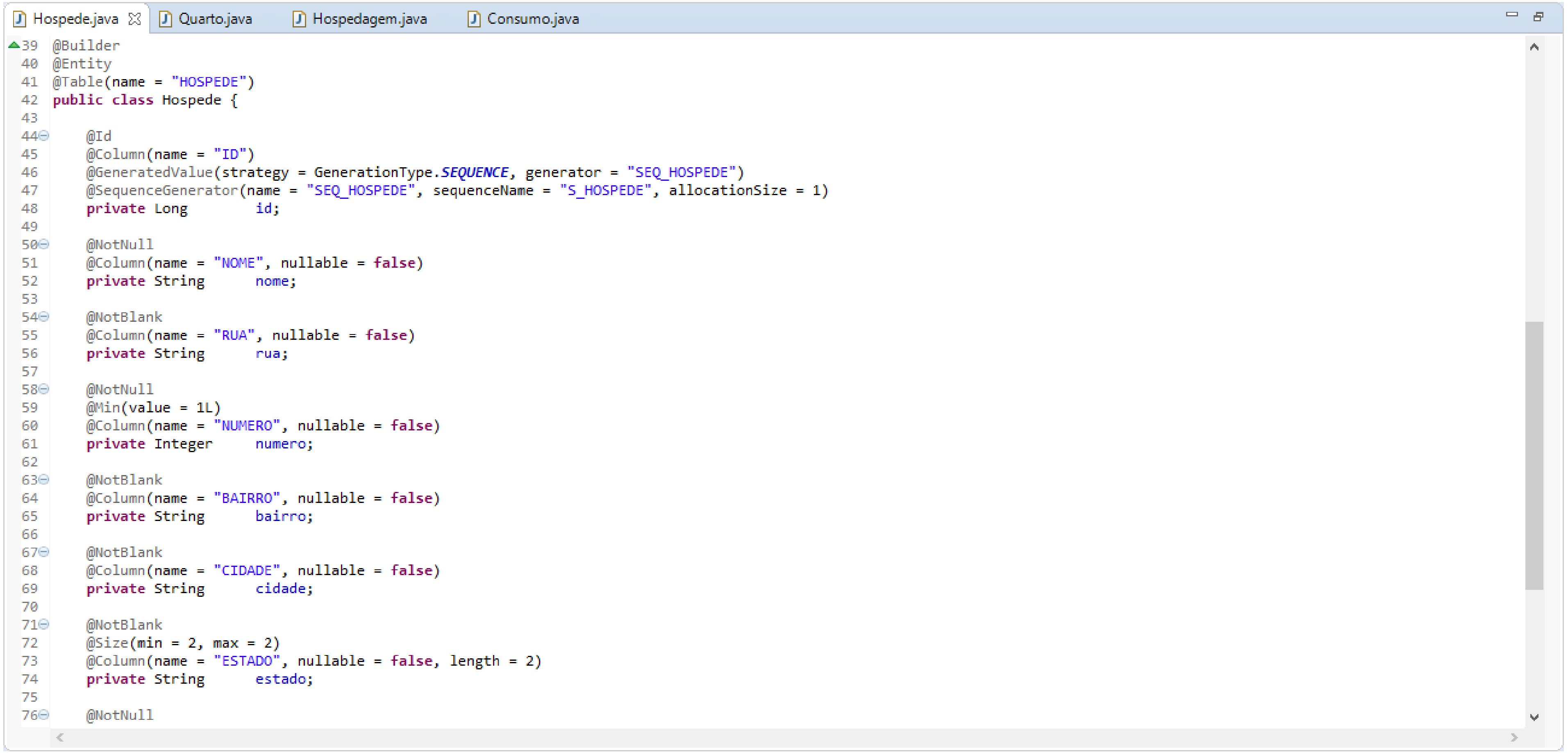
```xml
<Consumo>
    <id>1</id>
    <produto>Coca-Cola Lata</produto>
    <preco>4.5</preco>
    <quantidade>3</quantidade>
    <dataDoConsumo>1490473591716</dataDoConsumo>
    <hospedagem>
        <id>1</id>
        <dataDeEntrada>1490473591715</dataDeEntrada>
        <dataDeSaida/>
        <hospede>
            <id>1</id>
            <nome>Leonardo Carmona</nome>
            <rua>Av. Unisinos</rua>
            <numero>950</numero>
            <bairro>Cristo Rei</bairro>
            <cidade>São Leopoldo</cidade>
            <estado>RS</estado>
            <ddd>51</ddd>
            <telefone>35911112</telefone>
            <dataDeRegistro>1490473591673</dataDeRegistro>
        </hospede>
        <quarto>
            <id>1</id>
            <diaria>50.0</diaria>
            <descricao>Quarto 1</descricao>
        </quarto>
    </hospedagem>
</Consumo>
```

Mapeamento OR (Hospede) 1/3



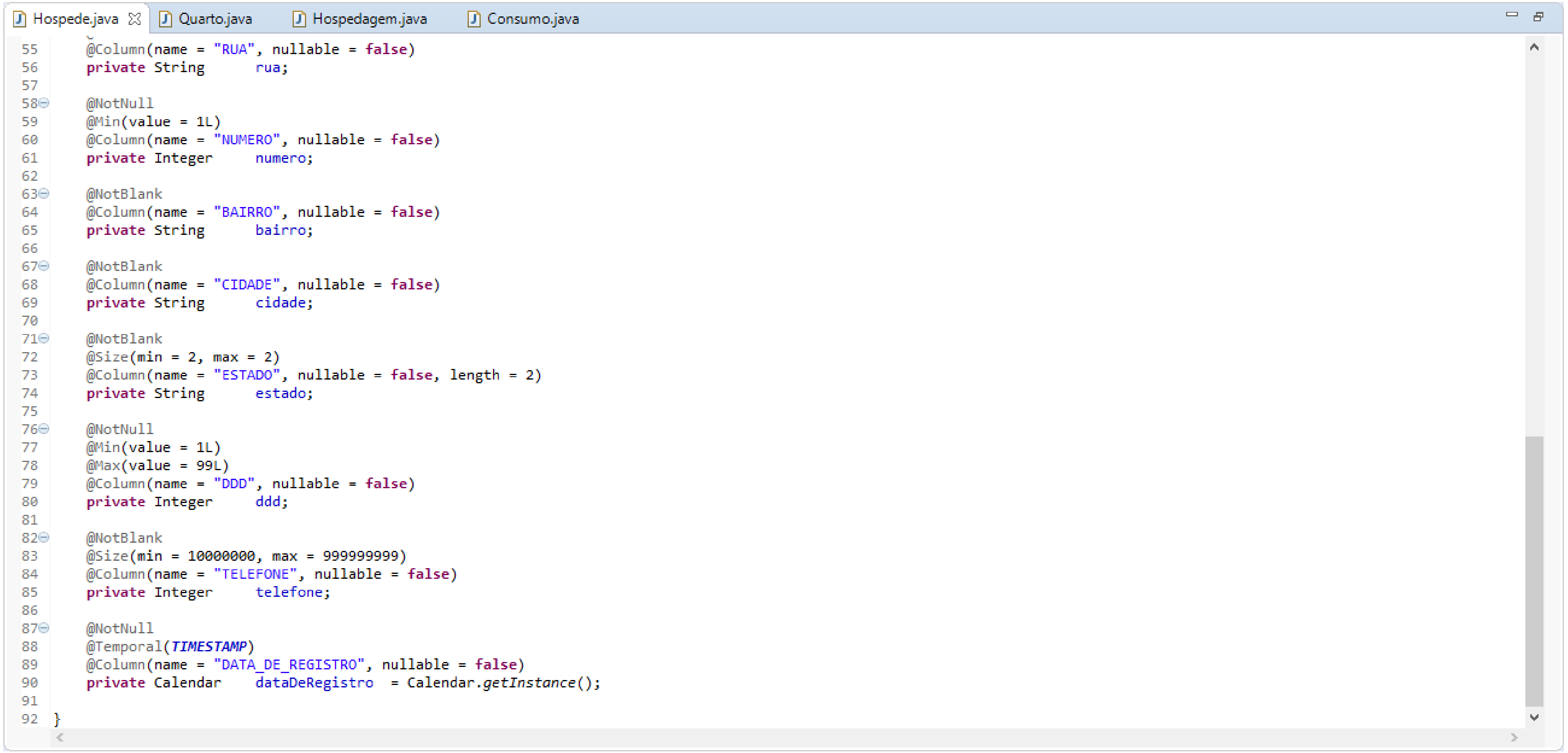
```java
package condominio.model;

import static javax.persistence.TemporalType.TIMESTAMP;

import java.util.Calendar;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.SequenceGenerator;
import javax.persistence.Table;
import javax.persistence.Temporal;
import javax.validation.constraints.Max;
import javax.validation.constraints.Min;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;

import org.hibernate.validator.constraints.NotBlank;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

/**
 *
 * @author Leonardo Carmona da Silva
 *         <ul>
 *         <li><a href="https://br.linkedin.com/in/l3ocarmona">https://br.linkedin.com/in/l3ocarmona</a></li>
 *         <li><a href="mailto:lcdesenv@gmail.com">lcdesenv@gmail.com</a></li>
 *         </ul>
 *
 */
@Data
@NoArgsConstructor
@AllArgsConstructor
```

Mapeamento OR (Hospede) 2/3



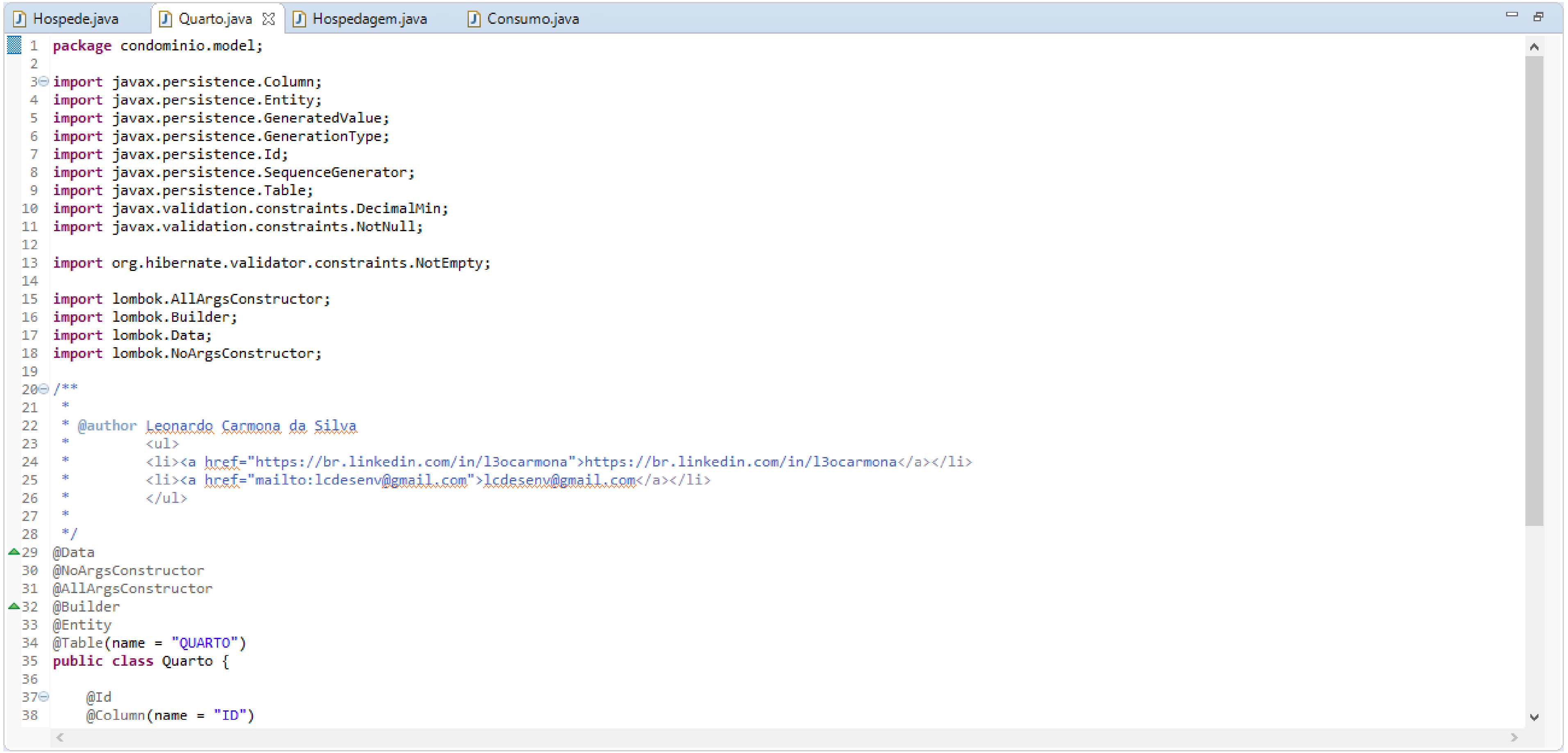
```java
@Builder
@Entity
@Table(name = "HOSPEDE")
public class Hospede {

    @Id
    @Column(name = "ID")
    @GeneratedValue(strategy = GenerationType.SEQUENCE, generator = "SEQ_HOSPEDE")
    @SequenceGenerator(name = "SEQ_HOSPEDE", sequenceName = "S_HOSPEDE", allocationSize = 1)
    private Long        id;

    @NotNull
    @Column(name = "NOME", nullable = false)
    private String      nome;

    @NotBlank
    @Column(name = "RUA", nullable = false)
    private String      rua;

    @NotNull
    @Min(value = 1L)
    @Column(name = "NUMERO", nullable = false)
    private Integer     numero;

    @NotBlank
    @Column(name = "BAIRRO", nullable = false)
    private String      bairro;

    @NotBlank
    @Column(name = "CIDADE", nullable = false)
    private String      cidade;

    @NotBlank
    @Size(min = 2, max = 2)
    @Column(name = "ESTADO", nullable = false, length = 2)
    private String      estado;

    @NotNull
```

Mapeamento OR (Hospede) 3/3


Hospede.java | Quarto.java | Hospedagem.java | Consumo.java

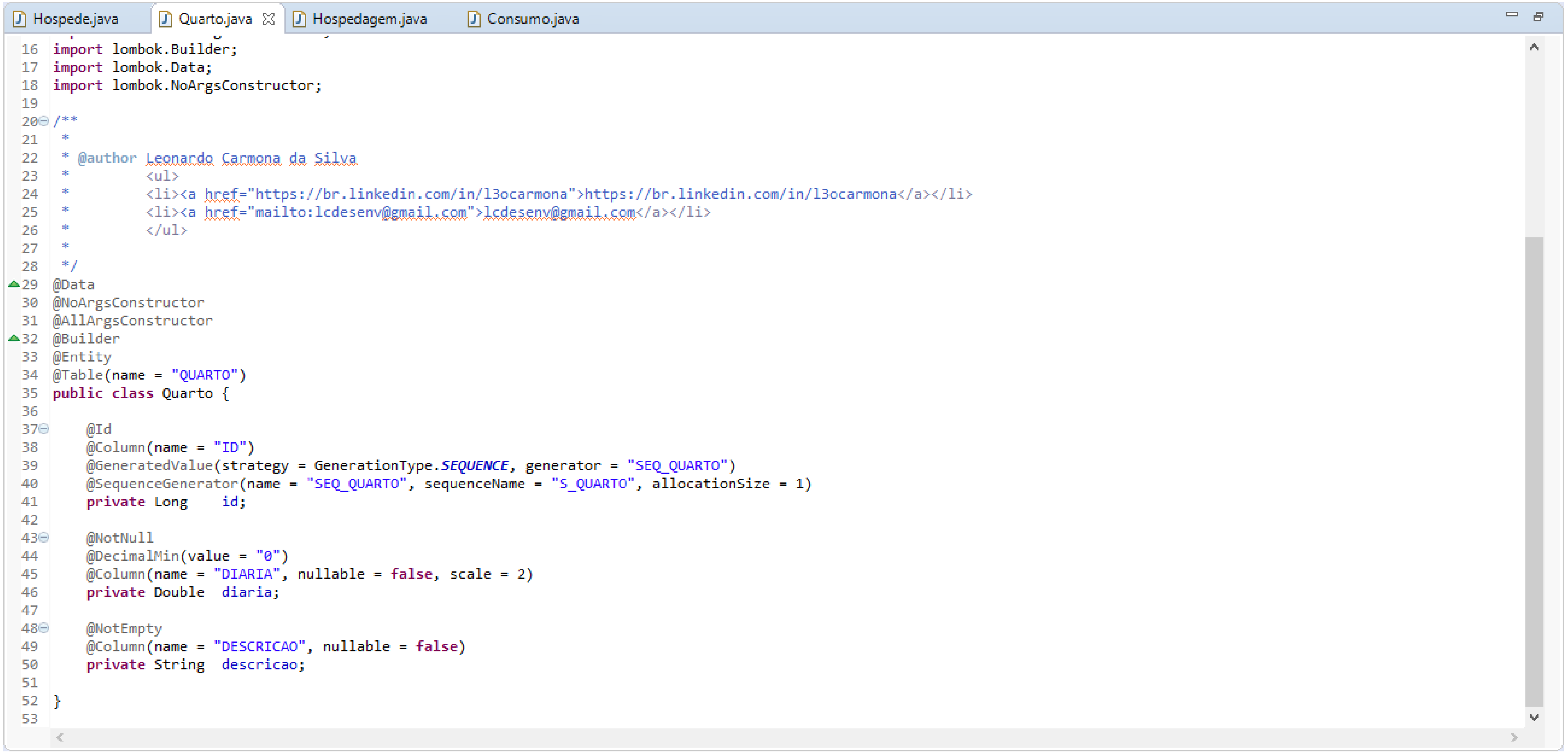
```java
    @Column(name = "RUA", nullable = false)
    private String      rua;

    @NotNull
    @Min(value = 1L)
    @Column(name = "NUMERO", nullable = false)
    private Integer     numero;

    @NotBlank
    @Column(name = "BAIRRO", nullable = false)
    private String      bairro;

    @NotBlank
    @Column(name = "CIDADE", nullable = false)
    private String      cidade;

    @NotBlank
    @Size(min = 2, max = 2)
    @Column(name = "ESTADO", nullable = false, length = 2)
    private String      estado;

    @NotNull
    @Min(value = 1L)
    @Max(value = 99L)
    @Column(name = "DDD", nullable = false)
    private Integer     ddd;

    @NotBlank
    @Size(min = 10000000, max = 999999999)
    @Column(name = "TELEFONE", nullable = false)
    private Integer     telefone;

    @NotNull
    @Temporal(TIMESTAMP)
    @Column(name = "DATA_DE_REGISTRO", nullable = false)
    private Calendar    dataDeRegistro  = Calendar.getInstance();

}
```

Mapeamento OR (Quarto) 1/2



```java
package condominio.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.SequenceGenerator;
import javax.persistence.Table;
import javax.validation.constraints.DecimalMin;
import javax.validation.constraints.NotNull;

import org.hibernate.validator.constraints.NotEmpty;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

/**
 *
 * @author Leonardo Carmona da Silva
 *          <ul>
 *          <li><a href="https://br.linkedin.com/in/l3ocarmona">https://br.linkedin.com/in/l3ocarmona</a></li>
 *          <li><a href="mailto:lcdesenv@gmail.com">lcdesenv@gmail.com</a></li>
 *          </ul>
 *
 */
@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
@Entity
@Table(name = "QUARTO")
public class Quarto {

    @Id
    @Column(name = "ID")
```

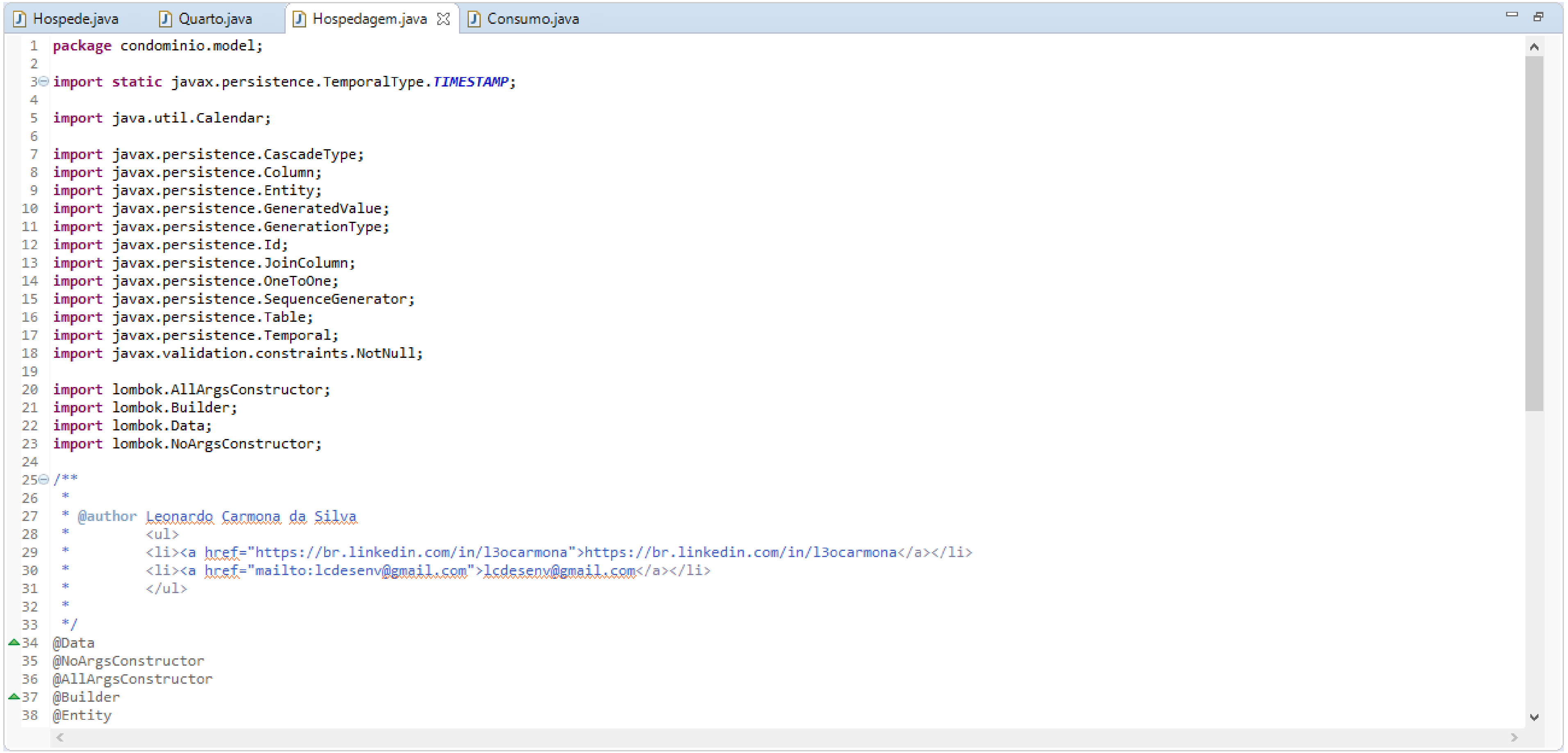
Mapeamento OR (Quarto) 2/2



```java
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

/**
 *
 * @author Leonardo Carmona da Silva
 *         <ul>
 *         <li><a href="https://br.linkedin.com/in/l3ocarmona">https://br.linkedin.com/in/l3ocarmona</a></li>
 *         <li><a href="mailto:lcdesenv@gmail.com">lcdesenv@gmail.com</a></li>
 *         </ul>
 *
 */
@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
@Entity
@Table(name = "QUARTO")
public class Quarto {

    @Id
    @Column(name = "ID")
    @GeneratedValue(strategy = GenerationType.SEQUENCE, generator = "SEQ_QUARTO")
    @SequenceGenerator(name = "SEQ_QUARTO", sequenceName = "S_QUARTO", allocationSize = 1)
    private Long    id;

    @NotNull
    @DecimalMin(value = "0")
    @Column(name = "DIARIA", nullable = false, scale = 2)
    private Double  diaria;

    @NotEmpty
    @Column(name = "DESCRICAO", nullable = false)
    private String  descricao;

}
```

Gerenciamento de Hotéis

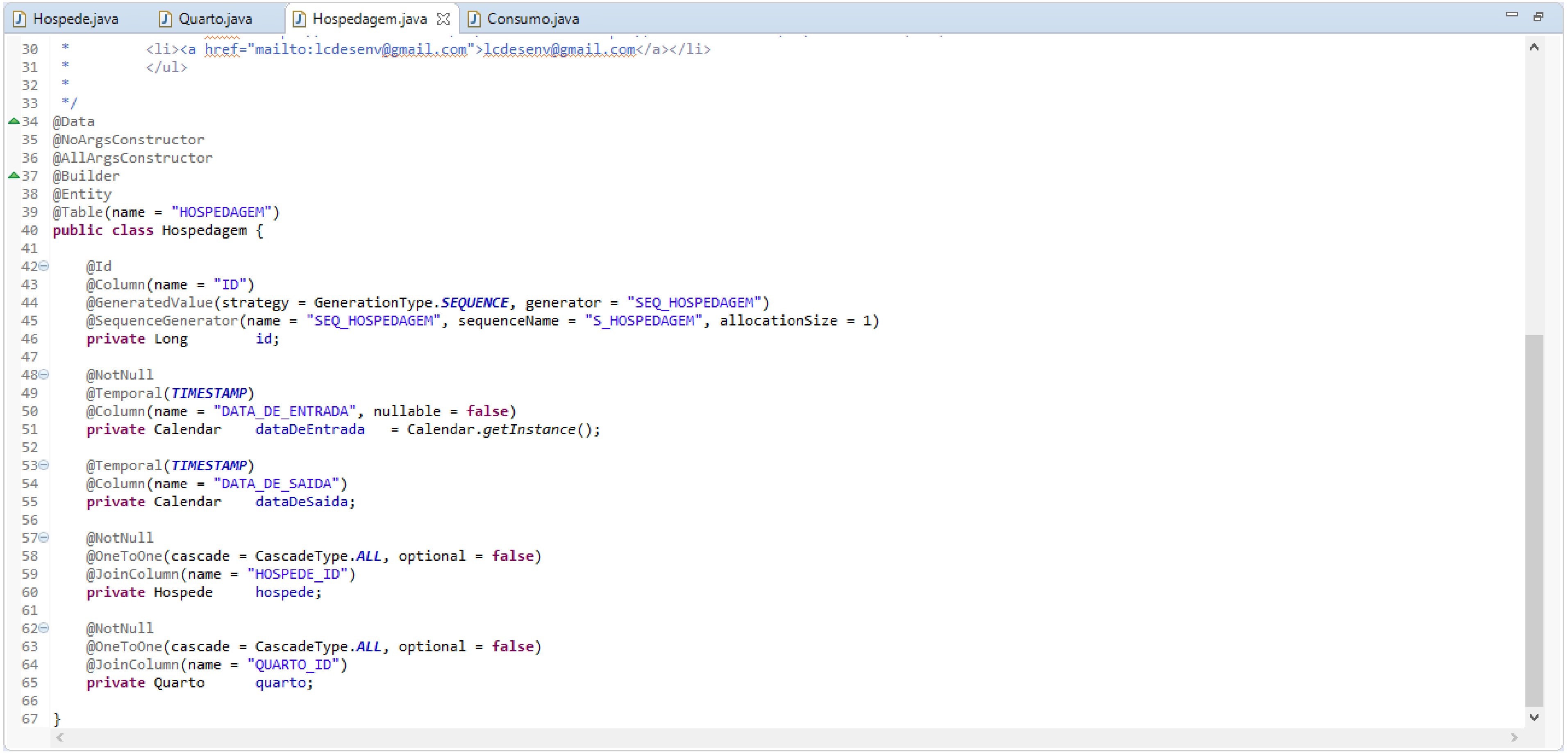
Mapeamento OR (Hospedagem) 1/2



Hospede.java Quarto.java Hospedagem.java Consumo.java

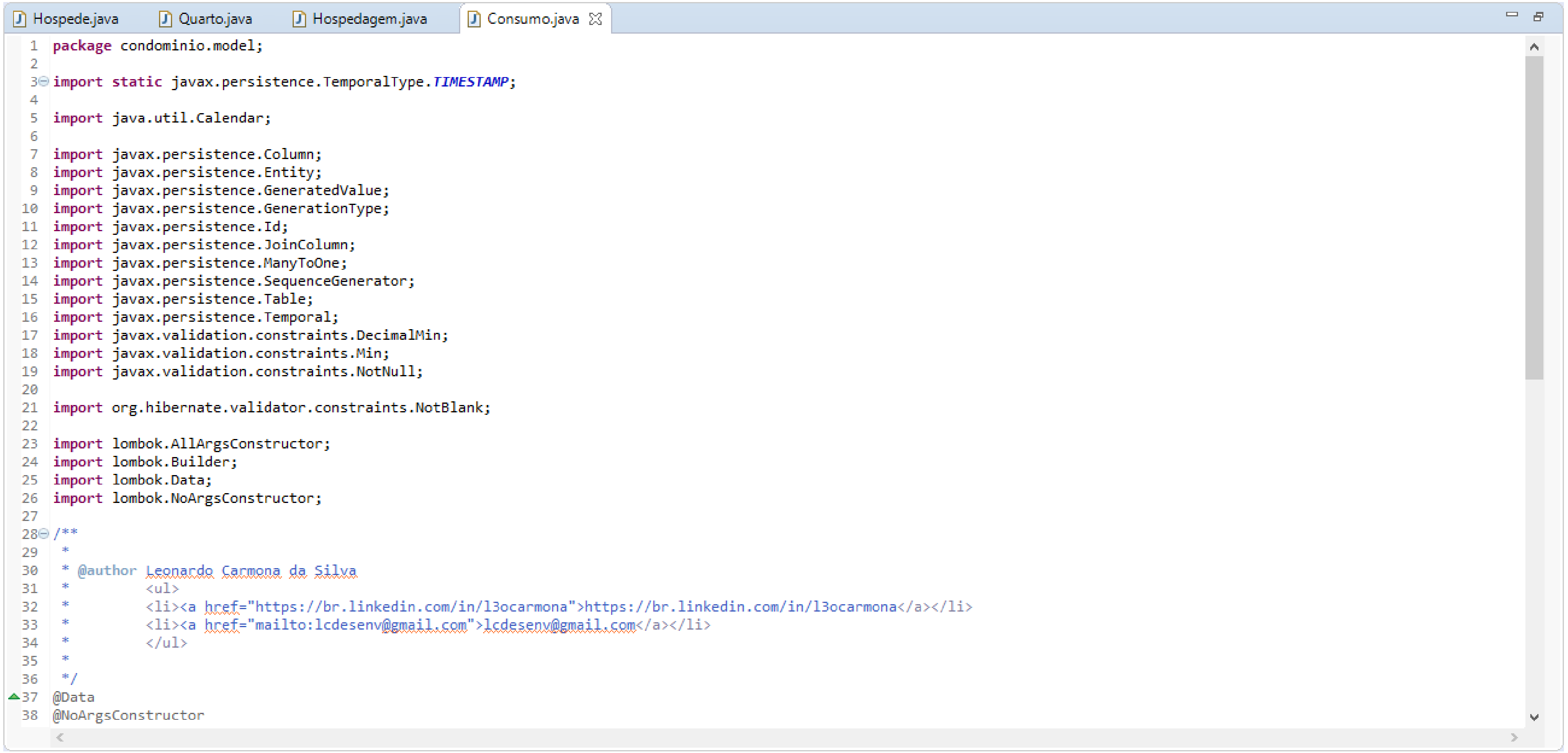
```java
package condominio.model;

import static javax.persistence.TemporalType.TIMESTAMP;

import java.util.Calendar;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.OneToOne;
import javax.persistence.SequenceGenerator;
import javax.persistence.Table;
import javax.persistence.Temporal;
import javax.validation.constraints.NotNull;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

/**
 *
 * @author Leonardo Carmona da Silva
 *          <ul>
 *          <li><a href="https://br.linkedin.com/in/l3ocarmona">https://br.linkedin.com/in/l3ocarmona</a></li>
 *          <li><a href="mailto:lcdesenv@gmail.com">lcdesenv@gmail.com</a></li>
 *          </ul>
 *
 */
@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
@Entity
```

Mapeamento OR (Hospedagem) 2/2


Hospede.java | Quarto.java | Hospedagem.java | Consumo.java

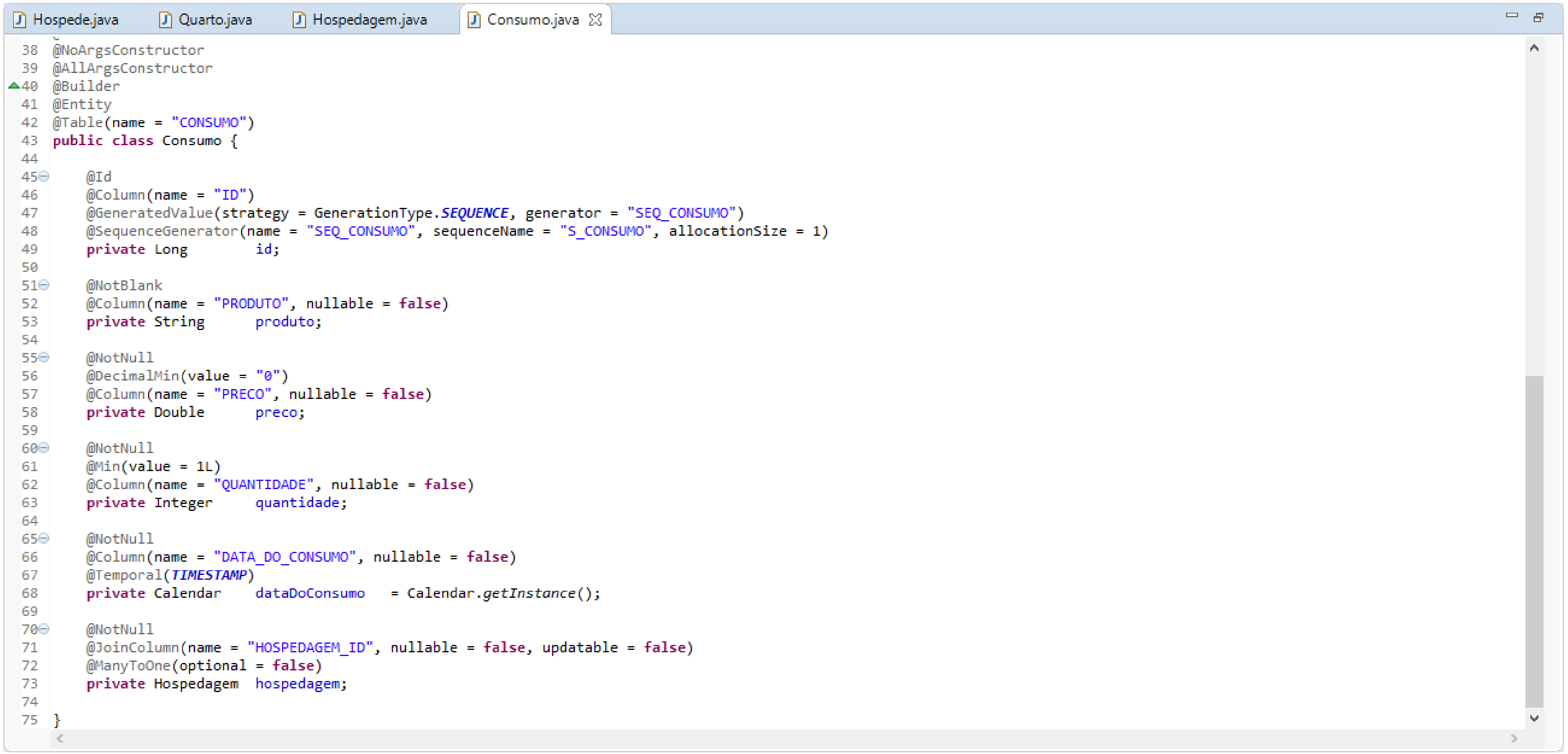
```
 *         <li><a href="mailto:lcdesenv@gmail.com">lcdesenv@gmail.com</a></li>
 *         </ul>
 *
 */
@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
@Entity
@Table(name = "HOSPEDAGEM")
public class Hospedagem {

    @Id
    @Column(name = "ID")
    @GeneratedValue(strategy = GenerationType.SEQUENCE, generator = "SEQ_HOSPEDAGEM")
    @SequenceGenerator(name = "SEQ_HOSPEDAGEM", sequenceName = "S_HOSPEDAGEM", allocationSize = 1)
    private Long        id;

    @NotNull
    @Temporal(TIMESTAMP)
    @Column(name = "DATA_DE_ENTRADA", nullable = false)
    private Calendar    dataDeEntrada    = Calendar.getInstance();

    @Temporal(TIMESTAMP)
    @Column(name = "DATA_DE_SAIDA")
    private Calendar    dataDeSaida;

    @NotNull
    @OneToOne(cascade = CascadeType.ALL, optional = false)
    @JoinColumn(name = "HOSPEDE_ID")
    private Hospede     hospede;

    @NotNull
    @OneToOne(cascade = CascadeType.ALL, optional = false)
    @JoinColumn(name = "QUARTO_ID")
    private Quarto      quarto;

}
```

Mapeamento OR (Consumo) 1/2


Hospede.java Quarto.java Hospedagem.java Consumo.java

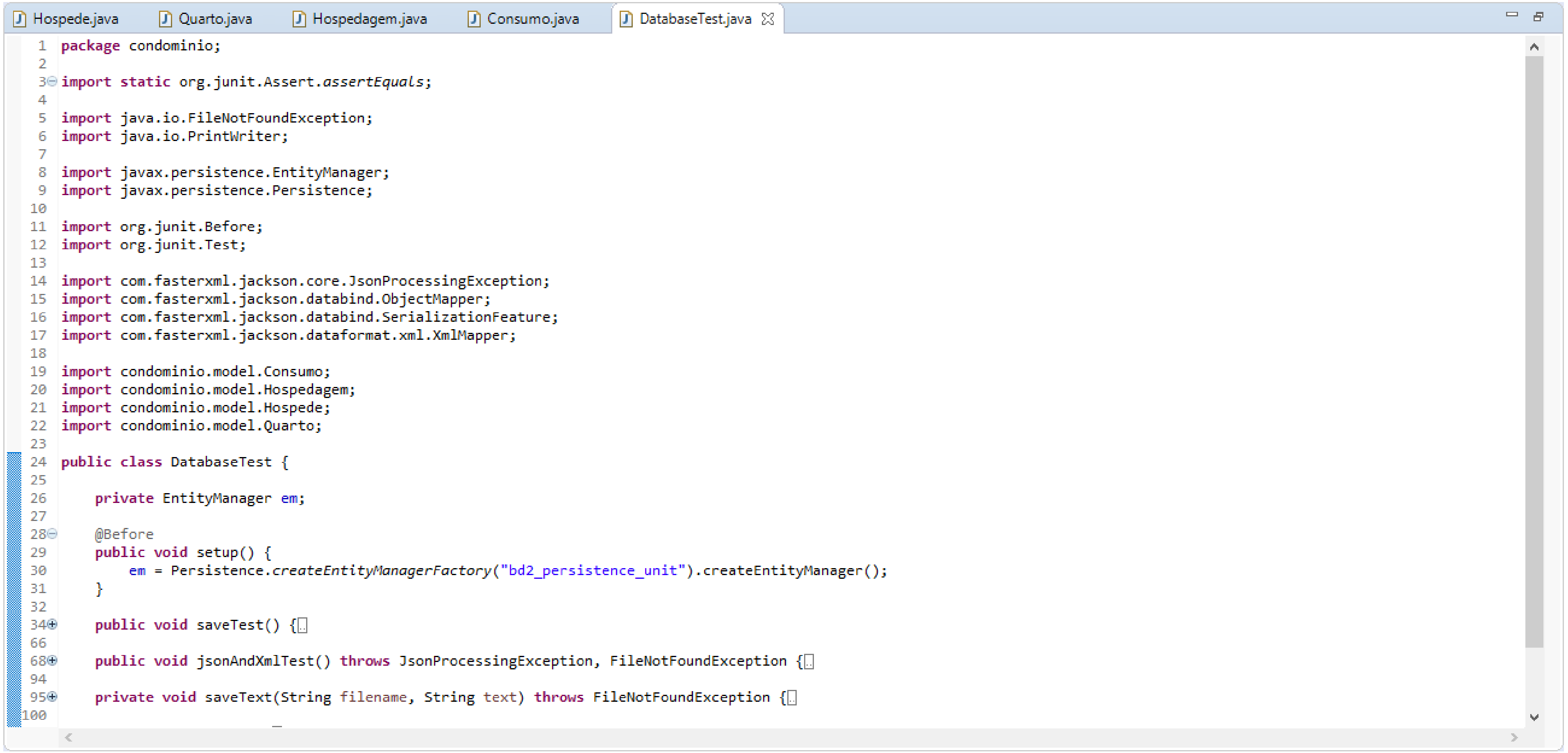
```
package condominio.model;

import static javax.persistence.TemporalType.TIMESTAMP;

import java.util.Calendar;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.SequenceGenerator;
import javax.persistence.Table;
import javax.persistence.Temporal;
import javax.validation.constraints.DecimalMin;
import javax.validation.constraints.Min;
import javax.validation.constraints.NotNull;

import org.hibernate.validator.constraints.NotBlank;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

/**
 *
 * @author Leonardo Carmona da Silva
 *         <ul>
 *         <li><a href="https://br.linkedin.com/in/l3ocarmona">https://br.linkedin.com/in/l3ocarmona</a></li>
 *         <li><a href="mailto:lcdesenv@gmail.com">lcdesenv@gmail.com</a></li>
 *         </ul>
 *
 */
@Data
@NoArgsConstructor
```

Mapeamento OR (Consumo) 2/2



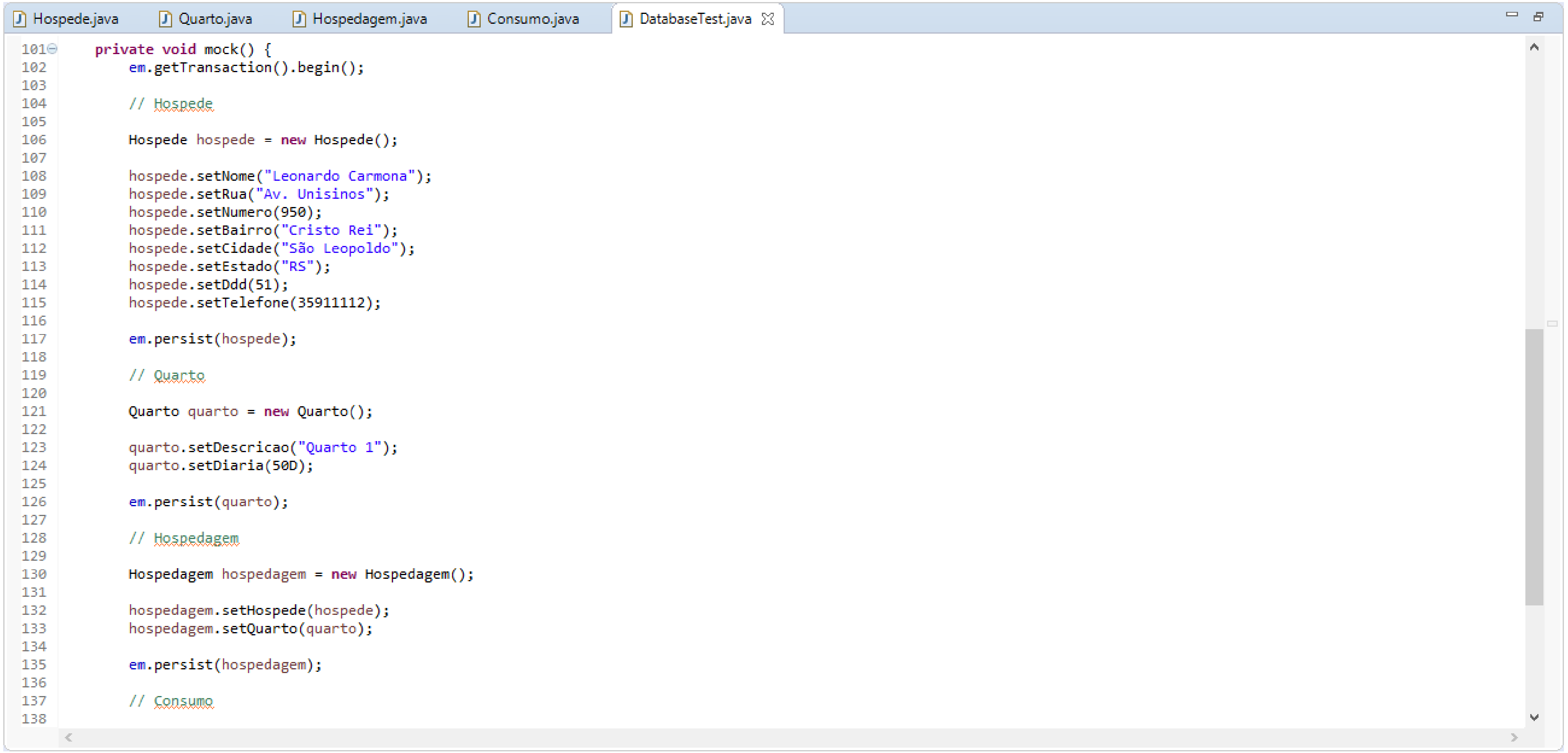
```java
@NoArgsConstructor
@AllArgsConstructor
@Builder
@Entity
@Table(name = "CONSUMO")
public class Consumo {

    @Id
    @Column(name = "ID")
    @GeneratedValue(strategy = GenerationType.SEQUENCE, generator = "SEQ_CONSUMO")
    @SequenceGenerator(name = "SEQ_CONSUMO", sequenceName = "S_CONSUMO", allocationSize = 1)
    private Long        id;

    @NotBlank
    @Column(name = "PRODUTO", nullable = false)
    private String      produto;

    @NotNull
    @DecimalMin(value = "0")
    @Column(name = "PRECO", nullable = false)
    private Double      preco;

    @NotNull
    @Min(value = 1L)
    @Column(name = "QUANTIDADE", nullable = false)
    private Integer     quantidade;

    @NotNull
    @Column(name = "DATA_DO_CONSUMO", nullable = false)
    @Temporal(TIMESTAMP)
    private Calendar    dataDoConsumo   = Calendar.getInstance();

    @NotNull
    @JoinColumn(name = "HOSPEDAGEM_ID", nullable = false, updatable = false)
    @ManyToOne(optional = false)
    private Hospedagem  hospedagem;

}
```

DatabaseTest (Dependências e Setup)



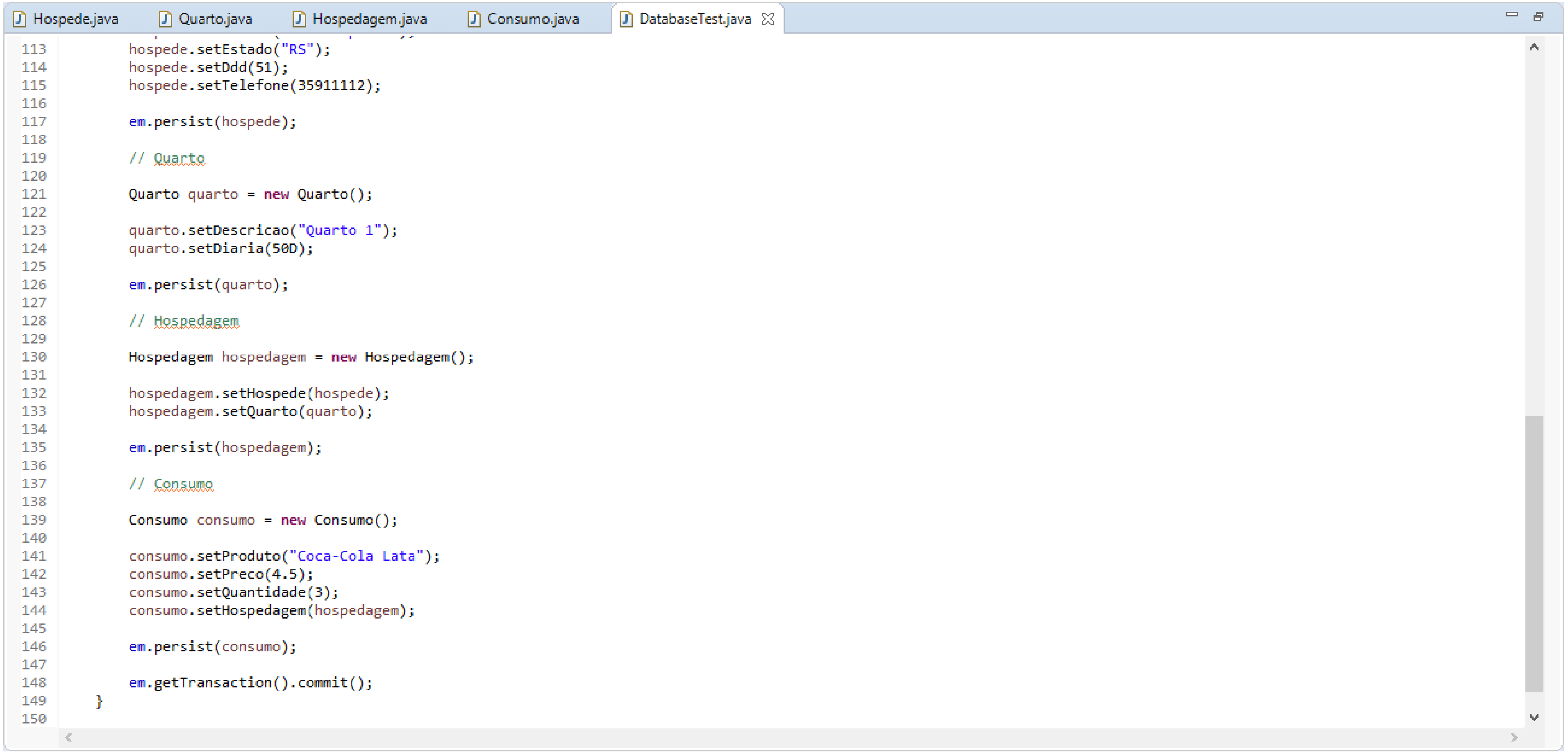
```java
package condominio;

import static org.junit.Assert.assertEquals;

import java.io.FileNotFoundException;
import java.io.PrintWriter;

import javax.persistence.EntityManager;
import javax.persistence.Persistence;

import org.junit.Before;
import org.junit.Test;

import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.fasterxml.jackson.databind.SerializationFeature;
import com.fasterxml.jackson.dataformat.xml.XmlMapper;

import condominio.model.Consumo;
import condominio.model.Hospedagem;
import condominio.model.Hospede;
import condominio.model.Quarto;

public class DatabaseTest {

    private EntityManager em;

    @Before
    public void setup() {
        em = Persistence.createEntityManagerFactory("bd2_persistence_unit").createEntityManager();
    }

    public void saveTest() {

    public void jsonAndXmlTest() throws JsonProcessingException, FileNotFoundException {

    private void saveText(String filename, String text) throws FileNotFoundException {
```

DatabaseTest (População dos Objetos) 1/2


Hospede.java Quarto.java Hospedagem.java Consumo.java DatabaseTest.java

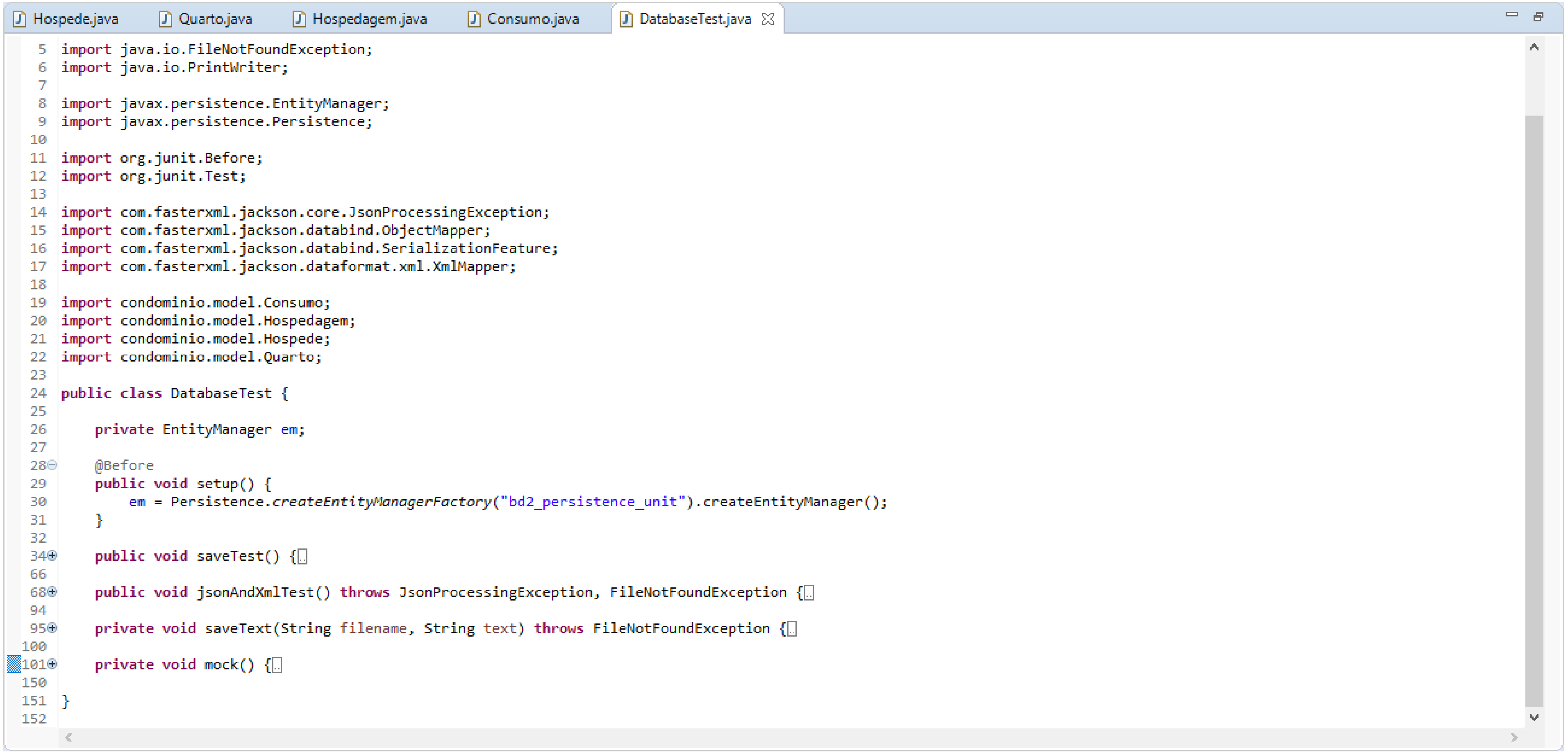
```
    private void mock() {
        em.getTransaction().begin();

        // Hospede

        Hospede hospede = new Hospede();

        hospede.setNome("Leonardo Carmona");
        hospede.setRua("Av. Unisinos");
        hospede.setNumero(950);
        hospede.setBairro("Cristo Rei");
        hospede.setCidade("São Leopoldo");
        hospede.setEstado("RS");
        hospede.setDdd(51);
        hospede.setTelefone(35911112);

        em.persist(hospede);

        // Quarto

        Quarto quarto = new Quarto();

        quarto.setDescricao("Quarto 1");
        quarto.setDiaria(50D);

        em.persist(quarto);

        // Hospedagem

        Hospedagem hospedagem = new Hospedagem();

        hospedagem.setHospede(hospede);
        hospedagem.setQuarto(quarto);

        em.persist(hospedagem);

        // Consumo

```

DatabaseTest (População dos Objetos) 2/2



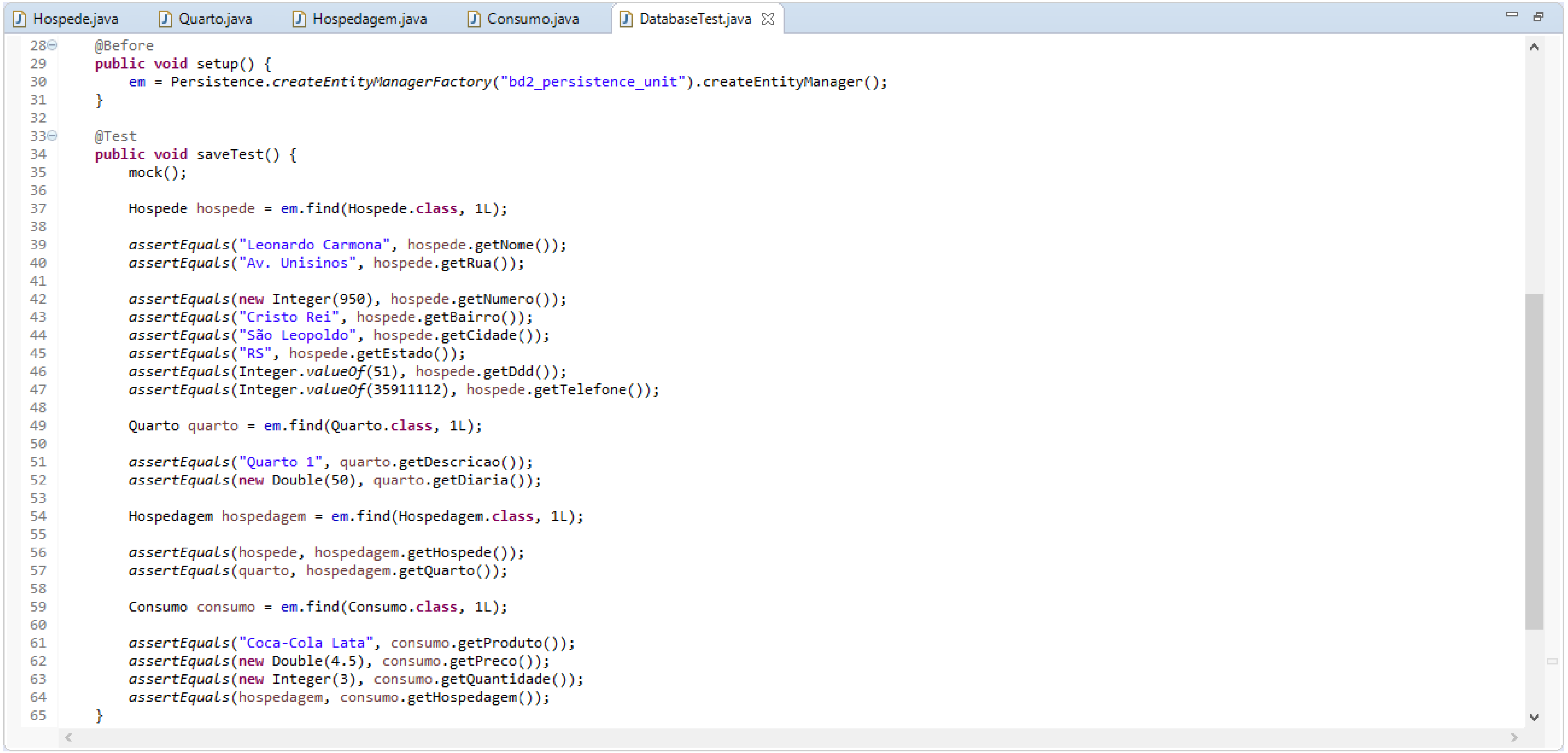
```
        hospede.setEstado("RS");
        hospede.setDdd(51);
        hospede.setTelefone(35911112);

        em.persist(hospede);

        // Quarto

        Quarto quarto = new Quarto();

        quarto.setDescricao("Quarto 1");
        quarto.setDiaria(50D);

        em.persist(quarto);

        // Hospedagem

        Hospedagem hospedagem = new Hospedagem();

        hospedagem.setHospede(hospede);
        hospedagem.setQuarto(quarto);

        em.persist(hospedagem);

        // Consumo

        Consumo consumo = new Consumo();

        consumo.setProduto("Coca-Cola Lata");
        consumo.setPreco(4.5);
        consumo.setQuantidade(3);
        consumo.setHospedagem(hospedagem);

        em.persist(consumo);

        em.getTransaction().commit();
    }

```

Testes Unitários (setup)



```
import java.io.FileNotFoundException;
import java.io.PrintWriter;

import javax.persistence.EntityManager;
import javax.persistence.Persistence;

import org.junit.Before;
import org.junit.Test;

import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.fasterxml.jackson.databind.SerializationFeature;
import com.fasterxml.jackson.dataformat.xml.XmlMapper;

import condominio.model.Consumo;
import condominio.model.Hospedagem;
import condominio.model.Hospede;
import condominio.model.Quarto;

public class DatabaseTest {

    private EntityManager em;

    @Before
    public void setup() {
        em = Persistence.createEntityManagerFactory("bd2_persistence_unit").createEntityManager();
    }

    public void saveTest() {

    public void jsonAndXmlTest() throws JsonProcessingException, FileNotFoundException {

    private void saveText(String filename, String text) throws FileNotFoundException {

    private void mock() {

}
```

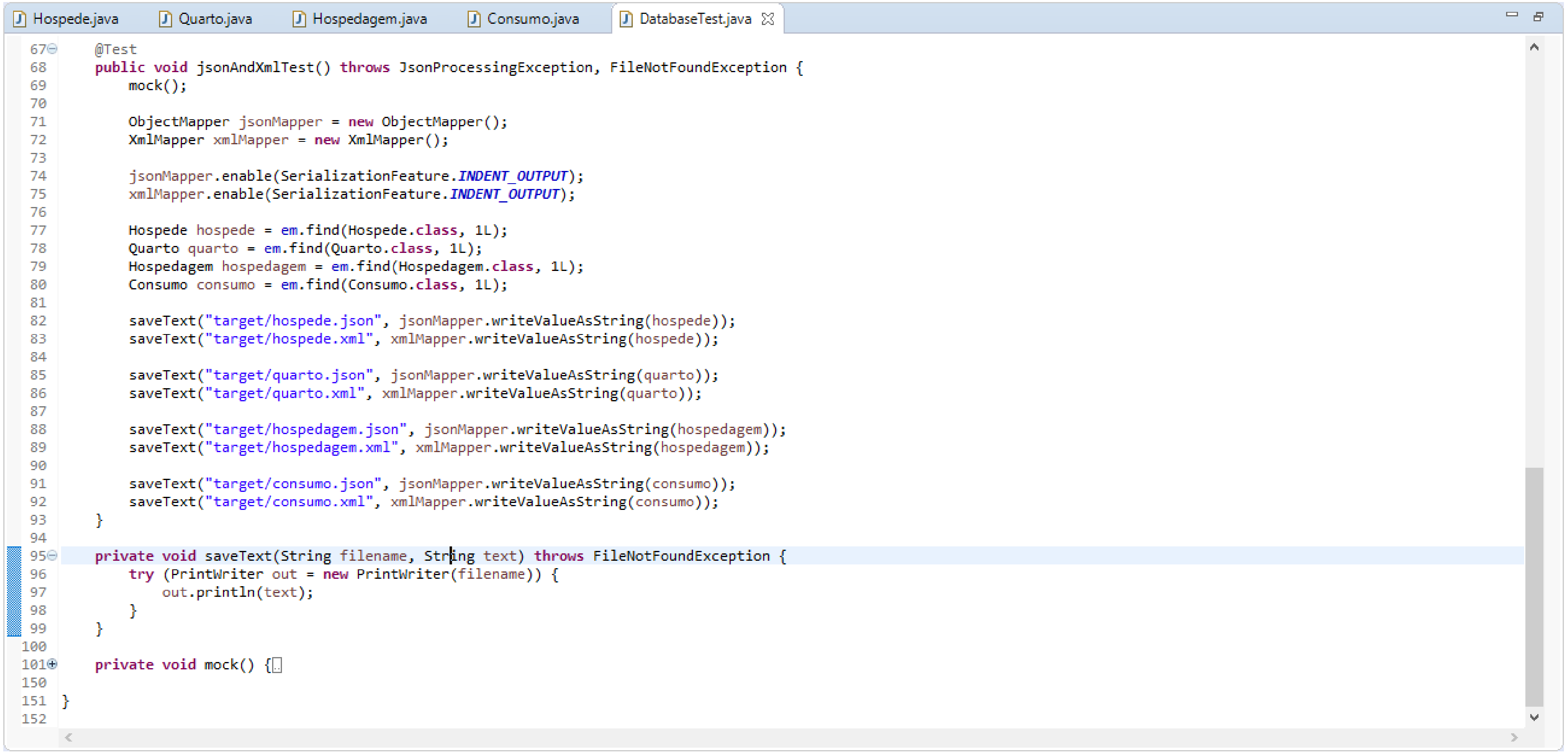
Testes Unitários (saveTest)



Hospede.java | Quarto.java | Hospedagem.java | Consumo.java | DatabaseTest.java

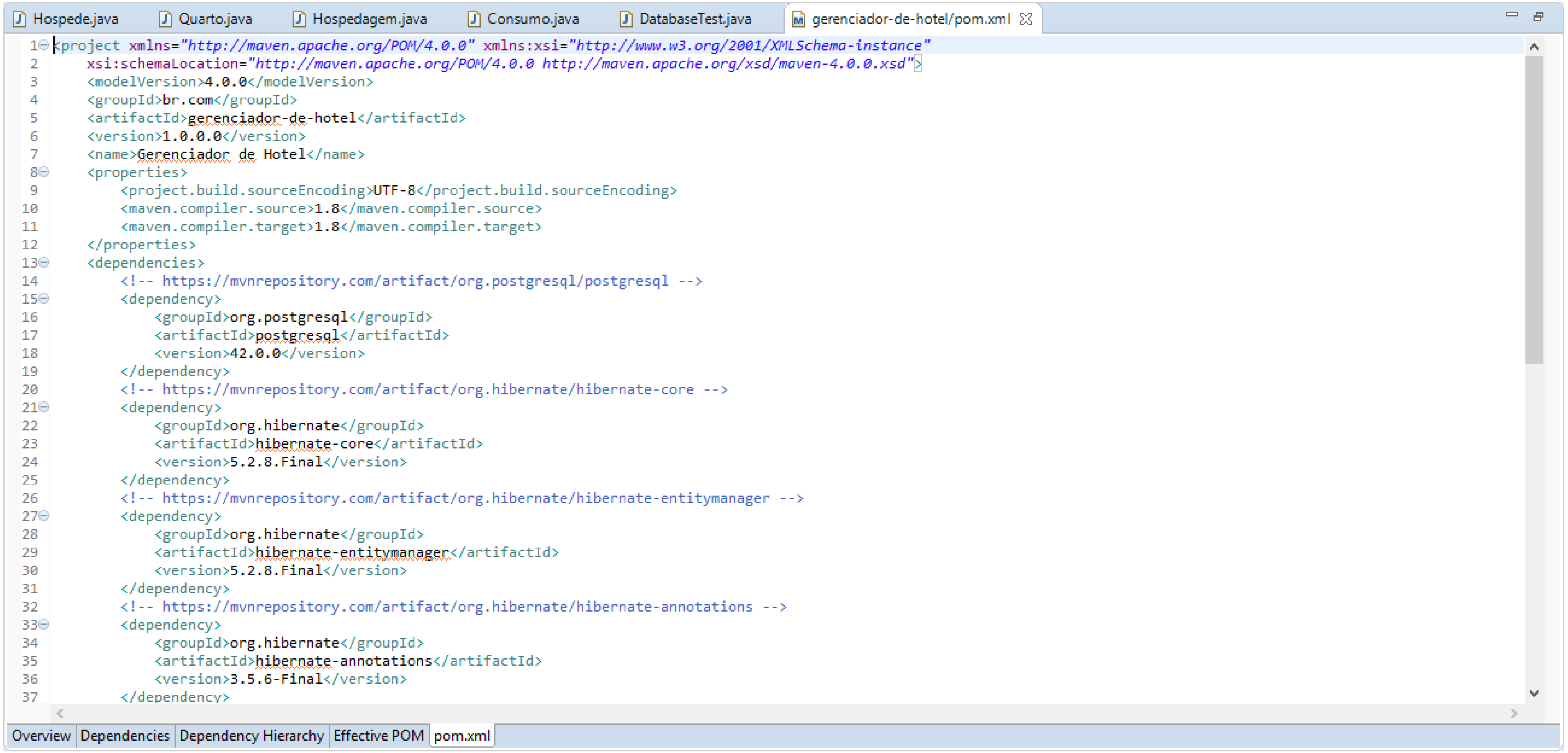
```java
    @Before
    public void setup() {
        em = Persistence.createEntityManagerFactory("bd2_persistence_unit").createEntityManager();
    }

    @Test
    public void saveTest() {
        mock();

        Hospede hospede = em.find(Hospede.class, 1L);

        assertEquals("Leonardo Carmona", hospede.getNome());
        assertEquals("Av. Unisinos", hospede.getRua());

        assertEquals(new Integer(950), hospede.getNumero());
        assertEquals("Cristo Rei", hospede.getBairro());
        assertEquals("São Leopoldo", hospede.getCidade());
        assertEquals("RS", hospede.getEstado());
        assertEquals(Integer.valueOf(51), hospede.getDdd());
        assertEquals(Integer.valueOf(35911112), hospede.getTelefone());

        Quarto quarto = em.find(Quarto.class, 1L);

        assertEquals("Quarto 1", quarto.getDescricao());
        assertEquals(new Double(50), quarto.getDiaria());

        Hospedagem hospedagem = em.find(Hospedagem.class, 1L);

        assertEquals(hospede, hospedagem.getHospede());
        assertEquals(quarto, hospedagem.getQuarto());

        Consumo consumo = em.find(Consumo.class, 1L);

        assertEquals("Coca-Cola Lata", consumo.getProduto());
        assertEquals(new Double(4.5), consumo.getPreco());
        assertEquals(new Integer(3), consumo.getQuantidade());
        assertEquals(hospedagem, consumo.getHospedagem());
    }
```

Testes Unitários (jsonAndXmlTest)


Hospede.java | Quarto.java | Hospedagem.java | Consumo.java | DatabaseTest.java

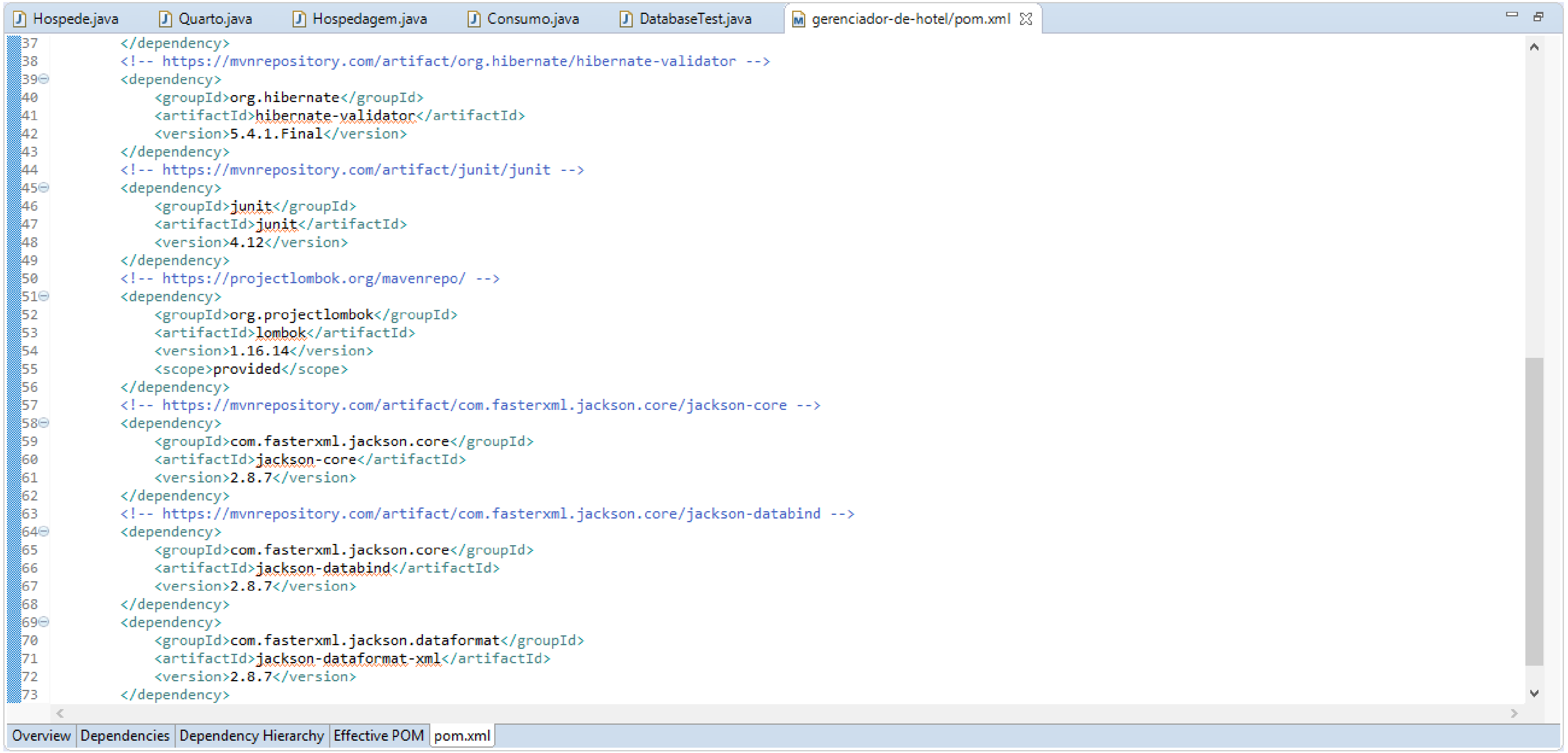
```java
    @Test
    public void jsonAndXmlTest() throws JsonProcessingException, FileNotFoundException {
        mock();

        ObjectMapper jsonMapper = new ObjectMapper();
        XmlMapper xmlMapper = new XmlMapper();

        jsonMapper.enable(SerializationFeature.INDENT_OUTPUT);
        xmlMapper.enable(SerializationFeature.INDENT_OUTPUT);

        Hospede hospede = em.find(Hospede.class, 1L);
        Quarto quarto = em.find(Quarto.class, 1L);
        Hospedagem hospedagem = em.find(Hospedagem.class, 1L);
        Consumo consumo = em.find(Consumo.class, 1L);

        saveText("target/hospede.json", jsonMapper.writeValueAsString(hospede));
        saveText("target/hospede.xml", xmlMapper.writeValueAsString(hospede));

        saveText("target/quarto.json", jsonMapper.writeValueAsString(quarto));
        saveText("target/quarto.xml", xmlMapper.writeValueAsString(quarto));

        saveText("target/hospedagem.json", jsonMapper.writeValueAsString(hospedagem));
        saveText("target/hospedagem.xml", xmlMapper.writeValueAsString(hospedagem));

        saveText("target/consumo.json", jsonMapper.writeValueAsString(consumo));
        saveText("target/consumo.xml", xmlMapper.writeValueAsString(consumo));
    }

    private void saveText(String filename, String text) throws FileNotFoundException {
        try (PrintWriter out = new PrintWriter(filename)) {
            out.println(text);
        }
    }

    private void mock() {
}
```

pom.xml 1/3



```xml
<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 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>br.com</groupId>
    <artifactId>gerenciador-de-hotel</artifactId>
    <version>1.0.0.0</version>
    <name>Gerenciador de Hotel</name>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
    </properties>
    <dependencies>
        <!-- https://mvnrepository.com/artifact/org.postgresql/postgresql -->
        <dependency>
            <groupId>org.postgresql</groupId>
            <artifactId>postgresql</artifactId>
            <version>42.0.0</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
        <dependency>
            <groupId>org.hibernate</groupId>
            <artifactId>hibernate-core</artifactId>
            <version>5.2.8.Final</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-entitymanager -->
        <dependency>
            <groupId>org.hibernate</groupId>
            <artifactId>hibernate-entitymanager</artifactId>
            <version>5.2.8.Final</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-annotations -->
        <dependency>
            <groupId>org.hibernate</groupId>
            <artifactId>hibernate-annotations</artifactId>
            <version>3.5.6-Final</version>
        </dependency>
```

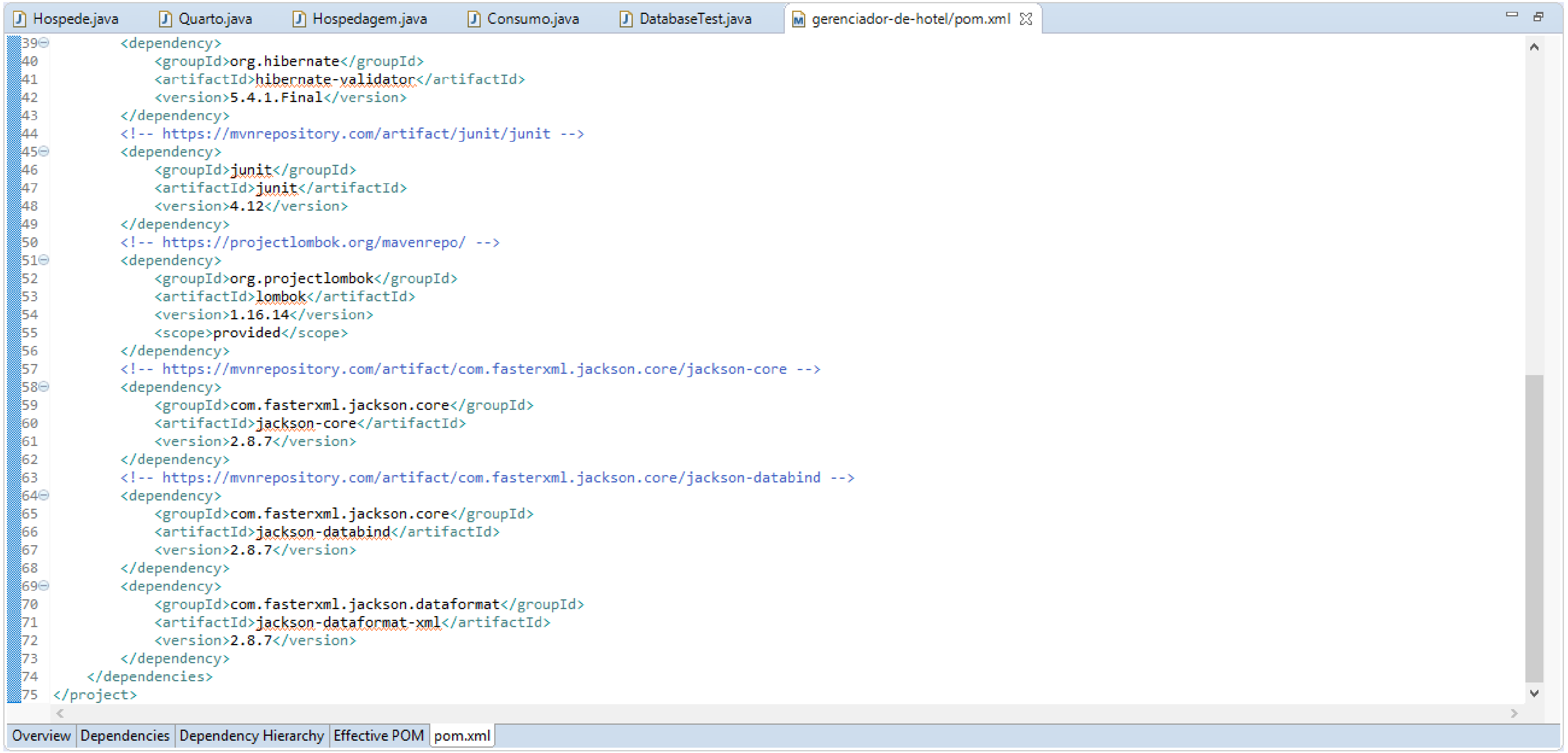
pom.xml 2/3



```xml
        </dependency>
        <!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-validator -->
        <dependency>
            <groupId>org.hibernate</groupId>
            <artifactId>hibernate-validator</artifactId>
            <version>5.4.1.Final</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/junit/junit -->
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.12</version>
        </dependency>
        <!-- https://projectlombok.org/mavenrepo/ -->
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <version>1.16.14</version>
            <scope>provided</scope>
        </dependency>
        <!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-core -->
        <dependency>
            <groupId>com.fasterxml.jackson.core</groupId>
            <artifactId>jackson-core</artifactId>
            <version>2.8.7</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-databind -->
        <dependency>
            <groupId>com.fasterxml.jackson.core</groupId>
            <artifactId>jackson-databind</artifactId>
            <version>2.8.7</version>
        </dependency>
        <dependency>
            <groupId>com.fasterxml.jackson.dataformat</groupId>
            <artifactId>jackson-dataformat-xml</artifactId>
            <version>2.8.7</version>
        </dependency>
```

Gerenciamento de Hotéis

pom.xml 3/3



```
        <dependency>
            <groupId>org.hibernate</groupId>
            <artifactId>hibernate-validator</artifactId>
            <version>5.4.1.Final</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/junit/junit -->
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.12</version>
        </dependency>
        <!-- https://projectlombok.org/mavenrepo/ -->
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <version>1.16.14</version>
            <scope>provided</scope>
        </dependency>
        <!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-core -->
        <dependency>
            <groupId>com.fasterxml.jackson.core</groupId>
            <artifactId>jackson-core</artifactId>
            <version>2.8.7</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-databind -->
        <dependency>
            <groupId>com.fasterxml.jackson.core</groupId>
            <artifactId>jackson-databind</artifactId>
            <version>2.8.7</version>
        </dependency>
        <dependency>
            <groupId>com.fasterxml.jackson.dataformat</groupId>
            <artifactId>jackson-dataformat-xml</artifactId>
            <version>2.8.7</version>
        </dependency>
    </dependencies>
</project>
```

Conclusão

- IDE: Eclipse
- Tipo de Projeto: Maven
- Bibliotecas:
  - Postgresql
  - Hibernate (core, entitymanager, annotations & validator)
  - Junit
  - Lombok
  - Jackson (core, databind & dataformat-xml)