

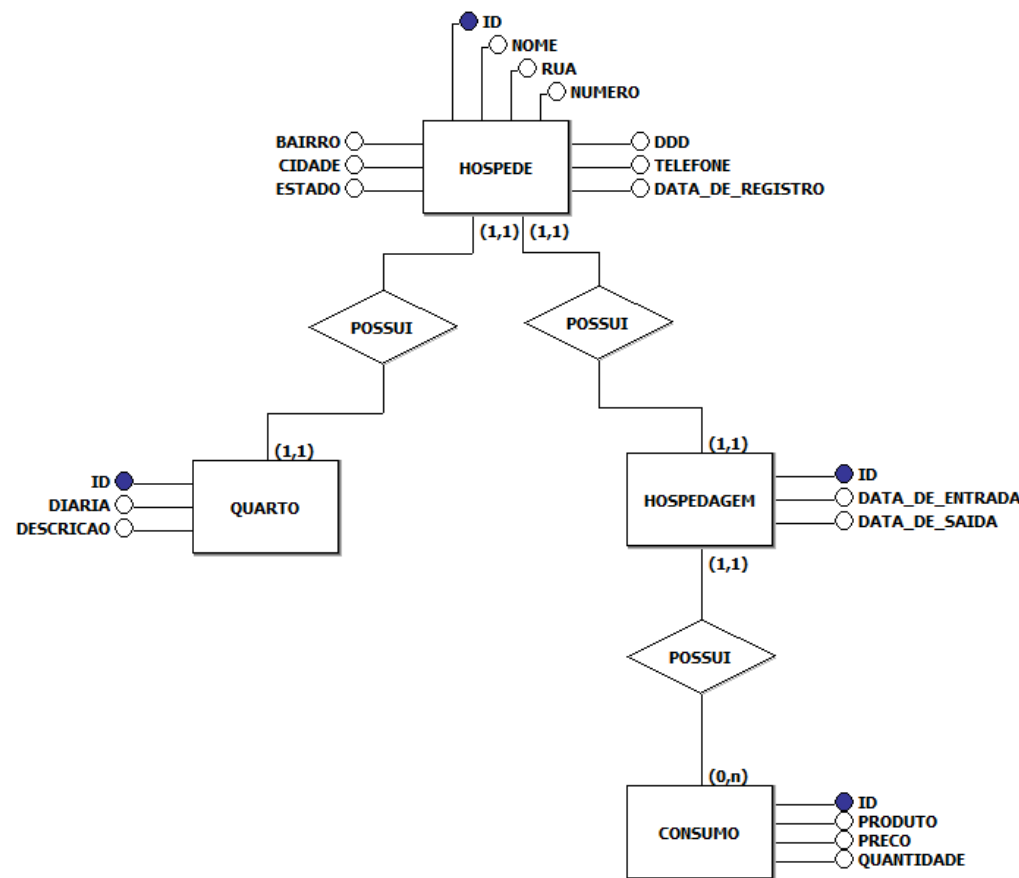
Gerenciamento de Hotéis

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

```

1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <xs:schema version="1.0"
3    xmlns:xs="http://www.w3.org/2001/XMLSchema">
4    <xs:complexType name="hospedagem">
5      <xs:sequence>
6        <xs:element name="dataDeEntrada" type="xs:dateTime" minOccurs="0"/>
7        <xs:element name="dataDeSaida" type="xs:dateTime" minOccurs="0"/>
8        <xs:element name="hospede" type="hospede" minOccurs="0"/>
9        <xs:element name="id" type="xs:long" minOccurs="0"/>
10       <xs:element name="quarto" type="quarto" minOccurs="0"/>
11     </xs:sequence>
12   </xs:complexType>
13   <xs:complexType name="hospede">
14     <xs:sequence>
15       <xs:element name="bairro" type="xs:string" minOccurs="0"/>
16       <xs:element name="cidade" type="xs:string" minOccurs="0"/>
17       <xs:element name="dataDeRegistro" type="xs:dateTime" minOccurs="0"/>
18       <xs:element name="ddd" type="xs:int" minOccurs="0"/>
19       <xs:element name="estado" type="xs:string" minOccurs="0"/>
20       <xs:element name="id" type="xs:long" minOccurs="0"/>
21       <xs:element name="nome" type="xs:string" minOccurs="0"/>
22       <xs:element name="numero" type="xs:int" minOccurs="0"/>
23       <xs:element name="rua" type="xs:string" minOccurs="0"/>
24       <xs:element name="telefone" type="xs:int" minOccurs="0"/>
25     </xs:sequence>
26   </xs:complexType>
27   <xs:complexType name="quarto">
28     <xs:sequence>
29       <xs:element name="descricao" type="xs:string" minOccurs="0"/>
30       <xs:element name="diaria" type="xs:double" minOccurs="0"/>
31       <xs:element name="id" type="xs:long" minOccurs="0"/>
32     </xs:sequence>
33   </xs:complexType>
34   <xs:complexType name="consumo">
35     <xs:sequence>
36       <xs:element name="dataDoConsumo" type="xs:dateTime" minOccurs="0"/>
37       <xs:element name="hospedagem" type="hospedagem" minOccurs="0"/>
38       <xs:element name="id" type="xs:long" minOccurs="0"/>
39       <xs:element name="preco" type="xs:double" minOccurs="0"/>
40       <xs:element name="produto" type="xs:string" minOccurs="0"/>
41       <xs:element name="quantidade" type="xs:int" minOccurs="0"/>
42     </xs:sequence>
43   </xs:complexType>
44 </xs:schema>

```

JSON / XML (Hospede)

```
1 {  
2   "id": 1,  
3   "nome": "Leonardo Carmona",  
4   "rua": "Av. Unisinos",  
5   "numero": 950,  
6   "bairro": "Cristo Rei",  
7   "cidade": "São Leopoldo",  
8   "estado": "RS",  
9   "ddd": 51,  
10  "telefone": 35911112,  
11  "dataDeRegistro": 1490473591673  
12 }
```

```
1 <Hospede>  
2   <id>1</id>  
3   <nome>Leonardo Carmona</nome>  
4   <rua>Av. Unisinos</rua>  
5   <numero>950</numero>  
6   <bairro>Cristo Rei</bairro>  
7   <cidade>São Leopoldo</cidade>  
8   <estado>RS</estado>  
9   <ddd>51</ddd>  
10  <telefone>35911112</telefone>  
11  <dataDeRegistro>1490473591673</dataDeRegistro>  
12 </Hospede>
```

JSON / XML (Quarto)

```
1 {  
2   "id": 1,  
3   "diaria": 50,  
4   "descricao": "Quarto 1"  
5 }
```

```
1 <Quarto>  
2   <id>1</id>  
3   <diaria>50.0</diaria>  
4   <descricao>Quarto 1</descricao>  
5 </Quarto>
```

JSON / XML (Hospedagem)

```
1 {  
2   "id": 1,  
3   "dataDeEntrada": 1490473591715,  
4   "dataDeSaida": null,  
5   "hospede": {  
6     "id": 1,  
7     "nome": "Leonardo Carmona",  
8     "rua": "Av. Unisinos",  
9     "numero": 950,  
10    "bairro": "Cristo Rei",  
11    "cidade": "São Leopoldo",  
12    "estado": "RS",  
13    "ddd": 51,  
14    "telefone": 35911112,  
15    "dataDeRegistro": 1490473591673  
16  },  
17  "quarto": {  
18    "id": 1,  
19    "diaria": 50,  
20    "descricao": "Quarto 1"  
21  }  
22 }
```

```
1 <Hospedagem>  
2   <id>1</id>  
3   <dataDeEntrada>1490473591715</dataDeEntrada>  
4   <dataDeSaida/>  
5   <hospede>  
6     <id>1</id>  
7     <nome>Leonardo Carmona</nome>  
8     <rua>Av. Unisinos</rua>  
9     <numero>950</numero>  
10    <bairro>Cristo Rei</bairro>  
11    <cidade>São Leopoldo</cidade>  
12    <estado>RS</estado>  
13    <ddd>51</ddd>  
14    <telefone>35911112</telefone>  
15    <dataDeRegistro>1490473591673</dataDeRegistro>  
16   </hospede>  
17   <quarto>  
18     <id>1</id>  
19     <diaria>50.0</diaria>  
20     <descricao>Quarto 1</descricao>  
21   </quarto>  
22 </Hospedagem>
```

JSON / XML (Consumo)

```

1 {
2   "id": 1,
3   "produto": "Coca-Cola Lata",
4   "preco": 4.5,
5   "quantidade": 3,
6   "dataDoConsumo": 1490473591716,
7   "hospedagem": {
8     "id": 1,
9     "dataDeEntrada": 1490473591715,
10    "dataDeSaida": null,
11    "hospede": {
12      "id": 1,
13      "nome": "Leonardo Carmona",
14      "rua": "Av. Unisinos",
15      "numero": 950,
16      "bairro": "Cristo Rei",
17      "cidade": "São Leopoldo",
18      "estado": "RS",
19      "ddd": 51,
20      "telefone": 35911112,
21      "dataDeRegistro": 1490473591673
22    },
23    "quarto": {
24      "id": 1,
25      "diaria": 50,
26      "descricao": "Quarto 1"
27    }
28  }
29 }

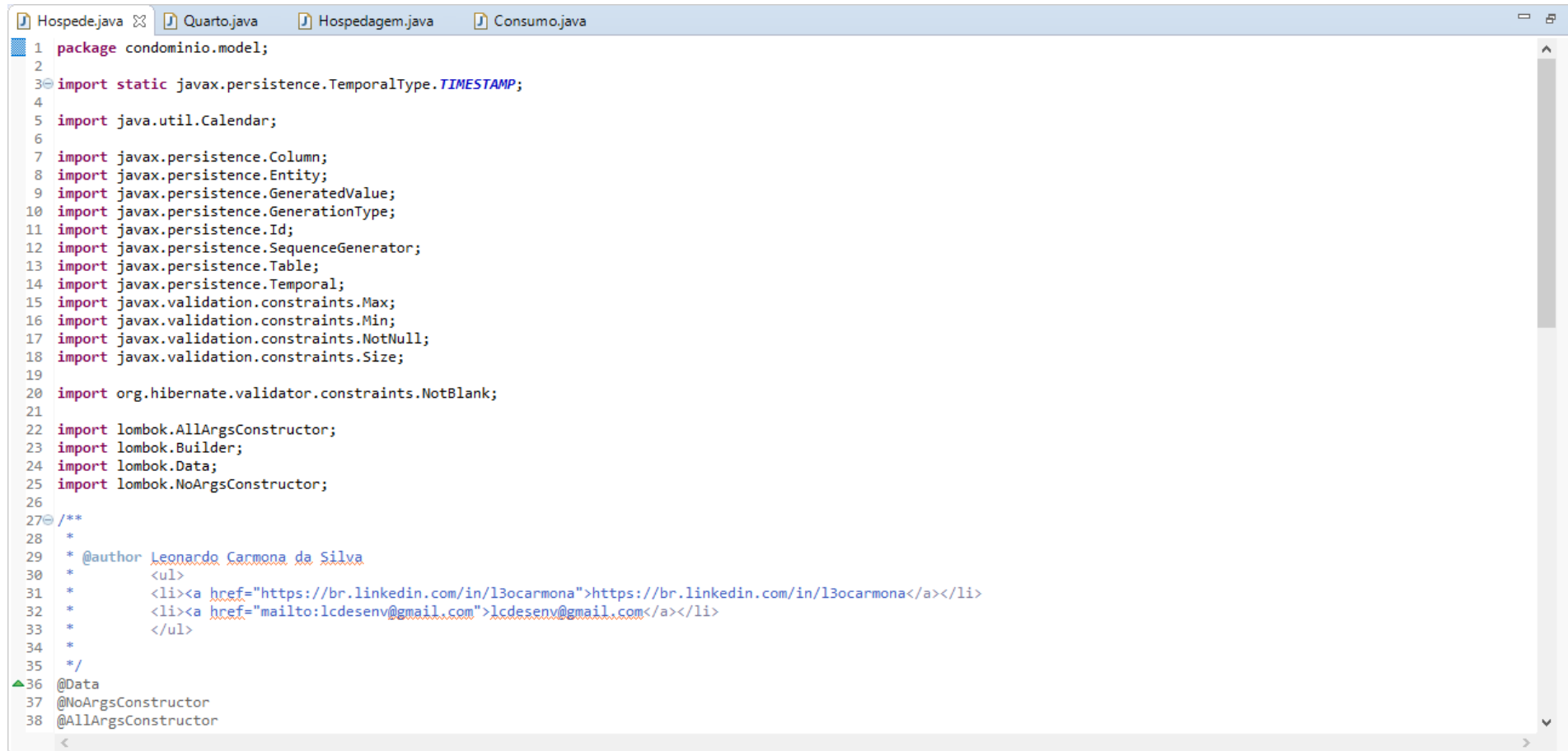
```

```

1 <Consumo>
2   <id>1</id>
3   <produto>Coca-Cola Lata</produto>
4   <preco>4.5</preco>
5   <quantidade>3</quantidade>
6   <dataDoConsumo>1490473591716</dataDoConsumo>
7   <hospedagem>
8     <id>1</id>
9     <dataDeEntrada>1490473591715</dataDeEntrada>
10    <dataDeSaida/>
11    <hospede>
12      <id>1</id>
13      <nome>Leonardo Carmona</nome>
14      <rua>Av. Unisinos</rua>
15      <numero>950</numero>
16      <bairro>Cristo Rei</bairro>
17      <cidade>São Leopoldo</cidade>
18      <estado>RS</estado>
19      <ddd>51</ddd>
20      <telefone>35911112</telefone>
21      <dataDeRegistro>1490473591673</dataDeRegistro>
22    </hospede>
23    <quarto>
24      <id>1</id>
25      <diaria>50.0</diaria>
26      <descricao>Quarto 1</descricao>
27    </quarto>
28  </hospedagem>
29 </Consumo>

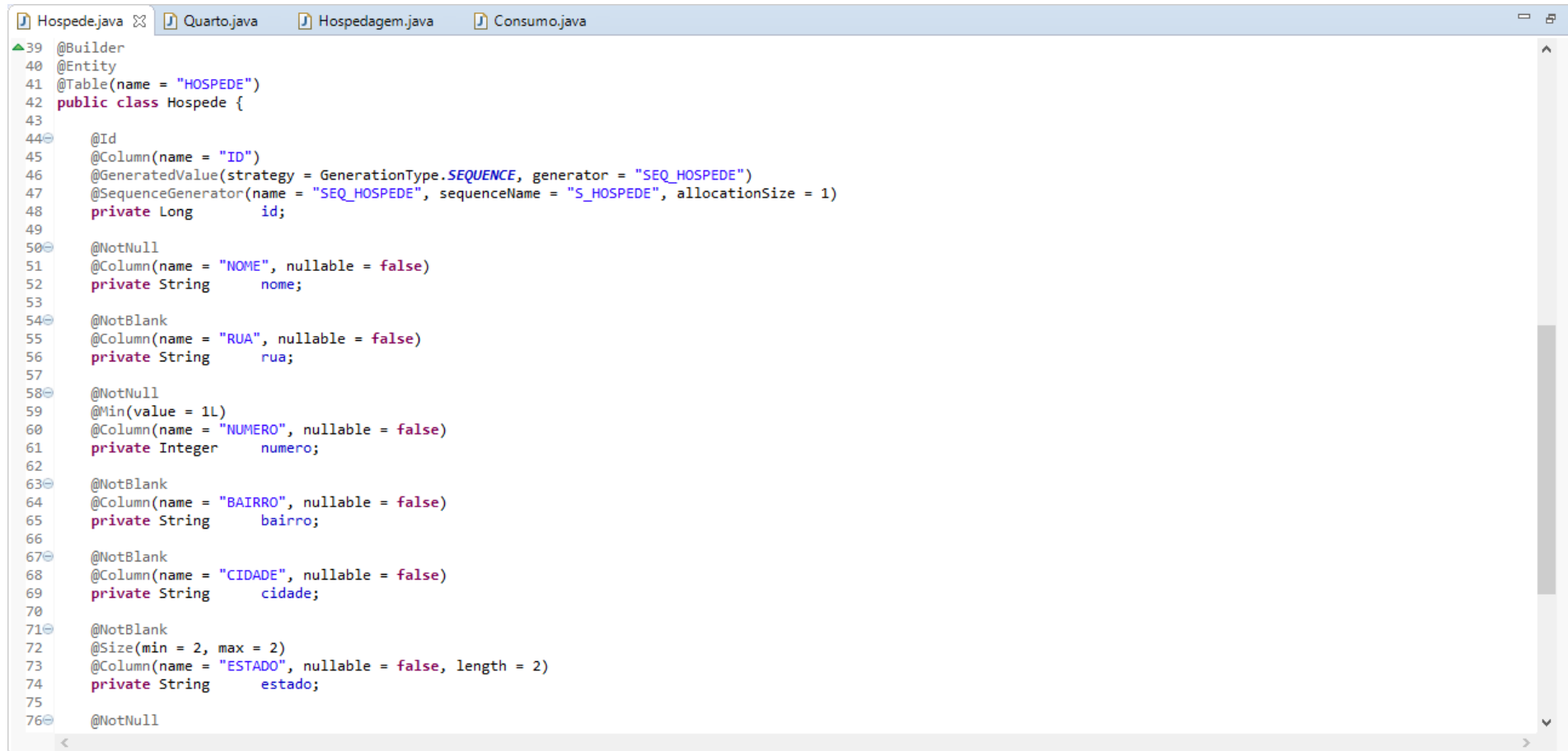
```


Mapeamento OR (Hospede) 1/3



```
1 package condominio.model;
2
3 import static javax.persistence.TemporalType.TIMESTAMP;
4
5 import java.util.Calendar;
6
7 import javax.persistence.Column;
8 import javax.persistence.Entity;
9 import javax.persistence.GeneratedValue;
10 import javax.persistence.GenerationType;
11 import javax.persistence.Id;
12 import javax.persistence.SequenceGenerator;
13 import javax.persistence.Table;
14 import javax.persistence.Temporal;
15 import javax.validation.constraints.Max;
16 import javax.validation.constraints.Min;
17 import javax.validation.constraints.NotNull;
18 import javax.validation.constraints.Size;
19
20 import org.hibernate.validator.constraints.NotBlank;
21
22 import lombok.AllArgsConstructor;
23 import lombok.Builder;
24 import lombok.Data;
25 import lombok.NoArgsConstructor;
26
27 /**
28  *
29  * @author Leonardo Carmona da Silva
30  *
31  * <ul>
32  * <li><a href="https://br.linkedin.com/in/l3ocarmona">https://br.linkedin.com/in/l3ocarmona</a></li>
33  * <li><a href="mailto:lcdesenv@gmail.com">lcdesenv@gmail.com</a></li>
34  * </ul>
35  */
36 @Data
37 @NoArgsConstructor
38 @AllArgsConstructor
```

Mapeamento OR (Hospede) 2/3



```
39 @Builder
40 @Entity
41 @Table(name = "HOSPEDE")
42 public class Hospede {
43
44     @Id
45     @Column(name = "ID")
46     @GeneratedValue(strategy = GenerationType.SEQUENCE, generator = "SEQ_HOSPEDE")
47     @SequenceGenerator(name = "SEQ_HOSPEDE", sequenceName = "S_HOSPEDE", allocationSize = 1)
48     private Long id;
49
50     @NotNull
51     @Column(name = "NOME", nullable = false)
52     private String nome;
53
54     @NotBlank
55     @Column(name = "RUA", nullable = false)
56     private String rua;
57
58     @NotNull
59     @Min(value = 1L)
60     @Column(name = "NUMERO", nullable = false)
61     private Integer numero;
62
63     @NotBlank
64     @Column(name = "BAIRRO", nullable = false)
65     private String bairro;
66
67     @NotBlank
68     @Column(name = "CIDADE", nullable = false)
69     private String cidade;
70
71     @NotBlank
72     @Size(min = 2, max = 2)
73     @Column(name = "ESTADO", nullable = false, length = 2)
74     private String estado;
75
76     @NotNull
```

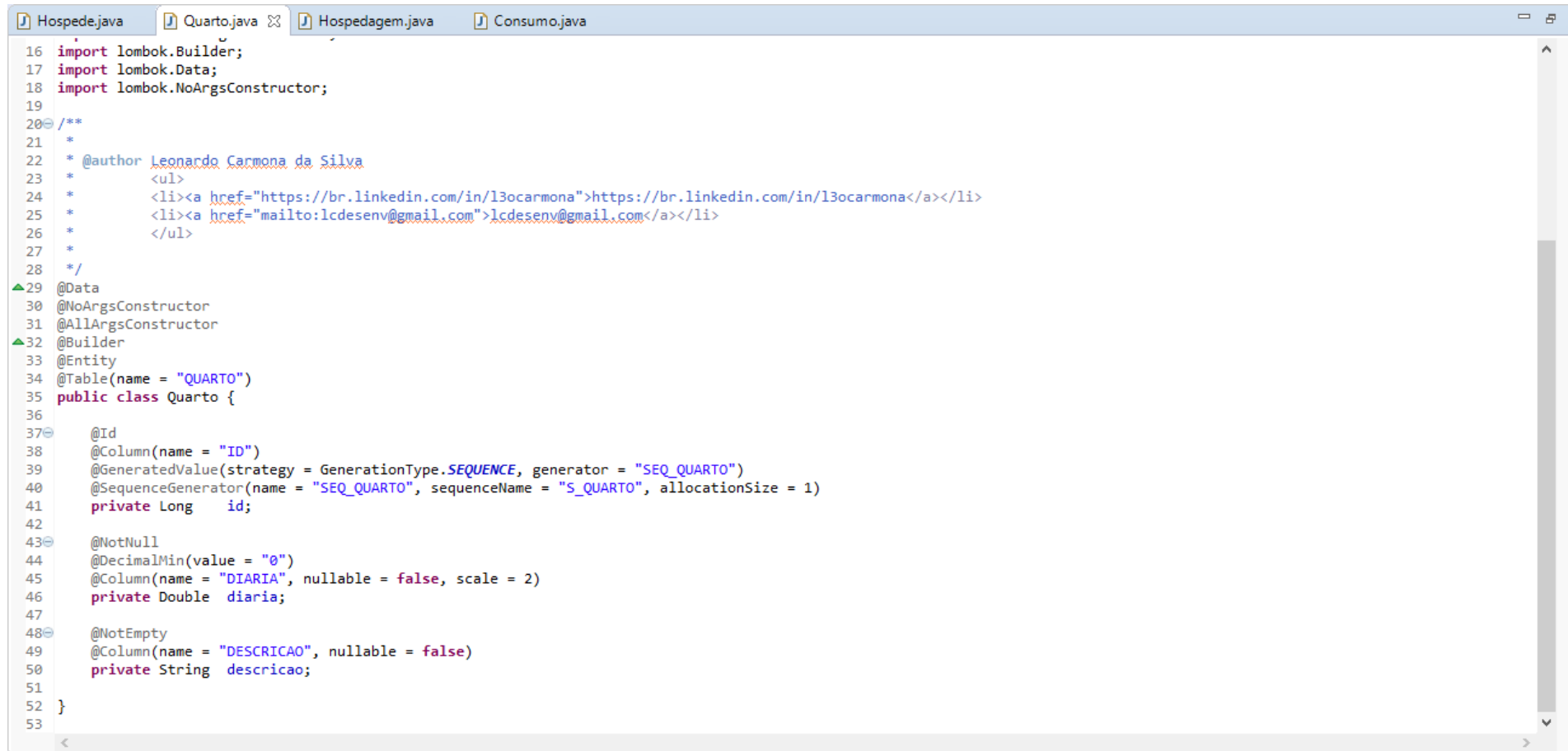
Mapeamento OR (Hospede) 3/3

```
Hospede.java  Quarto.java  Hospedagem.java  Consumo.java
55  ~
56  @Column(name = "RUA", nullable = false)
57  private String rua;
58
59  @NotNull
60  @Min(value = 1L)
61  @Column(name = "NUMERO", nullable = false)
62  private Integer numero;
63
64  @NotBlank
65  @Column(name = "BAIRRO", nullable = false)
66  private String bairro;
67
68  @NotBlank
69  @Column(name = "CIDADE", nullable = false)
70  private String cidade;
71
72  @NotBlank
73  @Size(min = 2, max = 2)
74  @Column(name = "ESTADO", nullable = false, length = 2)
75  private String estado;
76
77  @NotNull
78  @Min(value = 1L)
79  @Max(value = 99L)
80  @Column(name = "DDD", nullable = false)
81  private Integer ddd;
82
83  @NotBlank
84  @Size(min = 10000000, max = 999999999)
85  @Column(name = "TELEFONE", nullable = false)
86  private Integer telefone;
87
88  @NotNull
89  @Temporal(TIMESTAMP)
90  @Column(name = "DATA_DE_REGISTRO", nullable = false)
91  private Calendar dataDeRegistro = Calendar.getInstance();
92 }
```

Mapeamento OR (Quarto) 1/2

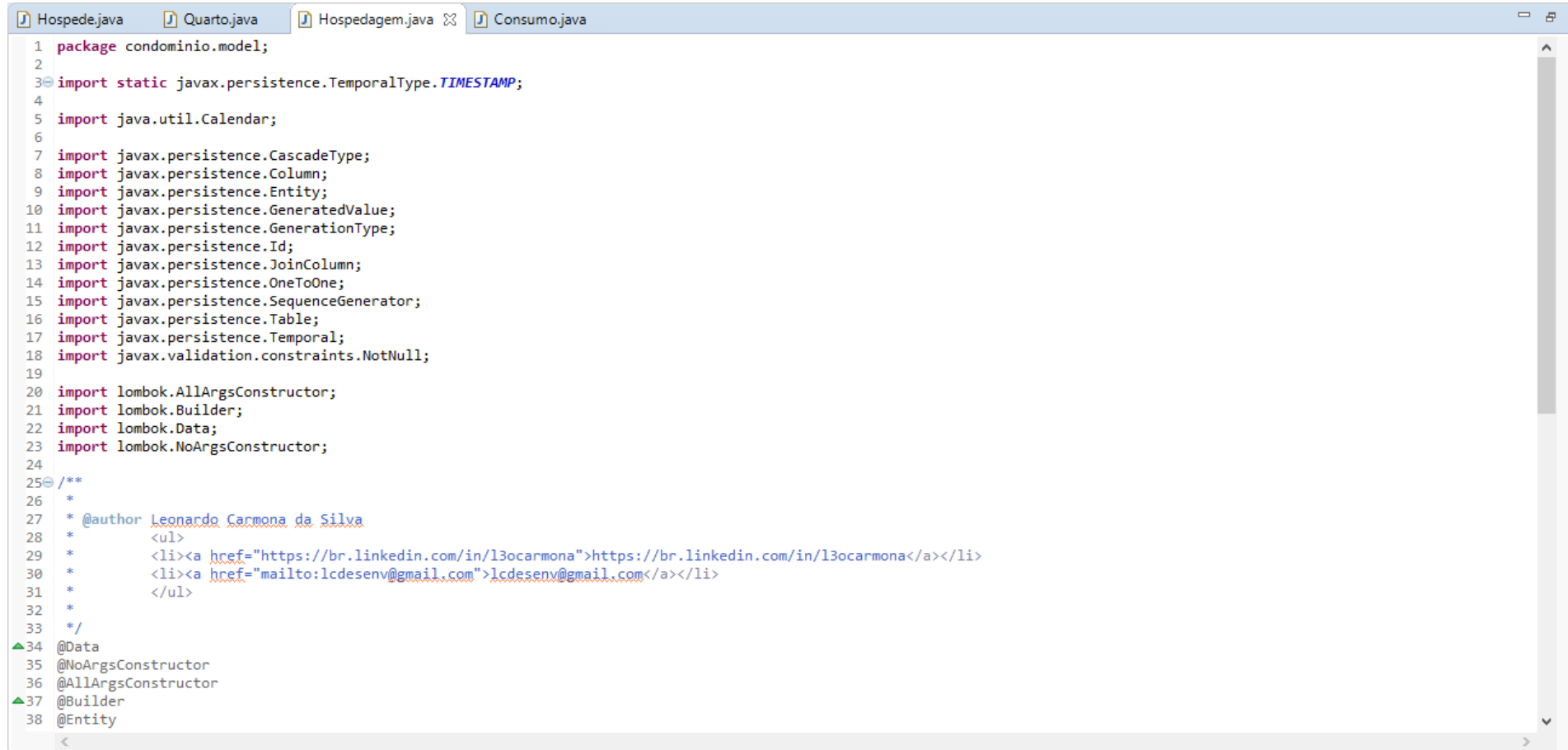
```
Hospede.java  Quarto.java  Hospedagem.java  Consumo.java
1 package condominio.model;
2
3 import javax.persistence.Column;
4 import javax.persistence.Entity;
5 import javax.persistence.GeneratedValue;
6 import javax.persistence.GenerationType;
7 import javax.persistence.Id;
8 import javax.persistence.SequenceGenerator;
9 import javax.persistence.Table;
10 import javax.validation.constraints.DecimalMin;
11 import javax.validation.constraints.NotNull;
12
13 import org.hibernate.validator.constraints.NotEmpty;
14
15 import lombok.AllArgsConstructor;
16 import lombok.Builder;
17 import lombok.Data;
18 import lombok.NoArgsConstructor;
19
20 /**
21  *
22  * @author Leonardo Carmona da Silva
23  * <ul>
24  * <li><a href="https://br.linkedin.com/in/l3ocarmona">https://br.linkedin.com/in/l3ocarmona</a></li>
25  * <li><a href="mailto:lcdesenv@gmail.com">lcdesenv@gmail.com</a></li>
26  * </ul>
27  *
28  */
29 @Data
30 @NoArgsConstructor
31 @AllArgsConstructor
32 @Builder
33 @Entity
34 @Table(name = "QUARTO")
35 public class Quarto {
36
37     @Id
38     @Column(name = "ID")
```

Mapeamento OR (Quarto) 2/2



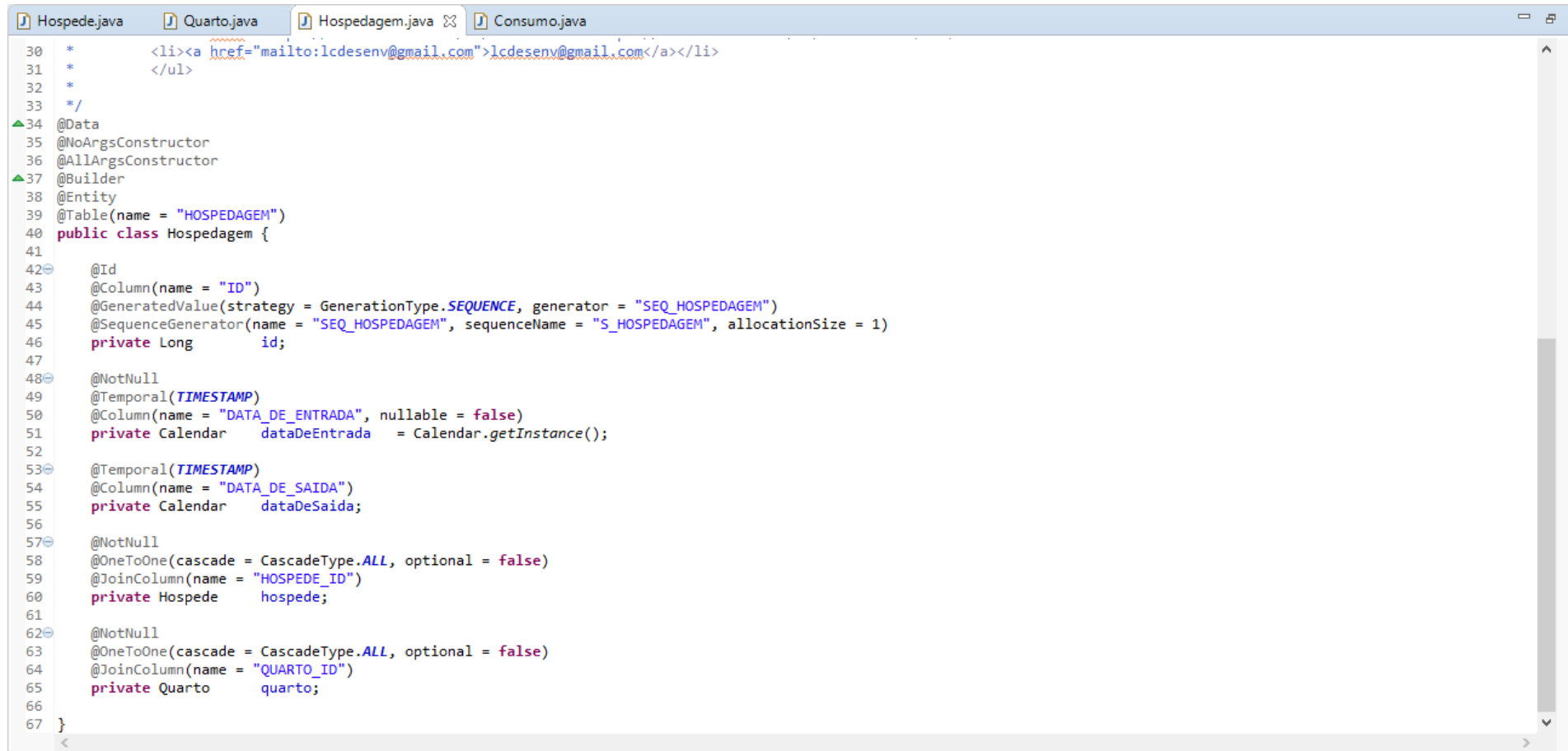
```
16 import lombok.Builder;
17 import lombok.Data;
18 import lombok.NoArgsConstructor;
19
20 /**
21  *
22  * @author Leonardo Carmona da Silva
23  *
24  * <li><a href="https://br.linkedin.com/in/l3ocarmona">https://br.linkedin.com/in/l3ocarmona</a></li>
25  * <li><a href="mailto:lcdesenv@gmail.com">lcdesenv@gmail.com</a></li>
26  *
27  * </ul>
28  */
29 @Data
30 @NoArgsConstructor
31 @AllArgsConstructor
32 @Builder
33 @Entity
34 @Table(name = "QUARTO")
35 public class Quarto {
36
37     @Id
38     @Column(name = "ID")
39     @GeneratedValue(strategy = GenerationType.SEQUENCE, generator = "SEQ_QUARTO")
40     @SequenceGenerator(name = "SEQ_QUARTO", sequenceName = "S_QUARTO", allocationSize = 1)
41     private Long id;
42
43     @NotNull
44     @DecimalMin(value = "0")
45     @Column(name = "DIARIA", nullable = false, scale = 2)
46     private Double diaria;
47
48     @NotEmpty
49     @Column(name = "DESCRICAO", nullable = false)
50     private String descricao;
51
52 }
53
```

Mapeamento OR (Hospedagem) 1/2



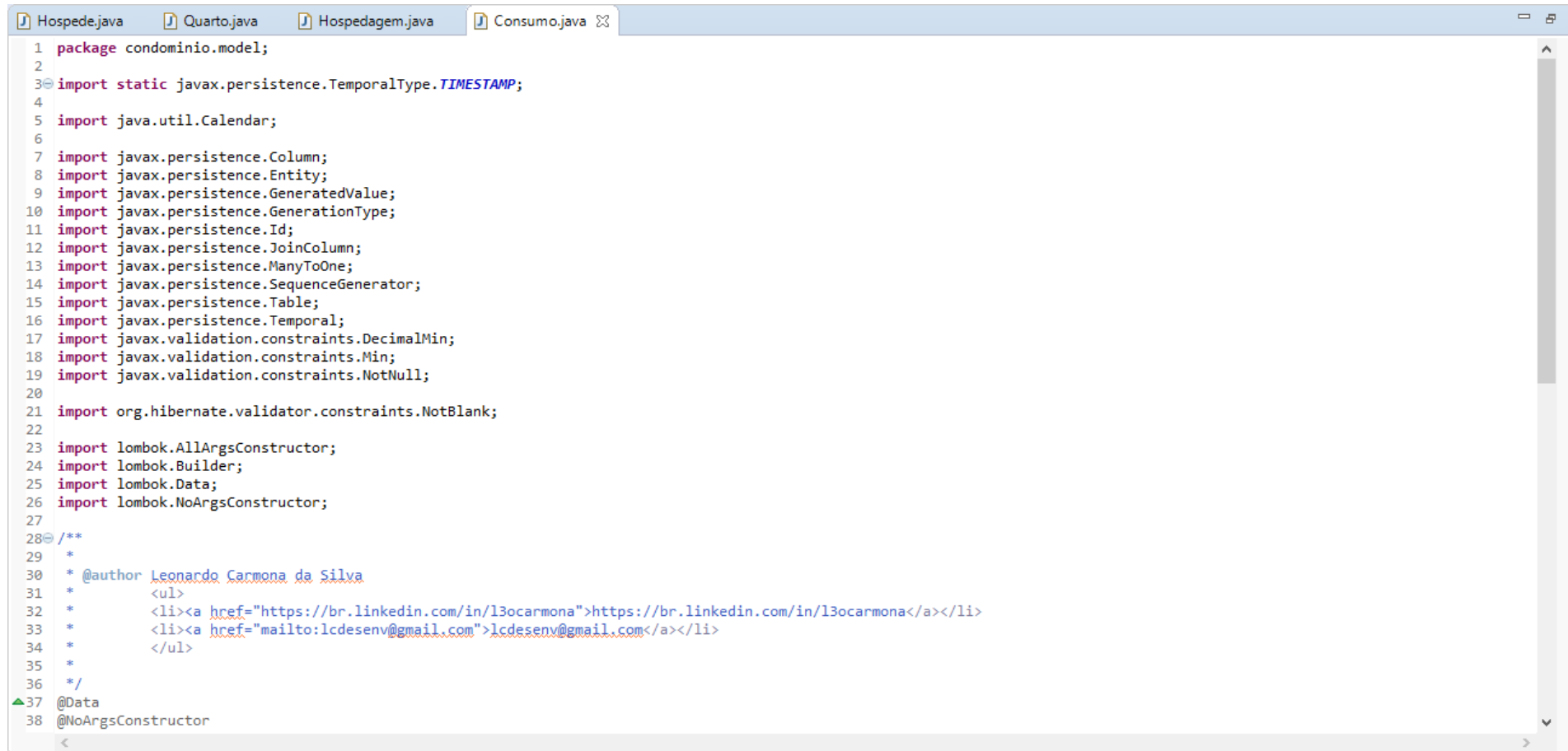
```
1 package condominio.model;
2
3 import static javax.persistence.TemporalType.TIMESTAMP;
4
5 import java.util.Calendar;
6
7 import javax.persistence.CascadeType;
8 import javax.persistence.Column;
9 import javax.persistence.Entity;
10 import javax.persistence.GeneratedValue;
11 import javax.persistence.GenerationType;
12 import javax.persistence.Id;
13 import javax.persistence.JoinColumn;
14 import javax.persistence.OneToOne;
15 import javax.persistence.SequenceGenerator;
16 import javax.persistence.Table;
17 import javax.persistence.Temporal;
18 import javax.validation.constraints.NotNull;
19
20 import lombok.AllArgsConstructor;
21 import lombok.Builder;
22 import lombok.Data;
23 import lombok.NoArgsConstructor;
24
25 /**
26  *
27  * @author Leonardo Carmona da Silva
28  *
29  * <ul>
30  * <li><a href="https://br.linkedin.com/in/l3ocarmona">https://br.linkedin.com/in/l3ocarmona</a></li>
31  * <li><a href="mailto:lcdesenv@gmail.com">lcdesenv@gmail.com</a></li>
32  * </ul>
33  */
34 @Data
35 @NoArgsConstructor
36 @AllArgsConstructor
37 @Builder
38 @Entity
```

Mapeamento OR (Hospedagem) 2/2



```
30 *      <li><a href="mailto:lcdesenv@gmail.com">lcdesenv@gmail.com</a></li>
31 *      </ul>
32 *
33 */
34 @Data
35 @NoArgsConstructor
36 @AllArgsConstructor
37 @Builder
38 @Entity
39 @Table(name = "HOSPEDAGEM")
40 public class Hospedagem {
41
42     @Id
43     @Column(name = "ID")
44     @GeneratedValue(strategy = GenerationType.SEQUENCE, generator = "SEQ_HOSPEDAGEM")
45     @SequenceGenerator(name = "SEQ_HOSPEDAGEM", sequenceName = "S_HOSPEDAGEM", allocationSize = 1)
46     private Long id;
47
48     @NotNull
49     @Temporal(TIMESTAMP)
50     @Column(name = "DATA_DE_ENTRADA", nullable = false)
51     private Calendar dataDeEntrada = Calendar.getInstance();
52
53     @Temporal(TIMESTAMP)
54     @Column(name = "DATA_DE_SAIDA")
55     private Calendar dataDeSaida;
56
57     @NotNull
58     @OneToOne(cascade = CascadeType.ALL, optional = false)
59     @JoinColumn(name = "HOSPEDE_ID")
60     private Hospede hospede;
61
62     @NotNull
63     @OneToOne(cascade = CascadeType.ALL, optional = false)
64     @JoinColumn(name = "QUARTO_ID")
65     private Quarto quarto;
66
67 }
```

Mapeamento OR (Consumo) 1/2

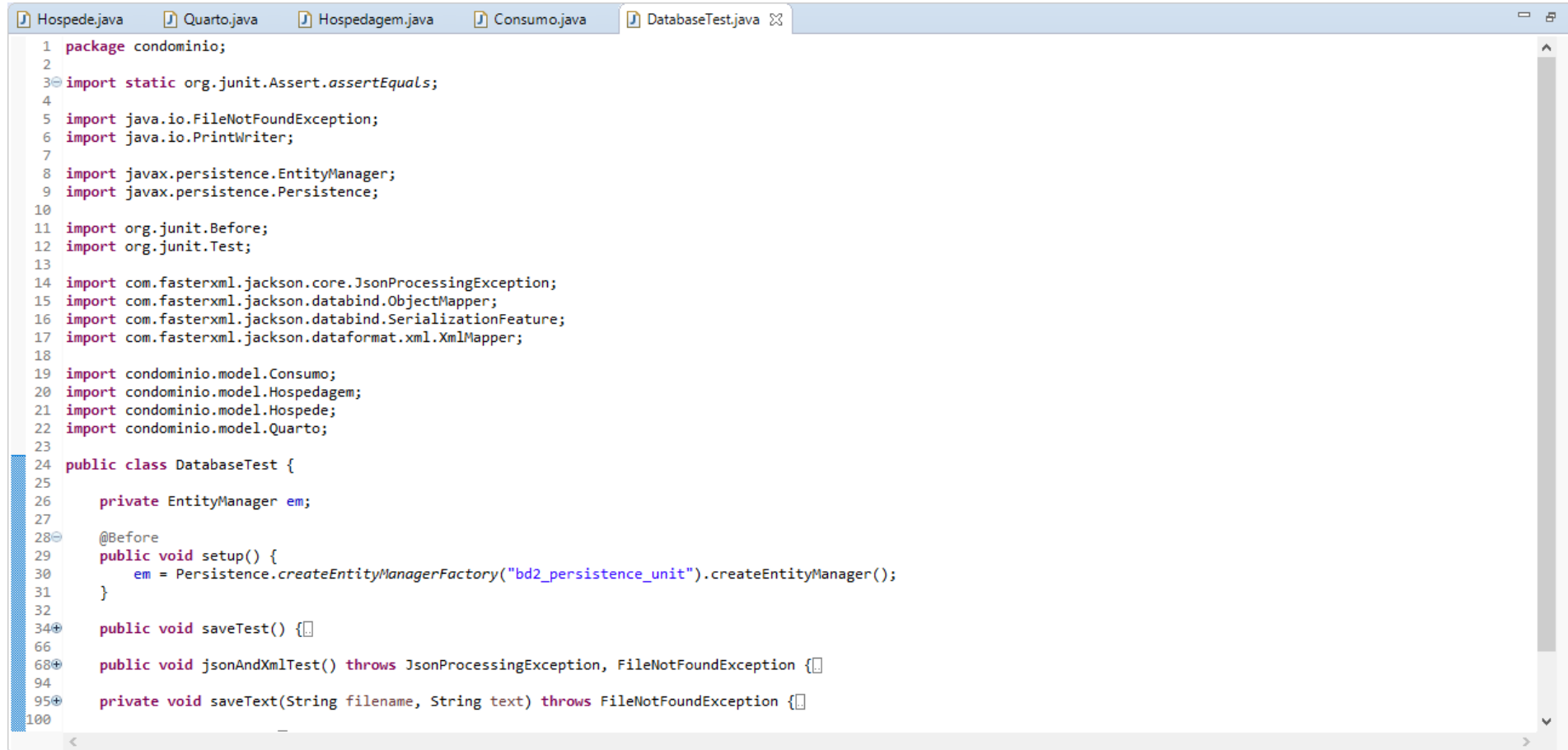


```
1 package condominio.model;
2
3 import static javax.persistence.TemporalType.TIMESTAMP;
4
5 import java.util.Calendar;
6
7 import javax.persistence.Column;
8 import javax.persistence.Entity;
9 import javax.persistence.GeneratedValue;
10 import javax.persistence.GenerationType;
11 import javax.persistence.Id;
12 import javax.persistence.JoinColumn;
13 import javax.persistence.ManyToOne;
14 import javax.persistence.SequenceGenerator;
15 import javax.persistence.Table;
16 import javax.persistence.Temporal;
17 import javax.validation.constraints.DecimalMin;
18 import javax.validation.constraints.Min;
19 import javax.validation.constraints.NotNull;
20
21 import org.hibernate.validator.constraints.NotBlank;
22
23 import lombok.AllArgsConstructor;
24 import lombok.Builder;
25 import lombok.Data;
26 import lombok.NoArgsConstructor;
27
28 /**
29 *
30 * @author Leonardo Carmona da Silva
31 *
32 * <li><a href="https://br.linkedin.com/in/l3ocarmona">https://br.linkedin.com/in/l3ocarmona</a></li>
33 * <li><a href="mailto:lcdesenv@gmail.com">lcdesenv@gmail.com</a></li>
34 *
35 * </ul>
36 */
37 @Data
38 @NoArgsConstructor
```


Mapeamento OR (Consumo) 2/2

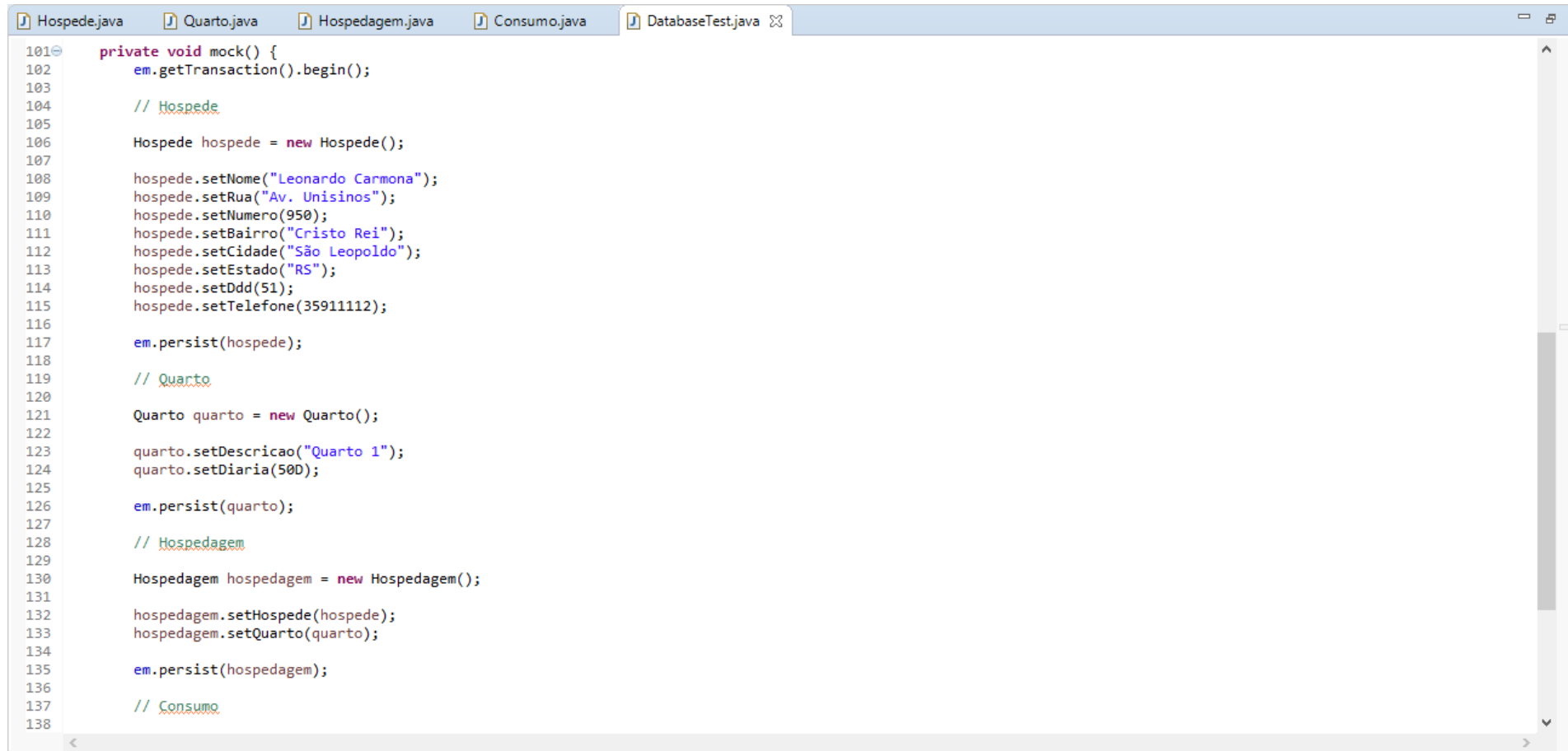
```
Hospede.java  Quarto.java  Hospedagem.java  Consumo.java
38 @NoArgsConstructor
39 @AllArgsConstructor
40 @Builder
41 @Entity
42 @Table(name = "CONSUMO")
43 public class Consumo {
44
45     @Id
46     @Column(name = "ID")
47     @GeneratedValue(strategy = GenerationType.SEQUENCE, generator = "SEQ_CONSUMO")
48     @SequenceGenerator(name = "SEQ_CONSUMO", sequenceName = "S_CONSUMO", allocationSize = 1)
49     private Long id;
50
51     @NotBlank
52     @Column(name = "PRODUTO", nullable = false)
53     private String produto;
54
55     @NotNull
56     @DecimalMin(value = "0")
57     @Column(name = "PRECO", nullable = false)
58     private Double preco;
59
60     @NotNull
61     @Min(value = 1L)
62     @Column(name = "QUANTIDADE", nullable = false)
63     private Integer quantidade;
64
65     @NotNull
66     @Column(name = "DATA_DO_CONSUMO", nullable = false)
67     @Temporal(TIMESTAMP)
68     private Calendar dataDoConsumo = Calendar.getInstance();
69
70     @NotNull
71     @JoinColumn(name = "HOSPEDAGEM_ID", nullable = false, updatable = false)
72     @ManyToOne(optional = false)
73     private Hospedagem hospedagem;
74
75 }
```

DatabaseTest (Dependências e Setup)



```
1 package condominio;
2
3 import static org.junit.Assert.assertEquals;
4
5 import java.io.FileNotFoundException;
6 import java.io.PrintWriter;
7
8 import javax.persistence.EntityManager;
9 import javax.persistence.Persistence;
10
11 import org.junit.Before;
12 import org.junit.Test;
13
14 import com.fasterxml.jackson.core.JsonProcessingException;
15 import com.fasterxml.jackson.databind.ObjectMapper;
16 import com.fasterxml.jackson.databind.SerializationFeature;
17 import com.fasterxml.jackson.dataformat.xml.XmlMapper;
18
19 import condominio.model.Consumo;
20 import condominio.model.Hospedagem;
21 import condominio.model.Hospede;
22 import condominio.model.Quarto;
23
24 public class DatabaseTest {
25
26     private EntityManager em;
27
28     @Before
29     public void setup() {
30         em = Persistence.createEntityManagerFactory("bd2_persistence_unit").createEntityManager();
31     }
32
33     public void saveTest() {}
34
35     public void jsonAndXmlTest() throws JsonProcessingException, FileNotFoundException {}
36
37     private void saveText(String filename, String text) throws FileNotFoundException {}
38 }
```

DatabaseTest (População dos Objetos) 1/2



```
101 private void mock() {
102     em.getTransaction().begin();
103
104     // Hospede
105
106     Hospede hospede = new Hospede();
107
108     hospede.setNome("Leonardo Carmona");
109     hospede.setRua("Av. Unisinos");
110     hospede.setNumero(950);
111     hospede.setBairro("Cristo Rei");
112     hospede.setCidade("São Leopoldo");
113     hospede.setEstado("RS");
114     hospede.setDdd(51);
115     hospede.setTelefone(35911112);
116
117     em.persist(hospede);
118
119     // Quarto
120
121     Quarto quarto = new Quarto();
122
123     quarto.setDescricao("Quarto 1");
124     quarto.setDiaria(500);
125
126     em.persist(quarto);
127
128     // Hospedagem
129
130     Hospedagem hospedagem = new Hospedagem();
131
132     hospedagem.setHospede(hospede);
133     hospedagem.setQuarto(quarto);
134
135     em.persist(hospedagem);
136
137     // Consumo
138 }
```

DatabaseTest (População dos Objetos) 2/2

```
Hospede.java  Quarto.java  Hospedagem.java  Consumo.java  DatabaseTest.java ✕
113     hospede.setEstado("RS");
114     hospede.setDdd(51);
115     hospede.setTelefone(35911112);
116
117     em.persist(hospede);
118
119     // Quarto
120
121     Quarto quarto = new Quarto();
122
123     quarto.setDescricao("Quarto 1");
124     quarto.setDiaria(500);
125
126     em.persist(quarto);
127
128     // Hospedagem
129
130     Hospedagem hospedagem = new Hospedagem();
131
132     hospedagem.setHospede(hospede);
133     hospedagem.setQuarto(quarto);
134
135     em.persist(hospedagem);
136
137     // Consumo
138
139     Consumo consumo = new Consumo();
140
141     consumo.setProduto("Coca-Cola Lata");
142     consumo.setPreco(4.5);
143     consumo.setQuantidade(3);
144     consumo.setHospedagem(hospedagem);
145
146     em.persist(consumo);
147
148     em.getTransaction().commit();
149 }
150
```

Testes Unitários (setup)



```
5 import java.io.FileNotFoundException;
6 import java.io.PrintWriter;
7
8 import javax.persistence.EntityManager;
9 import javax.persistence.Persistence;
10
11 import org.junit.Before;
12 import org.junit.Test;
13
14 import com.fasterxml.jackson.core.JsonProcessingException;
15 import com.fasterxml.jackson.databind.ObjectMapper;
16 import com.fasterxml.jackson.databind.SerializationFeature;
17 import com.fasterxml.jackson.dataformat.xml.XmlMapper;
18
19 import condominio.model.Consumo;
20 import condominio.model.Hospedagem;
21 import condominio.model.Hospede;
22 import condominio.model.Quarto;
23
24 public class DatabaseTest {
25
26     private EntityManager em;
27
28     @Before
29     public void setup() {
30         em = Persistence.createEntityManagerFactory("bd2_persistence_unit").createEntityManager();
31     }
32
33     public void saveTest() {}
34
35     public void jsonAndXmlTest() throws JsonProcessingException, FileNotFoundException {}
36
37     private void saveText(String filename, String text) throws FileNotFoundException {}
38
39     private void mock() {}
40 }
```

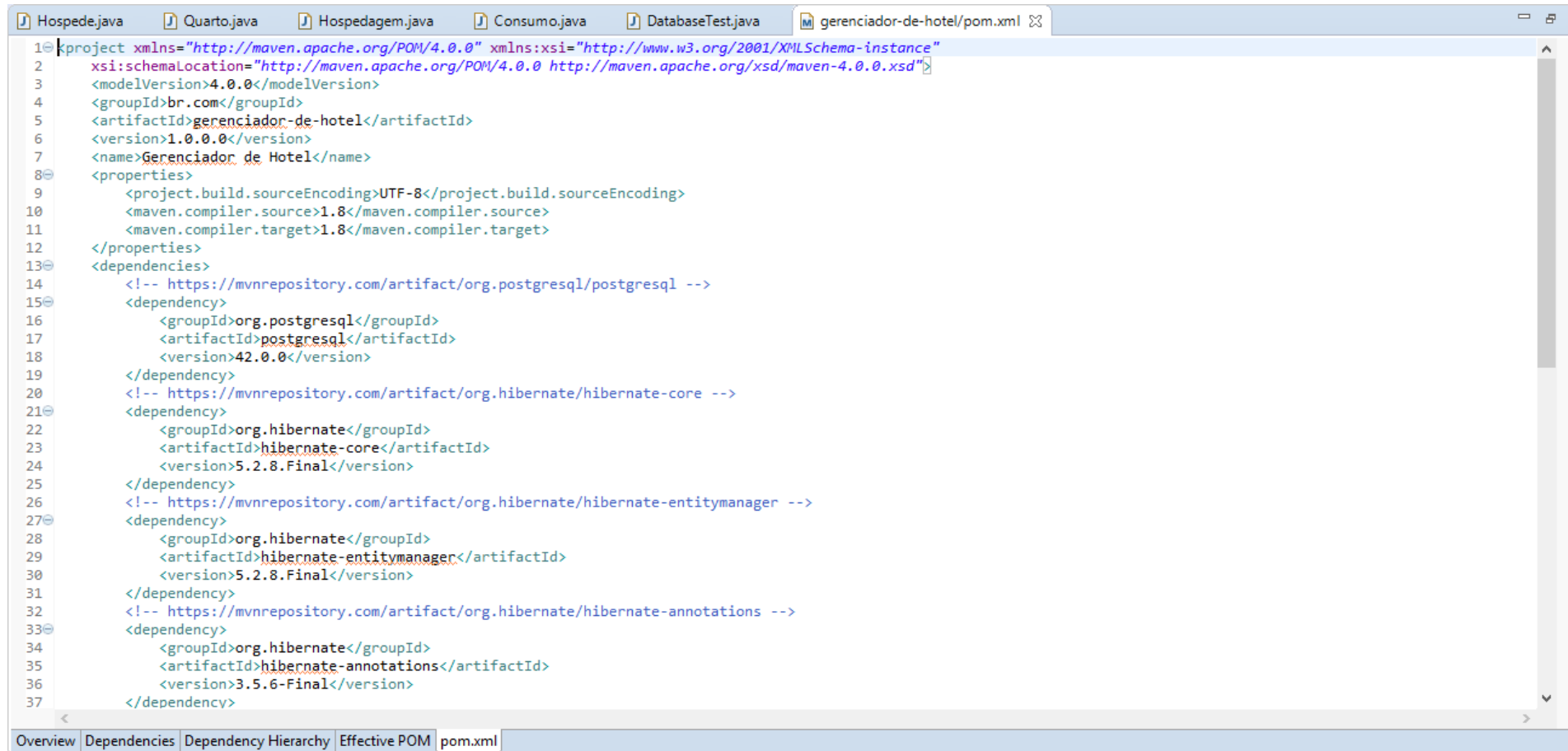
Testes Unitários (saveTest)

```
Hospede.java  Quarto.java  Hospedagem.java  Consumo.java  DatabaseTest.java ✕
28 @Before
29 public void setup() {
30     em = Persistence.createEntityManagerFactory("bd2_persistence_unit").createEntityManager();
31 }
32
33 @Test
34 public void saveTest() {
35     mock();
36
37     Hospede hospede = em.find(Hospede.class, 1L);
38
39     assertEquals("Leonardo Carmona", hospede.getNome());
40     assertEquals("Av. Unisinos", hospede.getRua());
41
42     assertEquals(new Integer(950), hospede.getNumero());
43     assertEquals("Cristo Rei", hospede.getBairro());
44     assertEquals("São Leopoldo", hospede.getCidade());
45     assertEquals("RS", hospede.getEstado());
46     assertEquals(Integer.valueOf(51), hospede.getDdd());
47     assertEquals(Integer.valueOf(35911112), hospede.getTelefone());
48
49     Quarto quarto = em.find(Quarto.class, 1L);
50
51     assertEquals("Quarto 1", quarto.getDescricao());
52     assertEquals(new Double(50), quarto.getDiaria());
53
54     Hospedagem hospedagem = em.find(Hospedagem.class, 1L);
55
56     assertEquals(hospede, hospedagem.getHospede());
57     assertEquals(quarto, hospedagem.getQuarto());
58
59     Consumo consumo = em.find(Consumo.class, 1L);
60
61     assertEquals("Coca-Cola Lata", consumo.getProduto());
62     assertEquals(new Double(4.5), consumo.getPreco());
63     assertEquals(new Integer(3), consumo.getQuantidade());
64     assertEquals(hospedagem, consumo.getHospedagem());
65 }
```

Testes Unitários (jsonAndXmlTest)

```
Hospede.java  Quarto.java  Hospedagem.java  Consumo.java  DatabaseTest.java X
67 @Test
68 public void jsonAndXmlTest() throws JsonProcessingException, FileNotFoundException {
69     mock();
70
71     ObjectMapper jsonMapper = new ObjectMapper();
72     XmlMapper xmlMapper = new XmlMapper();
73
74     jsonMapper.enable(SerializationFeature.INDENT_OUTPUT);
75     xmlMapper.enable(SerializationFeature.INDENT_OUTPUT);
76
77     Hospede hospede = em.find(Hospede.class, 1L);
78     Quarto quarto = em.find(Quarto.class, 1L);
79     Hospedagem hospedagem = em.find(Hospedagem.class, 1L);
80     Consumo consumo = em.find(Consumo.class, 1L);
81
82     saveText("target/hospede.json", jsonMapper.writeValueAsString(hospede));
83     saveText("target/hospede.xml", xmlMapper.writeValueAsString(hospede));
84
85     saveText("target/quarto.json", jsonMapper.writeValueAsString(quarto));
86     saveText("target/quarto.xml", xmlMapper.writeValueAsString(quarto));
87
88     saveText("target/hospedagem.json", jsonMapper.writeValueAsString(hospedagem));
89     saveText("target/hospedagem.xml", xmlMapper.writeValueAsString(hospedagem));
90
91     saveText("target/consumo.json", jsonMapper.writeValueAsString(consumo));
92     saveText("target/consumo.xml", xmlMapper.writeValueAsString(consumo));
93 }
94
95 private void saveText(String filename, String text) throws FileNotFoundException {
96     try (PrintWriter out = new PrintWriter(filename)) {
97         out.println(text);
98     }
99 }
100
101 private void mock() {}
150 }
151
152
```

pom.xml 1/3

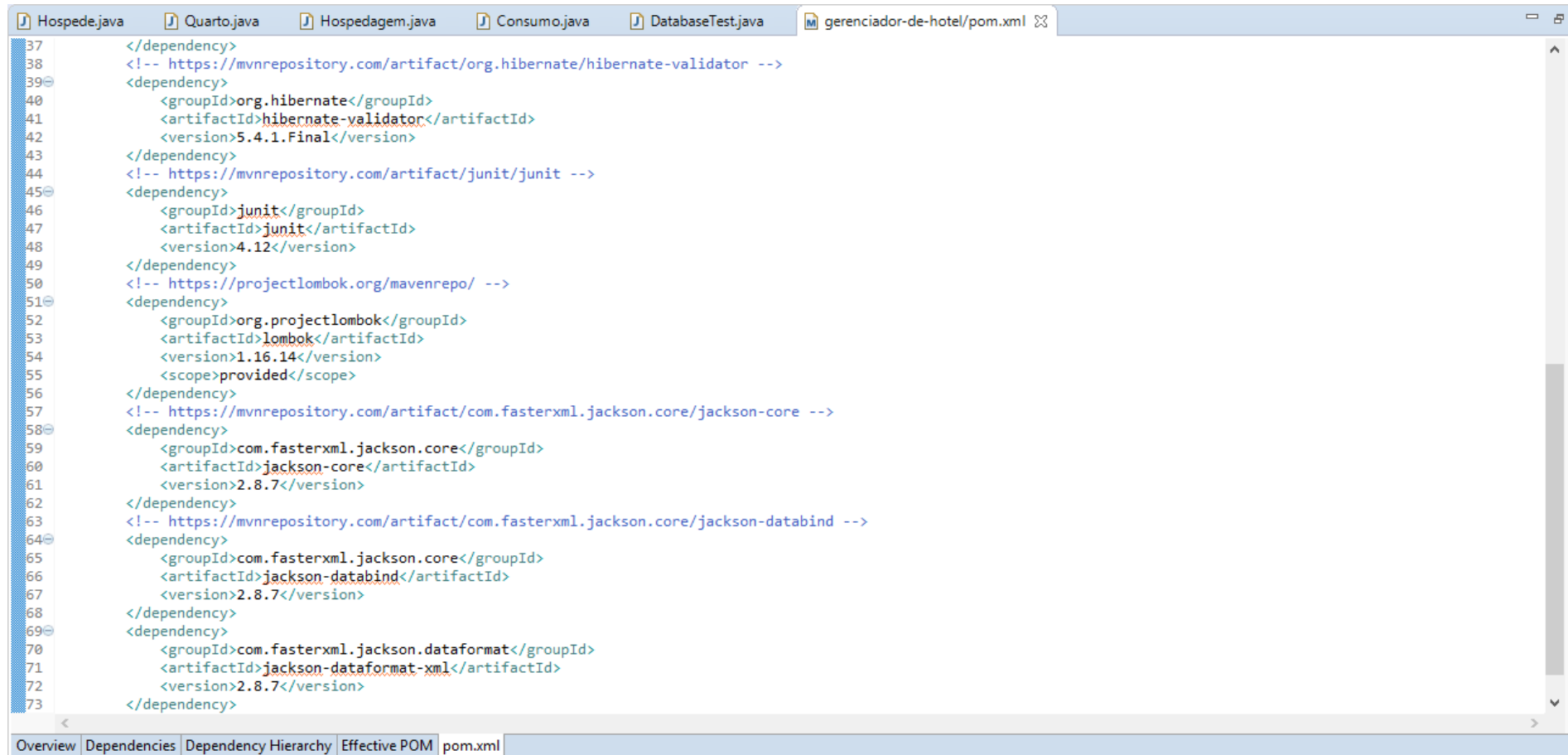


The screenshot shows an IDE with several tabs open: Hospede.java, Quarto.java, Hospedagem.java, Consumo.java, DatabaseTest.java, and gerenciador-de-hotel/pom.xml. The pom.xml file is selected and displays the following XML content:

```
1<?xml version="1.0" encoding="UTF-8" ?>
2<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
4  <modelVersion>4.0.0</modelVersion>
5  <groupId>br.com</groupId>
6  <artifactId>gerenciador-de-hotel</artifactId>
7  <version>1.0.0.0</version>
8  <name>Gerenciador de Hotel</name>
9  <properties>
10    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
11    <maven.compiler.source>1.8</maven.compiler.source>
12    <maven.compiler.target>1.8</maven.compiler.target>
13  </properties>
14  <dependencies>
15    <!-- https://mvnrepository.com/artifact/org.postgresql/postgresql -->
16    <dependency>
17      <groupId>org.postgresql</groupId>
18      <artifactId>postgresql</artifactId>
19      <version>42.0.0</version>
20    </dependency>
21    <!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
22    <dependency>
23      <groupId>org.hibernate</groupId>
24      <artifactId>hibernate-core</artifactId>
25      <version>5.2.8.Final</version>
26    </dependency>
27    <!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-entitymanager -->
28    <dependency>
29      <groupId>org.hibernate</groupId>
30      <artifactId>hibernate-entitymanager</artifactId>
31      <version>5.2.8.Final</version>
32    </dependency>
33    <!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-annotations -->
34    <dependency>
35      <groupId>org.hibernate</groupId>
36      <artifactId>hibernate-annotations</artifactId>
37      <version>3.5.6-Final</version>
38    </dependency>
39  </dependencies>
40</project>
```

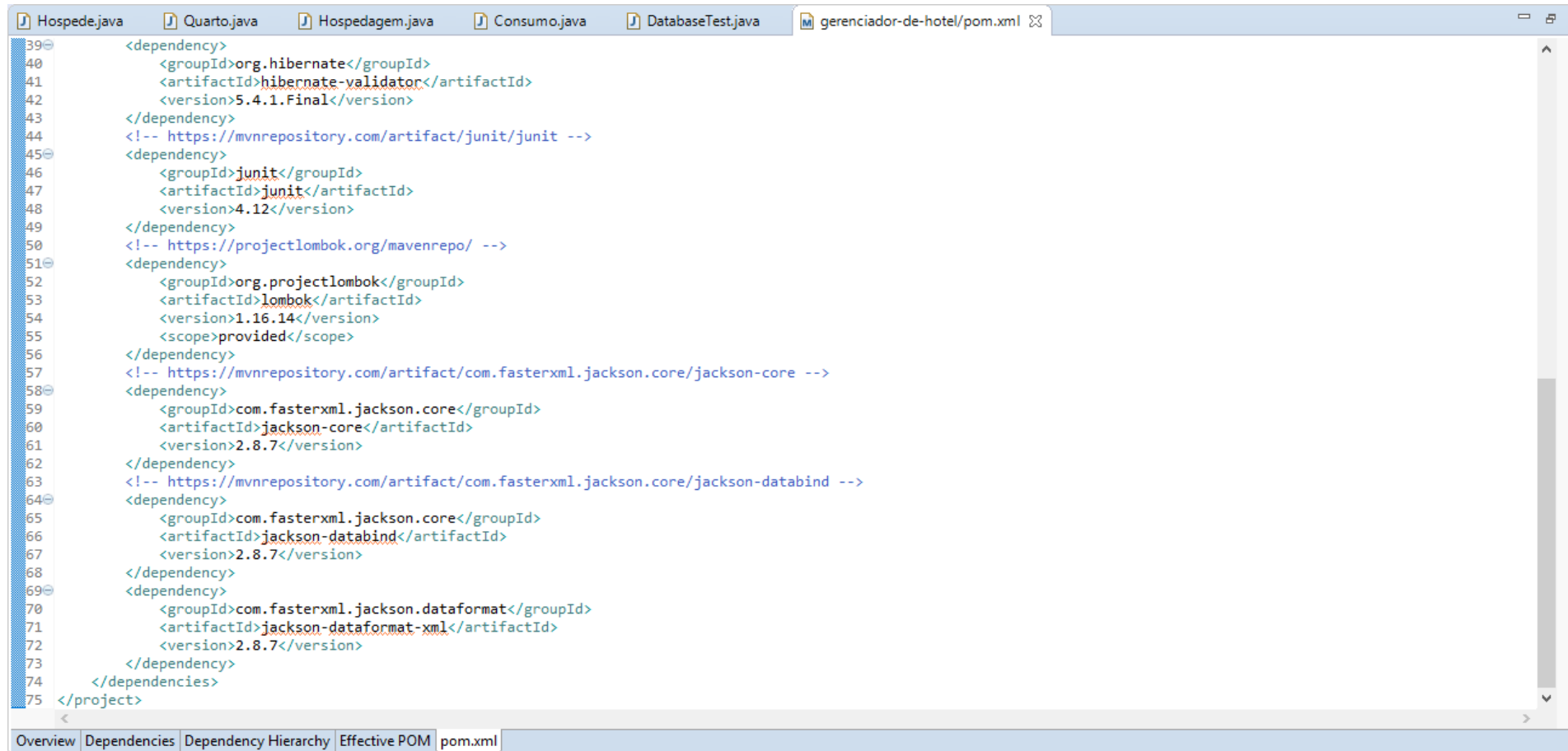
The IDE's bottom panel shows tabs for Overview, Dependencies, Dependency Hierarchy, Effective POM, and pom.xml, with pom.xml currently selected.

pom.xml 2/3



```
37     </dependency>
38     <!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-validator -->
39     <dependency>
40       <groupId>org.hibernate</groupId>
41       <artifactId>hibernate-validator</artifactId>
42       <version>5.4.1.Final</version>
43     </dependency>
44     <!-- https://mvnrepository.com/artifact/junit/junit -->
45     <dependency>
46       <groupId>junit</groupId>
47       <artifactId>junit</artifactId>
48       <version>4.12</version>
49     </dependency>
50     <!-- https://projectlombok.org/mavenrepo/ -->
51     <dependency>
52       <groupId>org.projectlombok</groupId>
53       <artifactId>lombok</artifactId>
54       <version>1.16.14</version>
55       <scope>provided</scope>
56     </dependency>
57     <!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-core -->
58     <dependency>
59       <groupId>com.fasterxml.jackson.core</groupId>
60       <artifactId>jackson-core</artifactId>
61       <version>2.8.7</version>
62     </dependency>
63     <!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-databind -->
64     <dependency>
65       <groupId>com.fasterxml.jackson.core</groupId>
66       <artifactId>jackson-databind</artifactId>
67       <version>2.8.7</version>
68     </dependency>
69     <dependency>
70       <groupId>com.fasterxml.jackson.dataformat</groupId>
71       <artifactId>jackson-dataformat-xml</artifactId>
72       <version>2.8.7</version>
73     </dependency>
```

pom.xml 3/3



The screenshot shows an IDE window with the file 'gerenciador-de-hotel/pom.xml' open. The file contains the following XML content:

```
39<dependency>
40  <groupId>org.hibernate</groupId>
41  <artifactId>hibernate-validator</artifactId>
42  <version>5.4.1.Final</version>
43</dependency>
44<!-- https://mvnrepository.com/artifact/junit/junit -->
45<dependency>
46  <groupId>junit</groupId>
47  <artifactId>junit</artifactId>
48  <version>4.12</version>
49</dependency>
50<!-- https://projectlombok.org/mavenrepo/ -->
51<dependency>
52  <groupId>org.projectlombok</groupId>
53  <artifactId>lombok</artifactId>
54  <version>1.16.14</version>
55  <scope>provided</scope>
56</dependency>
57<!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-core -->
58<dependency>
59  <groupId>com.fasterxml.jackson.core</groupId>
60  <artifactId>jackson-core</artifactId>
61  <version>2.8.7</version>
62</dependency>
63<!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-databind -->
64<dependency>
65  <groupId>com.fasterxml.jackson.core</groupId>
66  <artifactId>jackson-databind</artifactId>
67  <version>2.8.7</version>
68</dependency>
69<dependency>
70  <groupId>com.fasterxml.jackson.dataformat</groupId>
71  <artifactId>jackson-dataformat-xml</artifactId>
72  <version>2.8.7</version>
73</dependency>
74</dependencies>
75</project>
```

The IDE interface includes tabs for 'Hospede.java', 'Quarto.java', 'Hospedagem.java', 'Consumo.java', 'DatabaseTest.java', and 'gerenciador-de-hotel/pom.xml'. At the bottom, there is a navigation bar with tabs for 'Overview', 'Dependencies', 'Dependency Hierarchy', 'Effective POM', and 'pom.xml'.

Conclusão

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