

# Casos de Teste 2

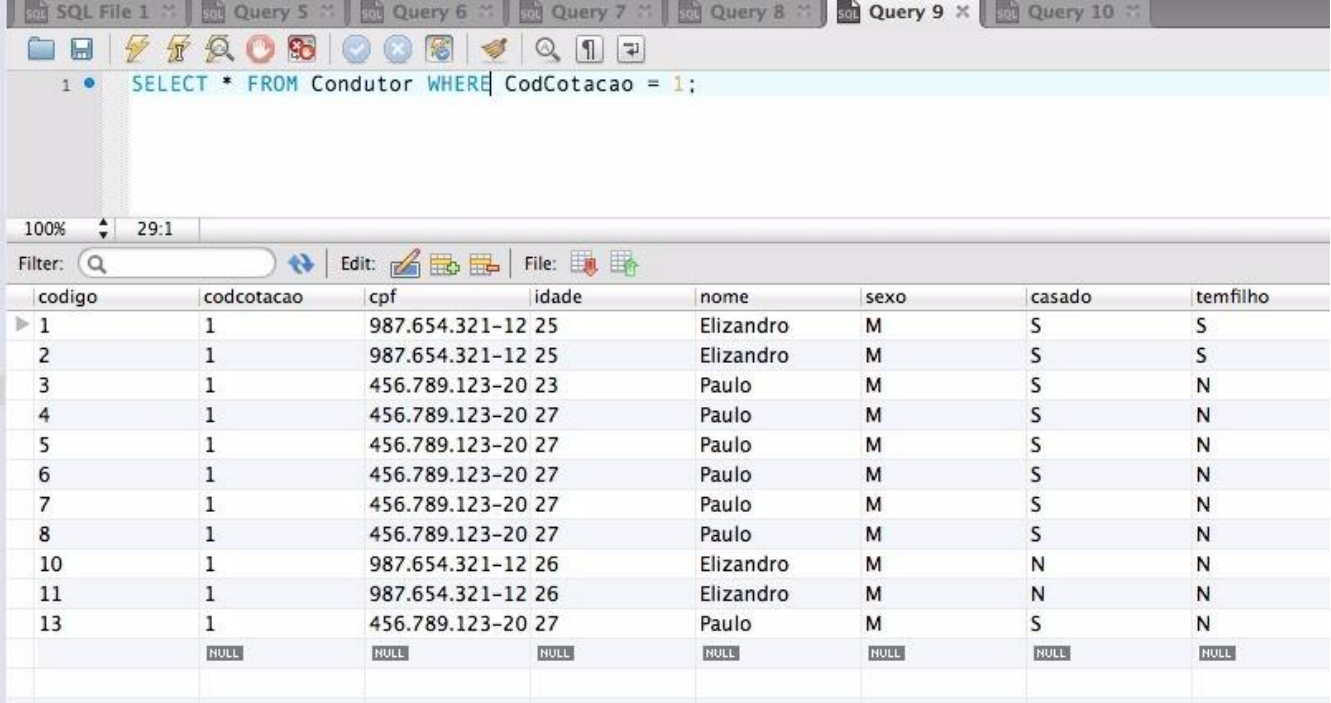
**Versão: 1.0**

## 1. Testes Unitários Automatizados

Nome da Classe	Nome do Método
ClausulaDAOTest	testLoadFromDBInt()
ClausulaDAOTest	testLoadFromDBString()
CotacaoDAOTest	testSaveToDB()
CotacaoDAOTest	testLoadFromDB()
CotacaoDAOTest	testClausulas()
RelatorioDAOTest	testLoadApolicesVencendo()
VeiculoDAOTest	testSaveToDB()
VeiculoDAOTest	testLoadFromDB()

## 2. Casos de Testes das Atividades da Sprint 1

Base de Dados Utilizada para realizar os casos de testes:



The screenshot shows a SQL query editor with a toolbar at the top. The query editor contains the following SQL query:

```
1 • SELECT * FROM Condutor WHERE CodCotacao = 1;
```

Below the query editor, there is a table displaying the results of the query. The table has 8 columns: **codigo**, **codcotacao**, **cpf**, **idade**, **nome**, **sexo**, **casado**, and **temfilho**. The results are as follows:

codigo	codcotacao	cpf	idade	nome	sexo	casado	temfilho
1	1	987.654.321-12	25	Elizandro	M	S	S
2	1	987.654.321-12	25	Elizandro	M	S	S
3	1	456.789.123-20	23	Paulo	M	S	N
4	1	456.789.123-20	27	Paulo	M	S	N
5	1	456.789.123-20	27	Paulo	M	S	N
6	1	456.789.123-20	27	Paulo	M	S	N
7	1	456.789.123-20	27	Paulo	M	S	N
8	1	456.789.123-20	27	Paulo	M	S	N
10	1	987.654.321-12	26	Elizandro	M	N	N
11	1	987.654.321-12	26	Elizandro	M	N	N
13	1	456.789.123-20	27	Paulo	M	S	N
	NULL	NULL	NULL	NULL	NULL	NULL	NULL

## ClausulaDAOTest

### Método:

```
@Test
public final void testLoadFromDBInt() {
    assertTrue(ClausulaDAO.loadFromDB(1).getTipo().equals("Carro reserva
7"));
}
```

### Objetivo do método:

Com base em valores pré-definidos na base, se o tipo da clausula for igual a carro reserva 7 o teste passou.

### Método:

```
public final void testLoadFromDBString() {
    assertTrue(ClausulaDAO.loadFromDB("Carro reserva 7").getCodigo() ==
1);
}
```

### Objetivo do método:

Com base em valores pré-definidos na base, se retornar a clausula de código 1 o teste passou.

## Classe CotacaoDAOTest

```
public class CotacaoDAOTest {  
  
    private int codigo;
```

### Método

```
@Test  
public final void testSaveToDB() {  
    CotacaoDAO cotacao = new CotacaoDAO();  
    assertTrue("Erro ao salvar a cotacao.", cotacao.saveToDB());  
    this.codigo = cotacao.getCodigo();  
    assertTrue("Erro ao salvar a cotacao.", this.codigo > -1);  
}
```

### Objetivo do método:

Com base em valores pré-definidos na base, insere no banco, se o código da cotação gerado após executar o método de salvar no banco, for -1 apresenta mensagem que ocorreu erro ao salvar a cotação.

### Método

```
@Test  
public final void testLoadFromDB() {  
    testSaveToDB();  
    CotacaoDAO cotacao = CotacaoDAO.loadFromDB(this.codigo);  
    assertTrue("Erro ao carregar a cotacao.", this.codigo ==  
cotacao.getCodigo());  
}
```

### Objetivo do método:

Com base em valores pré-definidos na base, pesquisa o código gerado no método testSaveToDB()., se o código carregado for diferente do código inserido, apresenta mensagem que ocorreu erro ao carregar cotação.

### Método

```
@Test
public final void testClausulas() {
    testSaveToDB();
    CotacaoDAO cotacao = CotacaoDAO.loadFromDB(this.codigo);
    cotacao.addClausula(ClausulaDAO.loadFromDB(1)); //adiciona uma
cobertura para carro reserva de 7 dias
    assertTrue(cotacao.getClausulas().get(0) == 1);
}
```

### Objetivo do método:

Com base em valores pré-definidos na base, adiciona e teste a clausula do carro reserva de 7 na cotação for igual a 1.

## Classe RelatorioDAOTest

### Método

```
public class RelatorioDAOTest {  
  
    @Test  
    public final void testLoadApolicesVencendo() {  
        System.out.println(RelatorioDAO.loadApolicesVencendo(367));  
    }  
  
}
```

### Objetivo do método:

Carrega as apólices com mais de 367.

## Classe VeiculoDAOTest

```
public class VeiculoDAOTest {  
  
    private int codigo;  
    private String modelo = "Celta";  
    private double valorFIP = 35000;
```

## Método

```
@Test  
public final void testSaveToDB() {  
    VeiculoDAO veiculo = new VeiculoDAO();  
    veiculo.setModelo(this.modelo);  
    veiculo.setValorFIP(this.valorFIP);  
    veiculo.setCotacao(1);  
    if(veiculo.saveToDB()){  
        this.codigo = veiculo.getCodigo();  
        assertTrue("Erro ao salvar o ve'culo.", this.codigo > -1);  
    } else {  
        fail("Erro ao salvar ve'culo.");  
    }  
}
```

## Objetivo do método:

Com base em valores pré-definidos na base, seta alguns dados do veiculo e insere no banco, se o código do veiculo gerado após executar o método de salvar no banco, for -1 apresenta mensagem que ocorreu erro ao salvar o veiculo.



## Método

```
@Test
public final void testLoadFromDB() {
    testSaveToDB();
    VeiculoDAO dao = VeiculoDAO.loadFromDB(this.codigo);
    assertTrue("Erro ao carregar ve'culo.",
this.modelo.equals(dao.getModelo()));
}
```

### Objetivo do método:

Com base em valores pré-definidos na base, pesquisa o código gerado no método testSaveToDB()., se o modelo for diferente de "Celta ", apresenta mensagem que ocorreu erro ao carregar veículo.