Casos de Teste 1

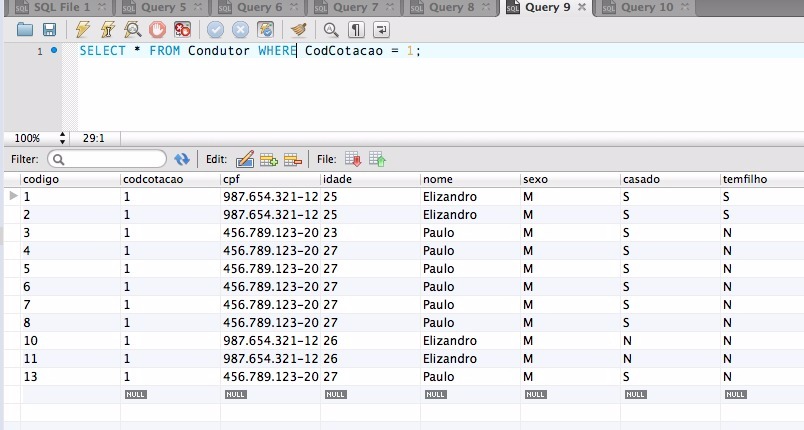
**Versão: 1.0**

1. **Testes Unitários Automatizados**

|  |  |
| --- | --- |
| **Nome da Classe** | **Nome do Método** |
| CotacaoControleTest | testGetCondutorPiorPerfil() |
| CotacaoControleTest | testCalculaPremio() |
| DBConnectionTest | testStatusConection() |
| DBConnectionTest | testFecharConexao() |
| DBConnectionTest | testCanExecuteCmd() |
| SeguradoDAOTest | testSaveToDB() |
| SeguradoDAOTest | testLoadFromDB() |
| CondutorDAOTest | testSaveToDB() |
| CondutorDAOTest | testLoadFromDB() |

1. **Casos de Testes das Atividades da Sprint 1**

Base de Dados Utilizada para realizar os casos de testes:



**Classe CotacaoControleTest**

**Método:**

@Test

**public** **final** **void** testGetCondutorPiorPerfil() {

**try** {

CotacaoControle co = **new** CotacaoControle(**new** Cotacao(1));

Condutor cn = co.getCondutorPiorPerfil();

assertTrue("N‹o testou certo - se tiver idade menor que 26", cn.getIdade() < 26);

} **catch** (Exception e) {

e.printStackTrace();

fail("Erro ao criar o controle");

}

}

**Objetivo do método:**

Com base em valores pré-definidos na base, se o pior condutor não for menor de 26 anos retorna mensagem que o teste falhou.

**Método:**

@Test

**public** **final** **void** testCalculaPremio(){

**try** {

CotacaoControle co = **new** CotacaoControle(**new** Cotacao(1));

**double** premio = co.calculaPremio();

assertTrue("N‹o testou certo - se premio nao for 1239", premio == 1239);

} **catch** (Exception e) {

e.printStackTrace();

fail("Erro ao criar o controle");

}

}

**Objetivo do método:**

Com base em valores pré-definidos na base, se o valor do premio não for igual a 1239 retorna mensagem que o teste falhou.

**Classe DBConnectionTest**

**Método**

@Test

**public** **final** **void** testStatusConection() {

DBConnection db = **new** DBConnection();

assertTrue(db.statusConection().equals("Conectado."));

}

**Objetivo do método:**

Com base em valores pré-definidos na base, se o status da conexão for conectado não terá erro. Do contrário apresenta mensagem que o teste falhou.

**Método**

@Test

**public** **final** **void** testFecharConexao() {

DBConnection db = **new** DBConnection();

assertTrue(db.FecharConexao());

}

**Objetivo do método:**

Com base em valores pré-definidos na base, se tiver sucesso ao fechar a conexão, não terá erro. Do contrário apresenta mensagem que o teste falhou.

**Método**

@Test

**public** **final** **void** testCanExecuteCmd() {

DBConnection db = **new** DBConnection();

assertTrue(db.canExecuteCmd());

}

**Objetivo do método:**

Com base em valores pré-definidos na base, se tiver sucesso ao executar um comando, não terá erro. Do contrário apresenta mensagem que o teste falhou.

**Classe SeguradoDAOTest**

**public** **class** SeguradoDAOTest {

**private** **int** codigo;

**private** String nome = "Maria";

**Método**

@Test

**public** **final** **void** testSaveToDB() {

SeguradoDAO segurado = **new** SeguradoDAO();

segurado.setNome(nome);

segurado.setCpf("123.456.789-20");

Calendar cal = **new** GregorianCalendar();

cal.set(1988, 9, 8);

segurado.setDataNascimento(**new** Date(cal.getTime().getTime()));

segurado.setSexo("F");

segurado.setTelefone("3333-8888"); //**TODO**: rever para funcionar com o DDD

**if**(segurado.saveToDB()){

**this**.codigo = segurado.getCodigo();

assertTrue("Erro ao salvar o segurado.", **this**.codigo > -1);

} **else** {

fail("Erro ao salvar o segurado.");

}

}

**Objetivo do método:**

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

**Método**

@Test

**public** **final** **void** testLoadFromDB() {

testSaveToDB();

SeguradoDAO segurado = SeguradoDAO.loadFromDB(codigo);

assertTrue("Erro ao carregar segurado.", nome.equals(segurado.getNome()));

}

**Objetivo do método:**

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

**Classe CondutorDAOTest**

**public** **class** CondutorDAOTest {

**private** **int** codigo;

**private** String nome = "Elizandro";

**Método**

@Test

**public** **final** **void** testSaveToDB() {

CondutorDAO dao = **new** CondutorDAO();

dao.setCodCotacao(1);

dao.setNome(nome);

dao.setCpf("987.654.321-12");

dao.setIdade(25);

dao.setSexo("M");

dao.setTemFilho("N");

dao.setCasado("N");

**if**(dao.saveToDB()){

**this**.codigo = dao.getCodigo();

assertTrue("Erro ao salvar o condutor.", **this**.codigo > -1);

} **else** {

fail("Erro ao salvar o condutor.");

}

}

**Objetivo do método:**

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

**Método**

@Test

**public** **final** **void** testLoadFromDB() {

testSaveToDB();

CondutorDAO dao = CondutorDAO.loadFromDB(**this**.codigo);

assertTrue("Erro ao carregar condutor.", nome.equals(dao.getNome()));

}

**Objetivo do método:**

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