Universidade Federal do Ceará
Campus Quixadá

Relatório de Testes
CliniVet

Felipe Raulino Lemos
Ítalo Rodrigues Nascimento
Mateus Gonçalves Loiola

ÍNDICE

1. Introdução

1.1. Descrição do documento

Esse documento pode ser destinado a todos que tenham interesse em conferir os testes unitários desenvolvidos para a aplicação CliniVet. A aplicação foi desenvolvida em conjunto pelos membros formadores da equipe, descritos na capa. O aplicativo faz parte do trabalho da cadeira de Projeto Integrado III, da Universidade Federal do Ceará - Campus Quixadá, ministrada pelo professor Camilo Camilo Almendra.

1.2. Objetivo dos testes

Os testes desenvolvidos têm por objetivo certificar que o aplicativo executa corretamente algumas de suas funções mais importantes, bem como garantir o uso das boas práticas existentes no mercado de software. Os testes foram feitos na linguagem de desenvolvimento JAVA, assim como toda a aplicação. Para os testes foram usadas bibliotecas como o JUnit, Mockito e PowerMock.

2. Testes

Todos os testes a seguir são do pacote Model. As classes Pet, Usuário, Reserva e Serviço foram testadas. Existem três grandes áreas de testes em aplicações móveis, são elas os testes unitários, os testes de integração e os testes de UI/Testes Instrumentais. Devido ao fato da nossa aplicação ter sido desenvolvida com o uso do Firebase, sendo assim Serverless, não foi possível testar algumas das principais chamadas ao Back-end da aplicação. Optamos, portanto, por testar aquilo que nos foi disponibilizado na área de testes unitários, ou seja, os Models da aplicação. Os Models da aplicação são utilizados na criação de novos objetos dos tipos Pet, Usuário, Reserva e Serviço, seus métodos são utilizados para atualização de alguns desses objetos, faz assim a necessidade de incluir os testes nesse módulo da aplicação.

2.1. Testes unitários da classe PET

```
@RunWith(PowerMockRunner.class)
@PowerMockRunnerDelegate(JUnit4.class)
@PrepareForTest({ FirebaseDatabase.class})
@PowerMockIgnore("jdk.internal.reflect.*")
public class PetUnitTest {
    private Pet pet;

    @Mock
    private DatabaseReference mockedDatabaseReference;

    @Before
    public void setUp(){
        mockedDatabaseReference = Mockito.mock(DatabaseReference.class);

        FirebaseDatabase mockedFirebaseDatabase = Mockito.mock(FirebaseDatabase.class);
        PowerMockito.when(mockedFirebaseDatabase.getReference()).thenReturn(mockedDatabaseReference);

        PowerMockito.mockStatic(FirebaseDatabase.class);
        PowerMockito.when(FirebaseDatabase.getInstance()).thenReturn(mockedFirebaseDatabase);

        pet = new Pet( id: "67477fed1789f17fa188337265ff245459776f2b",  nome: "Joao",  tipo: "Cachorro",  descricao: "Um cachorro bonito",
    }
```

Imagem 1: Definição da classe e inicialização de mocks.

```java
@Test
public void testSetId(){
    String idTest = "9o5S8agQu1u4rpw3R53g";
    pet.setNome(idTest);
    assertEquals(idTest, pet.getId());
}

@Test
public void testSetNome(){
    String nomeTest = "Bolt";
    pet.setNome(nomeTest);
    assertEquals(nomeTest, pet.getNome());
}

@Test
public void testAtualizandoTipoPet(){
    String tipoTest = "Bolt";
    pet.setTipo(tipoTest);
    assertEquals(tipoTest, pet.getTipo());
}
```

Imagem 2: Testes de atualização de id, nome e tipo de um Pet.

```java
@Test
public void testSetDescricao(){
    String descricaoTest = "Uma nova descrição";
    pet.setDescricao(descricaoTest);
    assertEquals(descricaoTest, pet.getDescricao());
}

@Test
public void testSetIdade(){
    String idadeTest = "2,5";
    pet.setIdade(idadeTest);
    assertEquals(idadeTest, pet.getIdade());
}
```

Imagem 3: Testes de atualização da descrição e idade de um Pet.

```java
@Test
public void testSetPeso(){
    String pesoTest = "2,8";
    pet.setPeso(pesoTest);
    assertEquals(pesoTest, pet.getPeso());
}

@Test
public void testSetAltura(){
    String alturaTest = "88";
    pet.setAltura(alturaTest);
    assertEquals(alturaTest, pet.getAltura());
}
```

Imagem 4: Testes de atualização do peso e altura de um Pet.

2.2. Testes unitários da classe USUÁRIO

```java
public class UsuarioUnitTest {
    Usuario usuario;

    @Before
    public void setUp() { usuario = new Usuario(); }
```

Imagem 5: Definição da classe e inicialização de objeto da classe Usuário.

```java
@Test
public void testSetId(){
    String idTest = "67477fed1789f17fa188337265ff245459776f2b";
    usuario.setId(idTest);
    assertEquals(idTest, usuario.getId());
}

@Test
public void testSetNome(){
    String nomeTest = "João";
    usuario.setNome(nomeTest);
    assertEquals(nomeTest, usuario.getNome());
}

@Test
public void testSetEmail(){
    String emailTest = "joão@gmail.com";
    usuario.setNome(emailTest);
    assertEquals(emailTest, usuario.getNome());
}
```

Imagem 6: Teste de atualização de id, nome e email de um Usuário.

```java
@Test
public void testSetTelefone(){
    String telefoneTest = "88928373829";
    usuario.setNome(telefoneTest);
    assertEquals(telefoneTest, usuario.getNome());
}

@Test
public void testSetSenha(){
    String senhaTest = "joao123";
    usuario.setNome(senhaTest);
    assertEquals(senhaTest, usuario.getNome());
}
```

Imagem 7: Teste de atualização de telefone e senha de um Usuário.

2.3. Testes unitários da classe RESERVA

```java
public class ReservaUnitTest {

    Reserva reserva;

    @Before
    public void setUp(){
        reserva = new Reserva( id: "9b89b52aa72ffba80d0315d6deb34357d48caf5d",  idpet: "bc5d47bde5b4640cd64e10ffc716dff878b41b33",
    }
```

Imagem 8: Definição e inicialização de objeto da classe Reserva.

```java
@Test
public void testAtualizandoNomeDoPet(){
    String nomePetTest = "Tuta";
    reserva.setPetNome(nomePetTest);
    assertEquals(nomePetTest, reserva.getPetNome());
}

@Test
public void testAtualizandoServicoDaReserva(){
    String idServicoTest = "RRyx1Ldobf78VZp97QmF";
    reserva.setIdServico(idServicoTest);
    assertEquals(idServicoTest, reserva.getIdServico());
}

@Test
public void testAtualizandoImagemDoPet(){
    String urlDaImagemDoPetTest = "https://nova_imagem_do_pet.com/01.png";
    reserva.setImgPet(urlDaImagemDoPetTest);
    assertEquals(urlDaImagemDoPetTest, reserva.getImgPet());
}
```

Imagem 9: Teste de atualização de nome, serviço e imagem do pet de uma Reserva.

```java
@Test
public void testAtualizandoDiaDaReserva(){
    String diaTest = "18";
    reserva.setDia(diaTest);
    assertEquals(diaTest, reserva.getDia());
}

@Test
public void testAtualizandoHorarioDaReserva(){
    String horarioTest = "15:30";
    reserva.setHora(horarioTest);
    assertEquals(horarioTest, reserva.getHora());
}

@Test
public void testAtualizandoStatusDaReserva(){
    String statusTest = "Concluída";
    reserva.setStatus(statusTest);
    assertEquals(statusTest, reserva.getStatus());
}
```

Imagem 10: Teste de atualização de dia, horário e status de uma Reserva.

2.4. Testes unitários da classe SERVIÇO

```java
public class ServicoUnitTest {

    Servico servico;

    @Before
    public void setUp(){
        servico = new Servico( id: "17Hj7enYG14s4W5AT1ZM1MyIP", nome: "Vacinação", pets: "Cachorros",
    }
```

Imagem 11: Definição e inicialização de objeto da classe Serviço.

```java
@Test
public void testAtualizandoNomeDeUmServico(){
    String nomeServicoTest = "Vacinação em filhotes";
    servico.setNome(nomeServicoTest);
    assertEquals(nomeServicoTest, servico.getNome());
}

@Test
public void testAtualizandoPetsDestinadosDeUmServico(){
    String petsServicoTest = "Gatos";
    servico.setPets(petsServicoTest);
    assertEquals(petsServicoTest, servico.getPets());
}
```

Imagem 12: Teste de atualização do nome de um serviço e dos pets destinados àquele serviço.

```java
@Test
public void testAtualizandoDescricaoDeUmServico(){
    String descricaoServicoTest = "Vacinação em filhotes";
    servico.setDescricao(descricaoServicoTest);
    assertEquals(descricaoServicoTest, servico.getDescricao());
}

@Test
public void testAtualizandoPrecoDeUmServico(){
    String precoServicoTest = "65";
    servico.setPreco(precoServicoTest);
    assertEquals(precoServicoTest, servico.getPreco());
}
```

Imagem 13: Teste de atualização da descrição e do preço de um serviço.

3. Referências

Documentação do Developers Android: Link para a documentação