Memoria de la Práctica 3

5 de mayo de 2024

Integrantes del grupo

- Carmen Chunyin Fernández Núñez
- Pablo García Guijosa
- Marta Xiaoyang Moraga Hernández
- Jesús Navarrete Caparrós

1. Índice

2. Descripcion

Esta práctica ha consistido en diseñar e implementar una serie de tests, relativos a la funcionalidad del modelo de la aplicación creada para la práctica 2. Muchos no han requerido cambios al modelo subyacente, aunque algunas funcionalidades sí han requerido modificaciones, para lograr que su funcionalidad satisfaga las especificaciones de los tests propuestos.

Estos tests comprueban el correcto funcionamiento del modelo, tanto de las funcionalidades más simples (como añadir un empleado o departamento por sí solos), como operaciones más complejas (como añadir empleados o departamentos con datos iguales a otros ya añadidos, o eliminar bloques de departamentos y empleados anidados).

Además, se ha decidido modificar un comportamiento del modelo que en un principio considerábamos como válido, pero que actualmente hemos decidido modificar. Se trata de la posibilidad de que un empleado o departamento, de los mismos datos, pueda pertenecer a más de un departamento a la vez. En la versión que se entrega en esta práctica, si se intenta añadir a otro departamento un empleado o departamento con los mismos datos que otro que ya se encuentra en un departamento, lo único que se producirá es que el empleado (o departamento) cambiará a pertenecer sólamente al nuevo departamento donde se ha añadido, y no a los dos.

3. Planificación de las pruebas del sistema

Las pruebas ideadas para comprobar el funcionamiento de la aplicación son las siguientes:

1. Añadir empleado a empleado
2. Añadir departamento a departamento
3. Añadir empleado de mismos datos en varios departamentos
4. Añadir departamento de mismos datos en varios departamentos
5. Añadir departamento a empleado
6. Añadir empleado a departamento
7. Añadir departamento a sí mismo
8. Añadir empleado o departamento con datos parciales
9. Añadir empleado fuera de departamento
10. Eliminar elemento
11. Eliminar bloque completo
12. Eliminar con nada seleccionado
13. Eliminar bloque elimina bien lo de dentro

4. Análisis de pruebas

Elemento a probar	**Condiciones**	**Datos requeridos**
Añadir empleado a empleado	Tener 1 empleado creado	Contenido de sus arrays "elementos" respectivos después de intentar realizar la acción
Añadir departamento a si mismo	Tener 1 departamento creado	Contenido de sus arrays "elementos" respectivos después de intentar realizar la acción
Añadir departamento a departamento	Tener 1 departamento creado	Contenido de sus arrays "elementos" respectivos después de intentar realizar la acción
Añadir empleado de mismos datos en varios departamentos	Tener varios departamentos creados	Contenido de sus arrays "elementos" respectivos después de intentar realizar la acción
Añadir departamento de mismos datos en varios departamentos	Tener varios departamentos creados	Contenido de sus arrays "elementos" respectivos después de intentar realizar la acción
Añadir departamento a empleado	Tener 1 empleado creado	Contenido de sus arrays "elementos" respectivos después de intentar realizar la acción
Añadir empleado a departamento	Tener 1 departamento creado	Contenido de sus arrays "elementos" respectivos después de intentar realizar la acción
Añadir empleado o departamento con datos parciales	—	—
Añadir empleado fuera de departamento	—	—
Eliminar elemento	Tener creado al menos 1 elemento	Contenido de sus arrays "elementos" respectivos después de intentar realizar la acción
Eliminar bloque completo	Tener creado al menos 1 departamento que, a su vez, tiene agregado como mínimo 1 departamento o 1 empleado	Contenido de sus arrays "elementos" respectivos después de intentar realizar la acción
Eliminar con nada seleccionado	—	—
Eliminar bloque elimina bien lo de dentro	Tener creado al menos 1 departamento que, a su vez, tiene agregado como mínimo 1 departamento o 1 empleado	Contenido de sus arrays "elementos" respectivos después de intentar realizar la acción

Cuadro 1: Análisis

5. Diseño de pruebas

Casos de prueba	**Entornos de prueba**	**Datos de prueba**	**Trazabilidad**
Añadir empleado a empleado	Ejecución normal del programa. No necesita herramientas adicionales	Elemento Empleado al que se le intenta añadir un empleado, dentro de "elementos"	Aparece un nuevo error de tipo ÜnimplementedError"
Añadir departamento a departamento	Ejecución normal del programa. No necesita herramientas adicionales	Elemento Departamento al que se le intenta añadir un departamento, dentro de "elementos"	"elementos" del departamento contiene un nuevo departamento, cuyo padre es el departamento donde queríamos incluirlo
Añadir empleado de mismos datos en varios departamentos	Ejecución normal del programa. No necesita herramientas adicionales	Elementos Departamento a los que se le intenta añadir un empleado, dentro de "elementos"	"elementos" del primer departamento ahora no contiene al empleado, mientras que "elementos" del segundo departamento ahora sí lo contiene
Añadir departamento de mismos datos en varios departamentos	Ejecución normal del programa. No necesita herramientas adicionales	Elementos Departamento a los que se le intenta añadir un departamento, dentro de "elementos"	"elementos" del primer departamento ahora no contiene al departamento, mientras que "elementos" del segundo departamento ahora sí lo contiene
Añadir departamento a empleado	Ejecución normal del programa. No necesita herramientas adicionales	Elemento Empleado al que se le intenta añadir un empleado, dentro de "elementos"	Aparece un nuevo error de tipo ÜnimplementedError"
Añadir empleado a departamento	Ejecución normal del programa. No necesita herramientas adicionales	Elemento Departamento al que se le intenta añadir un empleado, dentro de "elementos"	"elementos" contiene un nuevo empleado, cuyo padre es el departamento donde queríamos incluirlo
Añadir empleado o departamento con datos parciales	Ejecución normal del programa. No necesita herramientas adicionales	Elemento Empleado o Elemento Departamento al que se le intenta añadir al sistema sin todos los campos llenos	Los elementos incompletos no estarán contenidos en el array de elementos del director
Añadir empleado fuera de departamento	Ejecución normal del programa. No necesita herramientas adicionales	Añadir un Elemento Empleado al sistema sin tener asociado ningún Elemento Departamento	"elementos" contiene un empleado que no tiene asociado ningún padre
Eliminar elemento	Ejecución normal del programa. No necesita herramientas adicionales	Eliminar un Elemento del sistema	El elemento eliminado no se encontrará en el array
Eliminar bloque completo	Ejecución normal del programa. No necesita herramientas adicionales	Eliminar un Elemento Departamento que tenga subdepartamentos y/o empleados	Los elementos eliminados no estarán contenidos en el array de elementos del director
Eliminar con nada seleccionado	Ejecución normal del programa. No necesita herramientas adicionales	Intentar eliminar sin seleccionar ningún elemento	Ningún elemento se verá modificado, todos los elementos originales estarán contenidos en el array del director
Eliminar bloque elimina bien lo de dentro	Ejecución normal del programa. No necesita herramientas adicionales	Eliminar todos los "elementos" de todos los Departamentos y Empleados del bloque seleccionado	Verificar que los elementos fuera del bloque seleccionado no se ven afectados y que todos los elementos dentro del bloque se eliminan correctamente

Cuadro 2: Diseño

6. Implementación de los test

En esta sección, se van a explicar las modificaciones que se han tenido que realizar sobre el código para que las funcionalidades probadas a partir de los test estuviesen implementadas correctamente.

1. Añadir departamento a departamento

   Se ha tenido que modificar la función addElementoEmpresa(ElementoEmpresa elemento) de la clase Departamento. Pues no se modifica el Departamento Superior asociado al elemento que añadimos a la lista de elementos empresa.

```
@override
void addElementoEmpresa(ElementoEmpresa elemento) {
  // TODO: implement addElementoEmpresa
  elementos.add(elemento);
  elemento.cambiarSuperior(this);

}
```

2. Añadir empleado a departamento

   Lo explicado en el punto anterior también ha sido aplicado a la clase Empleado, además de implementar la función cambiarSuperior(), pues no estaba implementada.

```
@override
void cambiarSuperior(ElementoEmpresa? nuevoSuperior) {
  this.DepSuperior = nuevoSuperior;
}
```

3. Añadir departamento/empleado de mismos datos en varios departamentos

   Esta funcionalidad se enfoca desde el hecho de que no se pueden añadir el mismo empleado/departamento en dos departamentos diferentes, por ello al añadirlos a un departamento, si ya pertenecían a otro, se modificará el superior del elemento añadido y se eliminará del antiguo superior.

```
@override
void addElementoEmpresa(ElementoEmpresa elemento) {
  // TODO: implement addElementoEmpresa
  ElementoEmpresa? sup = elemento.getSuperior();
  if(sup!=null){   //Pertenece a otro departamento, hay que eliminarlo
    sup.getElementos().remove(elemento);
  }
  elementos.add(elemento);
  elemento.cambiarSuperior(this);

}
```

4. Añadir un departamento a sí mismo

Si un departamento intenta añadirse a su propia lista de departamentos, deberá lanzarse un error. Para ello, hemos modificado la función cambiarSuperior, para que compruebe si el elemento que se intenta añadir es uno mismo. Además de cambiar el orden de las operaciones dentro de la función addElementoEmpresa.

```
@override
void cambiarSuperior(ElementoEmpresa? nuevoSuperior) {
  if(nuevoSuperior == this){
    throw UnimplementedError();
  }
  else{
    this.DepSuperior = nuevoSuperior;
  }
}
```

```
@override
void addElementoEmpresa(ElementoEmpresa elemento) {
  // TODO: implement addElementoEmpresa
  ElementoEmpresa? sup = elemento.getSuperior();
  if(sup!=null){   //Pertenece a otro departamento, hay que eliminarlo
    sup.getElementos().remove(elemento);
  }
  elemento.cambiarSuperior(this);
  elementos.add(elemento);
}
```

5. La posibilidad de añadir un elemento empresa directamente con director.

   En nuestro modelo inicial, se ofrecía poder añadir un departamento o un empleado utilizando sus datos con las funciones de addDepartamento y addEmpleado, respectivamente, que se encargan de crear el elemento indicado. Sin embargo, el director carecía de un addElementoEmpresa para añadir elementos ya creados, que ahora añadimos. Puesto que queremos evitar que se añadan a la empresa elementos sin los datos adecuados, se añaden comprobaciones similares a las que tenían addDepartamento y addEmpleado.

```
void addElementoEmpresa(ElementoEmpresa elemento) {
  bool elemento_aceptable = false;
  if(elemento.toString().trim().isNotEmpty){
    if (elemento is Empleado) {
      if(elemento.getDni().trim().isNotEmpty
          && elemento.getTipoContrato().trim().isNotEmpty
          && elemento.getCargo().trim().isNotEmpty){
        elemento_aceptable = true;
      }
    } else {
      elemento_aceptable = true;
    }
  }

  if(elemento_aceptable){
    if(seleccionado == null){
      if(elemento.getSuperior() != null){
        elemento.getSuperior()?.removeElementoEmpresa(elemento);
        elemento.cambiarSuperior(null);
      }
      empresa.add(elemento);
    } else if (seleccionado is Departamento){
      // addElementoEmpresa del departamento ya se encarga de cambiarle el superior a él si hiciera falta
      seleccionado?.addElementoEmpresa(elemento);
    }
  }
}
```

Para facilitar la corrección, incluimos el código de los test.

```dart
import 'package:ejercicio_grupal/Model/Director.dart';
import 'package:ejercicio_grupal/Model/EmpleadoTiempoCompletoBuilder.dart';
import 'package:flutter_test/flutter_test.dart';
import 'package:ejercicio_grupal/Model/Empleado.dart';
import 'package:ejercicio_grupal/Model/Departamento.dart';

void main() {
  group('Test Práctica 3', () {
    late Empleado employee;
    late Empleado subEmployee;
    late Departamento department1;
    late Departamento department2;

    setUp(() {
      employee = Empleado('John Doe', '12345678A', 'Software Engineer',
          'Tiempo Completo', null);
      subEmployee = Empleado('Jane Doe', '98765432B', 'Software Engineer',
          'Tiempo Completo', null);
      department1 = Departamento('Department 1', null);
      department2 = Departamento('Department 2', null);
    });

    test('Añadir empleado a empleado', () {
      expect(() => employee.addElementoEmpresa(subEmployee),
          throwsUnimplementedError);
    });

    test('Añadir Departamento a Departamento', () {
      department1.addElementoEmpresa(department2);

      expect(department1.getElementos(), contains(department2));
      expect(department2.getSuperior(), equals(department1));
    });
```

```
  test('Añadir Departamento perteneciente a un Departamento a otro', () {
    Departamento childDepartment = Departamento('Child Department', null);

    department1.addElementoEmpresa(childDepartment);
    department2.addElementoEmpresa(childDepartment);

    expect(department1.getElementos(), isNot(contains(childDepartment)));
    expect(department2.getElementos(), contains(childDepartment));
    expect(childDepartment.getSuperior(), equals(department2));
  });
  test('Añadir Empleado perteneciente a un Departamento a otro', () {
    department1.addElementoEmpresa(employee);
    department2.addElementoEmpresa(employee);

    expect(department1.getElementos(), isNot(contains(employee)));
    expect(department2.getElementos(), contains(employee));
    expect(employee.getSuperior(), equals(department2));
  });

  test('Añadir Departamento a Empleado', () {
    expect(() => employee.addElementoEmpresa(department1),
        throwsUnimplementedError);
  });

  test('Añadir Empleado a Departamento', () {
    department1.addElementoEmpresa(employee);

    expect(department1.getElementos(), contains(employee));
    expect(employee.getSuperior(), equals(department1));
  });

  test('Añadir Departamento a sí mismo', () {
    expect(() => department1.addElementoEmpresa(department1),
        throwsUnimplementedError);
  });
});
```

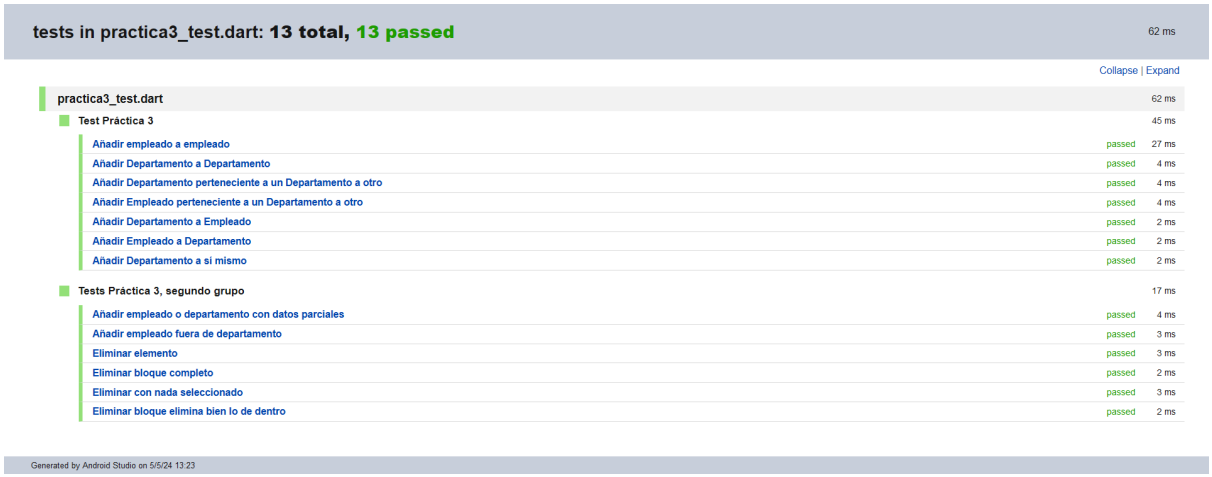
```
group('Tests Práctica 3, segundo grupo', () {
  late Director director;
  late Departamento departmentoA;
  late Departamento departmentoB;
  late Empleado empleado1;
  late Empleado empleado2;
  late Empleado empleado3;

  setUp(() {
    director = Director(EmpleadoTiempoCompletoBuilder(null));
    departmentoA = Departamento('A', null);
    departmentoB = Departamento('B', null);
    empleado1 =
        Empleado('1', '111111111L', 'Empleado', 'Tiempo Completo', null);
    empleado2 =
        Empleado('2', '999999999P', 'Empleado', 'Tiempo Completo', null);
    empleado3 =
        Empleado('3', '000000000A', 'Empleado', 'Tiempo Completo', null);

    director.addElementoEmpresa(departmentoA);
    director.addElementoEmpresa(empleado3);

    director.setElementoSeleccionado(departmentoA);
    director.addElementoEmpresa(departmentoB);
    director.addElementoEmpresa(empleado1);

    director.setElementoSeleccionado(departmentoB);
    director.addElementoEmpresa(empleado2);
    director.setElementoSeleccionado(null);
    /*
    Estructura de la empresa
      A
      -> B
      -> -> 2
      -> 1
      3
     */
  });
```

```
test('Añadir empleado o departamento con datos parciales', () {
  Departamento departmentIncompleto = Departamento('', null);
  Empleado empleadoIncompleto =
      Empleado('', '', '', 'Tiempo Completo', null);
  director.addElementoEmpresa(departmentIncompleto);
  director.addElementoEmpresa(empleadoIncompleto);
  expect(director.getEmpresa(), isNot(contains(departmentIncompleto)));
  expect(director.getEmpresa(), isNot(contains(empleadoIncompleto)));

  int n_elementos = director.getEmpresa().length;
  director.addEmpleado("", "", "", null);
  director.addDepartamento("", null);
  expect(director.getEmpresa().length, n_elementos);
});

test('Añadir empleado fuera de departamento', () {
  Empleado empleadoAux =
      Empleado('Aux', '000000000A', 'Posicion', 'Tiempo Completo', null);
  director.addElementoEmpresa(empleadoAux);
  expect(director.getEmpresa(), contains(empleadoAux));

  director.addEmpleado('Aux', '000000000A', 'Posicion', null);
  expect(director.getEmpresa().last.toString(), 'Aux');
});

test('Eliminar elemento', () {
  director.setElementoSeleccionado(empleado1);
  director.remove();
  expect(director.getEmpresa(), isNot(contains(empleado1)));

  director.setElementoSeleccionado(departmentoA);
  director.remove();
  expect(director.getEmpresa(), isNot(contains(departmentoA)));
});
```

```
    test('Eliminar bloque completo', () {
      director.setElementoSeleccionado(departmentoA);
      director.remove();

      expect(director.getEmpresa(), isNot(contains(departmentoA)));
    });

    test('Eliminar con nada seleccionado', () {
      int n_elementos = director.getEmpresa().length;
      director.setElementoSeleccionado(null);

      expect(director.getEmpresa().length, n_elementos);
      expect(director.getEmpresa(), contains(empleado3));
      expect(director.getEmpresa(), contains(departmentoA));
      expect(departmentoA.getElementos(), contains(departmentoB));
      expect(departmentoA.getElementos(), contains(empleado1));
      expect(departmentoB.getElementos(), contains(empleado2));
    });

    test('Eliminar bloque elimina bien lo de dentro', () {
      director.setElementoSeleccionado(departmentoA);
      director.remove();

      expect(director.getEmpresa(), isNot(contains(departmentoA)));
      expect(departmentoA.getElementos().length, 0);
      expect(departmentoB.getElementos().length, 0);
      expect(empleado1.getSuperior(), null);
      expect(empleado2.getSuperior(), null);
    });
  });
}
```

7. Resultados tests

El resultado de los tests:



tests in practica3_test.dart: 13 total, 13 passed — 62 ms

Collapse | Expand

practica3_test.dart		62 ms
Test Práctica 3		45 ms
Añadir empleado a empleado	passed	27 ms
Añadir Departamento a Departamento	passed	4 ms
Añadir Departamento perteneciente a un Departamento a otro	passed	4 ms
Añadir Empleado perteneciente a un Departamento a otro	passed	4 ms
Añadir Departamento a Empleado	passed	2 ms
Añadir Empleado a Departamento	passed	2 ms
Añadir Departamento a si mismo	passed	2 ms
Tests Práctica 3, segundo grupo		17 ms
Añadir empleado o departamento con datos parciales	passed	4 ms
Añadir empleado fuera de departamento	passed	3 ms
Eliminar elemento	passed	3 ms
Eliminar bloque completo	passed	2 ms
Eliminar con nada seleccionado	passed	3 ms
Eliminar bloque elimina bien lo de dentro	passed	2 ms

Generated by Android Studio on 5/5/24 13:23