Proyecto Bets22

Verificación del Software 22-23

UPV-EHU Ingeniería del Software II

Anal Lucía Durán Lengo

Jon Ortega Goikoetxea

UPV EHU

Contenido

Integrantes y Horas de Trabajo

- Ana Lucía Durán Lengo: 15h → Método 1: gertaerakSortu()
- Jon Ortega Goikoetxea: 15h → Método 2: EmaitzakIpini()

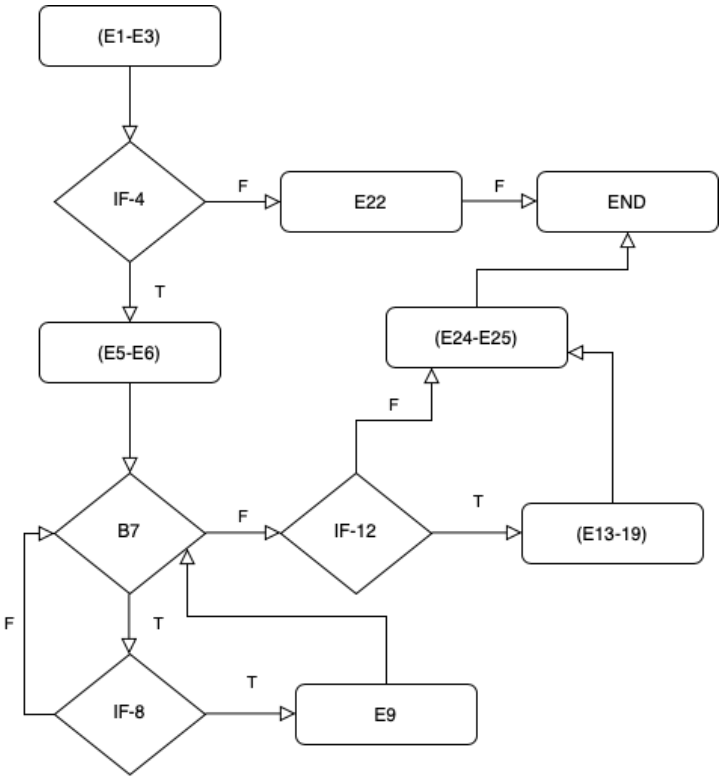
Método 1: gertaerakSortu()

Código

```
public boolean gertaerakSortu(String description,Date eventDate, String sport) {
    boolean b = true;
    db.getTransaction().begin();
    Sport spo =db.find(Sport.class, sport);
    if(spo!=null) {
        TypedQuery<Event> Equery = db.createQuery("SELECT e FROM Event e WHERE e.getEventDate() =?1 ",Event.class);
        Equery.setParameter(1, eventDate);
        for(Event ev: Equery.getResultList()) {
            if(ev.getDescription().equals(description)) {
                b = false;
            }
        }
        if(b) {
            String[] taldeak = description.split("-");
            Team lokala = new Team(taldeak[0]);
            Team kanpokoa = new Team(taldeak[1]);
            Event e = new Event(description, eventDate, lokala, kanpokoa);
            e.setSport(spo);
            spo.addEvent(e);
            db.persist(e);
        }
    }else {
        return false;
    }
    db.getTransaction().commit();
    return b;
}
```

Pruebas Unitarias

Diseño



Base de Datos Utilizada

Eventos	
Evento 1	("Sevilla- Betis", 14/03/2023, "Fútbol")
Evento 2	("Eibar-Sevilla", 28/02/2023, "Fútbol")
Evento 3	(null, 14/03/2023, "Fútbol")
Evento 4	("Sevilla- Betis", null, "Fútbol")
Evento 5	("Sevilla- Betis", 14/03/2023, null)
Evento 6	(12345, 14/03/2023, "Fútbol")
Evento 7	("Sevilla- Betis, "14/03/2023", "Fútbol")
Evento 8	("Sevilla- Betis", 14/03/2023, 12345)
Evento 9	("Sevilla- Betis", 14/03/2020, 12345)

Caja Blanca

Caminos
IF4(F)-END
IF4(T)-B(0)-IF12(T)-END
IF4(T)-B(N)-IF8(T)-IF12(F)-END
IF4(T)-B(N)-IF8(F)-IF12(T)-END

Nº caso	Camino seguido	Condición de entrada	Entrada	Salida
1	IF4(F)-END	Sport no está en la base de datos	sport==null	false
2	IF4(T)-B(0)-IF12(T)-END	No hay evento en esa fecha		True
3	IF4(T)-B(N)-IF8(T)-IF12(F)-END	Hay evento en esa fecha		True
4	IF4(T)-B(N)-IF8(F)-IF12(T)-END	Hay evento para la descripción dada		False

Caja Negra

Nº caso	Condiciones de entrada	Clases de equivalencia válidas	Clases de equivalencia no válidas
1	El evento está en la BD	Evento en BD(1)	Evento no está en BD(2)
2	Descripción no es null	description!=null(3)	description==null(4)
3	EventDate no es null	eventDate!=null(5)	eventDate==null(6)
4	sport no es null	sport!=null(7)	sport==null(8)
5	Descripción es de tipo String	description es String(9)	description no es String(10)
6	EventDate es de tipo date	eventDate es Date(11)	eventDate no es Date(12)
7	Sport es de tipo String	sport es String(13)	sport no es String(14)
8	Sport pertenece a sport.class	sport pertenece a sport.class(15)	sport no pertenece a sport.class(16)
9	La fecha es posterior a hoy	eventDate es posterior a hoy(17)	eventDate no es posterior a hoy(18)

Nº caso	Clases de equivalencia cubiertas	Entrada	Salida
1	1,3,5,7,9,11,13,15,17	Evento 1	El evento se crea, b=true
2	2	Evento 2	b=false
3	4	Evento 3	b=false
4	6	Evento 4	b=false
5	8	Evento 5	b=false
6	10	Evento 6	b=false
7	12	Evento 7	b=false
8	14	Evento 8	b=false
9	16	Evento 9	b=false
	18	Evento 9	b=false

Implementación

Caja Negra

```
@Test
public void test1() {
    boolean expected = true;

    descripcion = "Casper Ruud-Alexander Zverev";
    date = UtilDate.newDate(today.get(Calendar.YEAR), (today.get(Calendar.MONTH)+1), 17);
    s = "Tennis";

    boolean result = dt.gertaerakSortu(descripcion, date, s);

    assertEquals(expected, result);

    for (Event ev : dt.getEvents(date)) {
        if (ev.getDescription().equals(descripcion)) {
            dt.gertaeraEzabatu(ev);
            deleted = true;
            break;
        }
    }

    System.out.println("El evento (" + descripcion + ", " + date.toString() + ", " + s + ") creado correctamente")

    if (deleted)
        System.out.println("Evento borrado (" + descripcion + ", " + date.toString() + ", " + s + ")");
    else
        System.out.println("Evento no borrado (" + descripcion + ", " + date + ", " + s + ")");

    descripcion = "Madrid-Barca";
    date = UtilDate.newDate(today.get(Calendar.YEAR), today.get(Calendar.MONTH), 10);
    deleted = false;
    for (Event ev : dt.getEvents(date)) {
        if (ev.getDescription().equals(descripcion)) {
            dt.gertaeraEzabatu(ev);
            deleted = true;
            break;
        }
    }

    if (deleted)
        System.out.println("Evento borrado (" + descripcion + ", " + date.toString() + ", " + s + ")");
    else
        System.out.println("Evento no borrado (" + descripcion + ", " + date + ", " + s + ")");
}
```

```java
@Test
public void test2() {

    boolean esperado = false;

    descripcion = " ";
    date = UtilDate.newDate(today.get(Calendar.YEAR), (today.get(Calendar.MONTH)+1), 17);
    s = "Tennis";

    boolean res = dt.gertaerakSortu(descripcion, date, s);

    assertEquals(esperado, res);

    for (Event ev : dt.getEvents(date)) {
        if (ev.getDescription().equals(descripcion)) {
            dt.gertaeraEzabatu(ev);
            deleted = true;
            break;
        }
    }

    System.out.println("La descripción es null (" + descripcion + ", " + date.toString() + ", " + s + ")");

    if (deleted)
        System.out.println("Evento borrado (" + descripcion + ", " + date.toString() + ", " + s + ")");
    else
        System.out.println("Evento no borrado (" + descripcion + ", " + date + ", " + s + ")");

}
```

```java
@Test
public void test3() {

    boolean esperado = false;

    descripcion = "Casper Ruud-Alexander Zverev";
    date = UtilDate.newDate(today.get(Calendar.YEAR), (today.get(Calendar.MONTH)+1), 17);
    s = " ";

    boolean res = dt.gertaerakSortu(descripcion, date, s);

    assertEquals(esperado, res);

    for (Event ev : dt.getEvents(date)) {
        if (ev.getDescription().equals(descripcion)) {
            dt.gertaeraEzabatu(ev);
            deleted = true;
            break;
        }
    }

    System.out.println("El sport es null (" + descripcion + ", " + date.toString() + ", " + s + ")");

    if (deleted)
        System.out.println("Evento borrado (" + descripcion + ", " + date.toString() + ", " + s + ")");
    else
        System.out.println("Evento no borrado (" + descripcion + ", " + date + ", " + s + ")");

}
```

```java
@Test
public void test4() {

    boolean esperado = false;

    descripcion = "Casper Ruud-Alexander Zverev";
    date = null;
    s = "Tennis";

    boolean res = dt.gertaerakSortu(descripcion, date, s);

    assertEquals(esperado, res);

    for (Event ev : dt.getEvents(date)) {
        if (ev.getDescription().equals(descripcion)) {
            dt.gertaeraEzabatu(ev);
            deleted = true;
            break;
        }
    }

    System.out.println("La fecha es null (" + descripcion + ", " + date.toString() + ", " + s + ")");

    if (deleted)
        System.out.println("Evento borrado (" + descripcion + ", " + date.toString() + ", " + s + ")");
    else
        System.out.println("Evento no borrado (" + descripcion + ", " + date + ", " + s + ")");

}
```

```java
@Test
public void test5() {

    boolean expected = false;

    descripcion = "Real Madrid-Barcelona";
    date = UtilDate.newDate(today.get(Calendar.YEAR), (today.get(Calendar.MONTH)+1), 17);
    s = "Balonmano";

    boolean result = dt.gertaerakSortu(descripcion, date, s);

    assertEquals(expected, result);

    for (Event ev : dt.getEvents(date)) {
        if (ev.getDescription().equals(descripcion)) {
            dt.gertaeraEzabatu(ev);
            deleted = true;
            break;
        }
    }

    System.out.println("sport " + s +" no es un Sport");

    if (deleted)
        System.out.println("Evento borrado (" + descripcion + ", " + date + ", " + s + ")");
    else
        System.out.println("Evento no borrado (" + descripcion + ", " + date + ", " + s + ")");

}
```

```
@Test
public void test6() {

    boolean expected = false;

    descripcion = "Real Madrid-Barcelona";
    date = UtilDate.newDate(today.get(Calendar.YEAR), (today.get(Calendar.MONTH)-3), 17);
    s = "Balonmano";

    boolean result = dt.gertaerakSortu(descripcion, date, s);

    assertEquals(expected, result);

    for (Event ev : dt.getEvents(date)) {
        if (ev.getDescription().equals(descripcion)) {
            dt.gertaeraEzabatu(ev);
            deleted = true;
            break;
        }
    }

    System.out.println("La fecha no puede ser anterior a la actual");

    if (deleted)
        System.out.println("Evento borrado (" + descripcion + ", " + date + ", " + s + ")");
    else
        System.out.println("Evento no borrado (" + descripcion + ", " + date + ", " + s + ")");

}
```

Caja Blanca

```
@Test
public void test1() {

    boolean expected = false;

    descripcion = "Real Madrid-Barcelona";
    date = UtilDate.newDate(today.get(Calendar.YEAR), (today.get(Calendar.MONTH)+1), 17);
    s = "Balonmano";

    boolean result = dt.gertaerakSortu(descripcion, date, s);

    assertEquals(expected, result);

    for (Event ev : dt.getEvents(date)) {
        if (ev.getDescription().equals(descripcion)) {
            dt.gertaeraEzabatu(ev);
            deleted = true;
            break;
        }
    }

    System.out.println("sport " + s +" no es un Sport");

    if (deleted)
        System.out.println("Evento borrado (" + descripcion + ", " + date + ", " + s + ")");
    else
        System.out.println("Evento no borrado (" + descripcion + ", " + date + ", " + s + ")");

}
```

```java
@Test
public void test2() {

    boolean expected = true;

    descripcion = "Casper Ruud-Alexander Zverev";
    date = UtilDate.newDate(today.get(Calendar.YEAR) + 1, (today.get(Calendar.MONTH)+1), 2);
    s = "Tennis";

    boolean result = dt.gertaerakSortu(descripcion, date, s);

    assertEquals(expected, result);

    for (Event ev : dt.getEvents(date)) {
        if (ev.getDescription().equals(descripcion)) {
            dt.gertaeraEzabatu(ev);
            deleted = true;
            break;
        }
    }

    if (deleted)
        System.out.println("Evento borrado (" + descripcion + ", " + date + ", " + s + ")");
    else
        System.out.println("Evento no borrado (" + descripcion + ", " + date + ", " + s + ")");

}
```

```java
@Test
public void test3() {

    boolean esperado = false;

    descripcion = "Casper Ruud-Alexander Zverev";
    date = UtilDate.newDate(today.get(Calendar.YEAR), (today.get(Calendar.MONTH)+1), 17);
    s = " ";

    boolean res = dt.gertaerakSortu(descripcion, date, s);

    assertEquals(esperado, res);

    for (Event ev : dt.getEvents(date)) {
        if (ev.getDescription().equals(descripcion)) {
            dt.gertaeraEzabatu(ev);
            deleted = true;
            break;
        }
    }

    System.out.println("El sport es null (" + descripcion + ", " + date.toString() + ", " + s + ")");

    if (deleted)
        System.out.println("Evento borrado (" + descripcion + ", " + date.toString() + ", " + s + ")");
    else
        System.out.println("Evento no borrado (" + descripcion + ", " + date + ", " + s + ")");

}
```

```
@Test
public void test4() {

    boolean expected = false;

    descripcion = "Djokovic-Federer";
    date = UtilDate.newDate(today.get(Calendar.YEAR), (today.get(Calendar.MONTH)+1), 17);
    s = "Tennis";

    dt.gertaerakSortu(descripcion, date, s);
    boolean result = dt.gertaerakSortu(descripcion, date, s);

    assertEquals(expected, result);

    for (Event ev : dt.getEvents(date)) {
        if (ev.getDescription().equals(descripcion)) {
            dt.gertaeraEzabatu(ev);
            deleted = true;
            break;
        }
    }

    if (deleted)
        System.out.println("Evento borrado (" + descripcion + ", " + date + ", " + s + ")");
    else
        System.out.println("Evento no borrado (" + descripcion + ", " + date + ", " + s + ")");

}
```

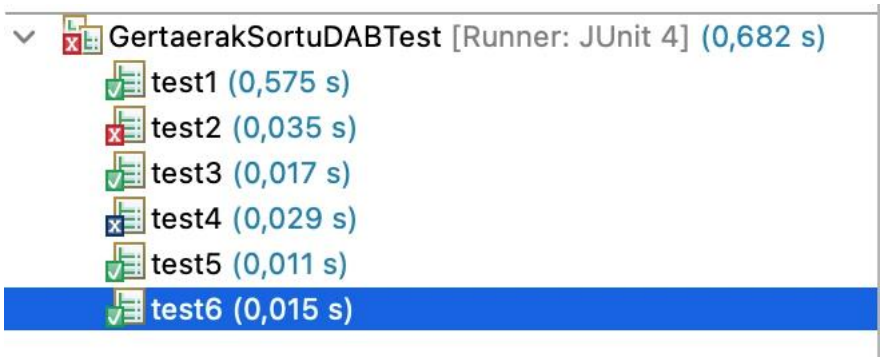
```
@Test
public void test6() {

    boolean esperado = false;

    descripcion = "Casper Ruud-Alexander Zverev";
    date = null;
    s = "Tennis";

    boolean res = dt.gertaerakSortu(descripcion, date, s);

    assertEquals(esperado, res);

    for (Event ev : dt.getEvents(date)) {
        if (ev.getDescription().equals(descripcion)) {
            dt.gertaeraEzabatu(ev);
            deleted = true;
            break;
        }
    }

    System.out.println("La fecha es null (" + descripcion + ", " + date.toString() + ", " + s + ")");

    if (deleted)
        System.out.println("Evento borrado (" + descripcion + ", " + date.toString() + ", " + s + ")");
    else
        System.out.println("Evento no borrado (" + descripcion + ", " + date + ", " + s + ")");

}
```

```
@Test
public void test5() {

    boolean expected = false;

    descripcion = null;
    date = UtilDate.newDate(today.get(Calendar.YEAR), (today.get(Calendar.MONTH)+1), 17);
    s = "Tennis";

    dt.gertaerakSortu(descripcion, date, s);
    boolean result = dt.gertaerakSortu(descripcion, date, s);

    assertEquals(expected, result);

    for (Event ev : dt.getEvents(date)) {
        if (ev.getDescription().equals(descripcion)) {
            dt.gertaeraEzabatu(ev);
            deleted = true;
            break;
        }
    }

    System.out.println("la descripción no puede ser null");

    if (deleted)
        System.out.println("Evento borrado (" + descripcion + ", " + date + ", " + s + ")");
    else
        System.out.println("Evento no borrado (" + descripcion + ", " + date + ", " + s + ")");


}
```

```
@Test
public void test7() {

    boolean expected = false;

    descripcion = "Real Madrid-Barcelona";
    date = UtilDate.newDate(today.get(Calendar.YEAR), (today.get(Calendar.MONTH)-3), 17);
    s = "Balonmano";

    boolean result = dt.gertaerakSortu(descripcion, date, s);

    assertEquals(expected, result);

    for (Event ev : dt.getEvents(date)) {
        if (ev.getDescription().equals(descripcion)) {
            dt.gertaeraEzabatu(ev);
            deleted = true;
            break;
        }
    }

    System.out.println("La fecha no puede ser anterior a la actual");

    if (deleted)
        System.out.println("Evento borrado (" + descripcion + ", " + date + ", " + s + ")");
    else
        System.out.println("Evento no borrado (" + descripcion + ", " + date + ", " + s + ")");

}
```

Resultados

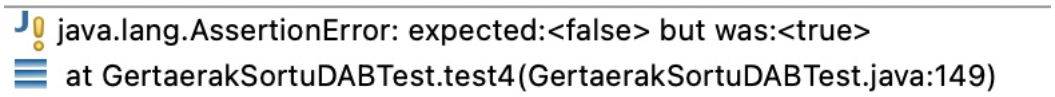
Caja Negra:



Test 4:

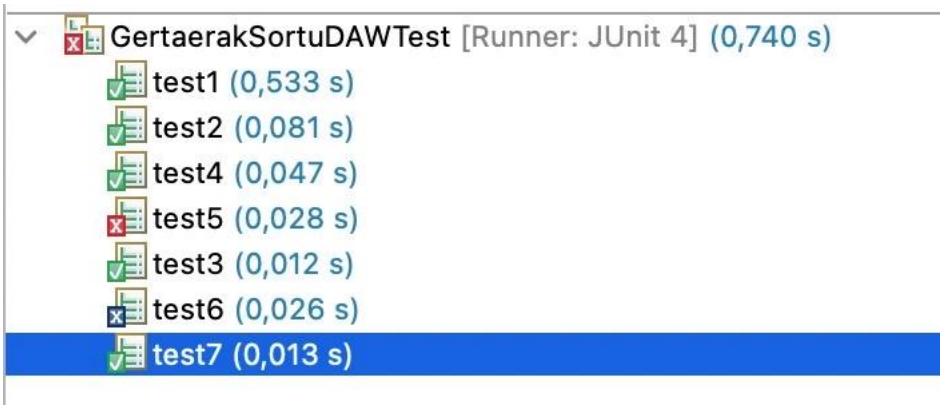
Valor esperado -> false

Valor obtenido -> true



Test 2:

Caja Blanca:



Test 6:

Valor esperado -> false

Valor obtenido -> true

Test 5:

Pruebas de Integración

Diseño

Implementación

Resultados

Método 2: EmaitzakIpini()

Código

```
public void EmaitzakIpini(Quote quote) throws EventNotFinished{

    Quote q = db.find(Quote.class, quote);
    String result = q.getForecast();

    if(new Date().compareTo(q.getQuestion().getEvent().getEventDate())<0)
        throw new EventNotFinished();

    Vector<Apustua> listApustuak = q.getApustuak();
    db.getTransaction().begin();
    Question que = q.getQuestion();
    Question question = db.find(Question.class, que);
    question.setResult(result);
    for(Quote quo: question.getQuotes()) {
        for(Apustua apu: quo.getApustuak()) {

            Boolean b=apu.galdutaMarkatu(quo);
            if(b) {
                apu.getApustuAnitza().setEgoera("galduta");
            }else {
                apu.setEgoera("irabazita");
            }
        }
    }
    db.getTransaction().commit();
    for(Apustua a : listApustuak) {
        db.getTransaction().begin();
        Boolean bool=a.getApustuAnitza().irabazitaMarkatu();
        db.getTransaction().commit();
        if(bool) {
            this.ApustuaIrabazi(a.getApustuAnitza());
        }
    }
}
```

Pruebas Unitarias

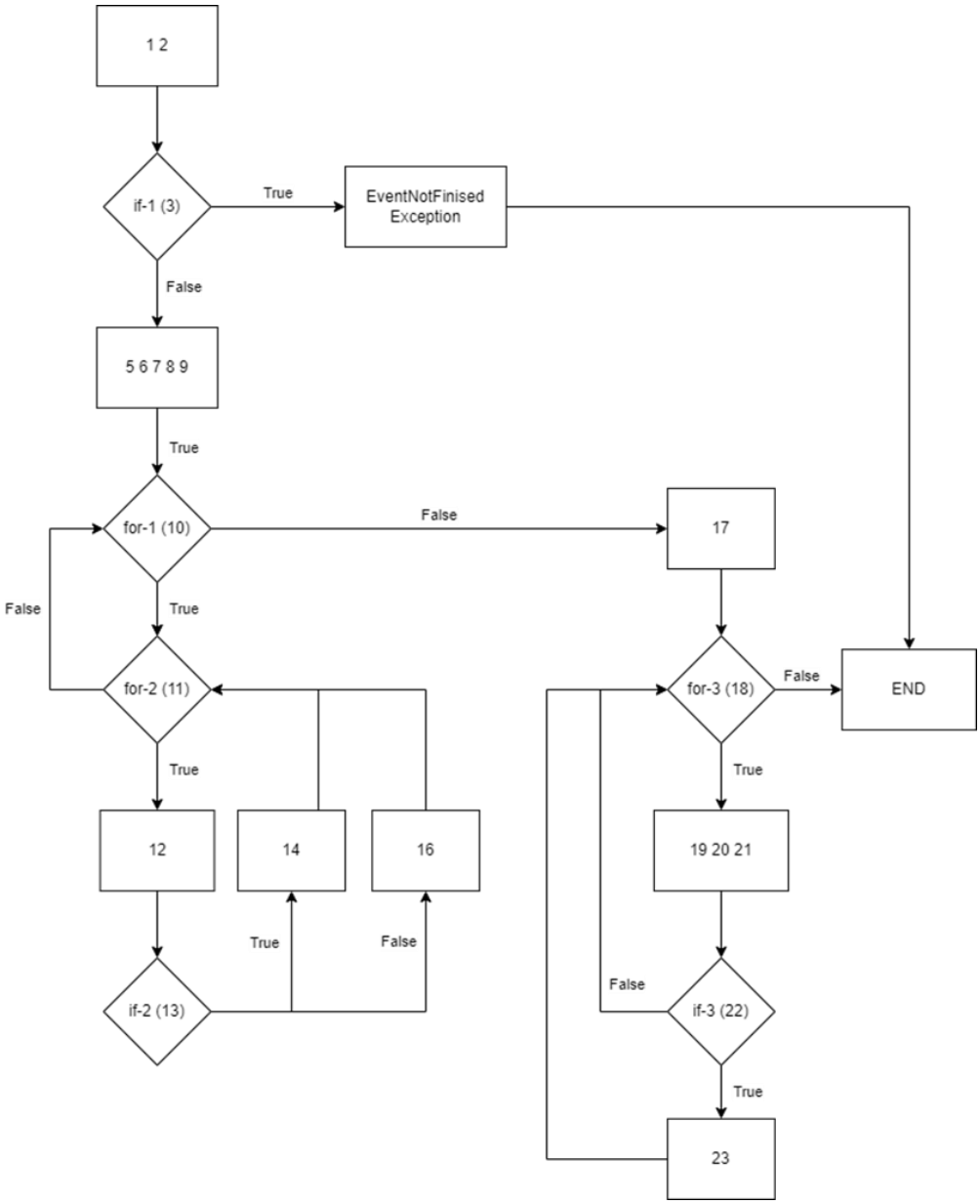
Diseño

Base de Datos Utilizada

Base de datos Caja negra					
Team1	Atletico (nombre)				
Team2	Athletic (nombre)				
Event1	1 (numero de evento)	Atletico-Athletic (descripcion)	17/01/2023 (fecha)	team1 (local)	team2 (visitante)
Event2	2 (numero de evneto)	Atletico-Athletic (descripcion)	17/01/2021 (fecha)	team1 (local)	team2 (visitante)
Question1	¿Quién ganara el partido? (pregunta)	1.0 (apuesta minima)	Event1 (event)		
Question2	¿Quién marcara primero? (pregunta)	2.0 (apuesta minima)	Event2 (event)		
Quote1	100 (valor)	2 (forecast)	Question1 (pregunta)		
Quote2	20 (valor)	2 (forecast)	Question2 (pregunta)		
Quote3	50 (valor)	1 (forecast)	Question1 (pregunta)		
ApustuAnitza1	registrado (estado)	5.0			
Apuesta1	ApustuAnitza1	Quote3			
ApustuAnitza2	registrado (estado)	3.0			
Apuesta2	ApustuAnitza2	Quote1			

Base de datos Caja blanca					
Team1	Atletico (nombre)				
Team2	Athletic (nombre)				
Event1	3 (numero de evento)	Atletico-Athletic (descripcion)	17/01/2024 (fecha)	team1 (local)	team2 (visitante)
Event2	2 (numero de evneto)	Atletico-Athletic (descripcion)	17/01/2021 (fecha)	team1 (local)	team2 (visitante)
Question1	¿Quién ganara? (pregunta)	1.0 (apuesta minima)	Event1 (event)		
Question2	¿Quién marcara primero? (pregunta)	2.0 (apuesta minima)	Event2 (event)		
Question3	¿Quién marcara primero? (pregunta)	2.0 (apuesta minima)	Event1 (event)		
Question4	¿Quién marcara ultimo? (pregunta)	2.0 (apuesta minima)	Event1 (event)		
Quote1	10 (valor	X (forecast)	Question1 (pregunta)		
Quote2	20 (valor)	2 (forecast)	Question2 (pregunta)		
Quote3	50 (valor)	2 (forecast)	Question1 (pregunta)		
Quote4	20 (valor)	2 (forecast)	Question3 (pregunta)		
Quote5	5 (valor)	2 (forecast)	Question4 (pregunta)		
Quote6	2 (valor)	1 (forecast)	Question4 (pregunta)		
ApustuAnitza1	registrado (estado)	5.0			
Apuesta1	ApustuAnitza1	Quote3			
ApustuAnitza2	registrado (estado)	2.0			
Apuesta2	ApustuAnitza2	Quote4			
ApustuAnitza3	registrado (estado)	40.0			
Apuesta3	ApustuAnitza3	Quote5			
Apuesta4	ApustuAnitza3	Quote6			

Caja Blanca



V(G) = 7

Caminos:

1- 1 2 if-1(T) Except. END

2- 1 2 if-1(F) 5 6 7 8 9 for-1(T) for-2(T) 12 if-2(F) 16 for-2(F) for-1(F) 17 for-3(T) 19 20 21 if-3(F) for-3(F) END

3- 1 2 if-1(F) 5 6 7 8 9 for-1(T) for-2(T) 12 if-2(T) 14 for-2(F) for-1(T) for-2(F) for-1(F) 17 for-3(F) END

4- 1 2 if-1(F) 5 6 7 8 9 for-1(T) for-2(T) 12 if-2(F) 16 for-2(F) for-1(F) 17 for-3(T) 19 20 21 if-3(T) for-3(F) END

5- 1 2 if-1(F) 5 6 7 8 9 for-1(T) for-2(T) 12 if-2(F) 16 for-2(F) for-1(T) for-2(T) 12 if-2(T) 14 for-2(F) for-1(F) 17 for-3(T) 19 20 21 if-3(F) for-3(F) END

#caso	Condición de Entrada	Valor Entrada	Valor Salida
1	todayDate < eventDate	quo1	EventNotFinished exception
2	question1.quotes.size() = 1	quo2	No hay salida
3	question1.quotes.size() = 2 quote3.apustuak.size() = 1 apuesta1.galdutaMarkatu(Quote1) = true	Quote1	No hay salida
4	question3.quotes.size() = 1 quote4.apustuak.size() = 1 apuesta2.galdutaMarkatu(Quote4) = false Quote4.apustuak.size() = 1 apuesta2.getApustuAnitza().irabazitaMarkatu() = true	Quote4	No hay salida
5	question4.quotes.size() = 2 quote5.apustuak.size() = 1 quote6.apustuak.size() = 1 apuesta3.galdutaMarkatu(Quote4) = false apuesta4.galdutaMarkatu(Quote4) = true Quote5.apustuak.size() = 1 apuesta3.getApustuAnitza().irabazitaMarkatu() = false	Quote5	No hay salida

Nota: Por razones de la implementación del método, estos caminos son inaccesibles. Esto hace que no se vaya a conseguir un 100% de cobertura del método mediante los tests.

Caja Negra

Condición de Entrada	Clases de Equivalencia Válidas	Clases de Equivalencia No Válidas
¿La fecha del evento es posterior?	true (1)	false (2)
¿Apuesta perdida?	true (3)	
¿Apuesta ganada?	true (4)	

		Contexto de Entrada		Contexto de Salida	
#	Clases Cubiertas	Estado BD	Parámetros	Estado BD	Salida
1	1, 3, 4	Quote siempre va a estar en BD (porque es un parámetro que se pasa desde la interfaz y no puede ser null i no estar en la BBDD)	Quote3	question1.resultado = 1 apuesta1.egoera = "irabazita" apuesta2.apustuAnitza2.egoera = "galduta" this.ApustuaIrabazi(apuesta1.getApustuAnitza())	No hay salida
2	2		Quote2	No cambia	EventNotFinished

Implementación

Caja Blanca

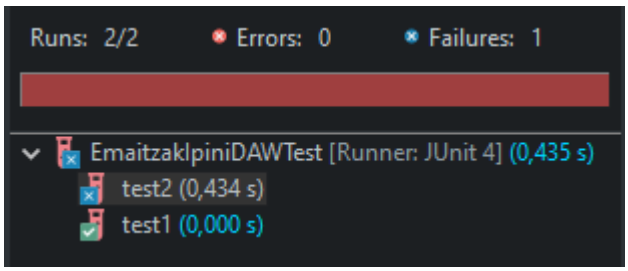
```
@Before
public void initialize() {

  eve1 = sut.getEventsAll().get(0);
  q1 = eve1.getQuestions().get(0);
  quo1 = q1.getQuotes().get(0);

  eve2 = sut.getEventsAll().get(0);
  q2 = eve2.getQuestions().get(1);
  quo2 = q2.getQuotes().get(0);
  ap2 = quo2.getApustuak().get(0);

  eve3 = sut.getEventsAll().get(21);
  q3 = eve3.getQuestions().get(0);
  quo3 = q3.getQuotes().get(0);
  ap4 = quo3.getApustuak().get(0);

}

/*Camino salta excepcion por fecha del
  evento todavia no finalizada **/
@Test
public void test1() {
  Date fecha = eve1.getEventDate();
  eve1.setEventDate(UtilDate.newDate(2023, 11, 23));

  try {
    sut.EmaitzakIpini(quo1);

  }catch(Exception e) {
      System.out.println("EventNotFinished");
      assertTrue(e instanceof EventNotFinished);
  }
  try {
    eve1.setEventDate(fecha);

  } catch (Exception e) {
    fail("No es posible");
  }
}
```

```
/* Camino para comprobar el resultado de la apuesta ganada y el resultado de la pregunta se ponen correctamente
 * (en el if(b) no va a entrar nunca debido a que llama al metodo galdutaMarcatu con el parametro quo) **/
@Test
public void test2() {

  String s1 = ap2.getEgoera();
  //String s2 = ap3.getApustuAnitza().getEgoera();

  ap2.setEgoera("jokoan");
  //ap3.setEgoera("jokoan");

  Date fecha = eve2.getEventDate();
  eve2.setEventDate(UtilDate.newDate(2021, 8, 7));

  try {
    sut.EmaitzakIpini(quo2);
  } catch (Exception EventNotFinished) {
    System.out.println("EventNotFinished");
    //assertTrue(true);
  }

  String expected = quo2.getForecast();
  String obtained = q2.getResult();

  String expected2 = "irabazita";
  String obtained2 = ap2.getEgoera();


  assertEquals(expected, obtained);
  assertEquals(expected2, obtained2);

  try {

    eve2.setEventDate(fecha);
    quo2.getQuestion().setResult("");
    ap2.setEgoera(s1);

  } catch (Exception e) {
    fail("No es posible");
  }
}


 @Test /* Camino para comprobar que la ApustuAnitza se gana **/
 public void test4() {
   System.out.println(ap4.getApustuAnitza().getApustuak().size());

   String s1 = ap4.getEgoera();
   String s2 = ap4.getApustuAnitza().getEgoera();

   ap4.setEgoera("jokoan");
   ap4.getApustuAnitza().setEgoera("jokoan");

   Date fecha = eve3.getEventDate();
   eve3.setEventDate(UtilDate.newDate(2021, 8, 7));

   try {
     sut.EmaitzakIpini(quo3);
   } catch (Exception EventNotFinished) {
     System.out.println("EventNotFinished");
     fail("No es posible")
   }


    String expected = quo3.getForecast();
    String obtained = q3.getResult();

    String expected2 = "irabazita";
    String obtained2 = ap4.getEgoera();

    String expected3 = "irabazita";
    String obtained3 = ap4.getApustuAnitza().getEgoera();

    assertEquals(expected, obtained);
    assertEquals(expected2, obtained2);
    assertEquals(expected3, obtained3);

    try {

      eve3.setEventDate(fecha);
      quo3.getQuestion().setResult("");
      ap4.setEgoera(s1);
      ap4.getApustuAnitza().setEgoera(s2);

    } catch (Exception e) {
      fail("No es posible");
    }
 }
```

Caja Negra

```
@Before
public void initialize() {

  eve1 = sut.getEventsAll().get(0);
  q1 = eve1.getQuestions().get(0);
  quo1 = q1.getQuotes().get(0);

  eve2 = sut.getEventsAll().get(0);
  q2 = eve2.getQuestions().get(1);
  quo2 = q2.getQuotes().get(0);
}


/* Camino en el que la cuota pasada por parametro no esta en BD **/
@Test
public void test0() {

  Quote quo = new Quote(1.5, "X", q1);

  try {
    sut.EmaitzakIpini(quo);
    fail("No es posible")
  } catch (Exception e) {
    assertTrue(true);
    System.out.println("Esa cuota no esta en la base de datos ");
  }
}

/* Camino para comprobar que el resultado
de la pregunta se pone correctamente  **/
@Test
public void test1() {

  Date fecha = eve2.getEventDate();
  eve2.setEventDate(UtilDate.newDate(2021, 8, 7));

  try {
    sut.EmaitzakIpini(quo2);
  } catch (Exception EventNotFinished) {
    System.out.println("EventNotFinished");
    fail("No es posible");
  }

  String expected = quo2.getForecast();
  String obtained = q2.getResult();

  assertEquals(expected, obtained);

  try {
    eve2.setEventDate(fecha);
    quo2.getQuestion().setResult("");
  } catch (Exception e) {
    fail("No es posible");
  }
}

/* Camino salta excepcion por fecha del
   evento todavia no finalizada **/
@Test
public void test2() {

  Date fecha = eve1.getEventDate();
  eve1.setEventDate(UtilDate.newDate(2023, 11, 23));

  try {
    sut.EmaitzakIpini(quo1);

  } catch (Exception EventNotFinished) {

    System.out.println("EventNotFinished");
    assertTrue(true);
  }

  try {
    eve1.setEventDate(fecha);
  } catch (Exception e) {
    fail("No es posible");
  }
}
```

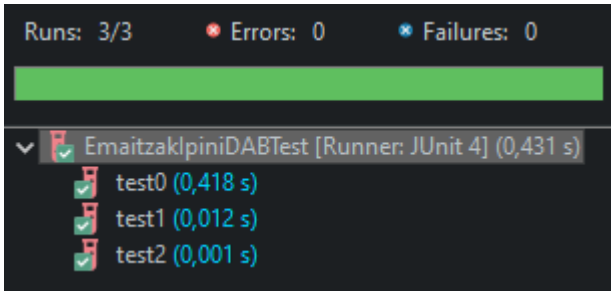
Resultados



En este caso no se ha podido pasar el test porque existen ramas inaccesibles en el método.

Valor esperado -> “galduta”

Valor obtenido -> “jokoan”



Pruebas de Integración

Diseño

Se comprobarán los mismos casos que en la caja negra del método `EmaitzakIpini()` de la clase DataAccess.

Implementación

```
public class EmaitzakIpiniMockIntTest {

  DataAccess mock = Mockito.mock(DataAccess.class);
  BLFacadeImplementation sut = new BLFacadeImplementation(mock);

  Team team1 = new Team("Atletico");
  Team team2 = new Team("Athletic");
  Event ev1 = new Event(1, "Atletico-Athletic", UtilDate.newDate(2022, 1, 17), team1, team2);
  Question q1 = new Question("¿Quién ganará el partido?", 1, ev1);
  Quote quote1 = q1.addQuote(1.3, "1", q1);

  @Test
  public void test1() {

    Registered reg1 = new Registered("registered", "123", 1234);
    ApustuAnitza apA1 = new ApustuAnitza(reg1, 5.0);
    Apustua ap1 = new Apustua(apA1, quote1);
    apA1.addApustua(ap1);

    ArgumentCaptor < Quote > cuota1 = ArgumentCaptor.forClass(Quote.class);

    try {
      Mockito.verify(mock, Mockito.times(1)).EmaitzakIpini(cuota1.capture());
    } catch (EventNotFinished e) {
      e.printStackTrace();
    }
    assertEquals(quote1, cuota1.getValue());

  }

  @Test
  public void test2() {

    try {
      Mockito.doReturn(new EventNotFinished()).when(mock).EmaitzakIpini(quote1);
      sut.EmaitzakIpini(quote1);

    } catch (EventNotFinished e1) {
      assertTrue(true);
      System.out.println("EventNotFinished");

    }

    ArgumentCaptor < Quote > cuota2 = ArgumentCaptor.forClass(Quote.class);
    try {
      Mockito.verify(mock, Mockito.times(1)).EmaitzakIpini(cuota2.capture());
    } catch (EventNotFinished e) {
      fail("No es posible");
    }
    assertEquals(quote1, cuota2.getValue());
  }

}
```

Enlaces

GitHub: https://github.com/anadurlen/IS2Bets

Sonarcloud: https://sonarcloud.io/project/overview?id=IS2Bets

Nota: Hemos realizado esta práctica junto con otro grupo, siendo sus integrantes Pablo Martinez Amunarri y Raúl Silva Alonso.