

# [ REFACTORIZACIÓN ]

**Ingeniería del Software II  
Grado en Ingeniería Informática  
Facultad de Informática  
Universidad del País Vasco**

Martinez Amunarriz, Pablo  
Silva Alonso, Raúl  
Viteri Sainz De Ugarte, Martín

16/10/2022

# Raúl Silva Alonso

## 1. Write short units of code

Código inicial:

```
I 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());
        }
    }
}
```

Código refactorizado:

```

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();
    marcarApuestas(q, result);
    calcularApuestas(listApustuak);
}

private void calcularApuestas(Vector<Apustua> listApustuak) {
    for(Apustua a : listApustuak) {
        db.beginTransaction().begin();
        Boolean bool=a.getApustuAnitza().irabazitaMarkatu();
        db.beginTransaction().commit();
        if(bool) {
            this.ApustuaIrabazi(a.getApustuAnitza());
        }
    }
}

private void marcarApuestas(Quote q, String result) {
    db.beginTransaction().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.beginTransaction().commit();
}

```

He creado dos métodos auxiliares para reducir el código del método “EmaitzakIpini” y que se entienda mejor. En el método “marcarApuestas” he dejado la parte del código donde asigna a cada apuesta si ha ganado o perdido y en el método “calcularApuestas” la parte donde se calcula la cantidad ganada de las apuestas que han ganado. Gracias a esto, podremos utilizar los métodos auxiliares desde otros métodos de la clase en caso necesario.

## 2. Write Simple Units Of Code

Código inicial:

```
I 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.beginTransaction().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("galdua");
            }else {
                apu.setEgoera("irabazita");
            }
        }
    }
    db.beginTransaction().commit();
    for(Apustua a : listApustuak) {
        db.beginTransaction().begin();
        Boolean bool=a.getApustuAnitza().irabazitaMarkatu();
        db.beginTransaction().commit();
        if(bool) {
            this.ApustuaIrabazi(a.getApustuAnitza());
        }
    }
}
```

Código refactorizado:

```

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();
    marcarApuestas(q, result);
    calcularApuestas(listApustuak);
}

private void calcularApuestas(Vector<Apustua> listApustuak) {
    for(Apustua a : listApustuak) {
        db.beginTransaction().begin();
        Boolean bool=a.getApustuAnitza().irabazitaMarkatu();
        db.beginTransaction().commit();
        if(bool) {
            this.ApustuaIrabazi(a.getApustuAnitza());
        }
    }
}

private void marcarApuestas(Quote q, String result) {
    db.beginTransaction().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.beginTransaction().commit();
}

```

Al igual que he realizado en el caso anterior para reducir las líneas de código del método en este caso también he creado dos métodos auxiliares que me han servido para minimizar la complejidad ciclomática del método emaitzaklpini a uno. Esto facilita el mantenimiento y uso de estas funciones.

### 3. Duplicate Code

Código inicial:

DataAccess.java	Define a constant instead of duplicating this literal "DiruaSartu" 4 times. [+4 locations]
-----------------	--------------------------------------------------------------------------------------------

```

this.DiruaSartu(reg1, 50.0, new Date(), "DiruaSartu");
this.DiruaSartu(reg2, 50.0, new Date(), "DiruaSartu");
this.DiruaSartu(reg3, 50.0, new Date(), "DiruaSartu");
this.DiruaSartu(reg4, 50.0, new Date(), "DiruaSartu");

```

Código refactorizado:

```

String meterDinero = "DiruaSartu";
this.DiruaSartu(reg1, 50.0, new Date(), meterDinero);
this.DiruaSartu(reg2, 50.0, new Date(), meterDinero);
this.DiruaSartu(reg3, 50.0, new Date(), meterDinero);
this.DiruaSartu(reg4, 50.0, new Date(), meterDinero);

```

He creado una variable con el valor que se repetía.

#### 4. Keep unit interfaces small

Código inicial:

```

public void DiruaSartu(Registered u, Double dirua, Date data, String mota) {
    Registered user = (Registered) db.find(Registered.class, u.getUsername());
    db.beginTransaction().begin();
    Transaction t = new Transaction(user, dirua, data, mota);
    System.out.println(t.getMota());
    user.addTransaction(t);
    user.updateDiruKontua(dirua);
    db.persist(t);
    db.getTransaction().commit();
}

@WebMethod
public void DiruaSartu(Registered u, Double dirua, String mota) {
    Date data = new Date();
    dbManager.open(false);
    dbManager.DiruaSartu(u, dirua, data, mota);
    dbManager.close();
}

```

Código refactorizado:

```

public void DiruaSartu(Registered u, Double dirua, String mota) {
    Date data = new Date();
    dbManager.open(false);
    DineroFechaMota d = new DineroFechaMota(dirua, data, mota);
    dbManager.DiruaSartu(u, d);
    dbManager.close();
}

public void DiruaSartu(Registered u, DineroFechaMota d) {
    Registered user = (Registered) db.find(Registered.class, u.getUsername());
    db.beginTransaction().begin();
    Transaction t = new Transaction(user, d.getDirua(), d.getData(), d.getMota());
    System.out.println(t.getMota());
    user.addTransaction(t);
    user.updateDiruKontua(d.getDirua());
    db.persist(t);
    db.getTransaction().commit();
}

```

Como solo había un método con más de 4 parámetros he elegido uno con 4. He creado la clase `DineroFechaMota` para almacenar 3 de estos cuatro parámetros. Además, he tenido que cambiar esto en las llamadas al método `DiruaSartu` en todas las clases, como por ejemplo `FacadeImplementation`.

# Pablo Martinez Amunarriz

## 1. Write short units of code

Código inicial:

```
0  public boolean gertaerakSortu(String description, Date eventDate, String sport) {
1      boolean b = true;
2      db.beginTransaction().begin();
3      Sport spo = db.find(Sport.class, sport);
4      if(spo!=null) {
5          TypedQuery<Event> Equery = db.createQuery("SELECT e FROM Event e WHERE e.getEventDate() =?1 ",Event.class);
6          Equery.setParameter(1, eventDate);
7          for(Event ev: Equery.getResultList()) {
8              if(ev.getDescription().equals(description)) {
9                  b = false;
10             }
11         }
12         if(b) {
13             String[] taldeak = description.split("-");
14             Team lokala = new Team(taldeak[0]);
15             Team kanpokoa = new Team(taldeak[1]);
16             Event e = new Event(description, eventDate, lokala, kanpokoa);
17             e.setSport(spo);
18             spo.addEvent(e);
19             db.persist(e);
20         }
21     }else {
22         return false;
23     }
24     db.getTransaction().commit();
25     return b;
END }
```

Código refactorizado:

```
public boolean sportIsNull(String sport) {
    Sport spo = db.find(Sport.class, sport);
    if(spo!=null) return true;
    else return false;
}

public boolean mismaDescripcion(String description, Date eventDate) {
    boolean b = true;
    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;
        }
    }
    return b;
}

public void createEvent(String description, Date eventDate, String sport) {
    db.beginTransaction().begin();
    String[] taldeak = description.split("-");
    Team lokala = new Team(taldeak[0]);
    Team kanpokoa = new Team(taldeak[1]);
    Event e = new Event(description, eventDate, lokala, kanpokoa);
    Sport spo = db.find(Sport.class, sport);
    e.setSport(spo);
    spo.addEvent(e);
    db.persist(e);
}

public boolean gertaerakSortu(String description, Date eventDate, String sport) {
    db.beginTransaction().begin();
    boolean b;
    boolean sp = this.sportIsNull(sport);
    if(sp) {
        b = this.mismaDescripcion(description, eventDate);
        if(b) {
            this.createEvent(description, eventDate, sport);
        }
    }else {
        return false;
    }
    db.getTransaction().commit();
    return b;
}
```

He creado tres métodos auxiliares, uno que comprueba que “sport” esté en la BD, otro para comprobar si hay algún “event” con la misma descripción en el día pasado como parámetro

y otro para crear el evento. De esta manera he podido reducir el código de 25 líneas a 13 líneas.

## 2. Write Simple Units Of Code

Código inicial:

```
0  public boolean gertaerakSortu(String description, Date eventDate, String sport) {
1      boolean b = true;
2      db.beginTransaction().begin();
3      Sport spo = db.find(Sport.class, sport);
4      if(spo!=null) {
5          TypedQuery<Event> Equery = db.createQuery("SELECT e FROM Event e WHERE e.getEventDate() =?1 ",Event.class);
6          Equery.setParameter(1, eventDate);
7          for(Event ev: Equery.getResultList()) {
8              if(ev.getDescription().equals(description)) {
9                  b = false;
10             }
11         }
12     if(b) {
13         String[] taldeak = description.split("-");
14         Team lokala = new Team(taldeak[0]);
15         Team kanpokoa = new Team(taldeak[1]);
16         Event e = new Event(description, eventDate, lokala, kanpokoa);
17         e.setSport(spo);
18         spo.addEvent(e);
19         db.persist(e);
20     }
21 }else {
22     return false;
23 }
24 db.beginTransaction().commit();
25 return b;
END }
```

Código refactorizado:

```
public boolean sportIsNull(String sport) {
    Sport spo = db.find(Sport.class, sport);
    if(spo!=null) return true;
    else return false;
}

public boolean mismaDescripcion(String description, Date eventDate) {
    boolean b = true;
    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;
        }
    }
    return b;
}

public void createEvent(String description, Date eventDate, String sport) {
    db.beginTransaction();
    String[] taldeak = description.split("-");
    Team lokala = new Team(taldeak[0]);
    Team kanpokoa = new Team(taldeak[1]);
    Event e = new Event(description, eventDate, lokala, kanpokoa);
    Sport spo = db.find(Sport.class, sport);
    e.setSport(spo);
    spo.addEvent(e);
    db.persist(e);
}

public boolean gertaerakSortu(String description, Date eventDate, String sport) {
    db.beginTransaction();
    boolean b;
    boolean sp = this.sportIsNull(sport);
    if(sp) {
        b = this.mismaDescripcion(description, eventDate);
        if(b) {
            this.createEvent(description, eventDate, sport);
        }
    }else {
        return false;
    }
    db.beginTransaction().commit();
    return b;
}
```

Implementando estos tres métodos mencionados antes, a parte de solucionar el anterior “Bad Smell” también he podido reducir la complejidad ciclomática de este método a 3.

## 3. Duplicate Code

Código inicial:

```
Transaction t1 = new Transaction(reg1, apA1.getBalioa(), new Date(), "ApustuaEgin");
Transaction t3 = new Transaction(reg2, apA4.getBalioa(), new Date(), "ApustuaEgin");
Transaction t4 = new Transaction(reg3, apA5.getBalioa(), new Date(), "ApustuaEgin");
Transaction t5 = new Transaction(reg4, apA3.getBalioa(), new Date(), "ApustuaEgin");
Transaction t6 = new Transaction(reg4, apA6.getBalioa(), new Date(), "ApustuaEgin");
Transaction t7 = new Transaction(reg1, apA7.getBalioa(), new Date(), "ApustuaEgin");
Transaction t8 = new Transaction(reg1, apA8.getBalioa(), new Date(), "ApustuaEgin");
Transaction t9 = new Transaction(reg2, apA9.getBalioa(), new Date(), "ApustuaEgin");
Transaction t10 = new Transaction(reg2, apA10.getBalioa(), new Date(), "ApustuaEgin");
Transaction t11 = new Transaction(reg3, apA11.getBalioa(), new Date(), "ApustuaEgin");
Transaction t12 = new Transaction(reg3, apA12.getBalioa(), new Date(), "ApustuaEgin");
```

Código refactorizado:

```
String hacerApuesta = "ApustuaEgin";
Transaction t1 = new Transaction(reg1, apA1.getBalioa(), new Date(), hacerApuesta);
Transaction t3 = new Transaction(reg2, apA4.getBalioa(), new Date(), hacerApuesta);
Transaction t4 = new Transaction(reg3, apA5.getBalioa(), new Date(), hacerApuesta);
Transaction t5 = new Transaction(reg4, apA3.getBalioa(), new Date(), hacerApuesta);
Transaction t6 = new Transaction(reg4, apA6.getBalioa(), new Date(), hacerApuesta);
Transaction t7 = new Transaction(reg1, apA7.getBalioa(), new Date(), hacerApuesta);
Transaction t8 = new Transaction(reg1, apA8.getBalioa(), new Date(), hacerApuesta);
Transaction t9 = new Transaction(reg2, apA9.getBalioa(), new Date(), hacerApuesta);
Transaction t10 = new Transaction(reg2, apA10.getBalioa(), new Date(), hacerApuesta);
Transaction t11 = new Transaction(reg3, apA11.getBalioa(), new Date(), hacerApuesta);
Transaction t12 = new Transaction(reg3, apA12.getBalioa(), new Date(), hacerApuesta);
```

He creado una variable con el valor que se repetía.

#### 4. Keep unit interfaces small

Debido a que en “dataAccess” no hay ningún método que tenga 5 parámetros he cogido un método de la clase “TeamRenderer” que pertenece al paquete “domain”.

Método inicial:

```
package domain;

import java.awt.Component;

public class TeamRenderer extends JLabel implements ListCellRenderer<Team> {

    private BLFacade businessLogic = MainGUI.getBusinessLogic();

    public TeamRenderer() {
        setOpaque(true);
    }

    public Component getListCellRendererComponent(JList<? extends Team> list, Team team, int index, boolean isSelected, boolean cellHasFocus) {

        ImageIcon imageIcon = new ImageIcon(".\\src\\main\\resources\\Equipos\\"+team.getIzena()+" .png"); // load the image to a ImageIcon
        Image image = imageIcon.getImage(); // transform it
        Image newimg = image.getScaledInstance(90, 90, Image.SCALE_SMOOTH); // scale it the smooth way
        imageIcon = new ImageIcon(newimg);
        setIcon(imageIcon);
        setText(team.getIzena());
        if(isSelected) {
            setBackground(list.getSelectionBackground().GRAY);
            setForeground(list.getSelectionForeground().WHITE);
        }else {
            setBackground(list.getSelectionBackground());
            setForeground(list.getSelectionForeground());
        }

        return this;
    }
}
```

En este método entra como parámetro un entero “index” el cual no es referenciado en todo el método por lo que he decidido quitarlo. El problema al quitar este parámetro es que la clase implementa un método de una interfaz. Para solucionarlo, he hecho que no

implemente el método de la interfaz. De esta manera el método seguirá teniendo el mismo funcionamiento.

```

package domain;

import java.awt.Component;

public class TeamRenderer extends JLabel {
    private BLFacade businessLogic = MainGUI.getBusinessLogic();

    public TeamRenderer() {
        setOpaque(true);
    }

    public Component getListCellRendererComponent(JList<? extends Team> list, Team team, boolean isSelected, boolean cellHasFocus) {

        ImageIcon imageIcon = new ImageIcon(".\\src/main/resources\\Equipos\\"+team.getIzena()+" .png"); // load the image to a ImageIcon
        Image image = imageIcon.getImage(); // transform it
        Image newimg = image.getScaledInstance(90, 90, Image.SCALE_SMOOTH); // scale it in the smooth way
        imageIcon = new ImageIcon(newimg);
        setIcon(imageIcon);
        setText(team.getIzena());
        if(isSelected) {
            setBackground(list.getSelectionBackground().GRAY);
            setForeground(list.getSelectionForeground().WHITE);
        }else {
            setBackground(list.getSelectionBackground());
            setForeground(list.getSelectionForeground());
        }

        return this;
    }
}

```

## Martín Viteri Sainz de Ugarte

### 1. "Write short units of code"

Código inicial:

```

public boolean borrarEvento(Event ev) {
    Event event = db.find(Event.class, ev);
    boolean resultB = true;
    List<Question> listQ = event.getQuestions();

    for(Question q : listQ) {
        if(q.getResult() == null) {
            resultB = false;
        }
    }
    if(resultB == false) {
        return false;
    }else if(new Date().compareTo(event.getEventDate())<0) {
        TypedQuery<Quote> Qquery = db.createQuery("SELECT q FROM Quote q WHERE q.getQuestion().getEvent().getEvent
        Qquery.setParameter(1, event.getEventNumber());
        List<Quote> listQUO = Qquery.getResultList();

        for(int j=0; j<listQUO.size(); j++) {
            Quote quo = db.find(Quote.class, listQUO.get(j));
            for(int i=0; i<quo.getApustuak().size(); i++) {
                ApustuAnitza apustuAnitza = quo.getApustuak().get(i).getApustuAnitza();
                ApustuAnitza ap1 = db.find(ApustuAnitza.class, apustuAnitza.getApustuAnitzaNumber());
                db.beginTransaction().begin();
                ap1.removeApustua(quo.getApustuak().get(i));
                db.getTransaction().commit();
                if(ap1.getApustuak().isEmpty() && !ap1.getEgoera().equals("galdua")) {
                    this.borrarApuesta(ap1.getUser(), ap1);
                }else if(!ap1.getApustuak().isEmpty() && ap1.irabazitaMarkatu()){
                    this.ganarApuesta(ap1);
                }
                db.beginTransaction().begin();
                Sport spo = quo.getQuestion().getEvent().getSport();

```

```

        spo.setApustuKantitatea(spo.getApustuKantitatea()-1);
        db.getTransaction().commit();
    }

}
db.beginTransaction();
db.remove(event);
db.beginTransaction().commit();
return true;
}

```

He creado un método nuevo que complete las apuestas del evento, y he eliminado un if al principio del método que era incensario.

Código refactorizado:

```

public boolean borrarEvento(Event ev) {
    Event event = db.find(Event.class, ev);
    boolean resultB = true;
    List<Question> listQ = event.getQuestions();
    for(Question q : listQ) {
        if(q.getResult() == null) {
            return false;
        }
    }
    if(new Date().compareTo(event.getEventDate())<0) {
        TypedQuery<Quote> Qquery = db.createQuery("SELECT q FROM Quote q WHERE q.getQuestion().getEvent(..));
        Qquery.setParameter(1, event.getEventNumber());
        List<Quote> listQUO = Qquery.getResultList();
        for(int j=0; j<listQUO.size(); j++) {
            Quote quo = db.find(Quote.class, listQUO.get(j));
            completarApuesta(quo);
        }
    }
    db.beginTransaction();
    db.remove(event);
    db.beginTransaction().commit();
    return true;
}

public void completarApuesta(Quote quo) {
    for(int i=0; i<quo.getApustuak().size(); i++) {
        ApustuAnitza apustuAnitza = quo.getApustuak().get(i).getApustuAnitza();
        ApustuAnitza ap1 = db.find(ApustuAnitza.class, apustuAnitza.getApustuAnitzaNumber());
        db.beginTransaction().begin();
        ap1.removeApustua(quo.getApustuak().get(i));
        if(ap1.getApustuak().isEmpty() && !ap1.getEgoera().equals("galduta")) {
            this.borrarApuesta(ap1.getUser(), ap1);
        }else if(!ap1.getApustuak().isEmpty() && ap1.irabazitaMarkatu()){
            this.ganarApuesta(ap1);
        }
        Sport spo = quo.getQuestion().getEvent().getSport();
        spo.setApustuKantitatea(spo.getApustuKantitatea()-1);
        db.beginTransaction().commit();
    }
}

```

## 2. Write Simple Units Of Code

Código inicial:

```

public boolean borrarEvento(Event ev) {
    Event event = db.find(Event.class, ev);
    boolean resultB = true;
    List<Question> listQ = event.getQuestions();

    for(Question q : listQ) {
        if(q.getResult() == null) {
            resultB = false;
        }
    }
    if(resultB == false) {
        return false;
    }else if(new Date().compareTo(event.getEventDate())<0) {
        TypedQuery<Quote> Qquery = db.createQuery("SELECT q FROM Quote q WHERE q.getQuestion().getEvent().getEvent
        Qquery.setParameter(1, event.getEventNumber());
        List<Quote> listQUO = Qquery.getResultList();

        for(int j=0; j<listQUO.size(); j++) {
            Quote quo = db.find(Quote.class, listQUO.get(j));
            for(int i=0; i<quo.getApustuak().size(); i++) {
                ApustuAnitza apustuAnitza = quo.getApustuak().get(i).getApustuAnitza();
                ApustuAnitza ap1 = db.find(ApustuAnitza.class, apustuAnitza.getApustuAnitzaNumber());
                db.getTransaction().begin();
                ap1.removeApustua(quo.getApustuak().get(i));
                db.getTransaction().commit();
                if(ap1.getApustuak().isEmpty() && !ap1.getEgoera().equals("galduta")) {
                    this.borrarApuesta(ap1.getUser(), ap1);
                }else if(!ap1.getApustuak().isEmpty() && ap1.irabazitaMarkatu()){
                    this.ganarApuesta(ap1);
                }
                db.getTransaction().begin();
                Sport spo = quo.getQuestion().getEvent().getSport();
                spo.setApustuKantitatea(spo.getApustuKantitatea()-1);
                db.getTransaction().commit();
            }
        }
        db.getTransaction().begin();
        db.remove(event);
        db.getTransaction().commit();
        return true;
    }
}

```

Eliminado el if del principio y creando el nuevo método, la complejidad ciclomática del método baja de 6 a 3.

```

public boolean borrarEvento(Event ev) {
    Event event = db.find(Event.class, ev);
    boolean resultB = true;
    List<Question> listQ = event.getQuestions();
    for(Question q : listQ) {
        if(q.getResult() == null) {
            return false;
        }
    }
    if(new Date().compareTo(event.getEventDate())<0) {
        TypedQuery<Quote> Qquery = db.createQuery("SELECT q FROM Quote q WHERE q.getQuestion().getEvent().getEvent
        Qquery.setParameter(1, event.getEventNumber());
        List<Quote> listQUO = Qquery.getResultList();
        for(int j=0; j<listQUO.size(); j++) {
            Quote quo = db.find(Quote.class, listQUO.get(j));
            completarApuesta(quo);
        }
    }
    db.getTransaction().begin();
    db.remove(event);
    db.getTransaction().commit();
    return true;
}

```

```

public void completarApuesta(Quote quo) {
    for(int i=0; i<quo.getApustuak().size(); i++) {
        ApustuAnitza apustuAnitza = quo.getApustuak().get(i).getApustuAnitza();
        ApustuAnitza ap1 = db.find(ApustuAnitza.class, apustuAnitza.getApustuAnitzaNumber());
        db.beginTransaction().begin();
        ap1.removeApustua(quo.getApustuak().get(i));
        if(ap1.getApustuak().isEmpty() && !ap1.getEgoera().equals("galdua")) {
            this.borrarApuesta(ap1.getUser(), ap1);
        }else if(!ap1.getApustuak().isEmpty() && ap1.irabazitaMarkatu()){
            this.ganarApuesta(ap1);
        }
        Sport spo =quo.getQuestion().getEvent().getSport();
        spo.setApustuKantitatea(spo.getApustuKantitatea()-1);
        db.beginTransaction().commit();
    }
}

```

### 3. Duplicate Code

Código inicial:

```

else if (Locale.getDefault().equals(new Locale("en"))) {
    Duplication
    q1=ev1.addQuestion("Who will win the match?",1);
    q2=ev1.addQuestion("Who will score first?",2);
    Duplication
    q3=ev11.addQuestion("Who will win the match?",1);
    q4=ev11.addQuestion("How many goals will be scored in the match?",2);
    Duplication
    q5=ev17.addQuestion("Who will win the match?",1);
    q6=ev17.addQuestion("Will there be goals in the first half?",2);

}

```

He creado una variable con la String que se repetía.

Código refactorizado:

```

else if (Locale.getDefault().equals(new Locale("en"))) {
    String quienGana= "Who will win the match?";
    Duplication
    q1=ev1.addQuestion(quienGana,1);
    q2=ev1.addQuestion("Who will score first?",2);
    Duplication
    q3=ev11.addQuestion(quienGana,1);
    q4=ev11.addQuestion("How many goals will be scored in the match?",2);
    Duplication
    q5=ev17.addQuestion(quienGana,1);
    q6=ev17.addQuestion("Will there be goals in the first half?",2);

}

```

### 4. Keep unit interfaces small

Como no había métodos con más de 4 parámetros, he creado una clase con los 4 parámetros que le entraban a la clase “apostar” y entrando así solo un parámetro, y cambiar las asignaciones de apostar.

Al BLFaciadlImplementation le entran 4 parámetros, crea un objeto con los 4 atributos y al método “apostar” entra solo 1 parámetro.

## Código inicial:

```
@WebMethod  
public boolean ApustuaEgin(Registered u, Vector<Quote> q, Double balioa, Integer apustuaGalarazi) {  
    dbManager.open(false);  
    boolean b = dbManager.apostar(u,q,balioa,apustuaGalarazi);  
    dbManager.close();  
    return b;  
}  
  
public boolean apostar(Registered u, Vector<Quote> q, Double balioa, Integer apustuaGalarazi) {  
  
    Registered user = (Registered) db.find(Registered.class, u.getUsername());  
    Boolean b;  
    if(user.getDirukop()>=balioa) {  
        db.beginTransaction().begin();  
        ApustuAnitza apustuAnitza = new ApustuAnitza(user, balioa);  
        db.persist(apustuAnitza);  
        for(Quote quo:q) {  
            Quote kuote = db.find(Quote.class, quo.getQuoteNumber());  
            Apustua ap = new Apustua(apustuAnitza, kuote);  
            db.persist(ap);  
            apustuAnitza.addApustua(ap);  
            kuote.addApustua(ap);  
        }  
        db.beginTransaction().commit();  
        db.beginTransaction().begin();  
        if(apustuBikoitzagalarazi== -1) {  
            apustuBikoitzagalarazi=apustuAnitza.getApustuAnitzaNumber();  
        }  
        apustuAnitza.setApustukopia(apustuBikoitzagalarazi);  
        user.updateDiruKontua(-balioa);  
        Transaction t = new Transaction(user, balioa, new Date(), "ApustuaEgin");  
        user.addApustuAnitza(apustuAnitza);  
        for(Apustua a: apustuAnitza.getApustuak()) {  
            Apustua apu = db.find(Apustua.class, a.getApustuaNumber());  
            Quote q = db.find(Quote.class, apu.getKuota().getQuoteNumber());  
            Sport spo = q.getQuestion().getEvent().getSport();  
        }  
    }  
}
```

## Código modificado:

```
@WebMethod  
public boolean ApustuaEgin(Registered u, Vector<Quote> q, Double balioa, Integer apustuaGalarazi) {  
    datosApuesta da= new datosApuesta(u,q,balioa,apustuaGalarazi);  
    dbManager.open(false);  
    boolean b = dbManager.apostar(da);  
    dbManager.close();  
    return b;  
}
```

```
1 package domain;  
2  
3 import java.util.Vector;  
4  
5 public class datosApuesta {  
6  
7     private Registered u;  
8     private Vector<Quote> q;  
9     private Double balioa;  
10    private Integer apustuaGalarazi;  
11  
12    public datosApuesta(Registered u, Vector<Quote> q, Double balioa, Integer apustuaGalarazi) {  
13        this.u=u;  
14        this.q=q;  
15        this.balioa=balioa;  
16        this.apustuaGalarazi=apustuaGalarazi;  
17    }  
18    public Registered getRegistered() {  
19        return this.u;  
20    }  
21    public Vector<Quote> getvector() {  
22        return this.q;  
23    }  
24    public Double getBalioa() {  
25        return this.balioa;  
26    }  
27    public Integer getApustuaGalarazi() {  
28        return this.apustuaGalarazi;  
29    }  
30}  
31}  
32}
```

```
public boolean apostar(datosApuesta da) {  
  
    Registered user = (Registered) db.find(Registered.class, da.getRegistered().getUsername());  
    Boolean b;  
    if(user.getDirukop()>=da.getBailoa()) {  
        db.beginTransaction().begin();  
        ApustuAnitza apustuAnitza = new ApustuAnitza(user, da.getBailoa());  
        db.persist(apustuAnitza);  
        for(Quote quo:da.getvector()) {  
            Quote kuote = db.find(Quote.class, quo.getQuoteNumber());  
            Apustua ap = new Apustua(apustuAnitza, kuote);  
            db.persist(ap);  
            apustuAnitza.addApustua(ap);  
            kuote.addApustua(ap);  
        }  
        db.beginTransaction().commit();  
        db.beginTransaction().begin();  
        if(apustuBikoitzaGalarazi== -1) {  
            apustuBikoitzaGalarazi=apustuAnitza.getApustuAnitzaNumber();  
        }  
        apustuAnitza.setApustuKopia(apustuBikoitzaGalarazi);  
        user.updateDiruKontua(-da.getBailoa());  
        Transaction t = new Transaction(user, da.getBailoa(), new Date(), "ApustuaEgin");  
        user.addApustuAnitza(apustuAnitza);  
        for(Apustua a: apustuAnitza.getApustuak()) {  
            Apustua apu = db.find(Apustua.class, a.getApostuaNumber());  
            Quote q = db.find(Quote.class, apu.getKuota().getQuoteNumber());  
            Sport spo =q.getQuestion().getEvent().getSport();  
        }  
    }  
}
```