

REFACTORIZACIÓN DEL PROYECTO BETS21

- "Write short units of code"

gertaeraEzabatu() código original:

```
public boolean gertaeraEzabatu(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().getEventNumber() =?1", Quote.class);
        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.apustuaEzabatu(ap1.getUser(), ap1);
                } else if(!ap1.getApustuak().isEmpty() && ap1.irabazitaMarkatu()){
                    this.ApustuaIrabazi(ap1);
                }
                db.getTransaction().begin();
                Sport spo =quo.getQuestion().getEvent().getSport();
                spo.setApustuKantitatea(spo.getApustuKantitatea()-1);
                KirolEstatistikak ke=ap1.getUser().kirolEstatistikakLortu(spo);
                ke.setKont(ke.getKont()-1);
                db.getTransaction().commit();
            }
        }
    }
    db.getTransaction().begin();
    db.remove(event);
    db.getTransaction().commit();
    return true;
}
```

Código refactorizado:

Para reducir el contenido de este metodo, lo he refactorizado separando partes de este en diferentes metodos más pequeños.

```

public boolean gertaeraEzabatu(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) {
        eguneratuQuotes(event);
    }

    db.getTransaction().begin();
    db.remove(event);
    db.getTransaction().commit();
    return true;
}

private void eguneratuQuotes(Event ev) {
    TypedQuery<Quote> Qquery = db.createQuery(
        "SELECT q FROM Quote q WHERE q.getQuestion().getEvent().getEventNumber() =?1", Quote.class);
    Qquery.setParameter(1, ev.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());

            eguneratuApostua(ap1, quo, i);
            eguneratuKirola(ap1, quo);
        }
    }
}

private void eguneratuApostua(ApustuAnitza ap, Quote q, int index) {
    db.getTransaction().begin();
    ap.removeApustua(q.getApustuak().get(index));
    db.getTransaction().commit();
    if (ap.getApustuak().isEmpty() && !ap.getEgoera().equals("galduta")) {
        this.apustuaEzabatu(ap.getUser(), ap);
    } else if (!ap.getApustuak().isEmpty() && ap.irabazitaMarkatu()) {
        this.ApustuaIrabazi(ap);
    }
}

private void eguneratuKirola(ApustuAnitza apA, Quote q) {
    db.getTransaction().begin();
    Sport spo = q.getQuestion().getEvent().getSport();
    spo.setApustuKantitatea(spo.getApustuKantitatea() - 1);
    KirolEstatistikak ke = apA.getUser().kirolEstatistikakLortu(spo);
    ke.setKont(ke.getKont() - 1);
    db.getTransaction().commit();
}

```

- "Write simple units of code"

Despues de los cambios anteriores en el metodo gertaeraEzabatu(...) la complejidad ciclomática es de $4+1=5$. Para simplificarlo, he realizado el siguiente cambio:

Código original:

```
public boolean gertaeraEzabatu(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) {
        eguneratuQuotes(event);
    }

    db.getTransaction().begin();
    db.remove(event);
    db.getTransaction().commit();
    return true;
}
```

Código simplificado:

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

    resultB = questionNull(resultB, listQ);

    if (resultB == false) {
        return false;
    } else if (new Date().compareTo(event.getEventDate()) < 0) {
        eguneratuQuotes(event);
    }

    db.getTransaction().begin();
    db.remove(event);
    db.getTransaction().commit();
    return true;
}

private boolean questionNull(boolean resultB, List<Question> listQ) {
    for (Question q : listQ) {
        if (q.getResult() == null) {
            resultB = false;
        }
    }
    return resultB;
}
```

- “Duplicate code”

Dentro del metodo initializeDB() del clase dataAcces he realizado la siguiente refactorización:

Código original:

```

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 apEgin = "ApustuaEgin";

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

```

- "Keep unit interfaces small"

Por último, para realizar esta refactorización no he encontrado ningún método con más de 4 parámetros. Por ello he escogido un método con 4 parámetros, concretamente el `DiruaSartu(...)`:

Código original:

```

public void DiruaSartu(User u, Double dirua, Date data, String mota) {
    Registered user = (Registered) db.find(User.class, u.getUsername());
    db.getTransaction().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();
}

```

Código refactorizado:

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

JOANES

- Write short units of code
'mezuaBidali' código original:

```
public boolean mezuaBidali(User igorlea, String hartzailea, String titulo, String test, Elkarrizketa elkarrizketa) {
    User igorle = db.find(User.class, igorlea.getUsername());
    User hartzaile = db.find(User.class, hartzailea);
    Elkarrizketa elk=null;
    if(hartzaile==null) {
        return false;
    }else {
        db.getTransaction().begin();
        Message m = new Message(igorle, hartzaile, test);
        db.persist(m);
        if(elkarrizketa!=null) {
            elk = db.find(Elkarrizketa.class, elkarrizketa.getElkarrizketaNumber());
        }else {
            elk= new Elkarrizketa(titulo, igorle, hartzaile);
            db.persist(elk);
            m.setElkarrizketa(elk);
            igorle.addElkarrizketak(elk);
            hartzaile.addElkarrizketak(elk);
        }
        elk.addMezua(m);
        igorle.addBidalitakoMezuak(m);
        hartzaile.addJasotakoMezuak(m);
        db.getTransaction().commit();
        return true;
    }
}
```

Código refactorizado:

```
public boolean mezuaBidali(User igorlea, String hartzailea, String titulo, String test, Elkarrizketa elkarrizketa) {
    User igorle = db.find(User.class, igorlea.getUsername());
    User hartzaile = db.find(User.class, hartzailea);
    if(hartzaile!=null) {
        db.getTransaction().begin();
        Message m = new Message(igorle, hartzaile, test);
        elkMezuarekin(titulo, igorle, hartzaile, m, elkarrizketa);
        db.getTransaction().commit();
        return true;
    }
    return false;
}
```

Para conseguir la refactorización he implementado este método nuevo:

```
private void elkMezuarekin(String titulo, User igorle, User hartzaile, Message m, Elkarrizketa elkarrizketa) {
    Elkarrizketa elk=null;
    db.getTransaction().begin();
    if(elkarrizketa==null) {
        elk = new Elkarrizketa(titulo, igorle, hartzaile);
        db.persist(elk);
        m.setElkarrizketa(elk);
        igorle.addElkarrizketak(elk);
        hartzaile.addElkarrizketak(elk);
    }else {
        elk = db.find(Elkarrizketa.class, elkarrizketa.getElkarrizketaNumber());
    }
    elk.addMezua(m);
    igorle.addBidalitakoMezuak(m);
    hartzaile.addJasotakoMezuak(m);
    db.getTransaction().commit();
}
```

- Write simple units of code
‘EmaitzakIpini’ código inicial:

```
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();
    db.getTransaction().begin();
    Question que = q.getQuestion();
    Question question = db.find(Question.class, que);
    question.setResult(result);

    galduAp(question);

    db.getTransaction().commit();

    irabaziAp(listApustuak);
}
```

Para conseguir esta refactorización he creado dos métodos nuevos, que han sido extraídos del método ‘EmaitzakIpini’:

```
private void galduAp(Question question) {
    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");
            }
        }
    }
}
```

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


- Duplicate code

En la clase `dataAccess`, en esta parte podemos ver que se repite “Who will win the match?” tres veces:

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

Para solucionar esto, vamos a definir un constante con el valor “Who will win the match?”, y vamos a utilizar este constante:

```
else if (Locale.getDefault().equals(new Locale("en"))) {
    String whoWillWin = "Who will win the match?";

    q1=ev1.addQuestion(whoWillWin,1);
    q2=ev1.addQuestion("Who will score first?",2);
    q3=ev11.addQuestion(whoWillWin,1);
    q4=ev11.addQuestion("How many goals will be scored in the match?",2);
    q5=ev17.addQuestion(whoWillWin,1);
    q6=ev17.addQuestion("Will there be goals in the first half?",2);
```

- Keep unit interfaces small

El método ‘`mezuaBidali`’ recibe 5 parámetros, entonces, para reducir el número de parámetros he hecho lo siguiente:

Entre los parámetros que recibía este método estaban ‘`igorlea`’ y ‘`hartzailea`’, que son la persona que envía el mensaje y el que recibe. He pensado en hacer una nueva clase llamada ‘`Participantes`’, y esta clase solo va tener dos atributos, uno para la persona que envía el mensaje, y otro para la persona que recibe el mensaje. La clase ‘`Elkarrizketa`’ tendrá un atributo de tipo ‘`Participantes`’ para guardar los participantes de la conversación.

La clase ‘`Participantes`’ será la siguiente(también tiene getters y setters):

```
1 package domain;
2
3 public class Participantes {
4     private String hartzailea;
5     private User igorlea;
6
7     public Participantes(String hartzailea, User igorlea) {
8         super();
9         this.hartzailea = hartzailea;
10        this.igorlea = igorlea;
11    }
12 }
```

El método 'mezuaBidali' de la clase 'dataAccess' quedará así:

```
public boolean mezuaBidali(Participantes p, String titulo, String test, Elkarrizketa elkarrizketa) {  
    User igorle = db.find(User.class, p.getIgorlea().getUsername());  
    User hartzaile = db.find(User.class, p.getHartzailea());  
    if(hartzaile!=null) {  
        db.getTransaction().begin();  
        Message m = new Message(igorle, hartzaile, test);  
        elkMezuarekin(titulo, igorle, hartzaile, m, elkarrizketa);  
        db.getTransaction().commit();  
        return true;  
    }  
    return false;  
}
```

Eneko

- Write short units of code
'gertaerakKopiatu' código original:

```
public boolean gertaerakKopiatu(Event e, Date date) {
    Boolean b = false;
    Event gertaera = db.find(Event.class, e.getEventNumber());
    db.getTransaction().begin();

    TypedQuery<Event> query = db.createQuery(
        "SELECT ev FROM Event ev WHERE ev.getDescription()=?1 and ev.getEventDate()=?2", Event.class);
    query.setParameter(1, gertaera.getDescription());
    query.setParameter(2, date);
    if (query.getResultList().isEmpty()) {
        b = true;
        String[] taldeak = gertaera.getDescription().split("-");
        Team lokala = new Team(taldeak[0]);
        Team kanpokoia = new Team(taldeak[1]);
        Event gertKopiatu = new Event(gertaera.getDescription(), date, lokala, kanpokoia);
        gertKopiatu.setSport(gertaera.getSport());
        gertaera.getSport().addEvent(gertKopiatu);
        db.persist(gertKopiatu);
        for (Question q : gertaera.getQuestions()) {
            Question que = new Question(q.getQuestion(), q.getBetMinimum(), gertKopiatu);
            gertKopiatu.listaraGehitu(que);
            Question galdera = db.find(Question.class, q.getQuestionNumber());
            db.persist(que);
            for (Quote k : galdera.getQuotes()) {
                Quote kuo = new Quote(k.getQuote(), k.getForecast(), que);
                que.listaraGehitu(kuo);
                db.persist(kuo);
            }
        }
    }
    db.getTransaction().commit();
    return b;
}
```

Código refactorizado:

```
public boolean gertaerakKopiatu(Event e, Date date) {
    Boolean b = false;
    Event gertaera = db.find(Event.class, e.getEventNumber());
    db.getTransaction().begin();
    TypedQuery<Event> query = db.createQuery("SELECT ev FROM Event ev WHERE ev.getDescription()=?1 and ev.getEventDate()=?2", Event.class);
    query.setParameter(1, gertaera.getDescription());
    query.setParameter(2, date);
    if (query.getResultList().isEmpty()) {
        b = true;
        Event gertKopiatu = this.gertaeraKoptiatuLaguntzailea1(gertaera, date);
        gertaera.getSport().addEvent(gertKopiatu);
        db.persist(gertKopiatu);
        for (Question q : gertaera.getQuestions()) {
            gertaeraKoptiatuLaguntzailea2(q, gertaera);
        }
    }
    db.getTransaction().commit();
    return b;
}
```

He creado otro método para hacer la copia del evento aparte y devolver directamente el evento creado:

```
public Event gertaeraKoptiatuLaguntzailea1(Event a, Date date) {
    String[] taldeak = a.getDescription().split("-");
    Team lokala = new Team(taldeak[0]);
    Team kanpokoia = new Team(taldeak[1]);
    Event gertKopiatu = new Event(a.getDescription(), date, lokala, kanpokoia);
    gertKopiatu.setSport(a.getSport());
    return gertKopiatu;
}
```

Y otro método para trabajar cada uno de los question de la lista en el for:

```
public void gertaeraKoptiatuLaguntzailea2(Question q, Event gertaera) {
    Question que = new Question(q.getQuestion(), q.getBetMinimum(), gertaera.getSport().getEvents().get(gertaera.getSport().getEvents().size()-1));
    gertaera.getSport().getEvents().get(gertaera.getSport().getEvents().size()-1).listaraGehitu(que);
    Question galdera = db.find(Question.class, q.getQuestionNumber());
    db.persist(que);
    for (Quote k : galdera.getQuotes()) {
        Quote kuo = new Quote(k.getQuote(), k.getForecast(), que);
        que.listaraGehitu(kuo);
        db.persist(kuo);
    }
}
```

- Write simple units of code
'gertaerakSortu' código inicial:

```
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 {
        db.getTransaction().commit(); // Error, no se cerraba la BD
        return false;
    }
    db.getTransaction().commit();
    return b;
}
```

Código refactorizado:


```
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);
        b=gertaerakSortuLaguntzailea1(Equery,description);
        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 {
        db.getTransaction().commit(); // Error, no se cerraba la BD
        return false;
    }
    db.getTransaction().commit();
    return b;
}
```

Despues de los cambios, el metodo principal tiene una complejidad ciclomatica de menos de 4, y para eso he implementado otro metodo para ejecutar el for que habia antes aparte:

```
public boolean gertaerakSortuLaguntzailea1(TypedQuery<Event> Equery,String description) {
    boolean b=true;
    for (Event ev : Equery.getResultList()) {
        if (ev.getDescription().equals(description)) {
            b = false;
        }
    }
    return b;
}
```

- “Duplicate code”

SonarLint:

 Define a constant instead of duplicating this literal "Zeinek irabaziko du partidua?" 3 times. [+3 locations]

Codigo original:

```
Duplication
q1 = ev1.addQuestion(1 "Zeinek irabaziko du partidua?", 1);
q2 = ev1.addQuestion("Zeinek sartuko du lehenengo gola?", 2);
Duplication
q3 = ev11.addQuestion(2 "Zeinek irabaziko du partidua?", 1);
q4 = ev11.addQuestion("Zenbat gol sartuko dira?", 2);
Duplication
q5 = ev17.addQuestion(3 "Zeinek irabaziko du partidua?", 1);
q6 = ev17.addQuestion("Golak sartuko dira lehenengo zatian?", 2);
```

Refactorizado:

```
String duplicated="Zeinek irabaziko du partidua?";
q1 = ev1.addQuestion(duplicated, 1);
q2 = ev1.addQuestion("Zeinek sartuko du lehenengo gola?", 2);
q3 = ev11.addQuestion(duplicated, 1);
q4 = ev11.addQuestion("Zenbat gol sartuko dira?", 2);
q5 = ev17.addQuestion(duplicated, 1);
q6 = ev17.addQuestion("Golak sartuko dira lehenengo zatian?", 2);
```

- Keep unit interfaces small (en la clase ElkarrizketaRender) en el metodo ‘getListCellRendererComponent’. Código inicial:

```
@Override
public Component getListCellRendererComponent(JList<? extends ElkarrizketaContainer> list, ElkarrizketaContainer elkarrizketa, int index,
    boolean isSelected, boolean cellHasFocus) {

    setText(elkarrizketa.toString());
    String code = "m1";

    ImageIcon imageIcon = new ImageIcon(".\\src\\main\\resources\\data\\"+elkarrizketa.getElka().denakIrakurrita(user)+".png"); // load the image to a ImageIcon
    Image image = imageIcon.getImage(); // transform it
    Image newimg = image.getScaledInstance(25, 25, Image.SCALE_SMOOTH); // scale it the smooth way
    imageIcon = new ImageIcon(newimg);
    setIcon(imageIcon);

    if (elkarrizketa.getElka().denakIrakurrita(user)==false && !isSelected) {
        setBackground(list.getSelectionBackground().PINK);
        setForeground(list.getSelectionForeground().BLACK);
    } else if(isSelected) {
        setBackground(list.getSelectionBackground().gray);
        setForeground(list.getSelectionForeground().WHITE);
    } else {
        setBackground(list.getBackground());
        setForeground(list.getForeground());
    }
    return this;
}
```

- Refactorizado:

Ya que en el método no se utilizaba el index he aprovechado para quitar ese parámetro. Pero esto no ha sido tarea fácil ya que esta clase (ElkarrikketaRender) implementa una interfaz (ListCellRenderer) y esa clase venía de la interfaz y entonces era obligatorio poner ese método con 5 parámetros porque así lo indicaba el método. Pero he entrado al código de la interfaz y he visto que solo tenía ese método y que si quitaba la interfaz y usaba el mismo método pero sin index podía funcionar igual y eso he hecho.

```
public Component getListCellRendererComponent(JList<? extends ElkarrikketaContainer> list, ElkarrikketaContainer elkarrikketa,
    boolean isSelected, boolean cellHasFocus) {

    setText(elkarrikketa.toString());
    String code = "01";

    ImageIcon imageIcon = new ImageIcon(".\\src\\main\\resources\\data\\"+elkarrikketa.getElka().denakIrakurrita(user)+".png"); // load the image to a ImageIcon
    Image image = imageIcon.getImage(); // transform it
    Image newimg = image.getScaledInstance(25, 25, Image.SCALE_SMOOTH); // scale it the smooth way
    ImageIcon = new ImageIcon(newimg);
    setIcon(imageIcon);

    if (elkarrikketa.getElka().denakIrakurrita(user)==false && !isSelected) {
        setBackground(list.getSelectionBackground().PINK);
        setForeground(list.getSelectionForeground().BLACK);
    } else if (isSelected) {
        setBackground(list.getSelectionBackground().gray);
        setForeground(list.getSelectionForeground().WHITE);
    } else {
        setBackground(list.getBackground());
        setForeground(list.getForeground());
    }
    return this;
}
```

Una vez quitado en index he cambiado los sitios donde se llamaba al método y les he quitado el index y así he conseguido que solo tuviera 4 parámetros.