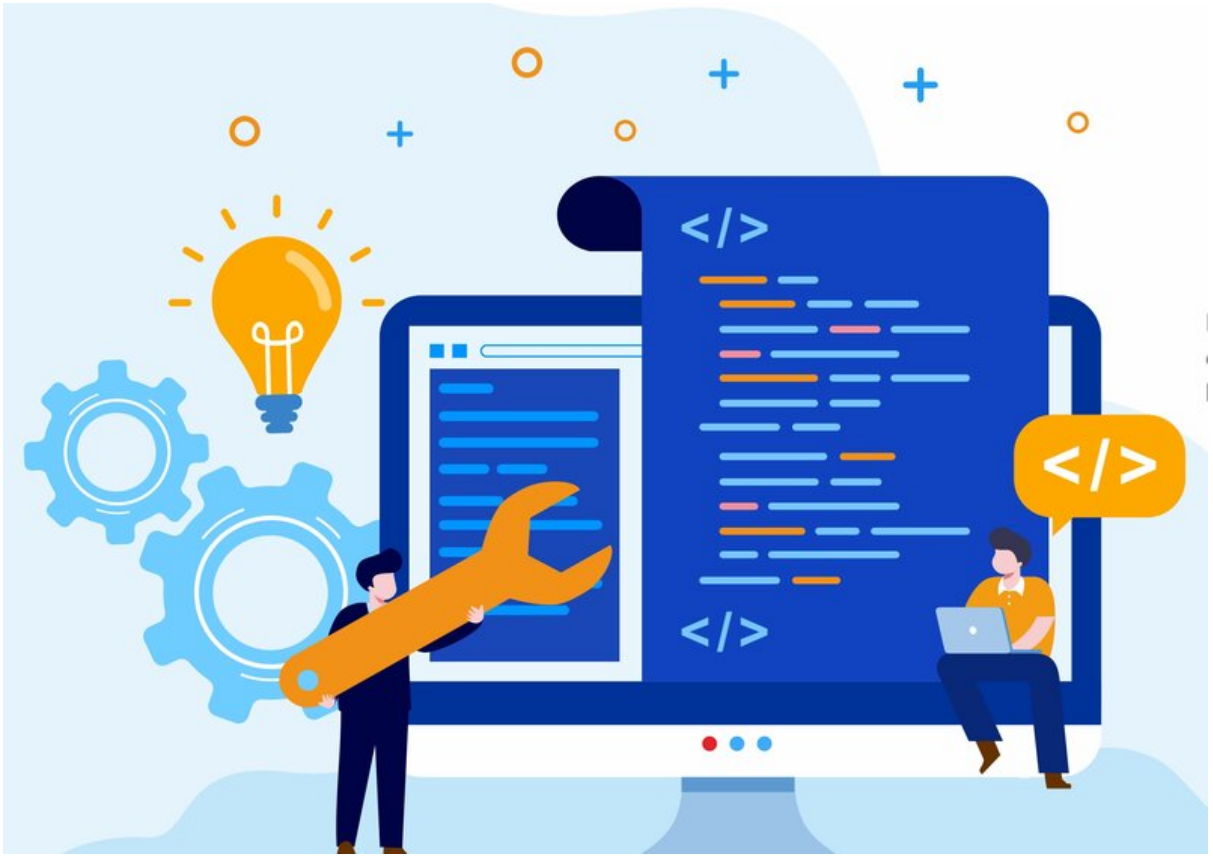
Errefaktorizazioa:



Txomin Errasti, Iosu Abal eta Alex Amenabar 16/10/2022

Alex:

➤ Kode unitate laburrak idatzi (15 lerro baino gutxiago):

EmtizakIpini() errefaktorizatu aurretik (23 lerro):

```
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();

    List<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());
        }
    }
}
```

Kodea errafaktorizatuta:

```
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();

    List<Apustua> listApustuak = q.getApustuak();
    db.getTransaction().begin();
    esleituEmaitzak(q, result);
    db.getTransaction().commit();
    for(Apustua a : listApustuak) {
        db.getTransaction().begin();
        Boolean bool=a.getApustuAnitza().irabazitaMarkatu();
        db.getTransaction().commit();
        if(bool) {
            this.ApustuaIrabazi(a.getApustuAnitza());
        }
    }
}

private void esleituEmaitzak(Quote q, String result) {
    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");
            }
        }
    }
}
```

Azalpena:

Metodotik ateratako atalak (esleituEmaitzak()), kuotari dagokion galderari emaitza esleitzen dio, eta aldi berean, galdera horretako kuota bakoitzeko, bere apostuak irabazi edo galdu egin diren ezartzen du. Zati horrek funtzio konkretu bat betetzen duenez hasierako metodoaren barruan, egokia da ateratzeko eta honela metodoa sinplifikatzeko.

➤Metodoen konplexutasuna murriztu:

Errefaktorizatu aurretik (10 erabaki nodo):

```
public boolean ApustuaEgin(User u, Vector<Quote> quote, Double balioa, Integer apustuBikoitzaGalarazi) {
    Registered user = (Registered) db.find(User.class, u.getUsername());
    Boolean b;
    if(user.getDirukop()>=balioa) {
        db.getTransaction().begin();
        ApustuAnitza apustuAnitza = new ApustuAnitza(user, balioa);
        db.persist(apustuAnitza);
        for(Quote quo: quote) {
            Quote kuote = db.find(Quote.class, quo.getQuoteNumber());
            Apustua ap = new Apustua(apustuAnitza, kuote);
            db.persist(ap);
            apustuAnitza.addApustua(ap);
            kuote.addApustua(ap);
        }
        db.getTransaction().commit();
        db.getTransaction().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.getApostuaNumber());
            Quote q = db.find(Quote.class, apu.getKuota().getQuoteNumber());
            Sport spo =q.getQuestion().getEvent().getSport();
            spo.setApustuKantitatea(spo.getApustuKantitatea()+1);
            if(!user.containsKirola(spo)) {
                KirolEstatistikak ke=new KirolEstatistikak(user, spo);
                ke.setKont(1);
                user.addKirolEstatistikak(ke);
                spo.addKirolEstatistikak(ke);
            }else {
                KirolEstatistikak ke=user.kirolEstatistikakLortu(spo);
                ke.eguneratuKont(1);
            }
        }

         user.addTransaction(t);
         db.persist(t);
         db.getTransaction().commit();
         for(Jarraitzailea reg:user.getJarraitzaileLista()) {
             Jarraitzailea erab=db.find(Jarraitzailea.class, reg.getJarraitzaileaNumber());
             b=true;
             for(ApustuAnitza apu: erab.getNork().getApustuAnitzak()) {
                 if(apu.getApustuKopia()==apustuAnitza.getApustuKopia()) {
                     b=false;
                 }
             }
             if(b) {
                 if(erab.getNork().getDiruLimitea()<balioa) {
                     this.ApustuaEgin(erab.getNork(), quote, erab.getNork().getDiruLimitea(), apustuBikoitzaGalarazi);
                 }else{
                     this.ApustuaEgin(erab.getNork(), quote, balioa, apustuBikoitzaGalarazi);
                 }
             }
         }
         return true;
     }else{
         return false;
     }

 }
```

Errefaktorizatu ostean:

```
public boolean ApustuaEgin(User u, Vector<Quote> quote, Double balioa, Integer apustuBikoitzaGalarazi) {
    Registered user = (Registered) db.find(User.class, u.getUsername());
    Boolean b;
    if(user.getDirukop()>=balioa) {
        db.getTransaction().begin();
        ApustuAnitza apustuAnitza = new ApustuAnitza(user, balioa);
        db.persist(apustuAnitza);
        for(Quote quo: quote) {
            Quote kuote = db.find(Quote.class, quo.getQuoteNumber());
            Apustua ap = new Apustua(apustuAnitza, kuote);
            db.persist(ap);
            apustuAnitza.addApustua(ap);
            kuote.addApustua(ap);
        }
        db.getTransaction().commit();
        db.getTransaction().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);
        kirolaEtaEstitistikakEguneratu(user, apustuAnitza);
        user.addTransaction(t);
        db.persist(t);
        db.getTransaction().commit();
        for(Jarraitzailea reg:user.getJarraitzaileLista()) {
            jarraitzaileekApostatu(quote, balioa, apustuBikoitzaGalarazi, apustuAnitza, reg);
        }
        return true;
    }else{
        return false;
    }

}

 private void kirolaEtaEstitistikakEguneratu(Registered user, ApustuAnitza 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();
         spo.setApustuKantitatea(spo.getApustuKantitatea()+1);
         if(!user.containsKirola(spo)) {
             KirolEstatistikak ke=new KirolEstatistikak(user, spo);
             ke.setKont(1);
             user.addKirolEstatistikak(ke);
             spo.addKirolEstatistikak(ke);
         }else {
             KirolEstatistikak ke=user.kirolEstatistikakLortu(spo);
             ke.eguneratuKont(1);
         }
     }
 }
 private void jarraitzaileekApostatu(Vector<Quote> quote, Double balioa, Integer apustuBikoitzaGalarazi,
         ApustuAnitza apustuAnitza, Jarraitzailea reg) {
     Boolean b;
     Jarraitzailea erab=db.find(Jarraitzailea.class, reg.getJarraitzaileaNumber());
     b=true;
     for(ApustuAnitza apu: erab.getNork().getApustuAnitzak()) {
         if(apu.getApustuKopia()==apustuAnitza.getApustuKopia()) {
             b=false;
         }
     }
     if(b) {
         if(erab.getNork().getDiruLimitea()<balioa) {
             this.ApustuaEgin(erab.getNork(), quote, erab.getNork().getDiruLimitea(), apustuBikoitzaGalarazi);
         }else{
             this.ApustuaEgin(erab.getNork(), quote, balioa, apustuBikoitzaGalarazi);
         }
     }
 }
```

Azalpena:

Metodoa nahiko konplexua izanik, behrrezkoa da hainbat zati ateratzea sinplifikatzeko. kirolaEtaEstatistikakEguneratu() metodoak, izenak dioena egiten du, apustua sortu ostean, apostua egin den kirolaren inguruko informazioa eguneratu, hain zuzen ere (kirol horretan apustu bat gehiago egin dela ezarri, erabiltzaileari kirolEstatistikak eguneratu...

Bigarren atalak, jarraitzaileekApostatu(), apustua sortu eta beharrezko eguneraketak egin ostean, erabiltzailearen jarraitzaileak lortzen ditu, eta jarraitzaileen kontuetan ere apostua egin, beharrezko baldintzak betetzen badira.

➢Errepikatutako kodea

Ez dut errepikatutako koderik aurkitu sonarLinten bidez, ez DataAccess klasean, ezta beste klaseren batean ere.

➢Metodoen interfazeak parametro gutxirekin mantendu:

Metodo honek 4 parametro bakarrik jasotzen ditu, baina ez dut aurkitu metodorik parametro gehiago jasotzen zituena dataAccess klasean.

Errefaktorizatu aurretik:

```
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();
}
```

Errefaktorizatu ostean:

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

Azalpena:

Metodoari pasatako parametro guztiek Transaction klaseko objektu bat sortzeko balio dute, beraz, interfazea txukun mantentzeko hobe da zuzenean Transaction objektua pasatzea eta ondoren, metodoak behar dituen parametroak objektutik lortzea.

Metodo honi egindako deialdi guztietan ere aldaketak egin behar izan dira, zuzenean Transaction klaseko objektu bat pasatzeko.

Txomin:

➤ Kode unitate laburrak idatzi (15 lerro baino gutxiago):

hasierako egoera:

```
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();
```

Errefaktorizatu ondoren:

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

    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().
        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));
            kuotaTratatu(quo);
        }

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

for iterazioko kode zatia kuotakTratatu izeneko metodo berri batera pasa dugu eta metodo horri deitu.

```java
public void kuotaTratatu(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.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();
    }
}
```

➤Metodoen konplexutasuna murriztu:

Kodearen lehen lerroetan, parametro moduan pasatako gertaerako galderek ea emaitzak dituzten konprobatzen da. Galderen zerrenda bat parametro moduan jasotzen duen emaitzak ditu metodoa sortu dugu, kodea errazago ulertzeko eta konplexutasuna baxutzeko

```java
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();
```

```java
public boolean emaitzakDitu(List<Question> listQ) {

    boolean resultB= true;
    for(Question q : listQ) {
        if(q.getResult() == null) {
            resultB = false;
        }
    }
    return resultB;

}
```

➤Errepikatutako kodea

Ez dut errepikatutako koderik topatu.

➤Metodoen interfazeak parametro gutxirekin mantendu:

```java
private void jarraitzaileekApostatu(Vector<Quote> quote, Double balioa, Integer apustuBikoitzaGalarazi,
        ApustuAnitza apustuAnitza, Jarraitzailea reg) {
    Boolean b;
    Jarraitzailea erab=db.find(Jarraitzailea.class, reg.getJarraitzaileaNumber());
    b=true;
    for(ApustuAnitza apu: erab.getNork().getApustuAnitzak()) {
        if(apu.getApustuKopia()==apustuAnitza.getApustuKopia()) {
            b=false;
        }
    }
    if(b) {
        if(erab.getNork().getDiruLimitea()<balioa) {
            this.ApustuaEgin(erab.getNork(), quote, erab.getNork().getDiruLimitea(), apustuBikoitzaGalarazi);
        }else{
            this.ApustuaEgin(erab.getNork(), quote, balioa, apustuBikoitzaGalarazi);
        }
    }
}
```

Metodo honek 5 parametro jasotzen ditu, baina balioa parametroa apustuAnitza parametrotik lor dezakegu getBalioa metodoa erabiliz, beraz soberan dago.

```java
private void jarraitzaileekApostatu(Vector<Quote> quote, Integer apustuBikoitzaGalarazi,
        ApustuAnitza apustuAnitza, Jarraitzailea reg) {
    Boolean b;
    Jarraitzailea erab=db.find(Jarraitzailea.class, reg.getJarraitzaileaNumber());
    b=true;
    Double balioa = apustuAnitza.getBalioa();
    for(ApustuAnitza apu: erab.getNork().getApustuAnitzak()) {
        if(apu.getApustuKopia()==apustuAnitza.getApustuKopia()) {
            b=false;
        }
    }
    if(b) {
        if(erab.getNork().getDiruLimitea()<balioa) {
            this.ApustuaEgin(erab.getNork(), quote, erab.getNork().getDiruLimitea(), apustuBikoitzaGalarazi);
        }else{
            this.ApustuaEgin(erab.getNork(), quote, balioa, apustuBikoitzaGalarazi);
        }
    }
}
```

IOSU ABAL

1 -Metodoen interfazeak parametro gutxirekin mantendu

Hasieran

```
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;
    }
}
```

Bukaeran

```
  public boolean mezuaBidali(User igorlea, Elkarrizketa elkarrizketa, Vector<String> osagaiak) {
      if(osagaiak.size()==3) {
          String hartzailea=osagaiak.get(0);
          String titulo=osagaiak.get(1);
          String testua=osagaiak.get(2);

          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, testua);
              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;
          }
      }//size!=3
      return false;

  }
```

Deskribapena:
Kasu honetan metodoak 5 parametro erabiltzen zituen, non 3 String motakoak ziren. Beraz, errefaktorizatu egin dut, 3 String horiek Lista batean sartzeko, *osagaiak* izenekoa Aldaketa txiki bat egin izan behar dut GUIan, String horiek lista gisa pasatzeko eta metodo berriak onar dezan. Beraz, orain metodoak bakarrik 3 parametro behar ditu, eta portaera berdina da.

2-Kode unitate laburrak idatzi (15 lerro baino gutxiago):

Hasieran:

```
public void apustuaEzabatu(User user1, ApustuAnitza ap) {
    Registered user = (Registered) db.find(User.class, user1.getUsername());
    ApustuAnitza apustuAnitza = db.find(ApustuAnitza.class, ap.getApustuAnitzaNumber());
    db.getTransaction().begin();
    user.updateDiruKontua(apustuAnitza.getBalioa());
    Transaction t = new Transaction(user, apustuAnitza.getBalioa(), new Date(), "ApustuaEzabatu");
    user.addTransaction(t);
    db.persist(t);
    user.removeApustua(apustuAnitza);
    int i;
    for(i=0; i<apustuAnitza.getApustuak().size(); i++) {
        apustuAnitza.getApustuak().get(i).getKuota().removeApustua(apustuAnitza.getApustuak().get(i));
        Sport spo =apustuAnitza.getApustuak().get(i).getKuota().getQuestion().getEvent().getSport();
        spo.setApustuKantitatea(spo.getApustuKantitatea()-1);
        KirolEstatistikak ke=user.kirolEstatistikakLortu(spo);
        ke.setKont(ke.getKont()-1);
    }
    db.remove(apustuAnitza);
    db.getTransaction().commit();
}
```

Bukaeran:

```
public void apustuaEzabatu(User user1, ApustuAnitza ap) {
    Registered user = (Registered) db.find(User.class, user1.getUsername());
    ApustuAnitza apustuAnitza = db.find(ApustuAnitza.class, ap.getApustuAnitzaNumber());
    db.getTransaction().begin();
    user.updateDiruKontua(apustuAnitza.getBalioa());
    Transaction t = new Transaction(user, apustuAnitza.getBalioa(), new Date(), "ApustuaEzabatu");
    user.addTransaction(t);
    db.persist(t);
    user.removeApustua(apustuAnitza);
    tratatuApustuAnitza(user, apustuAnitza);
    db.remove(apustuAnitza);
    db.getTransaction().commit();
}

private void tratatuApustuAnitza(Registered user, ApustuAnitza apustuAnitza) {
    for(int i=0; i<apustuAnitza.getApustuak().size(); i++) {
        apustuAnitza.getApustuak().get(i).getKuota().removeApustua(apustuAnitza.getApustuak().get(i));
        Sport spo =apustuAnitza.getApustuak().get(i).getKuota().getQuestion().getEvent().getSport();
        spo.setApustuKantitatea(spo.getApustuKantitatea()-1);
        KirolEstatistikak ke=user.kirolEstatistikakLortu(spo);
        ke.setKont(ke.getKont()-1);
    }
}
```

Deskribapena:
Metodo hau nahiko luzea denez (ia 30 lerro), for begizta beste metodo batean egitea aukera hobeagoa dela pentsatu dut. Extract method aukera erabili dut horretarako. Ikusten denez metodo berria tratatuapustuAnitza deitzen da, eta bi metodoen luzera 15 lerro baino gutxiago izatea lortu dugu.

3- Metodoen konplexutasuna murriztu:

Hasieran

```
public boolean mezuaBidali(User igorlea, Elkarrizketa elkarrizketa, Vector<String> osagaiak
    if(osagaiak.size()==3) {
        String hartzailea=osagaiak.get(0);
        String titulo=osagaiak.get(1);
        String testua=osagaiak.get(2);

        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, testua);
            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;
        }
    }//size!=3
    return false;

}
```

Bukaeran

```
public boolean mezuaBidali(User igorlea, Elkarrizketa elkarrizketa, Vector<String> osagaiak) {
    if(osagaiak.size()==3) {
        String hartzailea=osagaiak.get(0);
        String titulo=osagaiak.get(1);
        String testua=osagaiak.get(2);

        User igorle = db.find(User.class, igorlea.getUsername());
        User hartzaile = db.find(User.class, hartzailea);

        if(hartzaile==null) {
            return false;
        }else {
            List<User> users=new ArrayList<User>();
            users.add(igorle);
            users.add(hartzaile);
            return bidali(elkarrizketa, titulo, testua, users);
        }
    }//size!=3
    return false;

}

private boolean bidali(Elkarrizketa elkarrizketa, String titulo, String testua, List<User> users) {
    Elkarrizketa elk;
    User igorle=users.get(0);
    User hartzaile=users.get(1);
    db.getTransaction().begin();
    Message m = new Message(igorle, hartzaile, testua);
    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;
}
```

Deskribapena:
Hasierako metodoak, konplexutasun ziklomatikoa 5 zuen, eta barruko gorputza, beste metodo batera eraman eta gero (extract method erabilita) bere konplexutasuna 3koa izatea lortu dugu. Metodo honen testak eta erroreak kudeatzea askoz errazagoa izango da etorkizunean aldaketa hauekin

4- Errepikatutako kodea

Hasieran

```
else if (Locale.getDefault().equals(new Locale("en"))) {
    Duplication
    q1=ev1.addQuestion( 1 "Who will win the match?",1);
    q2=ev1.addQuestion("Who will score first?",2);
    Duplication
    q3=ev11.addQuestion( 2 "Who will win the match?",1);
    q4=ev11.addQuestion("How many goals will be scored in the match?",2);
    Duplication
    q5=ev17.addQuestion( 3 "Who will win the match?",1);
    q6=ev17.addQuestion("Will there be goals in the first half?",2);

}
```

Bukaeran

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

}
```

Deskribapena

Kasu honetan DatuBasea hasieratzen denean “code duplication” batzuk zeudela markatzen zigun Sonarlint-ek. Beraz, String berdina behin eta berriz errepikatu ordez, bakarrik behin erazagutu dugu aldagaia, eta aldagai hori behar dugun tokietan erabili dugu

Githubeko esteka: https://github.com/AlexAmena/BetCompleted22