Informe de refactorizaciones:

Mikel Lonbide:

1) **Write short units of code (capítulo 2):**

- **Problema: El método aplicarPromocíon contiene más de 15 lineas de código**
- Metodo: aplicarPromocion (en → dataAccess/DataAccess.java).

Código inicial:

```
903  public int aplicarPromocion(String text, Usuario actor) {
904      try {
905      TypedQuery<Promocion> query = db.createQuery("SELECT p FROM Promocion p",Promocion.class);
906      query.setParameter(1, text);
907      List<Promocion> resultados = query.getResultList();
908      Promocion resul= new Promocion("0");
909
910
911          for(Promocion e: resultados) {
912              if(e.getCode().equals(text)) {
913              resul= e;
914              }
915          }
916
917          for (Usuario a: resul.getUsuarios()) {
918              if (a.getDNI().equals(actor.getDNI())) return 1;
919          }
920
921          db.getTransaction().begin();
922          Promocion mod= db.find(Promocion.class, resul);
923          if(mod.getNum_veces()<=0) return 2;
924          mod.setNum_veces(mod.getNum_veces()-1);
925          Usuario user= db.find(Usuario.class, actor);
926          mod.anadirUsuario(user);
927
928          if(mod.getCompeticion()==null && !mod.isTipo()) {
929
930              user.setSaldo(user.getSaldo()+mod.getCant());
931
932          }else user.anadirPromo(mod);
933
934          db.persist(mod);
935          db.persist(user);
936          db.getTransaction().commit();
937
938
939      }catch(NullPointerException e) {
940          return 3;
941
942      }
943      return 0;
944  }
945
```

- Nos encontramos con el problema de que tenemos un método demasiado grande que realiza varias funciones.

Solución:

- Nuestro objetivo será que se repartan las distintas funciones en varios métodos y que este los llame.

- Para ello crearemos dos métodos nuevos:

    1) obtenerPromo → Obtendrá y nos devolverá la promoción que buscamos pasandole el id del mismo.
    2) PromoUsada → Nos indicará si la promoción ya ha sido utilizada por el usuario (con lo cual no se podrá utilizar) o no.

- Con estos añadidos, conseguimos reducir y simplificar el método al punto que queríamos (sin tener en cuenta los commit, ya que no son pasos imprescindibles).

Código refactorizado:

- **Método obtenerPromo:**

```
899
900    public Promocion obtenerPromo (List<Promocion> promociones, String text) {
901
902        Promocion resul= new Promocion("0");
903        for(Promocion e: promociones) {
904            if(e.getCode().equals(text)) {
905            resul= e;
906            }
907        }
908        return resul;
909    }
910
```

- **Método promoUsada:**

```
911    public boolean promoUsada(Promocion resul, Usuario actor) {
912
913        for (Usuario a: resul.getUsuarios()) {
914            if (a.getDNI().equals(actor.getDNI())) return true;
915        }
916
917        return false;
918    }
919
```

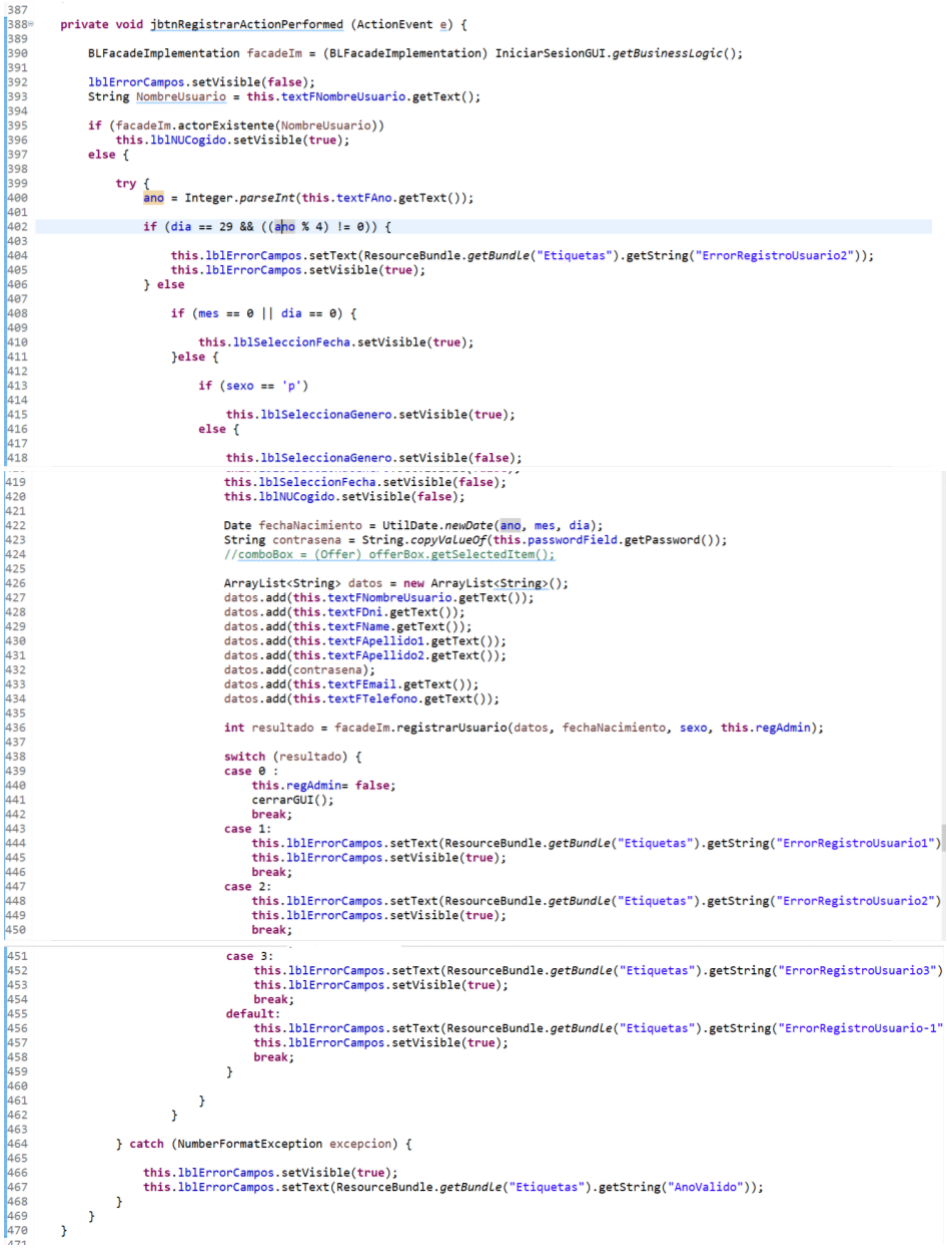
- **Método aplicarPromocion:**

```
919
920    public int aplicarPromocion(String text, Usuario actor) {
921        try {
922
923        TypedQuery<Promocion> query = db.createQuery("SELECT p FROM Promocion p",Promocion.class);
924        query.setParameter(1, text);
925        List<Promocion> resultados = query.getResultList();
926
927
928        Promocion resul = this.obtenerPromo(resultados,text);
929        boolean usado = this.promoUsada(resul, actor);
930        if(usado) return 1;
931
932            db.getTransaction().begin();
933            Promocion mod= db.find(Promocion.class, resul);
934            if(mod.getNum_veces()<=0) return 2;
935            mod.setNum_veces(mod.getNum_veces()-1);
936            Usuario user= db.find(Usuario.class, actor);
937            mod.anadirUsuario(user);
938
939            if(mod.getCompeticion()==null && !mod.isTipo()) user.setSaldo(user.getSaldo()+mod.getCant());
940            else user.anadirPromo(mod);
941
942            db.persist(mod);
943            db.persist(user);
944            db.getTransaction().commit();
945
946
947        }catch(NullPointerException e) {
948            return 3;
949
950        }
951        return 0;
952    }
```

2) Write simple units of code (capítulo 3):

- **Problema:** Complejidad ciclomática muy elevada (18).
- En: src/main/java/gui/RegistrarGUI.java.
- Metodo: private void jbtnRegistrarActionPerformed.

Código inicial:



```
387
388  private void jbtnRegistrarActionPerformed (ActionEvent e) {
389
390      BLFacadeImplementation facadeIm = (BLFacadeImplementation) IniciarSesionGUI.getBusinessLogic();
391
392      lblErrorCampos.setVisible(false);
393      String NombreUsuario = this.textFNombreUsuario.getText();
394
395      if (facadeIm.actorExistente(NombreUsuario))
396          this.lblNUCogido.setVisible(true);
397      else {
398
399          try {
400              ano = Integer.parseInt(this.textFAno.getText());
401
402              if (dia == 29 && ((ano % 4) != 0)) {
403
404                  this.lblErrorCampos.setText(ResourceBundle.getBundle("Etiquetas").getString("ErrorRegistroUsuario2"));
405                  this.lblErrorCampos.setVisible(true);
406              } else
407
408                  if (mes == 0 || dia == 0) {
409
410                      this.lblSeleccionFecha.setVisible(true);
411                  }else {
412
413                      if (sexo == 'p')
414
415                          this.lblSeleccionaGenero.setVisible(true);
416                      else {
417
418                          this.lblSeleccionaGenero.setVisible(false);
419                          this.lblSeleccionFecha.setVisible(false);
420                          this.lblNUCogido.setVisible(false);
421
422                          Date fechaNacimiento = UtilDate.newDate(ano, mes, dia);
423                          String contrasena = String.copyValueOf(this.passwordField.getPassword());
424                          //comboBox = (Offer) offerBox.getSelectedItem();
425
426                          ArrayList<String> datos = new ArrayList<String>();
427                          datos.add(this.textFNombreUsuario.getText());
428                          datos.add(this.textFDni.getText());
429                          datos.add(this.textFName.getText());
430                          datos.add(this.textFApellido1.getText());
431                          datos.add(this.textFApellido2.getText());
432                          datos.add(contrasena);
433                          datos.add(this.textFEmail.getText());
434                          datos.add(this.textFTelefono.getText());
435
436                          int resultado = facadeIm.registrarUsuario(datos, fechaNacimiento, sexo, this.regAdmin);
437
438                          switch (resultado) {
439                          case 0 :
440                              this.regAdmin= false;
441                              cerrarGUI();
442                              break;
443                          case 1:
444                              this.lblErrorCampos.setText(ResourceBundle.getBundle("Etiquetas").getString("ErrorRegistroUsuario1")
445                              this.lblErrorCampos.setVisible(true);
446                              break;
447                          case 2:
448                              this.lblErrorCampos.setText(ResourceBundle.getBundle("Etiquetas").getString("ErrorRegistroUsuario2")
449                              this.lblErrorCampos.setVisible(true);
450                              break;
451                          case 3:
452                              this.lblErrorCampos.setText(ResourceBundle.getBundle("Etiquetas").getString("ErrorRegistroUsuario3")
453                              this.lblErrorCampos.setVisible(true);
454                              break;
455                          default:
456                              this.lblErrorCampos.setText(ResourceBundle.getBundle("Etiquetas").getString("ErrorRegistroUsuario-1"
457                              this.lblErrorCampos.setVisible(true);
458                              break;
459                          }
460
461                      }
462                  }
463
464          } catch (NumberFormatException excepcion) {
465
466              this.lblErrorCampos.setVisible(true);
467              this.lblErrorCampos.setText(ResourceBundle.getBundle("Etiquetas").getString("AnoValido"));
468          }
469      }
470  }
```

- Vemos que el problema reside en la gran cantidad de condiciones que hay, con lo cual crearemos un método auxiliar que realice la comprobación de errores.

- **Solución:**

- comprobaciónErrores → Método de comprobación de errores que mostrará la etiqueta apropiada para cada caso.

- Conseguimos disminuir la complejidad hasta menos del límite establecido.

Código refactorizado:

- **Método comprobaciónErrores:**

```
388
389    public void comprobacionErrores(int resultado) {
390
391        switch (resultado) {
392        case 0 :
393            this.regAdmin= false;
394            cerrarGUI();
395            break;
396        case 1:
397            this.lblErrorCampos.setText(ResourceBundle.getBundle("Etiquetas").getString("ErrorRegistroUsuario1"));
398            this.lblErrorCampos.setVisible(true);
399            break;
400        case 2:
401            this.lblErrorCampos.setText(ResourceBundle.getBundle("Etiquetas").getString("ErrorRegistroUsuario2"));
402            this.lblErrorCampos.setVisible(true);
403            break;
404        case 3:
405            this.lblErrorCampos.setText(ResourceBundle.getBundle("Etiquetas").getString("ErrorRegistroUsuario3"));
406            this.lblErrorCampos.setVisible(true);
407            break;
408        default:
409            this.lblErrorCampos.setText(ResourceBundle.getBundle("Etiquetas").getString("ErrorRegistroUsuario-1"));
410            this.lblErrorCampos.setVisible(true);
411            break;
412        }
413    }
```

- **Método principal a refactorizar:**

```
415    private void jbtnRegistrarActionPerformed (ActionEvent e) {
416
417        BLFacadeImplementation facadeIm = (BLFacadeImplementation) IniciarSesionGUI.getBusinessLogic();
418
419        lblErrorCampos.setVisible(false);
420        String NombreUsuario = this.textFNombreUsuario.getText();
421
422        if (facadeIm.actorExistente(NombreUsuario))
423            this.lblNUCogido.setVisible(true);
424        else {
425
426            try {
427                ano = Integer.parseInt(this.textFAno.getText());
428
429                if (dia == 29 && ((ano % 4) != 0)) {
430
431                    this.lblErrorCampos.setText(ResourceBundle.getBundle("Etiquetas").getString("ErrorRegistroUsuario2"));
432                    this.lblErrorCampos.setVisible(true);
433                } else
434
435                    if (mes == 0 || dia == 0) {
436
437                        this.lblSeleccionFecha.setVisible(true);
438                    }else {
439
440                        if (sexo == 'p')
441
442                            this.lblSeleccionaGenero.setVisible(true);
443                        else {
444
445                            this.lblSeleccionaGenero.setVisible(false);
446                            this.lblSeleccionFecha.setVisible(false);
445                            this.lblSeleccionaGenero.setVisible(false);
446                            this.lblSeleccionFecha.setVisible(false);
447                            this.lblNUCogido.setVisible(false);
448
449                            Date fechaNacimiento = UtilDate.newDate(ano, mes, dia);
450                            String contrasena = String.copyValueOf(this.passwordField.getPassword());
451
452                            ArrayList<String> datos = new ArrayList<String>();
453                            datos.add(this.textFNombreUsuario.getText());
454                            datos.add(this.textFDni.getText());
455                            datos.add(this.textFName.getText());
456                            datos.add(this.textFApellido1.getText());
457                            datos.add(this.textFApellido2.getText());
458                            datos.add(contrasena);
459                            datos.add(this.textFEmail.getText());
460                            datos.add(this.textFTelefono.getText());
461
462                            int resultado = facadeIm.registrarUsuario(datos, fechaNacimiento, sexo, this.regAdmin);
463
464                            this.comprobacionErrores(resultado);
465                        }
466                    }
467
468            } catch (NumberFormatException excepcion) {
469
470                this.lblErrorCampos.setVisible(true);
471                this.lblErrorCampos.setText(ResourceBundle.getBundle("Etiquetas").getString("AnoValido"));
472            }
473        }
474    }
```

3) Duplicate code (capítulo 4):

- **Problema:** Define a constant instead of duplicating this literal "Etiquetas" 22 times.
- **En:** src/main/java/gui/ApostarGUI.java

Código inicial:

```
77
78          lblSlctComp = new JLabel(ResourceBundle.getBundle("Etiquetas").getString("SeleccionaCompeticion"));
79          lblSlctComp.setBounds(10, 28, 185, 19);
80          getContentPane().add(lblSlctComp);
81
82          lblSlctEvent = new JLabel(ResourceBundle.getBundle("Etiquetas").getString("SeleccionaEvento"));
83          lblSlctEvent.setBounds(10, 58, 185, 19);
84          getContentPane().add(lblSlctEvent);
85          lblSlctEvent.setVisible(false);
86
87          lblSlctPreg = new JLabel(ResourceBundle.getBundle("Etiquetas").getString("SeleccionaPregunta"));
88          lblSlctPreg.setBounds(10, 88, 185, 19);
89          getContentPane().add(lblSlctPreg);
90          lblSlctPreg.setVisible(false);
91
92          lblSlctPron = new JLabel(ResourceBundle.getBundle("Etiquetas").getString("SeleccionaPronostico"));
93          lblSlctPron.setBounds(10, 118, 185, 19);
94          getContentPane().add(lblSlctPron);
95          lblSlctPron.setVisible(false);
96
97          comboBoxCompeticiones = new JComboBox<String>();
98          comboBoxCompeticiones.setBounds(205, 26, 233, 22);
99          getContentPane().add(comboBoxCompeticiones);
00
01          comboBoxEventos = new JComboBox<String>();
02          comboBoxEventos.setBounds(205, 56, 233, 22);
03          getContentPane().add(comboBoxEventos);
04          comboBoxEventos.setVisible(false);
05
```

Solución:

- Nos encontramos en muchas partes del código perteneciente a la clase ApostarGUI con duplicaciones de código que se pueden resolver definiendo una constante y sustituyéndola por las duplicaciones.

Código refactorizado:

```
57      private JButton btnVolver;
58
59      private Vector<Competicion> competiciones;
60      private Vector<Evento> eventos;
61      private Vector<Pregunta> preguntas;
62      private Vector<Pronostico> pronosticos;
63      BLFacadeImplementation facade = (BLFacadeImplementation) IniciarSesionGUI.getBusinessLogic();
64
65      static final String etiquetas = "Etiquetas";
66      static final String a = "Tahoma";
67      static final String o = "SeleccionaOpcion";
68
```

Nota: Además de la constante etiquetas, hemos añadido dos constantes más para eliminar todas las duplicaciones de la clase ApostarGUI.

4) Keep unit interfaces small (capítulo 5):

- **Problema:** Method has 11 parameters, which is greater than 7 authorized.
- **Metodo: registrarUsuario (en → dataAccess/DataAccess.java).**

Código inicial:

```
195⊖ /**
196   * Crea y anade a la base de datos un nuevo usuario a la base de datos con los datos introducidos
197   *
198   * @param sexo (char): Masculino = 'M'; Femenino = 'F'; Otro = 'O'
199   * @param admin :si es true est registrando un administrador.
200   */
201⊖ public void registrarUsuario(String nombreUsuario, String Dni, String nombre, String apellido1, String apellido2, Date fechaNacimi
202
203       db.getTransaction().begin();
204
205       if(!admin) { //se anade un usuario
206           Actor a = new Usuario(nombreUsuario,Dni,nombre,apellido1,apellido2,fechaNacimiento,contrasena,sexo,email,tlfn);
207           db.persist(a);
208       }
209       else { //se anade un administrador
210           Actor a = new Administrador(nombreUsuario,Dni,nombre,apellido1,apellido2,fechaNacimiento,contrasena,sexo,email,tlfn);
211           db.persist(a);
212       }
213
214       db.getTransaction().commit();
215   }
216
```

```
198
199
200
201  String apellido1, String apellido2, Date fechaNacimiento, String contrasena, char sexo, String email,String tlfn, boolean admin) {
202
203
204
205
206 haNacimiento,contrasena,sexo,email,tlfn);
207
208
209
210 lo2,fechaNacimiento,contrasena,sexo,email,tlfn);
211
212
213
```

- Nos encontramos con el problema de que tenemos demasiados argumentos para el método de registrarUsuario, concretamente 11.
- Nuestro objetivo será reducirlo hasta 4.

Solución:

- Para solucionar este bad smell, lo que haremos es crear un nuevo array (datos), donde almacenaremos todos los datos que recibimos como Strings.
- Ese array será el nuevo parámetro que reemplazará a los 8 string que había antes.

- Para ello debemos realizar los cambios pertinentes en cada una de las clases que participan a la hora de registrar un usuario, concretamente en el RegistrarGUI, BLFacadeImplementation y en el DataAccess.

Código refactorizado:

- **RegistrarGui:**

```
415                     this.lblSeleccionaGenero.setVisible(true);
416                 else {
417
418                     this.lblSeleccionaGenero.setVisible(false);
419                     this.lblSeleccionFecha.setVisible(false);
420                     this.lblNUCogido.setVisible(false);
421
422                     Date fechaNacimiento = UtilDate.newDate(ano, mes, dia);
423                     String contrasena = String.copyValueOf(this.passwordField.getPassword());
424                     //comboBox = (Offer) offerBox.getSelectedItem();
425
426                     ArrayList<String> datos = new ArrayList<String>();
427                     datos.add(this.textFNombreUsuario.getText());
428                     datos.add(this.textFDni.getText());
429                     datos.add(this.textFName.getText());
430                     datos.add(this.textFApellido1.getText());
431                     datos.add(this.textFApellido2.getText());
432                     datos.add(contrasena);
433                     datos.add(this.textFEmail.getText());
434                     datos.add(this.textFTelefono.getText());
435
436                     int resultado = facadeIm.registrarUsuario(datos, fechaNacimiento, sexo, this.regAdmin);
437
438                     switch (resultado) {
439                     case 0 :
440                         this.regAdmin= false;
441                         cerrarGUI();
442                         break;
443                     case 1:
444                         this.lblErrorCampos.setText(ResourceBundle.getBundle("Etiquetas").getString("ErrorRegistroUsuario1"
445                         this.lblErrorCampos.setVisible(true);
446                         break;
```

- **BLFacadeImplementation:**

```
74      * @return 0: Todx correcto; 1: Fecha es incorrecta; 2: Saldo es negativo
75      */
76     public int registrarUsuario(ArrayList<String> datos, Date fechaN, char sexo, boolean admin) {
77
78
79         dbManager.open(false);
80
81         Date fechaHoy = UtilDate.currentDate();
82         Date fechaHace18 = UtilDate.fechaMayorEdad();
83
84         if (dbManager.actorExistente(datos.get(0))) {
85
86             dbManager.close();
87             return 1; //El usuario que intenta anadir ya existe
88         }
89         else
90
91             if (fechaHoy.before(fechaN)) {
92                 dbManager.close();
93                 return 2; //Comprueba que la fecha es anterior al momento actual (Para que no se pueda poner que has nacido en 2025)
94             }
95
96             else
97                 if (fechaN.before(fechaHace18))
98                     dbManager.registrarUsuario(datos, fechaN, sexo, admin);
99                 else
100                    return 3;
101
102        dbManager.close();
103        return 0;
104    }
105
```

- **DataAccess:**

```
/**
 * Crea y anade a la base de datos un nuevo usuario a la base de datos con los datos introducidos
 *
 * @param sexo (char): Masculino = 'M'; Femenino = 'F'; Otro = 'O'
 * @param admin :si es true est registrando un administrador.
 */
public void registrarUsuario(ArrayList<String> datos, Date fechaNacimiento, char sexo, boolean admin) {

    db.getTransaction().begin();

    if(!admin) { //se anade un usuario
        Actor a = new Usuario(datos.get(0),datos.get(1),datos.get(2),datos.get(3),datos.get(4),fechaNacimiento,datos.get(5),sexo,
        db.persist(a);
    }
    else { //se anade un administrador
        Actor a = new Usuario(datos.get(0),datos.get(1),datos.get(2),datos.get(3),datos.get(4),fechaNacimiento,datos.get(5),sexo,
    }

    db.getTransaction().commit();
}
```

Ander Caro:

Para realizar los primeros tres apartados, me di cuenta que en el método crearApuesta(…) de la clase DataAccess incumplia las tres primeras reglas, y por tanto, decidi “arreglar” dicho método, realizando de esa manera la corrección de las tres primeras .

1) **Write short units of code (capítulo 2):**
2) **Write simple units of code (capítulo 3):**
3) **Duplicate code (capítulo 4):**

Se puede observar como claramente se superan las quince líneas de código. Para solucionar dicho problema, dividí el método en diferentes funcionalidades que tenía dentro implementadas, como por ejemplo comprobar que la apuesta se ha realizado de manera correcta. Para ello, cree nuevos métodos.

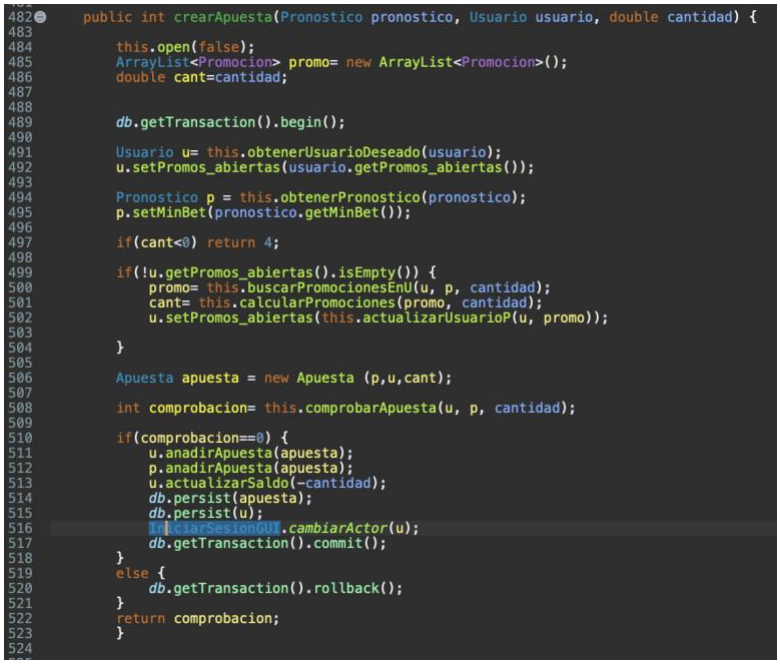
A su vez, la complejidad ciclomática ascendía a nueve, y mediante la solución del primero apartado, conseguí, reducir dicha complejidad.

También, se puede observas como entre las líneas 457 y 463 hay código que hace la misma función solo que con un *return* disitinto. Como en los anteriores apartados, creando un método aparte que devolviera el valor que debía estar en el *return*, se solucionó el problema.

Código inicial:

```
407    public int crearApuesta(Pronostico pronostico, Usuario usuario, double cantidad) {
408
409        this.open(false);
410        ArrayList<Promocion> promo= new ArrayList<Promocion>();
411        double cant=cantidad;
412        db.getTransaction().begin();
413
414        Usuario u= this.obtenerUsuarioDeseado(usuario);
415
416        Pronostico p = db.find(Pronostico.class, pronostico.getId());
417
418        if(u==null | p==null | cantidad < 0.0) return 4;
419
420        if(!usuario.getPromos_abiertas().isEmpty()) {
421        for(Promocion e: usuario.getPromos_abiertas()) {
422            if(p.getComp().equals(e.getNombreComo())) {
423                promo.add(e);
424            }
425        }
426
427
428        for(Promocion b: promo) {
429            if(!b.isTipo()) {
430                double aux2=b.getCant();
431                cant=(cantidad + aux2);
432            }
433            else {
434                double aux= b.getCant();
435                cant =(cantidad + (cantidad*(aux/100.0)));
436            }
437            int index= usuario.getPromos_abiertas().indexOf(b);
438            usuario.getPromos_abiertas().remove(index);
439        }
440
441        }
442
443        Apuesta apuesta = new Apuesta (pronostico,usuario,cant);
444        u.setPromos_abiertas(usuario.getPromos_abiertas());
445
446        if (u.getSaldo() >= cantidad) {
447            if ((p.getPregunta().getMinBet()) < cant) {
448                System.out.println(p.getPregunta().getMinBet());
449                u.anadirApuesta(apuesta);
450                p.anadirApuesta(apuesta);
451                u.actualizarSaldo(-cantidad);
452                db.persist(apuesta);
453                db.persist(u);
454                IniciarSesionGUI.cambiarActor(u);
455                db.getTransaction().commit();
456                return 0;
457            } else {
458                db.getTransaction().rollback();
459                return 1;
460            }
461        }else {
462            db.getTransaction().rollback();
463            return 2;
464        }
465
```

El método crearApuesta(…), quedó así, donde se puede ver la considerable reducción de código:



```
482     public int crearApuesta(Pronostico pronostico, Usuario usuario, double cantidad) {
483
484         this.open(false);
485         ArrayList<Promocion> promo= new ArrayList<Promocion>();
486         double cant=cantidad;
487
488
489         db.getTransaction().begin();
490
491         Usuario u= this.obtenerUsuarioDeseado(usuario);
492         u.setPromos_abiertas(usuario.getPromos_abiertas());
493
494         Pronostico p = this.obtenerPronostico(pronostico);
495         p.setMinBet(pronostico.getMinBet());
496
497         if(cant<0) return 4;
498
499         if(!u.getPromos_abiertas().isEmpty()) {
500             promo= this.buscarPromocionesEnU(u, p, cantidad);
501             cant= this.calcularPromociones(promo, cantidad);
502             u.setPromos_abiertas(this.actualizarUsuarioP(u, promo));
503
504         }
505
506         Apuesta apuesta = new Apuesta (p,u,cant);
507
508         int comprobacion= this.comprobarApuesta(u, p, cantidad);
509
510         if(comprobacion==0) {
511             u.anadirApuesta(apuesta);
512             p.anadirApuesta(apuesta);
513             u.actualizarSaldo(-cantidad);
514             db.persist(apuesta);
515             db.persist(u);
516             IniciarSesionGUI.cambiarActor(u);
517             db.getTransaction().commit();
518         }
519         else {
520             db.getTransaction().rollback();
521         }
522         return comprobacion;
523         }
524
```

A su vez, estos son los métodos creados para ello:

```
470
471     public int comprobarApuesta(Usuario u, Pronostico p, double cantidad) {
472
473         if(u.getSaldo() >= cantidad) {
474             if((p.getPregunta().getMinBet())<cantidad) {
475                 return 0;
476             }
477             return 1;
478         }
479         return 2;
480     }
481
```

```
452     public double calcularPromociones(ArrayList<Promocion> promos,double cantidad) {
453         double cant=cantidad;
454
455         for(Promocion e: promos) {
456             if(!e.isTipo()) {
457                 double aux2=e.getCant();
458                 cant=(cantidad + aux2);
459                 }
460                 else {
461                 double aux= e.getCant();
462                 cant =(cantidad + (cantidad*(aux/100.0)));
463                 }
464         }
465
466
467         return cant;
468
469     }
```

```
20    public ArrayList<Promocion> buscarPromocionesEnU (Usuario us, Pronostico p, double cantidad) {
21
22        double cant=cantidad;
23        ArrayList<Promocion> promos = new ArrayList<Promocion>();
24
25        Usuario u= this.obtenerUsuarioDeseado(us);
26        u.setPromos_abiertas(us.getPromos_abiertas());
27
28        for(Promocion e: u.getPromos_abiertas()) {
29            if(p.getComp().equals(e.getNombreComo())) {
30                promos.add(e);
31            }
32        }
33            return promos;
34            }
35
```

```
436
437    public ArrayList<Promocion> actualizarUsuarioP (Usuario us, ArrayList<Promocion> pro
438
439        Usuario u= this.obtenerUsuarioDeseado(us);
440        u.setPromos_abiertas(us.getPromos_abiertas());
441
442        for(Promocion e: promo) {
443            u.getPromos_abiertas().remove(e);
444            System.out.println(u.getPromos_abiertas().size());
445        }
446        db.persist(u);
447
448        return u.getPromos_abiertas();
449    }
450
451
```

```
98    public Usuario obtenerUsuarioDeseado(Usuario u) {
99
00        TypedQuery<Usuario> query = db.createQuery("SELECT u FROM Usuario u",Usuario.class);
01        List<Usuario> prontc = query.getResultList();
02
03
04        for(Usuario e: prontc) {
05            if(u.getNombreUsuario().equals(e.getNombreUsuario())) {
06                return e;
07            }
08        }
09        return null;
10
11    }
12
13    public Pronostico obtenerPronostico(Pronostico p) {
14        Pronostico pr=db.find(Pronostico.class, p.getId());
15        return pr;
16
17    }
18
```

4) Keep unit interfaces small (capítulo 5):

El método crear apuesta si que cumplía con esta regla, por tanto, al buscar otro método que no la cumpliera y así poder solventarlo, encontré el método crearCompeticion(…).

Codigo incial:

```
337     public Competicion crearCompeticion(String nombre, String deporte, char genero, String temporada, String descripcion, Actor admin) {
338
339         db.getTransaction().begin();
340         Administrador admin0= (Administrador) db.find(Actor.class, admin.getNombreUsuario());
341         Competicion competicion = new Competicion(nombre,  deporte,  genero,  temporada, descripcion, admin0);
342         db.persist(competicion);
343         db.getTransaction().commit();
344
345         return competicion;
346     }
```

Como se puede ver, los parámetros que se le pasan al método son más que cuatro, y por tanto, no cumple esta regla. Para solucionarlo, cambie todos los parámetros de tipo String que tenía la función por un Array List de Strings:

```
336
337     public Competicion crearCompeticion( char genero, Actor admin, ArrayList<String> data) {
338
339         db.getTransaction().begin();
340         Administrador admin0= (Administrador) db.find(Actor.class, admin.getNombreUsuario());
341         Competicion competicion = new Competicion(data.get(0),  data.get(2),  genero,  data.get(2), data.get(3), admin0);
342         db.persist(competicion);
343         db.getTransaction().commit();
344
345         return competicion;
346     }
347
```

Sin embargo, al hacer este cambio, generaba conflictos con las clases que llamaban a dicho método (BLFacadeImplementation y crearCompeticionGUI). Para solventarlos tuve que hacer los siguientes cambios:

BLFacadeImplementation original:

```
160     public Competicion crearCompeticion(String nombre, String deporte, char genero, String temporada, String descripcion, Actor admin) {
161
162         dbManager.open(false);
163
164         Competicion competicion = null;
165         if (!dbManager.existeCompeticion("Competicion:" + nombre)) {
166             competicion = dbManager.crearCompeticion(nombre, deporte, genero, temporada, descripcion, admin);
167         }
168         //Si la competicion ya est@ a@adida.
169         dbManager.close();
170         return competicion;
171     }
172
```

BLFacadeImplementation solución:

```
160     public Competicion crearCompeticion(char genero, Actor admin, ArrayList<String> data) {
161
162         dbManager.open(false);
163
164         Competicion competicion = null;
165         if (!dbManager.existeCompeticion("Competicion:" + data.get(0))) {
166             competicion = dbManager.crearCompeticion( genero, admin, data);
167         }
168         //Si la competicion ya est@ a@adida.
169         dbManager.close();
170         return competicion;
171     }
172
```

crearCompeticionGUI original:

```
188             facade.crearCompeticion(this.textFNombre.getText(), this.textFDeporte.getText(), sexo, temporada, this.textFDeporte.getText(), IniciarSesionGUI.actor());
```

crearCompeticionGUI solución:

```
189             data.add(this.textFNombre.getText());
190             data.add(this.textFDeporte.getText());
191             data.add(temporada);
192             data.add(this.textFDeporte.getText());
193
194
195             facade.crearCompeticion(sexo, IniciarSesionGUI.actor(), data);
```