# Patrones de Diseño IS2

## 2023/2024

## Proyecto Bets

**Autores:**
Unai Artano
Asier Contreras
Martin Ian Horsfield

eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea

# ÍNDICE



# ENLACE PROYECTO

https://github.com/asiercontreras/IS2-RepBets

# SIMPLE FACTORY

La ventaja que nos da el patrón de diseño Factory es facilitar el trabajo, en este caso, a la clase ApplicatioLauncer; es decir, a la que ejecuta la aplicación. En vez de que la aplicación cree la lógica del negocio, usaremos una clase externa llamada BLFactory, que será la clase Factory, para crear la lógica de negocio. Una vez creada, en la ApplicationLauncher llamaremos a dicha clase para crear la lógica del negocio que estaba antes implementada.
En dicha clase del factory, se ha puesto un método *crearBLFacade*; donde se le pasa un objeto de la clase ConfigXML para que a la hora de crear la lógica del negocio tenga en cuenta el fichero config.xml. Para ello, en la clase ApplicationLauncher, creamos un objeto ConfigXML para que haga una instancia a dicha clase, y para poder llamar al método *crearBLFacade*.

A continuación, se mostrarán los respectivos cambios que se han mencionado anteriormente:

**Clase ApplicationLauncher:**

```
private static BLFactory blfact = new BLFactory();

public static void main(String[] args) {

    ConfigXML c = ConfigXML.getInstance();
    appFacadeInterface = blfact.crearBFFacade(c);

    char[] passwd = { '1', '1' };
    String passS = "";
    for (char ch : passwd) {
        passS += ch;
    }
    try {
        appFacadeInterface.createUser(new User("asier", "contreras", "11",
                new Date(2000, 5, 1), passS, false));
        appFacadeInterface
                .createUser(new User("Adminitrador", "Administrador", "00",
                        new Date(2000, 5, 1), passS, true));
    } catch (Exception e) {
        // TODO: handle exception
    }

    // a.pack();

    i = new InitGUI();
    i.setVisible(true);
}
```

**Clase BLFactory:**

(Ignorando el package y los imports)

```
public class BLFactory {

    private static InitGUI i;

    private static BLFacade appFacadeInterface;

    public BLFacade crearBFFacade(ConfigXML c) {

        System.out.println(c.getLocale());

        Locale.setDefault(new Locale(c.getLocale()));

        System.out.println("Locale: " + Locale.getDefault());

        try {

            // UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsClassicLookAndFeel");
            // UIManager.setLookAndFeel("com.sun.java.swing.plaf.motif.MotifLookAndFeel");
            UIManager.setLookAndFeel("javax.swing.plaf.metal.MetalLookAndFeel");

            if (c.isBusinessLogicLocal()) {
                // In this option the DataAccess is created by FacadeImplementationWS
                // appFacadeInterface=new BLFacadeImplementation();
                // In this option, you can parameterize the DataAccess (e.g. a Mock DataAccess
                // object)
                DataAccess da = new DataAccess(c.getDataBaseOpenMode().equals("initialize"));
                appFacadeInterface = new BLFacadeImplementation(da);
            } else { // If remote
                String serviceName = "http://" + c.getBusinessLogicNode() + ":" + c.getBusinessLogicPort() + "/ws/"
                        + c.getBusinessLogicName() + "?wsdl";
                // URL url = new URL("http://localhost:9999/ws/ruralHouses?wsdl");
                URL url = new URL(serviceName);

              // 1st argument refers to wsdl document above
              // 2nd argument is service name, refer to wsdl document above
              // QName qname = new QName("http://businessLogic/",
              // "FacadeImplementationWSService");
              QName qname = new QName("http://businessLogic/", "BLFacadeImplementationService");
              Service service = Service.create(url, qname);
              appFacadeInterface = service.getPort(BLFacade.class);
            }
            return appFacadeInterface;
            /*
             * if (c.getDataBaseOpenMode().equals("initialize"))
             * appFacadeInterface.initializeBD();
             */

        } catch (Exception e) {
            i.getIJlabelSelectOption().setText("Error: " + e.toString());
            i.getIJlabelSelectOption().setForeground(Color.RED);

            System.out.println("Error in ApplicationLauncher: " + e.toString());
        }

        return null;
}
```

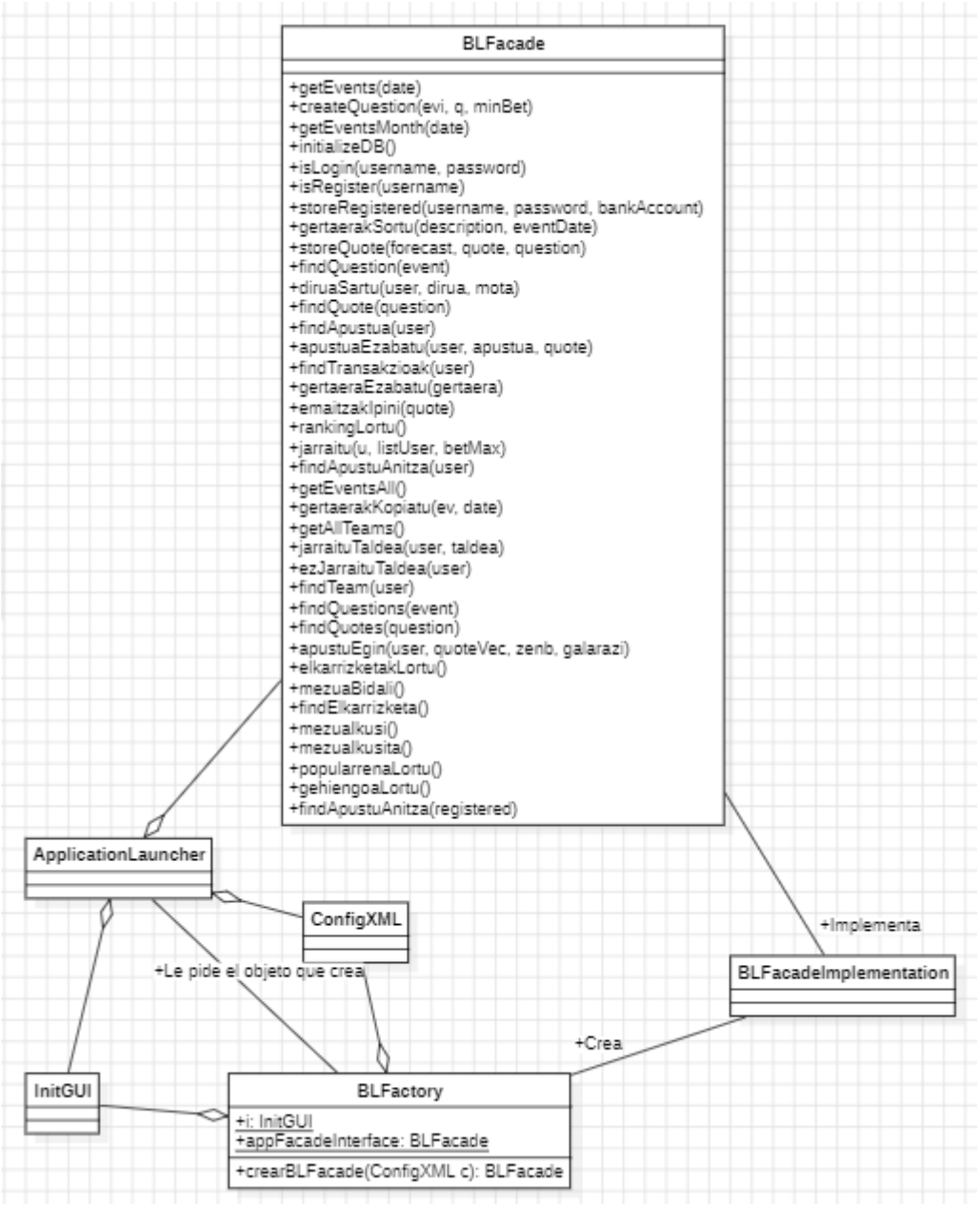
Un cambio extra que se ha aplicado para poder hacer la clase Factory, es a la hora de llamar al JLabel de la clase InitGUI. Al ser protected en la clase BLFactory daba error cuando la llamábamos. Para ello, se ha hecho un getter en la clase InitGUI a dicho JLabel.

**Clase InitGUI:**

```
protected JLabel jLabelSelectOption;

public JLabel getIJlabelSelectOption() {
    return this.jLabelSelectOption;
}
```

El respectivo UML del Factory:



# ITERATOR

El método implementado con la signatura dada es el siguiente:

```
    public  ExtendedIterator<Event> getEventsIterator(Date date){
        dbManager.open(false);
        Vector<Event> events = dbManager.getEvents(date);
        dbManager.close();
        ArrayList<Event> list = new ArrayList<Event>();
        for(Event e: events){
            list.add(e);
        }
        return new ExtendedIterator<Event>(list);
    }
```

Este método nos devolverá un iterador de eventos que será la clase ExtendedIterator en vez de un vector de Eventos como lo hace el método getEvents. La implementación de la clase ExtendedIterator es la siguiente:

```
package businessLogic;

import java.util.Iterator;
import java.util.List;

public class ExtendedIterator<Object> implements Iterator<Object> {
    private List<Object> eventList;
    private int currentIndex;

    public ExtendedIterator(List<Object> events) {
        this.eventList = events;
        this.currentIndex = 0;
    }

    public Object previous() {
        if (hasPrevious()) {
            return eventList.get(--currentIndex);
        }
        return null;
    }

    public boolean hasPrevious() {
        return currentIndex > 0;
    }

    public void goFirst() {
        currentIndex = -1;
    }

    public void goLast() {
        currentIndex = eventList.size();
    }

    public boolean hasNext() {
        return currentIndex < eventList.size() - 1;
    }

    public Object next() {
        if (hasNext()) {
            return eventList.get(++currentIndex);
        }
        return null; // O puedes lanzar una excepción aquí si lo prefieres
    }

}
```

Para probar este método se ha hecho dentro de la carpeta test un nuevo package llamado test.patrones donde se testean todos los métodos relacionados con los patrones.
Para este método se ha creado la clase de testing llamada iteratorTest. Una vez realizada la ejecución la consola de eclipse muestra lo siguiente:

```
_____________________
RECORRIDO       HACIA   ATRÁS
16;Betis-Real Madrid
15;Real Sociedad-Levante
14;Girona-Leganés
13;Malaga-Valencia
12;Las Palmas-Sevilla
11;Español-Villareal
10;Alavés-Deportivo
9;Getafe-Celta
8;Eibar-Barcelona
7;Atlético-Athletic


_____________________
RECORRIDO       HACIA   ADELANTE
7;Atlético-Athletic
8;Eibar-Barcelona
9;Getafe-Celta
10;Alavés-Deportivo
11;Español-Villareal
12;Las Palmas-Sevilla
13;Malaga-Valencia
14;Girona-Leganés
15;Real Sociedad-Levante
16;Betis-Real Madrid
```

# PATRÓN ADAPTER

Para este ejercicio se pide implementar una nueva GUI con el patrón adapter, enseñando las apuestas que tiene una persona. El GUI nuevo está implementado de la siguiente manera:

```
1  package gui;
2
3  import java.awt.BorderLayout;
4  import java.awt.Dimension;
5
6  import javax.swing.JFrame;
7  import javax.swing.JTable;
8  import javax.swing.JScrollPane;
9
10 import domain.*;
11
12 public class WindowTable extends JFrame{
13     private JTable  tabla;
14
15     public WindowTable(User user){
16         super("Apuestas realizadas  por "+  user.getName()+":");
17         this.setBounds(100, 100,    700,    200);
18         UserAdapter adapt   =   new UserAdapter(user);
19         tabla = new JTable(adapt);
20         tabla.setPreferredScrollableViewportSize(new Dimension(500, 70));
21         //Creamos un JscrollPane    y   le agregamos la JTable
22         JScrollPane scrollPane =    new JScrollPane(tabla);
23         //Agregamos el  JScrollPane al contenedor
24         getContentPane().add(scrollPane,    BorderLayout.CENTER);
25     }
26 }
```

Así se especifica que se implemente la clase en el ejercicio. Para que funcione el JTable, eclipse recomienda implementar el formato de tabla en UserAdapter, así que lo realizamos.

La clase UserAdapter se ha definido de la siguiente manera:

```java
package domain;

import java.util.ResourceBundle;
import java.util.Vector;

import javax.swing.event.TableModelListener;
import javax.swing.table.TableModel;

import businessLogic.BLFacade;
import gui.ApplicationLauncher;

public class UserAdapter implements TableModel {

    private User user;
    private int columnas = 4;
    private String[] columnBets = { ResourceBundle.getBundle("Etiquetas").getString("Event") + ":",
            ResourceBundle.getBundle("Etiquetas").getString("Question") + ":",
            ResourceBundle.getBundle("Etiquetas").getString("Date"),
            ResourceBundle.getBundle("Etiquetas").getString("Bet") + ":" };
    private BLFacade facade = ApplicationLauncher.getBusinessLogic();
    private Vector<Bet> bets;


    public UserAdapter(User user) {
        this.user=user;
        bets = user.getBets(); //No creo que haya una mejor manera. Se podría hacer otro método.

    }

    @Override
    public int getRowCount() {
        // TODO Auto-generated method stub
        return user.getBets().size();
    }

    @Override
    public int getColumnCount() {
        // TODO Auto-generated method stub
        return columnas;
    }
```

```java
    @Override
    public String getColumnName(int columnIndex) {
        // TODO Auto-generated method stub
        return columnBets[columnIndex];
    }

    @Override
    public Object getValueAt(int rowIndex, int columnIndex) {
        // TODO Auto-generated method stub
        switch(columnIndex) {
        case 0:
            return bets.get(rowIndex).getFr().getQuestion().getEvent().getDescription();
        case 1:
            return bets.get(rowIndex).getFr().getQuestion().getQuestion();
        case 2:
            return bets.get(rowIndex).getFr().getQuestion().getEvent().getEventDate();
        case 3:
            return bets.get(rowIndex).getBet();
        default:
            return null;
        }
    }

    @Override
    public Class<?> getColumnClass(int columnIndex) {
        // TODO Auto-generated method stub
        return columnBets[columnIndex].getClass();
    }

    @Override
    public boolean isCellEditable(int rowIndex, int columnIndex) {
        // TODO Auto-generated method stub
        return false;
    }

    @Override
    public void setValueAt(Object aValue, int rowIndex, int columnIndex) {
        // TODO Auto-generated method stub

    }
```

Todas las clases las define eclipse automáticamente y solo hace falta implementarlas nosotros. En la gran mayoría de los casos, la implementación es obvia pero en otros, tenemos que realizar llamadas a la lógica de negocio. Cabe destacar que hay algunas llamadas que pueden resultar raras, por ejemplo:

bets.get(rowIndex).getFr().getQuestion().getEvent().getDescription()

Parece una manera compleja de llegar al nombre del evento pero como la apuesta no guarda el evento directamente, tenemos que seguir todos los pasos para llegar a la información que queremos.

Para poder mostrar la tabla hemos creado un botón en la página del usuario en MainGUI con el siguiente código:

```
556    private JButton getJButtonAdapterTest() {
557        if (adapterBets == null) {
558            adapterBets = new JButton();
559            adapterBets.setText(ResourceBundle.getBundle("Etiquetas").getString("AdapterTest"));
560            adapterBets.setBounds(350, 250, 100, 30);
561            adapterBets.setBorder(BorderFactory.createEmptyBorder());
562            adapterBets.setBackground(new Color(180, 180, 180));
563            adapterBets.addMouseListener(new MouseAdapter() {
564                @Override
565                public void mouseEntered(MouseEvent e) {
566                    adapterBets.setBackground(new Color(70, 130, 180));
567                    adapterBets.setFocusPainted(false);
568                }
569
570                @Override
571                public void mouseExited(MouseEvent e) {
572                    adapterBets.setBackground(new Color(180, 180, 180));
573                }
574            });
575
576            adapterBets.addActionListener(new ActionListener() {
577                public void actionPerformed(ActionEvent e) {
578                    JFrame a = new WindowTable(appFacadeInterface.getCurrentUser());
579
580                    a.setVisible(true);
581                }
582            });
583        }
584        return adapterBets;
585    }
586 }
```

Con el botón creado y definido, lo hemos tenido que mostrar:

```
103     private JPanel getJContentPane() {
104         if (jContentPane == null) {
105             jContentPane = new JPanel();
106             jContentPane.setLayout(null);
107             jContentPane.add(getLblNewLabel());
108             jContentPane.add(getPanel());
109             jContentPane.add(getBtnLogOut());
110             jContentPane.add(this.getBtnChangeProfile());
111             if (appFacadeInterface.getCurrentUser().isAdmin()) {
112                 jContentPane.add(this.getJButtonCreateQuery());
113                 jContentPane.add(this.getJButtonCreateEvent());
114                 jContentPane.add(this.getjButtonCloseEvent());
115                 jContentPane.add(this.getJButtonCreateForecast());
116             } else {
117                 jContentPane.add(this.getJButtonQueryQueries());
118                 jContentPane.add(this.getJButtonMakeBet());
119                 jContentPane.add(this.getJButtonAddMoney());
120                 jContentPane.add(this.getJButtonLookBets());
121                 jContentPane.add(this.getJButtonAdapterTest());
122             }
123         }
124 
125         return jContentPane;
126     }
```

También hemos tenido que modificar las etiquetas para que se mostrase el botón de manera correcta.

Tanto el botón como el GUI se muestran correctamente y funcionan correctamente.