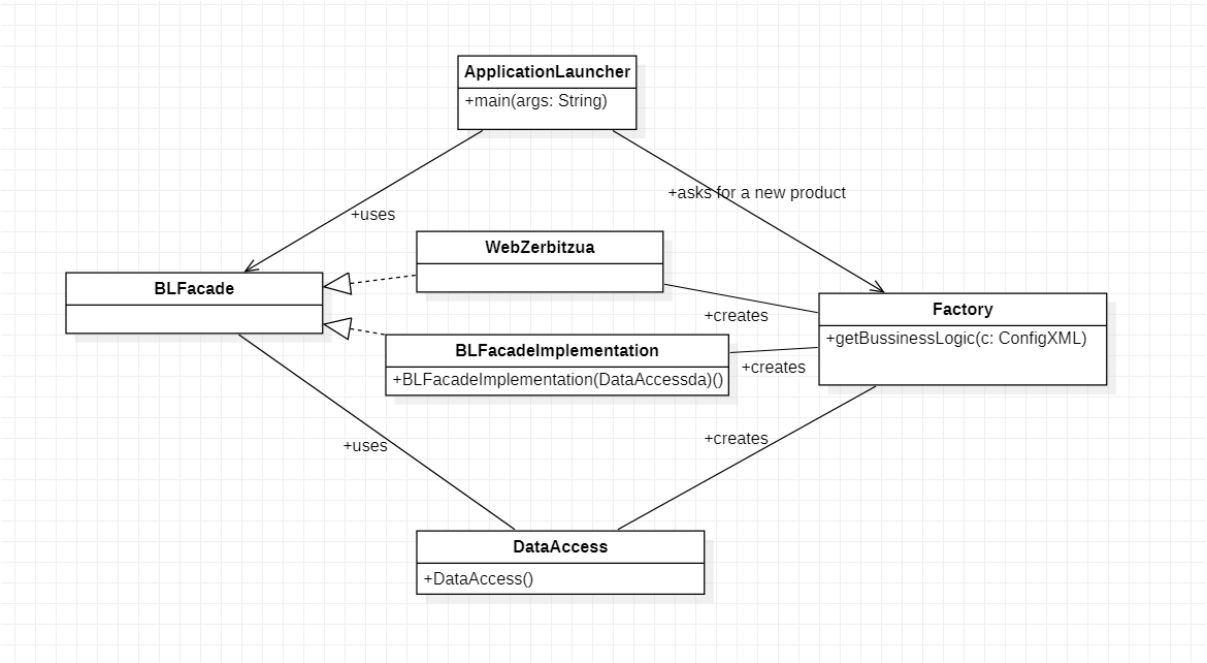
PROIEKTUAREN DISEINU PATROIAK

EGILEAK: EKAITZ PINEDO, BEÑAT ERCIBENGOA, URTZI ETXEGARAI
GITHUB ESTEKA:https://github.com/UrtziE/SI2Proiektua

Factory

1. UML diagrama



Egin dugun aldaketa izan da ApplicationLauncher-en zegoen negozio logika objektuaren sorkuntzaren logika BLFactory-ra mugitu. Beraz, orain ApplicationLauncher-ek BLFactor objektu bat sortuko du eta honen getBussinessLogic metodoaren bitartez aplikazioa irekitzerakoan erabili behar duen negozio logika eskuratuko du. UML diagraman argi ikusten da BLFactory Creator dela, ConcreteProduct WebZerbitzua eta BLFacadeImplementation direla, eta azkenik, Product BLFacade dela. Beste hitz batzueta, ApplicationLauncher-ek Product bat eskuratu nahi du. Lan hori BLFactory-ri ematen dio. Honek, ConfigXML fitxategiaren edukiaren arabera ConcreteProduct bat edo bestea itzuliko du, zeinak Product-ren inplementazioak diren, eta beraz, ApplicationLauncherrentzat baliagarriak direnak.

2.Kodearen azalpena

```
package gui;

import java.awt.Color;

public class ApplicationLauncher {



    public static void main(String[] args) {

        ConfigXML c=ConfigXML.getInstance();

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

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

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

        Driver driver=new Driver("driver3@gmail.com","Test Driver");


        MainGUI a=new MainGUI();
        a.setVisible(true);


        try {
            UIManager.setLookAndFeel("javax.swing.plaf.metal.MetalLookAndFeel");

            BLFactory factory = new BLFactory();
            BLFacade appFacadeInterface = factory.getBusinessLogicFactory(c);

            MainGUI.setBussinessLogic(appFacadeInterface);

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

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


    }

}
```

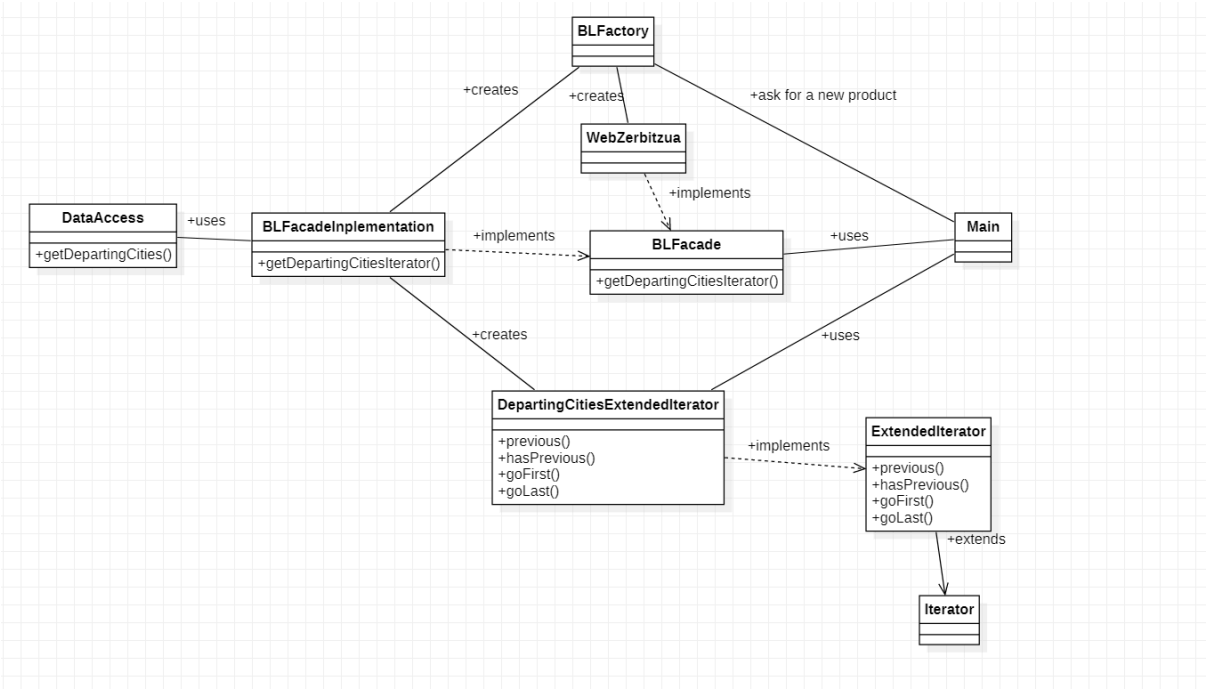
ApplicationLauncherren BLFacade instantzia sortzeko lana BLFactory-ri eman diogu.

```java
package businessLogic;

import java.net.URL;

public class BLFactory {

    public BLFacade getBusinessLogicFactory(ConfigXML c) throws Exception{
        BLFacade appFacadeInterface;
        UIManager.setLookAndFeel("javax.swing.plaf.metal.MetalLookAndFeel");

        //If locale
        if (c.isBusinessLogicLocal()) {
            DataAccess da= new DataAccess();
            appFacadeInterface = new BLFacadeImplementation(da);
        }else { //If remote

            String serviceName= "http://"+c.getBusinessLogicNode() +":"+ c.getBusinessLogicPort()+"/ws/"+c.getBusinessLogicName()+"?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/", "BLFacadeImplementationService");

            Service service = Service.create(url, qname);

             appFacadeInterface = service.getPort(BLFacade.class);
        }

        return appFacadeInterface;
    }

}
```

Lehen ApplicationLauncherren exekutatzen zen kodea BLFactory-ren getBussinessLogic metodora mugitu dugu, aldaketa nabarmenik gabe. Garrantzitsua da kontuan hartzea Exception salbuespena altxatzen duela (ApplicationLauncherrek hau altxatzen bada jada zuzen maneiatzen du egoera).

Iterator

1. UML diagrama



UML diagrama honetan ikusi daiteke aurreko ataleko BLFactory-ren inplementazioa eta gero Iterator-ak gehituta. Extendediterator iterator klaseko semea da eta Departing CitiesExtendedIterator-ek inplementatzen du Extended Iterator. Gainera, BLFacadeInplementation-ek getDepartingCitiesIterator() metodoan DepartingCitiesExtendedIterator- bat sortzen du. Azkenik, main metodoan BLFactory eta BLfacade erabiltzen ditu DepartingCitiesExtendedIterator bat sortzeko, eta Main klaseak azken klase hau erabili egiten du eragiketak gauzatzeko.

2. Kodearen azalpena

```
import java.util.Iterator;

public interface ExtendedIterator<Object>   extends Iterator<Object>    {

    public Object   previous();

    public boolean hasPrevious();

    public void goFirst();

    public void goLast();
    }
```

ExtendedIterator interfazea sortu dugu, hau da, nolako iterator-a inplementatuko den programan, iterator honek aurretik atzera edo atzetik aurrera korritu dezake

```
import java.util.ArrayList;
import java.util.List;
import java.util.Vector;

public class DepartingCitiesExtendedIterator<String> implements ExtendedIterator<Object> {
    List<String> depCities= new ArrayList<String>();
    int i=0;
    public DepartingCitiesExtendedIterator(List<String> departingCities) {
        this.depCities=departingCities;
    }

    @Override
    public boolean hasNext() {
        return i<=depCities.size()-1;

    }

    @Override
    public Object next() {
        String itzuli= depCities.get(i);
        i=i+1;
        return itzuli;
    }

    @Override
    public Object previous() {
        String itzuli= depCities.get(i);
        i=i-1;
        return itzuli;
    }

    @Override
    public boolean hasPrevious() {

        return i>=0;
    }

    @Override
    public void goFirst() {
        i=0;

    }

    @Override
    public void goLast() {
        i=depCities.size()-1;

    }
}
```

Gero, sortu egin dugu DepartingCitiesExtendedIterator-a hau da, extendedIterator bat da, Departing Cities korritzeko erabiliko dena, enuntziatuan getDepartingCities metodoak bektore bat itzultzen zuen, guk lehenagotik implementatuta genuenak lista bat itzultzen zuen(bektoreekin egiteko bakarrik Vector jarri arrayList beharrean), beraz, guk listarekin egin dugu lan bektorearekin beharren. Klase honetan espezifikatutko da departing cities-en nola inplementatuko diren heredatutako funtzioak.

```
@WebMethod
   public     ExtendedIterator<String>     getDepartingCitiesIterator();
```

BLFacade-n gehitu behar izan dugu getDepartingCitiesIterator() metodoa, honek itzuli egiten du departing cities-en oinarritutako extendedIterator bat.

```
@WebMethod
public ExtendedIterator<String> getDepartingCitiesIterator() {
      List<String> departLocations = this.getDepartCities();
      ExtendedIterator<String> iterator = new DepartingCitiesExtendedIterator(departLocations);
      return iterator;

}
```

Kode hau da nola inplementatuta dagoen BLFacadeInplementation klasean getDepartingCitiesIterator metodoa, ikusten den bezala departing cities guztiak lortu eta horiek baliatuz sortu egiten du Extended Iterator motako objektua eta hau itzuli egiten du.

```
package iterator;

import businessLogic.BLFacade;
import businessLogic.BLFactory;
import configuration.ConfigXML;

public class main {

    public static void main(String[] args) {
        // the BL is local
        ConfigXML config=ConfigXML.getInstance();

        BLFacade blFacade;
        try {
            blFacade = new BLFactory().getBussinessLogic(config);
            ExtendedIterator<String> i = blFacade.getDepartingCitiesIterator();
            String c;
            System.out.println("_____");
            System.out.println("FROM    LAST    TO  FIRST");
            i.goLast(); // Go to last element
            while (i.hasPrevious()) {
                c = i.previous();
                System.out.println(c);
            }
            System.out.println();
            System.out.println("_____");
            System.out.println("FROM    FIRST   TO  LAST");
            i.goFirst(); // Go to first element
            while (i.hasNext()) {
                c = i.next();
                System.out.println(c);
            }
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

    }

}
```

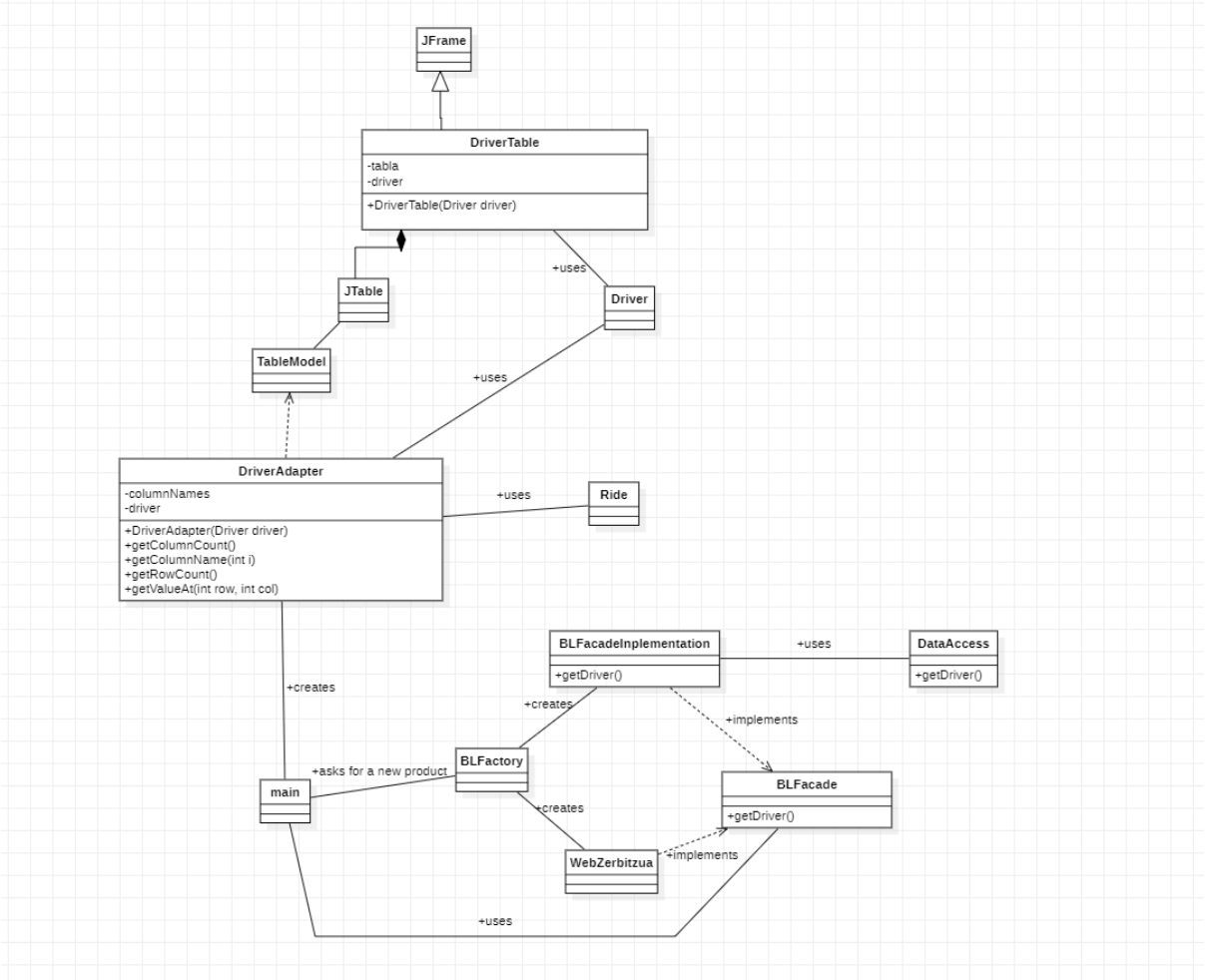
Azkenik main klase bat sortu dugu hau probatzeko. Blfacadea aukeratu eta gero lortu egiten dugu ExtendedIterator-a departing cities-ak korritzeko eta gero aurretik atzera eta atzetik aurrera pantailaratzen dira.

3. Exekuzioaren irudia

```
Read from config.xml:    businessLogicLocal=true databaseLocal=true      dataBaseInitialized=false
DataAccess opened => isDatabaseLocal: true
DataAccess created => isDatabaseLocal: true isDatabaseInitialized: false
DataAcess closed
Creating BLFacadeImplementation instance with DataAccess parameter
DataAccess opened => isDatabaseLocal: true
DataAcess closed
_____________________
FROM    LAST    TO      FIRST
Errenteria
Etxalar
Lesaka
Bera


_____________________
FROM    FIRST   TO      LAST
Bera
Lesaka
Etxalar
Errenteria
```

Adapter

1. UML diagrama



UML diagrama honetan ikusi daiteke nola aplikatu dugu adapter patroia Driver baten Rideak interfaze batean erakusteko. DriverAdapter klasea sortu dugu, AbstractTableModel klastearen extentsio bat da eta honekin lortuko dugu egin dituen bidaien informazio guztia taula batean txertatzeko. Gero, sortu dugu DriverTable klasea, honek DriverAdapter-ekin lortutako informazioa pantailaratuko du taula formatuan. Azkenik, main metodoan BLFactory bat sortzen dugu, Driver bat bilatzen dugu datu basean eta bere bidaien informazioa pantailaratzen dugu taula moduan adapter patroia erabiliz. Beharrezkoak ziren metodoak, driverra bilatzek, gehitu ditugu DataAccess-ean eta BLFacade-an.

2. Kodearen azalpena

```java
package adapter;

import javax.swing.table.AbstractTableModel;

public class DriverAdapter extends AbstractTableModel {

    private Driver driver;

    private String[] columnNames = {"From", "To", "Date","Places","Price"};


    public DriverAdapter(Driver driver) {
        this.driver = driver;
    }

    public int getColumnCount() {
        return columnNames.length;
    }

    public String getColumnName(int col) {
        return columnNames[col];
    }

    public int getRowCount() {

        return driver.getRides().size();
    }

    public Object getValueAt(int rowIndex, int columnIndex) {

        Ride ride = driver.getRides().get(rowIndex);
        switch (columnIndex) {
            case 0: return ride.getFrom();
            case 1: return ride.getTo();
            case 2: return ride.getDate();
            case 3: return ride.getnPlaces();
            case 4: return ride.getPrice();
        }
        return null;
    }
}
```

Sortu dugun DriverAdapter klasea hau da. Driver baten bidaietako informazioa zutabeetan banatu dugu. Gero, getValueAt metodoarekin informazio hori lortuko dugu taula betetzeko.

```
package adapter;

import java.awt.BorderLayout;

public class DriverTable extends JFrame {

    private Driver driver;
    private JTable  tabla;

    public DriverTable(Driver driver){
        super(driver.getUser()+"'s  rides ");
        this.setBounds(100, 100,    700,    200);
        this.driver =   driver;
        DriverAdapter   adapt   =   new DriverAdapter(driver);
        tabla = new JTable(adapt);
        tabla.setPreferredScrollableViewportSize(new Dimension(500, 70));
        JScrollPane scrollPane = new JScrollPane(tabla);
        getContentPane().add(scrollPane,    BorderLayout.CENTER);
    }


}
```

DriverTable klaseak taula sortuko du DriverAdapter-ekin lortzen dugu informazioarekin.

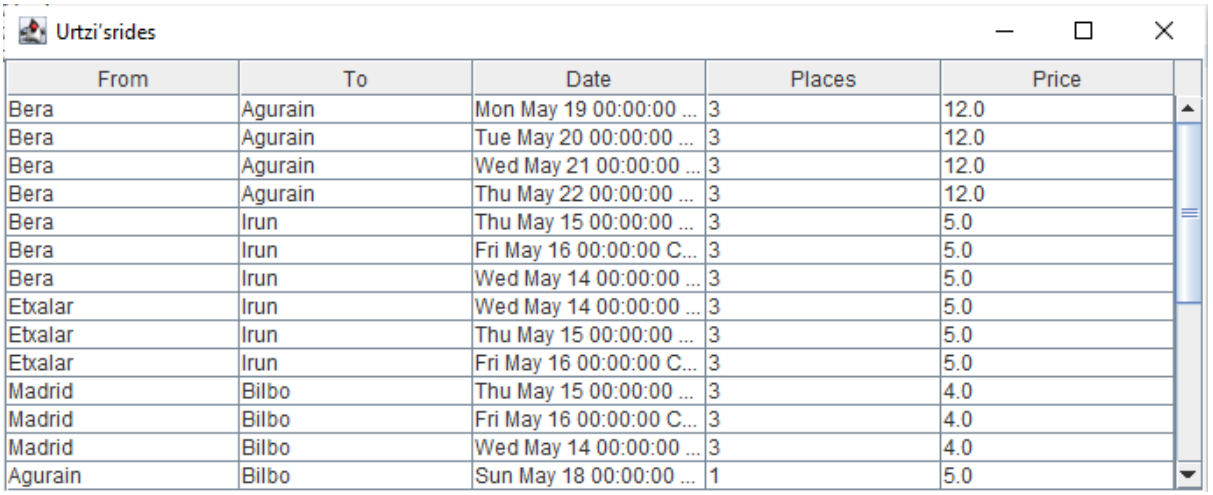
```
package adapter;

import businessLogic.BLFacade;

public class main {

    public static void main(String[]args)   {
        ConfigXML config=ConfigXML.getInstance();
        BLFacade blFacade;
        try {
            blFacade = new BLFactory().getBusinessLogicFactory(config);
            Driver  d= blFacade.getDriver("Urtzi");
            DriverTable dt=new  DriverTable(d);
            dt.setVisible(true);
        } catch (Exception e) {
            e.printStackTrace();
        }

    }

}
```

Main-en BLFactorylocal bat sortzen dugu. Nahi dugun Driver-a bilatzen dugu eta bere bidaien informazioa taula batean pantailaratzen dugu.

Hurrengo metodoak gehitu ditugu Driver-a bilatzeko:

```
@WebMethod
public Driver getDriver(String izena) {
    dbManager.open();
    Driver d = dbManager.getDriver(izena);
    dbManager.close();
    return d;
}


public Driver getDriver(String izena) {
    return db.find(Driver.class, izena);
}
```

3. Exekuzioaren irudia



From	To	Date	Places	Price
Bera	Agurain	Mon May 19 00:00:00 ...	3	12.0
Bera	Agurain	Tue May 20 00:00:00 ...	3	12.0
Bera	Agurain	Wed May 21 00:00:00 ...	3	12.0
Bera	Agurain	Thu May 22 00:00:00 ...	3	12.0
Bera	Irun	Thu May 15 00:00:00 ...	3	5.0
Bera	Irun	Fri May 16 00:00:00 C...	3	5.0
Bera	Irun	Wed May 14 00:00:00 ...	3	5.0
Etxalar	Irun	Wed May 14 00:00:00 ...	3	5.0
Etxalar	Irun	Thu May 15 00:00:00 ...	3	5.0
Etxalar	Irun	Fri May 16 00:00:00 C...	3	5.0
Madrid	Bilbo	Thu May 15 00:00:00 ...	3	4.0
Madrid	Bilbo	Fri May 16 00:00:00 C...	3	4.0
Madrid	Bilbo	Wed May 14 00:00:00 ...	3	4.0
Agurain	Bilbo	Sun May 18 00:00:00 ...	1	5.0