

Proyecto Patrones

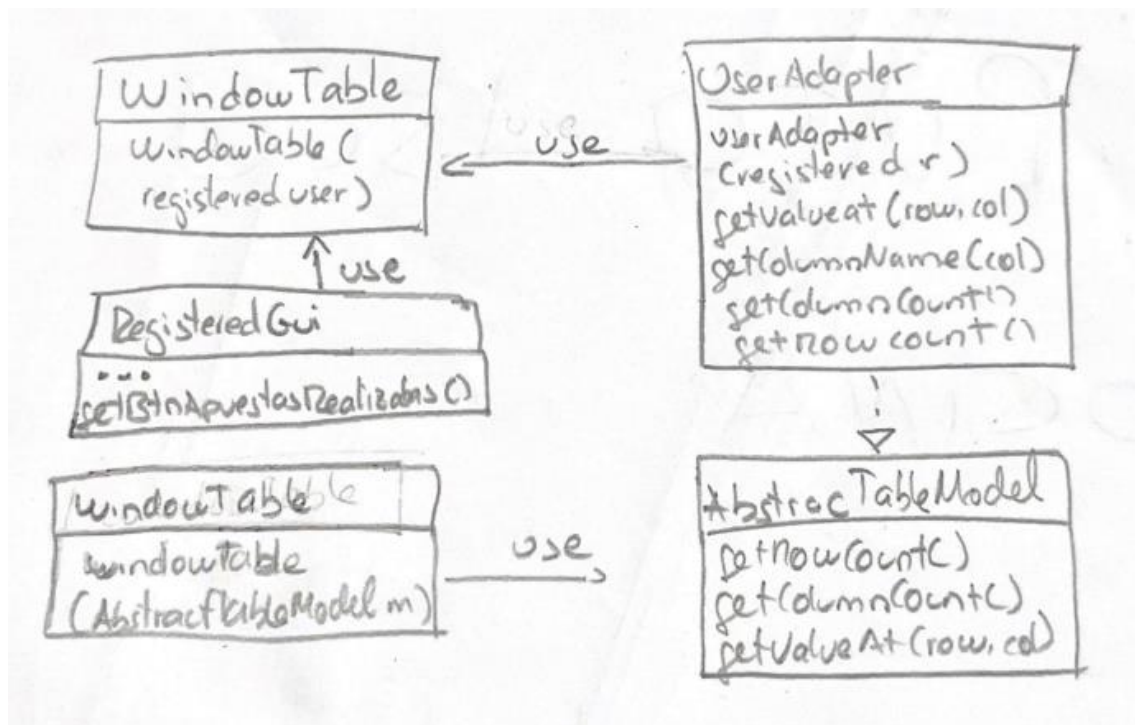
Contenido

Proyecto Patrones	1
EJERCICIO 3 PATRON ADAPTER.....	1
UML con las nuevas clases	1
Código modificado	2
Capturas mostrando la ejecución.....	4
EJERCICIO 2 PATRON ITERATOR	6
UML con las nuevas clases	6
Código modificado	7
Capturas mostrando la ejecución.....	10

EJERCICIO 3 PATRON ADAPTER

Realizado por Iñaki Inda Araguás

UML con las nuevas clases



Código modificado

En RegisteredGUI.java he creado un botón para poder acceder a la ventana que muestra la información con todas las apuestas realizadas por un usuario

```
private JButton getBtnApuestasRealizadas() {
    if (btnApuestasRealizadas == null) {
        btnApuestasRealizadas = new
JButton(ResourceBundle.getBundle("Etiquetas").getString("APREALIZADAS"));
//$NON-NLS-1$ //$NON-NLS-2$
        btnApuestasRealizadas.setFont(new Font("Tahoma",
Font.PLAIN, 16));
        btnApuestasRealizadas.setForeground(Color.DARK_GRAY);
        btnApuestasRealizadas.setBackground(Color.PINK);
        btnApuestasRealizadas.addActionListener(new
ActionListener() {
            public void actionPerformed(ActionEvent e) {
                WindowTable vt = new WindowTable(user);
                vt.setVisible(true);
            }
        });
        btnApuestasRealizadas.setBounds(10, 391, 282, 68);
    }
    return btnApuestasRealizadas;
}
```

Este botón crea una nueva instancia de la clase WindowTable

```
package gui;

public class WindowTable extends JFrame {

    private Registered user;

    private JTable tabla;

    public WindowTable(Registered user) {
super("Apuestas realizadas por " + user.getUsername() + ":");
        this.setBounds(100, 100, 700, 200);
        this.user = user;
        UserAdapter adapt = new UserAdapter(user);
        tabla = new JTable(adapt);
        tabla.setPreferredSize(new Dimension(500,
70));

        JScrollPane scrollPane = new JScrollPane(tabla);
        getContentPane().add(scrollPane, BorderLayout.CENTER);
    }
}
```

La cual crea una ventana e instancia la clase UserAdapter.

En esta clase en el método getValueAt accedo a las apuestas de un usuario y para cada apuesta que ha realizado recupero los datos

```
package adapter;

import java.util.ArrayList;

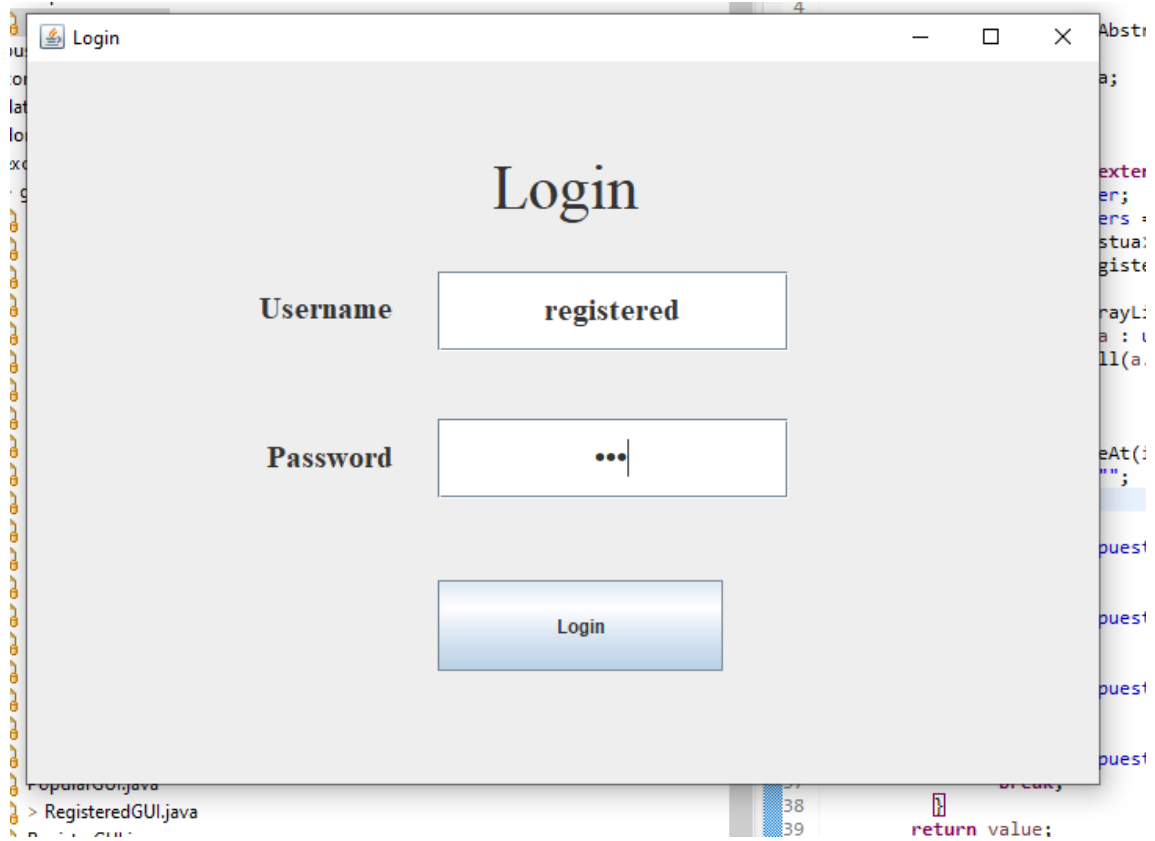
import javax.swing.table.AbstractTableModel;

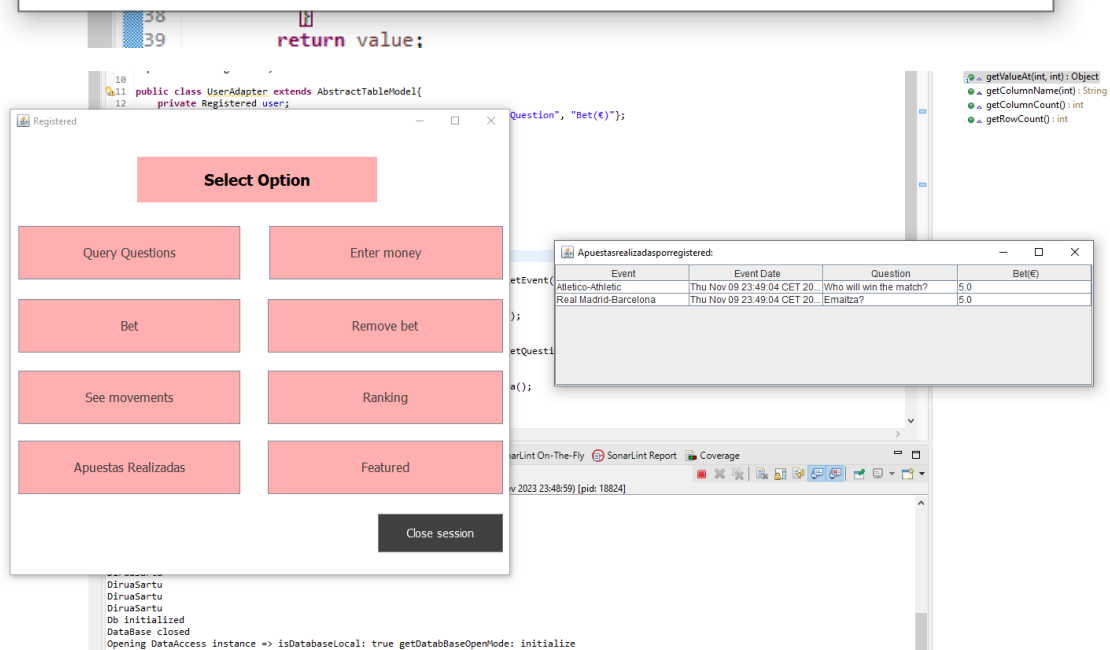
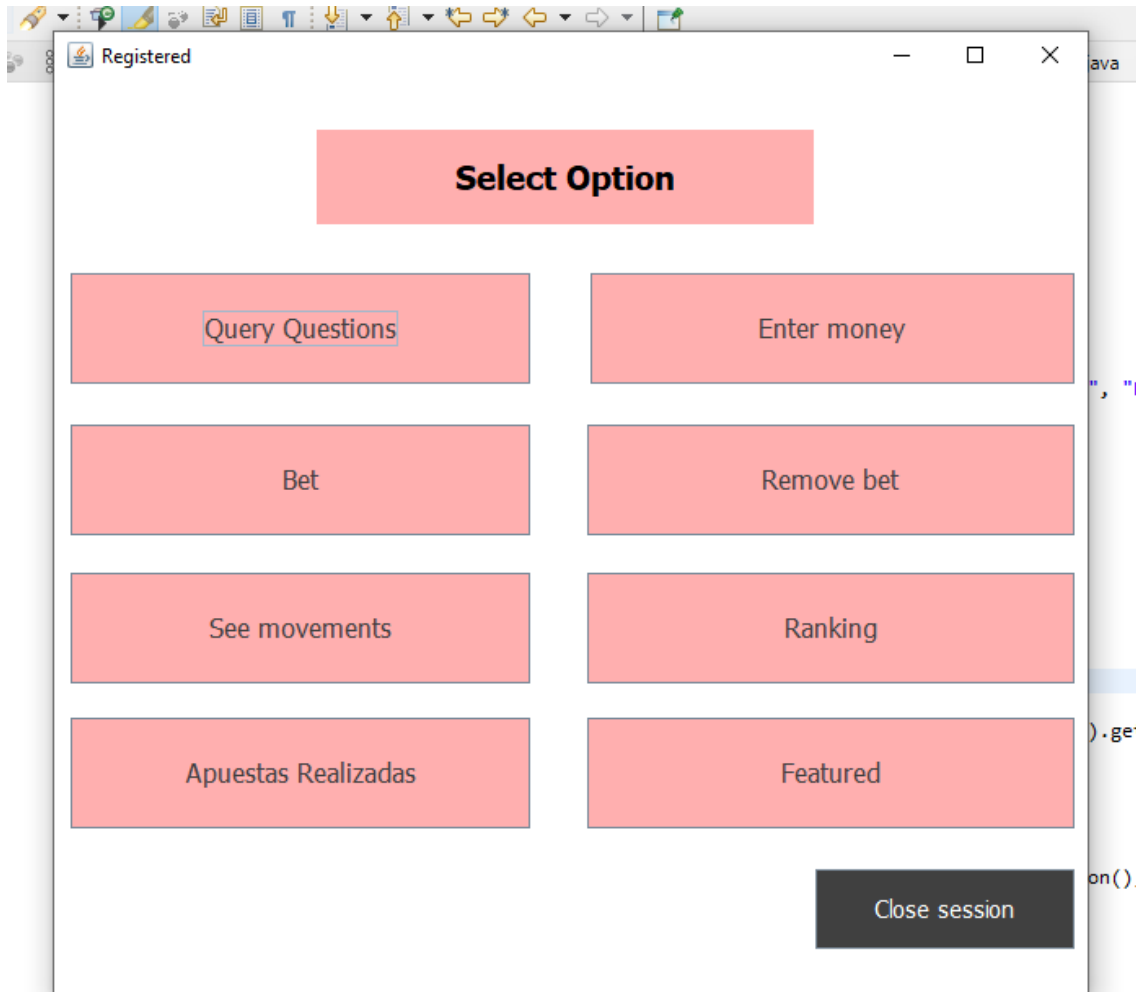
import domain.ApustuAnitza;
import domain.Apustua;
import domain.Registered;

public class UserAdapter extends AbstractTableModel{
    private Registered user;
    private String[] headers = new String[]{"Event", "Event Date",
"Question", "Bet(€)"};
    private ArrayList<Apustua> apuestas;
    public UserAdapter(Registered r) {
        this.user=r;
        apuestas = new ArrayList<Apustua>();
        for(ApustuAnitza a : user.getApustuAnitzak()) {
            apuestas.addAll(a.getApustuak());
        }
    }
    @Override
    public Object getValueAt(int row, int col) {
        Object value = "";
        switch (col) {
            case 0:
                value =
apuestas.get(row).getKuota().getQuestion().getEvent().getDescription();
                break;
            case 1:
                value = apuestas.get(row).getApustuAnitza().getData();
                break;
            case 2:
                value =
apuestas.get(row).getKuota().getQuestion().getQuestion();
                break;
            case 3:
                value = apuestas.get(row).getApustuAnitza().getBalioa();
                break;
        }
        return value;
    }
    @Override
    public String getColumnName(int col) {
        return headers[col];
    }
    @Override
    public int getColumnCount() {
        return 4;
    }
    @Override
    public int getRowCount() {
        return apuestas.size();
    }
}
```

```
}  
}
```

Capturas mostrando la ejecución

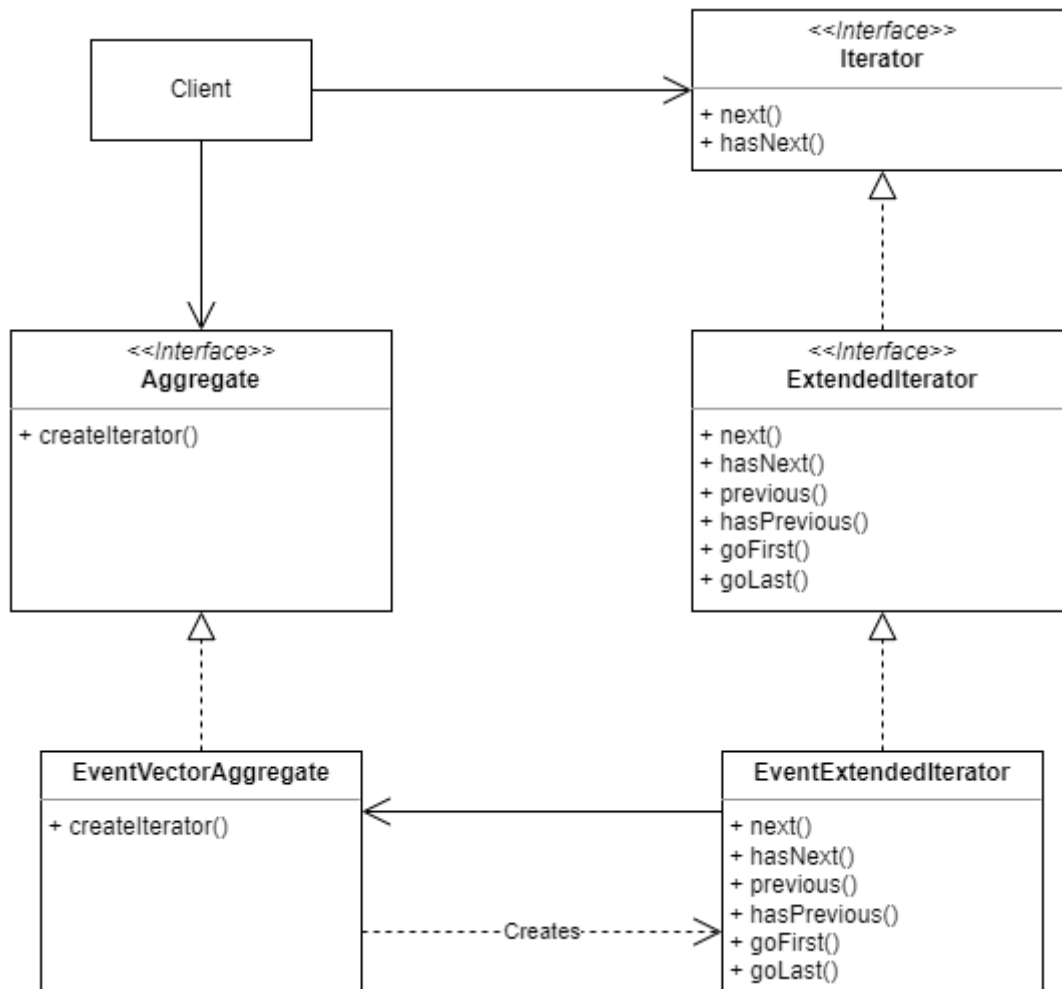




EJERCICIO 2 PATRON ITERATOR

Realizado por Josu Sáez Moreno

UML con las nuevas clases



Código modificado

Nuevo método en la lógica de negocio:

```
public ExtendedIterator<Event> getEventsIterator(Date date) {  
    dbManager.open(false);  
    Vector<Event> events=dbManager.getEvents(date);  
    dbManager.close();  
    EventVectorAggregate p=new EventVectorAggregate(events);  
    return p.createIterator();  
}
```

Interfaz Aggregate y clase EventVectorAggregate:

```
public interface Aggregate<T> {  
    public ExtendedIterator<T> createIterator();  
}  
  
public class EventVectorAggregate implements Aggregate<Event>{  
  
    protected Vector<Event> events;  
  
    public EventVectorAggregate (Vector<Event> events) {  
        this.events=events;  
    }  
  
    @Override  
    public ExtendedIterator<Event> createIterator() {  
        return new EventExtendedIterator(events);  
    }  
}
```

Interfaz ExtendedIterator:

```
public interface ExtendedIterator<T> extends Iterator<T> {  
  
    //return the actual element and go to the previous  
    public T previous();  
  
    //true if there is a previous element  
    public boolean hasPrevious();  
  
    //It is placed in the first element  
    public void goFirst();  
  
    // It is placed in the last element  
    public void goLast();  
}
```

Clase EventExtendedIterator:

```
public class EventExtendedIterator implements ExtendedIterator<Event>{

    protected Vector<Event> events;
    protected int position;
    protected int lastIndex;

    public EventExtendedIterator(Vector<Event> events) {
        this.events=events;
        this.lastIndex=events.size()-1;
        goFirst();
    }

    @Override
    public boolean hasNext() {
        return position<lastIndex;
    }

    @Override
    public Event next() {
        this.position+=1;
        return events.elementAt(position);
    }

    @Override
    public Event previous() {
        this.position-=1;
        return events.elementAt(position);
    }

    @Override
    public boolean hasPrevious() {
        return position>0;
    }

    @Override
    public void goFirst() {
        this.position=-1;
    }

    @Override
    public void goLast() {
        this.position=lastIndex+1;
    }

}
```


Clase IteratorMain:

```
public class IteratorMain {

    public static void main(String[] args) {
        BLFacade blFacade = new BLFacadeImplementation();
        SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");
        Date date;
        try {
            date = sdf.parse("17/12/2023"); // 17 del mes que viene
            ExtendedIterator<Event> i = blFacade.getEventsIterator(date);
            Event e;
            System.out.println("_____");
            System.out.println("RECORRIDO HACIA ATRAS");
            i.goLast(); // Hacia atrás
            while (i.hasPrevious()) {
                e = i.previous();
                System.out.println(e.toString());
            }
            System.out.println();
            System.out.println("_____");
            System.out.println("RECORRIDO HACIA ADELANTE");
            i.goFirst(); // Hacia adelante
            while (i.hasNext()) {
                e = i.next();
                System.out.println(e.toString());
            }
        } catch (Exception e) {
            System.out.println("Problems with date?? " + "17/12/2020");
        }
    }

}
```

Capturas mostrando la ejecución

```
Console X
<terminated> IteratorMain [Java Application] C:\Users\josus\Downloads\eclipse-jee-2021-09-R-win32-x86_64\eclipse\plugins\org.eclipse.justjop
Creating BLFacadeImplementation instance
Read from config.xml:      businessLogicLocal=true      databaseLocal=true      dataBaseOpenMode=initialize
Creating DataAccess instance => isDatabaseLocal: true getDatabBaseOpenMode: initialize
Opening DataAccess instance => isDatabaseLocal: true getDatabBaseOpenMode: initialize
Deleting the DataBase
newDate: Sun Dec 17 00:00:00 CET 2023
newDate: Sun Dec 17 00:00:00 CET 2023
newDate: Sun Dec 17 00:00:00 CET 2023
newDate: Sun Dec 17 00:00:00 CET 2023
newDate: Sun Dec 17 00:00:00 CET 2023
newDate: Sun Dec 17 00:00:00 CET 2023
newDate: Sun Dec 17 00:00:00 CET 2023
newDate: Sun Dec 17 00:00:00 CET 2023
newDate: Sun Dec 17 00:00:00 CET 2023
newDate: Sun Dec 17 00:00:00 CET 2023
newDate: Fri Dec 01 00:00:00 CET 2023
newDate: Fri Dec 01 00:00:00 CET 2023
newDate: Fri Dec 01 00:00:00 CET 2023
newDate: Fri Dec 01 00:00:00 CET 2023
newDate: Fri Dec 01 00:00:00 CET 2023
newDate: Sun Jan 28 00:00:00 CET 2024
newDate: Sun Jan 28 00:00:00 CET 2024
newDate: Sun Jan 28 00:00:00 CET 2024
newDate: Sun Jan 28 00:00:00 CET 2024
newDate: Sat Oct 21 00:00:00 CEST 2023
newDate: Sun Dec 17 00:00:00 CET 2023
newDate: Sun Dec 17 00:00:00 CET 2023
newDate: Sun Dec 17 00:00:00 CET 2023
newDate: Fri Dec 01 00:00:00 CET 2023
newDate: Fri Dec 01 00:00:00 CET 2023
newDate: Sun Dec 17 00:00:00 CET 2023
DiruaSartu
DiruaSartu
DiruaSartu
DiruaSartu
Db initialized
DataBase closed
Opening DataAccess instance => isDatabaseLocal: true getDatabBaseOpenMode: initialize
>> DataAccess: getEvents
1;Atletico-Athletic
2;Eibar-Barcelona
3;Getafe-Celta
4;Alaves-Deportivo
5;Espanol-Villareal
6;Las Palmas-Sevilla
7;Malaga-Valencia
8;Girona-Leganés
9;Real Sociedad-Levante
10;Betis-Real Madrid
22;LA Lakers-Phoenix Suns
23;Atlanta Hawks-Houston Rockets
24;Miami Heat-Chicago Bulls
27;Djokovic-Federer
DataBase closed
RECORBITIO      HACTA      ATBAS
```

RECORRIDO	HACIA	ATRAS
27;	Djokovic-Federer	
24;	Miami Heat-Chicago Bulls	
23;	Atlanta Hawks-Houston Rockets	
22;	LA Lakers-Phoenix Suns	
10;	Betis-Real Madrid	
9;	Real Sociedad-Levante	
8;	Girona-Leganes	
7;	Malaga-Valencia	
6;	Las Palmas-Sevilla	
5;	Espanol-Villareal	
4;	Alaves-Deportivo	
3;	Getafe-Celta	
2;	Eibar-Barcelona	
1;	Atletico-Athletic	

RECORRIDO	HACIA	ADELANTE
1;	Atletico-Athletic	
2;	Eibar-Barcelona	
3;	Getafe-Celta	
4;	Alaves-Deportivo	
5;	Espanol-Villareal	
6;	Las Palmas-Sevilla	
7;	Malaga-Valencia	
8;	Girona-Leganes	
9;	Real Sociedad-Levante	
10;	Betis-Real Madrid	
22;	LA Lakers-Phoenix Suns	
23;	Atlanta Hawks-Houston Rockets	
24;	Miami Heat-Chicago Bulls	
27;	Djokovic-Federer	