

**ARGENTINA PROGRAMA 4.0**  
**UNIVERSIDAD TECNOLÓGICA NACIONAL**

**Trabajo práctico integrador**

**Pronósticos Deportivos**

**Comisión 150 – Grupo 5**

**Integrantes:**

Alvarez, Ariel

Buen, Federico

**Docente:** Fernando Torres

**Tutor TP:** Federico Octavio Prandi

**Año:** 2023

## Tecnologías utilizadas

Para el desarrollo del tp se utilizaron las siguientes herramientas:

Lenguaje de Programación: **Java**

IDE: **Eclipse**

Motor de Base de Datos: **SQLite**

Se realizo utilizando **Maven**, **lombok** y **GitHub** como repositorio

Se aplicó el paradigma de programación orientado a objetos.

## Objetivo del proyecto

Se encomendó realizar un proyecto, para la gestión de un sistema de apuestas deportivas  
Dicho sistema debe poseer las siguientes características.

Procesar la información proveniente de una fuente de datos, con dicha información se debe poder realizar los siguientes enunciados

Acceder al origen de datos, tablas: Equipos, partidos y resultados

Listar participantes

Listado de rondas.

Permitir realizar varias rondas por fase, y suministrar a través de la fuente de datos los parámetros de puntos y opciones especiales para aplicar a las apuestas.

El análisis de dicha información debe arrojar un listado de los puntos obtenidos por los participantes como así también los puntos extras adquiridos según las condiciones especiales informadas al respecto. (Ej puntos extras por rondas perfectas, fase perfecta, etc)

El modelo propuesto, permite gestionar el origen de datos, a través de dos fuentes

Archivos csv, o conexión a un motor de BD SQL

Se detalla a continuación diseño de la BD Sql con sus tablas

Nombre	Tipo	Esquema
▼ Tablas (3)		
▼ partidos		CREATE TABLE "partidos" ( "nroRonda" INTEGER, "nroPartido" INTEGER, "equipoLocal" TEXT, "equipoVisitante" TEXT, "golesLocal" INTEGER, "golesVisitante" INTEGER )
nroRonda	INTEGER	"nroRonda" INTEGER
nroPartido	INTEGER	"nroPartido" INTEGER
equipoLocal	TEXT	"equipoLocal" TEXT
equipoVisitante	TEXT	"equipoVisitante" TEXT
golesLocal	INTEGER	"golesLocal" INTEGER
golesVisitante	INTEGER	"golesVisitante" INTEGER
▼ personas		CREATE TABLE "personas" ( "id" TEXT, "nombre" TEXT, "telefono" TEXT, "email" TEXT )
id	TEXT	"id" TEXT
nombre	TEXT	"nombre" TEXT
telefono	TEXT	"telefono" TEXT
email	TEXT	"email" TEXT
▼ pronosticos		CREATE TABLE "pronosticos" ( "nroRonda" INTEGER, "nroPartido" INTEGER, "nombre" TEXT, "resultado" TEXT )
nroRonda	INTEGER	"nroRonda" INTEGER
nroPartido	INTEGER	"nroPartido" INTEGER
nombre	TEXT	"nombre" TEXT
resultado	TEXT	"resultado" TEXT

Ejemplo contenido tabla ***pronostico***

nroRonda	nroPartido	nombre	resultado
1	1	28478966	EMPATE
1	2	28478966	LOCAL
1	3	28478966	EMPATE
2	1	28478966	VISITANTE
2	2	28478966	VISITANTE
2	3	28478966	LOCAL
2	4	28478966	VISITANTE
1	1	29785451	EMPATE
1	2	29785451	LOCAL
1	3	29785451	VISITANTE
2	1	29785451	VISITANTE
2	2	29785451	VISITANTE
2	3	29785451	LOCAL
2	4	29785451	VISITANTE
1	1	32566698	EMPATE
1	2	32566698	VISITANTE
1	3	32566698	EMPATE

Contenido tabla ***personas***

id	nombre	telefono	email
22078657	Carola	54654564	carola@gmail.com
28478966	laura	54564546	laura@yahoo.com
29785451	Juan	66988989	juan@juan.com
24230593	Ariel	3212123	ariel@gmail.com
32566698	Fede	13212132	fede@hotmail.com

Contenido tabla ***partidos***

nroRonda	nroPartido	equipoLocal	equipoVisitante	golesLocal	golesVisitante
1	1	River	Boca	2	2
1	2	Banfield	San Lorenzo	3	1
1	3	Independiente	Lanus	2	4
2	1	Talleres	Quilmes	1	2
2	2	Velez	Gimnasia	0	2
2	3	Union	Colon	2	0
2	4	Racing	Rosario Central	1	2
2	5	Boca	River	1	0

## Contenido de la Tabla *configuración.csv*

(define el puntaje a asignar según sistema de aciertos)

PUNTOS\_GANA,1

PUNTOS\_EXTRA,3

PUNTOS\_RONDA\_PERFECTA,5

PUNTOS\_FASE\_PERFECTA,10

Ejemplo consulta que muestra las apuestas realizadas por cada persona

select tb1.\*,tb2.nroRonda,tb2.nroPartido,tb2.resultado from personas as tb1

inner join pronosticos as tb2 on tb1.id=tb2.nombre

order by nombre












id	nombre	telefono	email	nroRonda	nroPartido	resultado
32566698	Fede	13212132	fede@hotmail.com	1	1	EMPATE
32566698	Fede	13212132	fede@hotmail.com	1	2	VISITANTE
32566698	Fede	13212132	fede@hotmail.com	1	3	EMPATE
29785451	Juan	66988989	juan@juan.com	1	1	EMPATE
29785451	Juan	66988989	juan@juan.com	1	2	LOCAL
29785451	Juan	66988989	juan@juan.com	1	3	VISITANTE
29785451	Juan	66988989	juan@juan.com	2	1	VISITANTE
29785451	Juan	66988989	juan@juan.com	2	2	VISITANTE
29785451	Juan	66988989	juan@juan.com	2	3	LOCAL
29785451	Juan	66988989	juan@juan.com	2	4	VISITANTE
28478966	laura	54564546	laura@yahoo.com	1	1	EMPATE
28478966	laura	54564546	laura@yahoo.com	1	2	LOCAL
28478966	laura	54564546	laura@yahoo.com	1	3	EMPATE
28478966	laura	54564546	laura@yahoo.com	2	1	VISITANTE
28478966	laura	54564546	laura@yahoo.com	2	2	VISITANTE
28478966	laura	54564546	laura@yahoo.com	2	3	LOCAL
28478966	laura	54564546	laura@yahoo.com	2	4	VISITANTE

## Estructura del proyecto Java












La estructura esta realizada sobre la base de un proyecto **Maven**



## Paquetes y clases

- ▼  modelo
  - ▶  Equipo.java
  - ▶  Juego.java
  - ▶  Main.java
  - ▶  Participantes.java
  - ▶  Partido.java
  - ▶  Persona.java
  - ▶  Pronostico.java
  - ▶  Puntos.java
  - ▶  Resultado.java
  - ▶  Ronda.java

El package **modelo** contiene las clases :

- ▼  modelo
  - ▶  Equipo.java
  - ▶  Juego.java
  - ▶  Main.java
  - ▶  Participantes.java
  - ▶  Partido.java
  - ▶  Persona.java
  - ▶  Pronostico.java
  - ▶  Puntos.java
  - ▶  Resultado.java
  - ▶  Ronda.java

**Main** : Esta clase es la principal del proyecto, aquí es donde inicia la ejecución del mismo

```
package modelo;

import menus.Menu;

/*****
 * Clase Main del Trabajo Integrador Final
 * del curso Argentina Programa 4.0 2023 - Java Inicial
 * -----
 * Comision 150 - Grupo 5
 *
 * Desarrollado en lenguaje Java utilizando el IDE Eclipse
 * Base de datos SQLite
 *
 * *****/

public class Main {

    public static void main(String[] args) {

        Menu menu = new MenuConfiguracion();
        menu.iniciar();

    }

}
```

Tiene el método **main** donde se instancia un objeto de la clase **MenuConfiguracion** y llama al método **iniciar**.

**Puntos** : Esta clase contiene los atributos que contendrán los valores de los puntajes

se utilizó la librería **lombok** y la notación **@Data** que nos permite acceder a los getter y setter

```
1 package modelo;
2
3 import lombok.Data;
4
5 @Data
6 public class Puntos {
7
8     private int puntosGana;
9     private int puntosExtra;
10    private int puntosRondaPerfecta;
11    private int puntosFasePerfecta;
12
13 }
14
```

**Resultado** : enum que representa a los distintos resultados que puede tener un partido

```
1 package modelo;
2
3 public enum Resultado {
4     LOCAL, EMPATE, VISITANTE
5 }
6
```

**Juego** : la clase juego es el esqueleto central del proyecto, donde se encuentra la lógica de la resolución de la consigna

```
1 package modelo;
2
3 import java.util.ArrayList;
4
18 public class Juego {
19
20     private ArrayList<Pronostico> pronosticos = new ArrayList<Pronostico>();
21     private HashMap<String, Integer> puntuacion = new HashMap<String, Integer>();
22     private Participantes participantes = new Participantes();
23     private Ronda rondas = new Ronda();
24     private Puntos puntosGanados = new Puntos();
25
26     /**
```

Contiene las colecciones donde se almacenarán los datos que son leídos desde los archivos .csv o de las tablas de la base de datos. Con ellos se realizarán los cálculos de los puntajes requeridos.

Los métodos principales de esta clase son :

**public void leerDatosDesdeArchivos()** : Lee los datos de los archivos .csv y los almacena en las distintas colecciones.

**public void** leerDatosDesdeBD() : Lee los datos desde las tablas de la base de datos y los almacena en las distintas colecciones.

**public void** resolverJuego() : se encarga de recorrer las colecciones, hacer las comparaciones necesarias y almacenar los resultados.

**public void** ImprimirResultados() : muestra los resultados de los puntajes por pantalla

**public void** listarRondas() : lista las rondas de los partidos por pantalla

**public void** mostrarParticipantes() : lista los participantes por pantalla

Las clases **Equipo**, **Persona**, **Participantes**, **Pronostico**, **Partido** y **Ronda** serán las que contendrán los atributos y métodos necesarios para parsear las estructuras de los datos leídos de los archivos o las tablas de las base de datos.

**Package conexionDB** : este package contiene la clase **ConexionDB**

esta clase es la que se encarga de realizar la conexión a la base de datos SQLite “prode” y de devolver un objeto Connection

```
1 package conexionDB;
2
3 import java.sql.Connection;
4
5 public class ConexionDB {
6
7     public static final String DB = "prode";
8     public static final String DB_URL = "jdbc:sqlite:src/main/resources/" + DB;
9
10    public static Connection getConexion() {
11        Connection miConexion = null;
12        try {
13            // crear conexion
14            miConexion = DriverManager.getConnection(DB_URL);
15            System.out.println();
16            System.out.println("Conexion con la base de datos *" + DB + "*" realizada con exito");
17
18        } catch (Exception e) {
19            System.out.println();
20            System.out.println("No se puede conectar a la base de datos: " + DB);
21        }
22        return miConexion;
23    }
24 }
```

**Package exceptions** : en este package estan todas la excepciones requeridas por la aplicación

- ▾ exceptions
  - CantidadDatosException.java
  - GolesNegativosException.java
  - RondaNegativaException.java



**Package comparators** : contiene la clase **PuntajeComparator** utilizada para ordenamiento de un Map a través de su value. En este caso esta ordenando de mayor a menor. *Si bien en la ultima entrega del proyecto no es utilizado si fue útil en las entregas preliminares. Por motivos académicos y para mostrar la evolución del proyecto no fue quitada del mismo.*

```
package comparators;

/**
 * clase definida que implementa la interfaz del Comparator y le pasamos su
 * objeto al TreeMap para que se ordene Map por valor
 */
import java.util.Comparator;
import java.util.Map;









public class PuntajeComparator implements Comparator<Object> {

    Map<String, Integer> map;

    public PuntajeComparator(Map<String, Integer> map) {
        this.map = map;
    }

    public int compare(Object o1, Object o2) {
        if (map.get(o2) == map.get(o1))
            return 1;
        else
            return ((Integer) map.get(o1)).compareTo((Integer) map.get(o2)) * -1;
    }
}
```

**Package gestores** : contiene las clases que se encargan de leer los archivos o las tablas de la base de datos, parsear los datos a los distintos objetos y agregarlos a las colecciones de la clase Juego

- ▼  gestores
  - ▶  GestorConfiguracionPuntos.java
  - ▶  GestorParticipantes.java
  - ▶  GestorParticipantesSQL.java
  - ▶  GestorPartidos.java
  - ▶  GestorPartidosSQL.java
  - ▶  GestorPronosticos.java
  - ▶  GestorPronosticosSQL.java

La clase **GestorConfiguracionPuntos** se encarga de leer el archivo `configuracion.csv` y de cargar los valores en un objeto de la clase **Puntos**

```

1 package gestores;
2
3 import java.io.BufferedReader;
4
5 /**
6  * Esta clase se va a encargar de leer desde un archivo csv la configuracion de puntajes
7  * y almacenarlos en un objeto de la clase Puntos
8  */
9
10 public class GestorConfiguracionPuntos {
11
12     private Puntos puntos = new Puntos();
13
14     public Puntos cargarPuntosDesdeArchivo(String path) {
15         try {
16             System.out.println("-----");
17             System.out.println("Cargando datos del archivo: " + path);
18             System.out.println("-----");
19             FileReader filePuntos = new FileReader(path);
20             BufferedReader brPuntos = new BufferedReader(filePuntos);
21             /*
22              * Formato de archivo de configuracion de puntos:
23              * PUNTOS_GANA,(int) = puntos acierto normal
24              * PUNTOS_EXTRA,(int) = puntos extra
25              * PUNTOS_RONDA_PERFECTA,(int) = puntos extra por ronda perfecta
26              * PUNTOS_FASE_PERFECTA,(int) = puntos extra por fase perfecta
27              */
28         }
29     }
30 }

```

La clase **GestorPartidos** se encarga de leer el archivo `partidos.csv`, de parsear los datos linea a linea y de cargarlos en la colección correspondiente a los partidos un objeto del tipo **Rondas**

```

1 /**
2  * Esta clase se va a encargar de leer desde un archivo csv los partidos y
3  * almacenarlos en una Ronda
4  */
5
6 public class GestorPartidos {
7
8     private Ronda ronda = new Ronda();
9     private Partido partido;
10
11     /**
12      * Metodo que lee el archivo de partidos y los carga en una Ronda
13      *
14      * @return ronda
15      */
16     public Ronda cargarPartidosDesdeArchivo(String path) {
17         try {
18             System.out.println("-----");
19             System.out.println("Cargando datos del archivo: " + path);
20             System.out.println("-----");
21             FileReader filePartidos = new FileReader(path);
22             BufferedReader brPartidos = new BufferedReader(filePartidos);
23             String unPartido = brPartidos.readLine();
24         }
25     }
26 }

```

Las clases **GestorParticipantes** y **GestorPronosticos** son similares a esta clase leyendo los respectivos archivos `.csv`

La clase **GestorPartidosSQL** se encarga de leer la tabla partidos de la base de datos, parsear los datos y de cargarlos en la colección correspondiente a los partidos un objeto del tipo **Rondas**

```

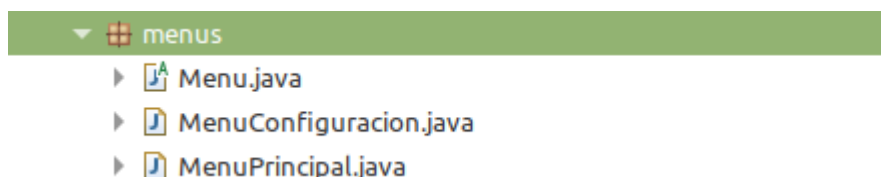
9 public class GestorPartidosSQL {
10
11     private Ronda ronda = new Ronda();
12     private Partido partido;
13
14     /**
15      * Metodo que lee la tabla de partidos y los carga en una Ronda
16      *
17      * @return ronda
18      */
19     public Ronda cargarPartidosDesdeTable() {
20         Connection conexion = null;
21         Statement consulta = null;
22
23         try {
24             // Creando una conexion a la DB
25             conexion = ConexionDB.getConexion();
26             consulta = conexion.createStatement();
27             String consultaSql = "SELECT * FROM partidos ORDER BY nroRonda, nroPartido";
28             ResultSet rsPartidos = consulta.executeQuery(consultaSql);
29             System.out.println("Actualizando datos de la tabla *partidos*");
30             int nroRonda;
31             while (rsPartidos.next()) {
32                 nroRonda = rsPartidos.getInt("nroRonda");

```

Las clases **GestorParticipantesSQL** y **GestorPronosticosSQL** son similares a esta clase leyendo los respectivas tablas de la base de datos.

### Aplicando Herencia

**Package menu** : contiene las clases que mostraran por pantalla los distintos menús de la aplicación. Estos menús le permiten al usuario interactuar con la aplicación.



Clase **Menu** : es una clase **abstracta**. Contiene los metodos:

**public void iniciar()** : muestra la pantalla del menu y contiene un ciclo para permitir al usuario ingresos por pantalla

**public void setSalir()** : setea la variable para terminar el ciclo y salir del método iniciar

```
1 package menus;
2
3 import java.util.Scanner;
4
5 public abstract class Menu {
6
7     private String opcion;
8     private boolean salir;
9
10    Scanner scn = new Scanner(System.in);
11    public void iniciar() {
12
13        mostrarPantalla();
14
15        salir = false;
16        while (!salir) {
17            opcion = scn.nextLine();
18            switch (opcion) {
19                case "1":
20                    opcion1();
21                    break;
22                case "2":
23                    opcion2();
24                    break;
25                case "3":
26                    opcion3();
27                    break;
```

Contiene los métodos abstractos que deberán ser implementados en los menús hijos de esta clase

```
public abstract void mostrarPantalla();
public abstract void mensajeErrorOpcionIngresada();
public abstract void opcion1();
public abstract void opcion2();
public abstract void opcion3();
public abstract void opcion4();
public abstract void opcion5();
public abstract void opcion0();
```

**MenuConfiguracion** : esta clase hereda de la clase **Menu**, tiene la implementación de los métodos de las opciones para que el usuario pueda elegir el origen de los datos que se van a cargar ya sea desde archivos CSV o de la Base de Datos

```
1 package menus;
2
3 import modelo.Juego;
4
5 public class MenuConfiguracion extends Menu{
6
7     /**
8      * instancia del juego
9      */
10    Juego juego = new Juego();
11
12    @Override
13    public void mostrarPantalla() {
14        System.out.println("=====");
15        System.out.println("|          MENU DE CONFIGURACION          |");
16        System.out.println("|-----|");
17        System.out.println("| 1 - Cargar datos desde archivos CSV  |");
18        System.out.println("| 2 - Cargar datos desde BASE DE DATOS |");
19        System.out.println("| 0 - Salir de la aplicacion           |");
20        System.out.println("=====");
21        System.out.println("Presione una de las opciones: ");
22    }
23
24    @Override
25    public void opcion1() {
26        juego.leerDatosDesdeArchivos();
27        Menu menuPrincipal = new MenuPrincipal(juego);
28        menuPrincipal.iniciar();
29        setSalir();
30    }
31
32    @Override
33    public void opcion2() {
34        juego.leerDatosDesdeBD();
35        Menu menuPrincipal = new MenuPrincipal(juego);
36        menuPrincipal.iniciar();
37        setSalir();
38    }
39 }
```



La clase **MenuPrincipal** hereda de la clase **Menu**, se encarga de la iteración entre el usuario y las distintas opciones que tiene la aplicación.

```
1 package menus;
2
3 import modelo.Juego;
4
5 public class MenuPrincipal extends Menu {
6
7     /**
8      * instancia del juego
9      */
10    Juego juego = new Juego();
11
12    public MenuPrincipal(Juego juego) {
13        this.juego = juego;
14    }
15
16    @Override
17    public void mostrarPantalla() {
18        System.out.println("=====");
19        System.out.println("          MENU PRINCIPAL          ");
20        System.out.println("-----");
21        System.out.println(" 1 - Calcular Puntajes          ");
22        System.out.println(" 2 - Imprimir Ganadores        ");
23        System.out.println(" 3 - Imprimir Rondas           ");
24        System.out.println(" 4 - Imprimir Participantes     ");
25        System.out.println(" 0 - Salir de la aplicacion     ");
26        System.out.println("=====");
27        System.out.println("Presione una de las opciones: ");
28    }
```

```
@Override
public void opcion1() {
    juego.resolverJuego();
    mostrarPantalla();
}

@Override
public void opcion2() {
    juego.ImprimirResultados();
    mostrarPantalla();
}

@Override
public void opcion3() {
    juego.listarRondas();
    mostrarPantalla();
}

@Override
public void opcion4() {
    juego.mostrarParticipantes();
    mostrarPantalla();
}
```

## Corriendo la aplicación

Al iniciar la aplicación se mostrara por pantalla en menú de configuración

```
=====
|          MENU DE CONFIGURACION          |
|-----|
| 1 - Cargar datos desde archivos CSV    |
| 2 - Cargar datos desde BASE DE DATOS   |
| 0 - Salir de la aplicacion             |
|-----|
=====
```

Presione una de las opciones:

El usuario podrá elegir el origen de los datos

**la opción 1** – leerá los datos de los archivos CSV.

El sistema mostrar el avance de la carga de datos de los archivos y mostrara por pantalla si hay errores en los datos.

```
Presione una de las opciones:
1
-----
BIENVENIDO AL SISTEMA DE PRONOSTICOS DEPORTIVOS
-----
Cargando datos del archivo: src/main/resources/archivos/configuracion.csv
-----
El id y Nombre son datos requeridos
-----
Cargando datos del archivo: src/main/resources/archivos/partidos.csv
-----
Cargando datos del archivo: src/main/resources/archivos/pronosticos.csv
-----
```

**la opción 2** – leerá los datos de la base de datos

el sistema mostrara los mensajes del estado de la conexión a la base de datos y el avance de la carga de los datos. Mostrara los mensajes de errores correspondientes

```
-----
BIENVENIDO AL SISTEMA DE PRONOSTICOS DEPORTIVOS
-----
Cargando datos del archivo: src/main/resources/archivos/configuracion.csv
-----
Conexion con la base de datos *prode* realizada con exito
Actualizando datos de la tabla *personas*

Conexion con la base de datos *prode* realizada con exito
Actualizando datos de la tabla *partidos*
El valor de los goles debe ser mayor o igual a 0.

Conexion con la base de datos *prode* realizada con exito
Actualizando datos de la tabla *pronosticos*
```

Si presiona el 0 – saldrá de la aplicación

Cualquiera de las opciones 1 y 2, al terminar la carga de los datos, mostrara por pantalla el menú principal de la aplicación

```
=====
|          MENU PRINCIPAL          |
|-----|
| 1 - Calcular Puntajes           |
| 2 - Imprimir Ganadores         |
| 3 - Imprimir Rondas            |
| 4 - Imprimir Participantes     |
| 0 - Salir de la aplicacion      |
|-----|
| Presione una de las opciones:  |
=====
```

**La opción 1** – realizara el proceso del calculo de puntajes de los participantes. Si no hay errores mostrara el mensaje al finalizar

```
1
-----
| Se realizo el calculo de los puntajes exitosamente |
-----
```

**La opción 2** – esta opción muestra por pantalla los puntajes obtenidos por los participantes

```
2
-----
|          Puntajes          |
|-----|
#---> Ariel
**** Puntos Totales Acumulados : 0 ****
-----
#---> Juan
Ronda: 1 Aciertos :3/3 Puntos: 3
Ronda: 2 Aciertos :4/4 Puntos: 4
Puntos extras por ronda(1) perfecta ---> 5
Puntos extras por ronda(2) perfecta ---> 5
Puntos extras por Fase perfecta ---> 10
**** Puntos Totales Acumulados : 27 ****
-----
#---> Fede
**** Puntos Totales Acumulados : 0 ****
-----
#---> Carola
**** Puntos Totales Acumulados : 0 ****
-----
#---> laura
Ronda: 1 Aciertos :2/3 Puntos: 2
Ronda: 2 Aciertos :4/4 Puntos: 4
```



Si el usuario introduce la opción 2 sin haber calculado los puntajes previamente obtendrá por pantalla los nombres de los participantes con puntuación 0

```
-----  
Puntajes  
-----  
#--> Ariel  
**** Puntos Totales Acumulados : 0 ****  
-----  
#--> Juan  
**** Puntos Totales Acumulados : 0 ****  
-----  
#--> Fede  
**** Puntos Totales Acumulados : 0 ****  
-----  
#--> Carola  
**** Puntos Totales Acumulados : 0 ****  
-----  
#--> laura  
**** Puntos Totales Acumulados : 0 ****  
-----
```

**La opción 3** – mostrara por pantalla el listado de los partidos por rondas con sus resultados

```
3  
-----  
Listado de partidos por rondas  
-----  
Ronda : 1  
1- River(2) VS Boca(2) Resultado : EMPATE  
2- Banfield(3) VS San Lorenzo(1) Resultado : LOCAL  
3- Independiente(2) VS Lanus(4) Resultado : VISITANTE  
-----  
Ronda : 2  
1- Talleres(1) VS Quilmes(2) Resultado : VISITANTE  
2- Velez(0) VS Gimnasia(2) Resultado : VISITANTE  
3- Union(2) VS Colon(0) Resultado : LOCAL  
4- Racing(1) VS Rosario Central(2) Resultado : VISITANTE
```

**La opción 4** – Mostrara el listado de los participantes

```
4  
-----  
Listado de participantes  
-----  
Clave --> 24230593 Nombre: Ariel  
TE: 3212123 E-mail: ariel@gmail.com  
Clave --> 29785451 Nombre: Juan  
TE: 66988989 E-mail: juan@juan.com  
Clave --> 32566698 Nombre: Fede  
TE: 13212132 E-mail: fede@hotmail.com  
Clave --> 22078657 Nombre: Carola  
TE: 54654564 E-mail: carola@gmail.com  
Clave --> 28478966 Nombre: laura  
TE: 54564546 E-mail: laura@yahoo.com
```

Por ultimo **la opción 0** – Cierra la aplicación

```
=====
|                MENU PRINCIPAL                |
|-----|
| 1 - Calcular Puntajes                        |
| 2 - Imprimir Ganadores                      |
| 3 - Imprimir Rondas                        |
| 4 - Imprimir Participantes                  |
| 0 - Salir de la aplicacion                  |
|-----|
Presione una de las opciones:
0
-----
ha salido con exito del sistema
-----
```

### Conclusiones

En el desarrollo de la aplicación se utilizaron herramientas y técnicas adquiridas en el curso de Java Inicial de Argentina Programa 4.0 – Fue un corto pero intenso recorrido donde se trataron muchos temas. Muchas gracias por la dedicación de los docentes, tanto en las clases sincrónicas dictadas por Fernando como en el feedback que tuvimos siempre de Federico, del tutor del TP. Esperemos continuar nuestro crecimiento en el próximo nivel.