Martín Gómez, Javier 47231977M javi040898@gmail.com-Ayala Urbanos, Christian 51196082Z christianayala.saramago@gmail.com

UNIVERSIDAD DE ALCALÁ GII PATRONES SOFTWARE GRUPO 5

SISTEMA GESTOR DE PLANTILLAS DE UNA LIGA DE BALONCESTO (SGPLB)

Contenido

[INTRODUCCIÓN 3](#_Toc63528934)

[REQUISITOS 3](#_Toc63528935)

[Requisitos Funcionales 3](#_Toc63528936)

[Requisitos No Funcionales 4](#_Toc63528937)

[Excepciones 4](#_Toc63528938)

[Distinción de Usuarios 5](#_Toc63528939)

[Usuarios de la aplicación 5](#_Toc63528940)

[Jugadores 5](#_Toc63528941)

[Cuerpo técnico 6](#_Toc63528942)

[Directiva 6](#_Toc63528943)

[Socio 6](#_Toc63528944)

[Dueño 6](#_Toc63528945)

[DIAGRAMA DE CASOS DE USO 7](#_Toc63528946)

[PATRONES UTILIZADOS 9](#_Toc63528947)

[Patrones de creación 9](#_Toc63528948)

[Patrón Factory Method 9](#_Toc63528949)

[Patrón Singleton 10](#_Toc63528950)

[Patrones estructurales 11](#_Toc63528951)

[Patrón Fachada 11](#_Toc63528952)

[Patrón Proxy 13](#_Toc63528953)

[Patrón Adapter 13](#_Toc63528954)

[Patrón DAO 14](#_Toc63528955)

[Patrones de comportamiento 15](#_Toc63528956)

[Patrón Iterator 15](#_Toc63528957)

[Patrón State 16](#_Toc63528958)

[Patrón Strategy 17](#_Toc63528959)

[MANUAL DE INSTALACIÓN 18](#_Toc63528960)

[Base de datos 20](#_Toc63528961)

[Inserción de datos 20](#_Toc63528962)

[MANUAL DE USUARIO 20](#_Toc63528963)

[Login 21](#_Toc63528964)

[Interfaz Dueño 25](#_Toc63528965)

[Interfaz Directiva 28](#_Toc63528966)

[Interfaz Jugador 33](#_Toc63528967)

[Interfaz Miembro del cuerpo técnico 34](#_Toc63528968)

[Interfaz Socio 34](#_Toc63528969)

[Interfaz Registro Socio 35](#_Toc63528970)

[Botones volver y cerrar sesión 36](#_Toc63528971)

[Iconos 36](#_Toc63528972)

# INTRODUCCIÓN

Desde una Liga de Baloncesto nos han solicitado el desarrollo de una aplicación informática para la gestión de las plantillas de los equipos.

Se va a desarrollar un sistema software para la gestión de plantillas de una liga de baloncesto.

El sistema almacenará los datos en una base de datos PostgreSQL con todos los datos de las plantillas de la liga, incluidos datos de directiva, contratos, socios... Dependiendo del tipo de usuario que acceda al sistema, podrá realizar unas operaciones u otras.

El sistema contará con una interfaz gráfica sencilla para con esto facilitar su uso de cara a los usuarios. Esto se debe a que se usará el lenguaje de programación Java para el desarrollo de dicha interfaz gráfica. De cara a una implementación venidera, se desarrollará una aplicación web y/o aplicación móvil para con ello facilitar más aún el acceso de los usuarios.

Esta interfaz gráfica será diferente para cada tipo de usuario. Por ejemplo, un socio no podrá realizar las mismas operaciones que un directivo del equipo, esto se basará en los distintos roles, como se ha explicado anteriormente.

El sistema se desarrollará aplicando patrones software, incluyendo de creación, estructurales, comportamiento, etc. También, se usará el patrón DAO (Data Access Object) para separar la lógica de negocio con la lógica de los datos, ya que usaremos una base de datos PostgreSQL.

Se incluirá un manual de usuario con la explicación del funcionamiento del sistema y, además, los pasos detallados de la instalación.

# REQUISITOS

## Requisitos Funcionales

|  |  |
| --- | --- |
| **Identificativo** | **Descripción** |
| RF1 | Pedir autenticarse al usuario para acceder al sistema |
| RF2 | Comprobar que el usuario y la contraseña coinciden con lo registrado en la base de datos |
| RF3 | Dar de alta, baja o modificar contratos de jugadores |
| RF4 | Distinguir los diferentes tipos de usuarios |
| RF5 | Registrar las plantillas de los equipos |
| RF6 | Actualizar la información de las plantillas |
| RF7 | Permitir a los usuarios consultar la diferente información sobre equipos, jugadores, etc. (Dependiendo del rol esta información será diferente) |
| RF8 | Limitar el número de jugadores por plantilla a 16 jugadores como máximo |
| RF9 | Limitar el número de jugadores por plantilla a 11 jugadores como mínimo |

## Requisitos No Funcionales

|  |  |
| --- | --- |
| **Identificativo** | **Descripción** |
| RNF1 | Los datos se almacenarán en una base de datos PostgreSQL |
| RNF2 | Se desarrollará el proyecto usando el lenguaje de programación Java a través del entorno de desarrollo NetBeans IDE. |
| RNF3 | Los nombres de usuarios será el DNI, por lo que será único |
| RNF4 | El sistema tendrá una interfaz gráfica utilizando la librería Swing |
| RNF5 | Se entregará un manual de usuario que detallará el uso de la aplicación |
| RNF6 | La aplicación se desarrollará siguiendo unos patrones software |
| RNF7 | El sistema debe ser accesible a cualquier hora del día |
| RNF8 | El sistema se podrá ejecutar en el sistema operativo Windows |
| RFN9 | El sistema tendrá que ser escalable, permitiendo la implementación de futuras funcionalidades |
| RNF10 | El sistema deberá proporcionar advertencias de error informativos y orientados a los usuarios. |
| RNF11 | El sistema permite las operaciones de selección, copiado y pegado de texto |

## Excepciones

Además de las excepciones predefinidas, la aplicación deberá controlar al menos las siguientes excepciones:

* Alta de un directivo, jugador o miembro de cuerpo técnico existente.
* Baja de un directivo, jugador o miembro de cuerpo técnico no existente.
* Solicitud de renovación de un jugador cedido.
* Alta de un socio que ya está dado de alta en la aplicación.
* Los equipos no pueden tener más de 16 jugadores en plantilla (RF8).
* Los equipos no pueden tener menos de 11 jugadores en plantilla (RF9).
* Solo se puede ser socio de un equipo.
* Alta de un jugador con dorsal ya existente en el equipo.
* Ausencia de campos en el alta de un usuario.

# Distinción de Usuarios

## Usuarios de la aplicación

En el sistema se verán representados distintos tipos de usuarios, los cuales se pueden representar por roles. Por un lado, se encuentra el personal de cada club y por otro lado los usuarios socios de cada equipo, los cuales podrán consultar los distintos tipos de datos o estadísticas de los jugadores, clasificación, etc. Los usuarios deberán identificarse para así dar unos permisos u otros, para ello dispondrán de un usuario diferenciador. Estos nombres de usuario vendrán acompañados obviamente de una clave personal

Por su parte los socios entrarán en la aplicación mediante su correo y clave y tendrán posibilidad de ver las plantillas de cada equipo.

La aplicación tendrá diferentes tipos de usuarios:

1. Jugadores.
2. Cuerpo técnico.
3. Directiva.
4. Socio.
5. Dueño

Los datos a considerar para cada usuario son comunes sin tener distinciones de roles:

* DNI
* Nombre
* Nacionalidad
* Correo electrónico
* Contraseña

También, habrá roles comunes a todos los usuarios:

* Consulta de plantillas
* Ver líderes de la liga
* Consulta de clasificación
* Modificación datos personales

### Jugadores

Para todos los jugadores, se deben añadir los siguientes datos:

* Posición
* Dorsal
* Fecha fin de contrato
* Estado
* Salario
* Media de puntos
* Media de rebotes
* Media de asistencias

Roles:

* Solicitar renovación
* Solicitar traspaso
* Renuncia de contrato

### Cuerpo técnico

Para todos los entrenadores, se deben añadir los siguientes datos:

* Puesto dentro del cuerpo técnico
* Fecha fin de contrato
* Táctica favorita
* Salario

Roles:

* Solicitar renovación
* Renuncia de contrato

### Directiva

Para todo el personal, se deben añadir:

* Puesto
* Fecha fin de contrato
* Salario

Roles:

* Solicitar renovación
* Renuncia de contrato
* Alta jugador o miembro del cuerpo técnico
* Baja jugador o miembro del cuerpo técnico
* Modificación contrato jugador o miembro del cuerpo técnico

### Socio

Para todos los socios, se deben añadir:

* Equipo

Roles:

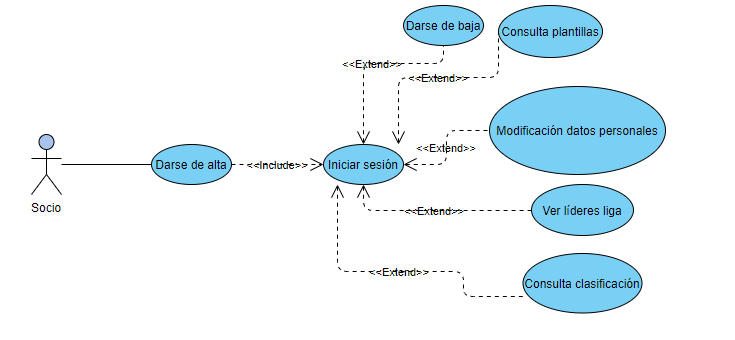
* Darse de baja del equipo

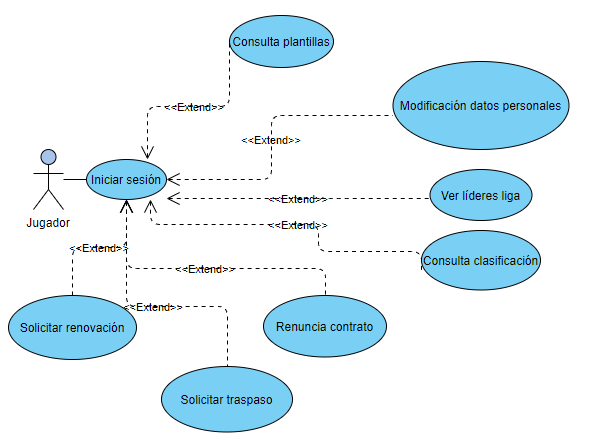
### Dueño

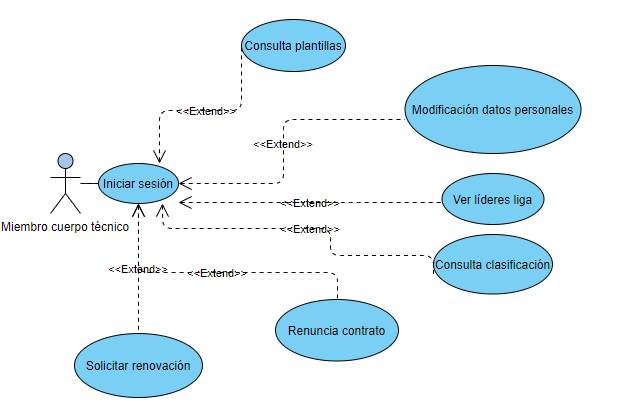
Para todos los dueños, se deben añadir los siguientes roles:

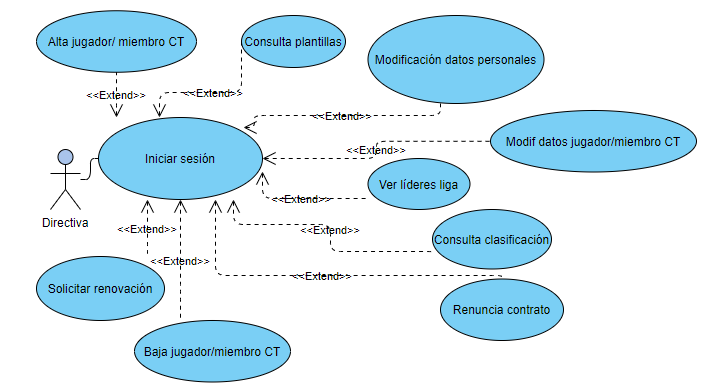
* Alta directivos
* Baja directivos

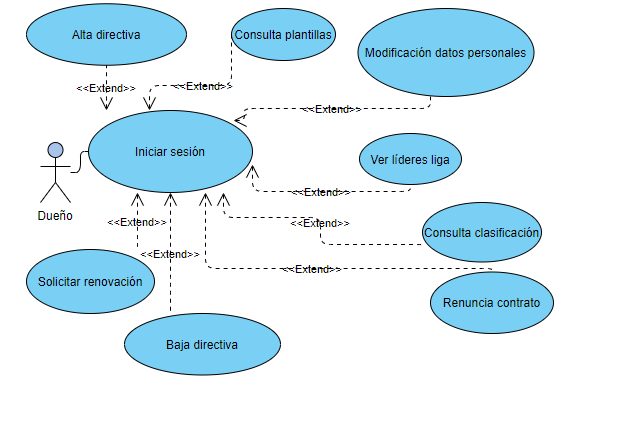
# DIAGRAMA DE CASOS DE USO











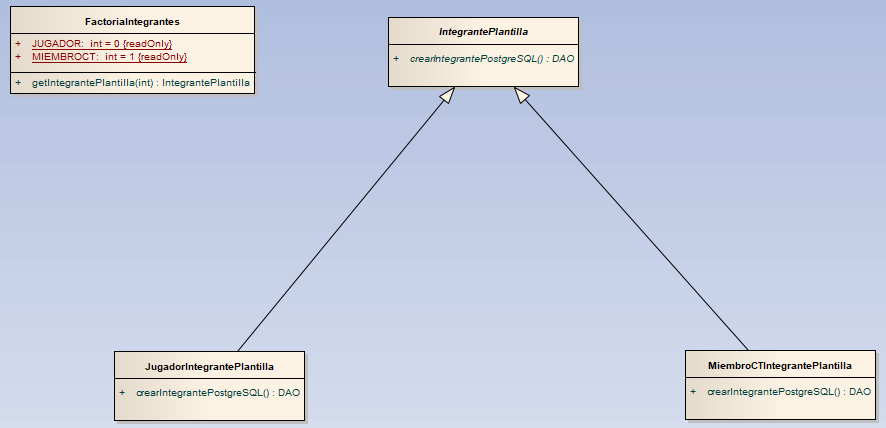
# PATRONES UTILIZADOS

## Patrones de creación

### Patrón Factory Method

Este patrón se utiliza definir una interfaz que cree un objeto, pero delega a las subclases que decidan qué método instanciar. En este caso queremos crear integrantes de plantilla (Jugador y Miembro del Cuerpo Técnico).

El UML es el siguiente:



Producto: IntegrantePlantilla

ProductoConcreto: JugadorIntegrantePlantilla y MiembroCTIntegrantePlantilla

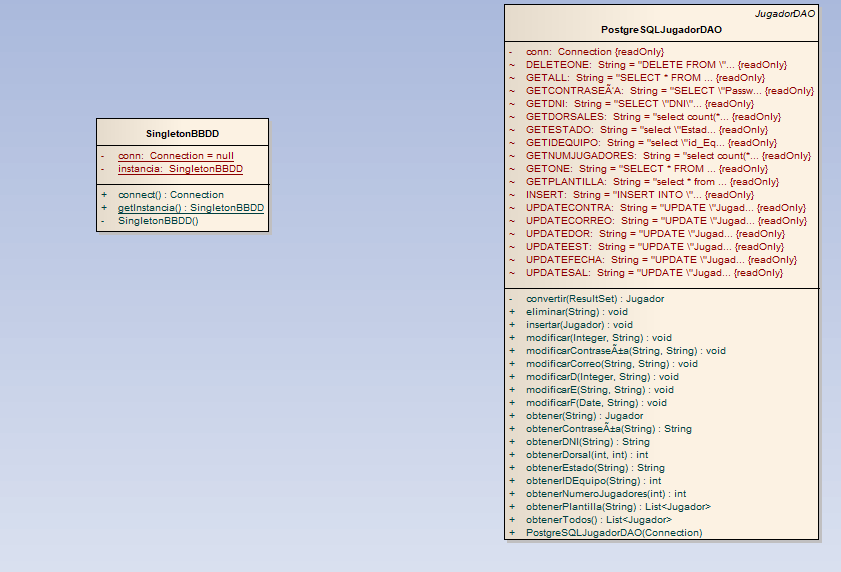
Creador: FactoriaIntegrantes.

Las subclases de la fachada (explicada posteriormente) serán quienes utilicen estas clases, haciendo de cliente.

### Patrón Singleton

Este patrón se utiliza para garantizar que las clases relacionadas con la base de datos solo tengan una instancia y así proporcionar un acceso global.

El UML es el siguiente:



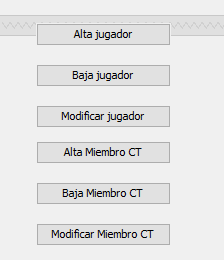
Cuando se instancien objetos de las clases del paquete SQL, se usará la única instancia del singleton a través del método getInstancia().

## Patrones estructurales

### Patrón Fachada

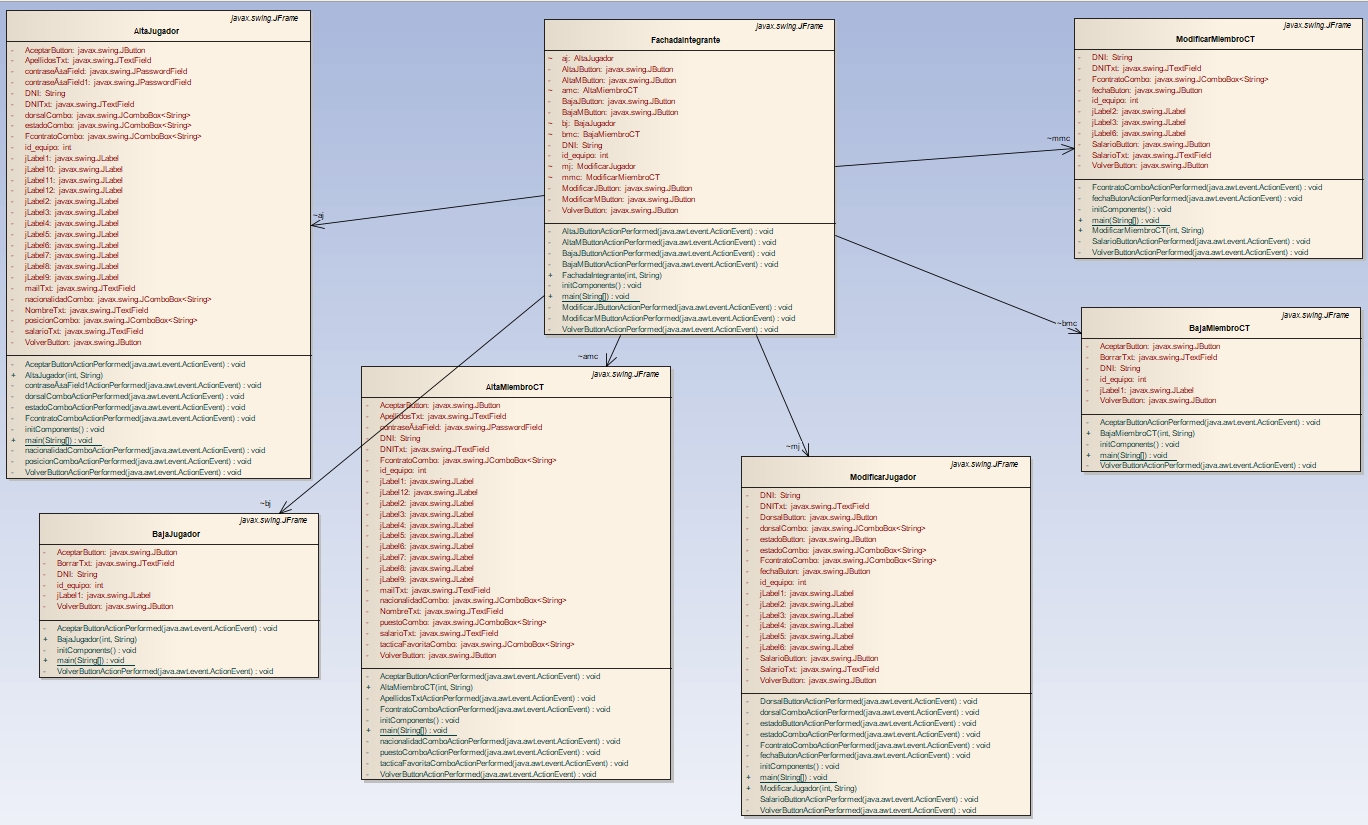
Se usa el patrón fachada en la interfaz FachadaIntegrante. Se utiliza para tener un acceso más sencillo a otras interfaces más complejas. Estas interfaces son AltaJugador, BajaJugador, BajaMiembroCT, AltaMiembroCT, ModificarJugador y ModificarMiembroCT.

La interfaz de la fachada es la siguiente:



Dependiendo del botón que se pulse, se accederá a una clase u otra.

El UML es el siguiente:



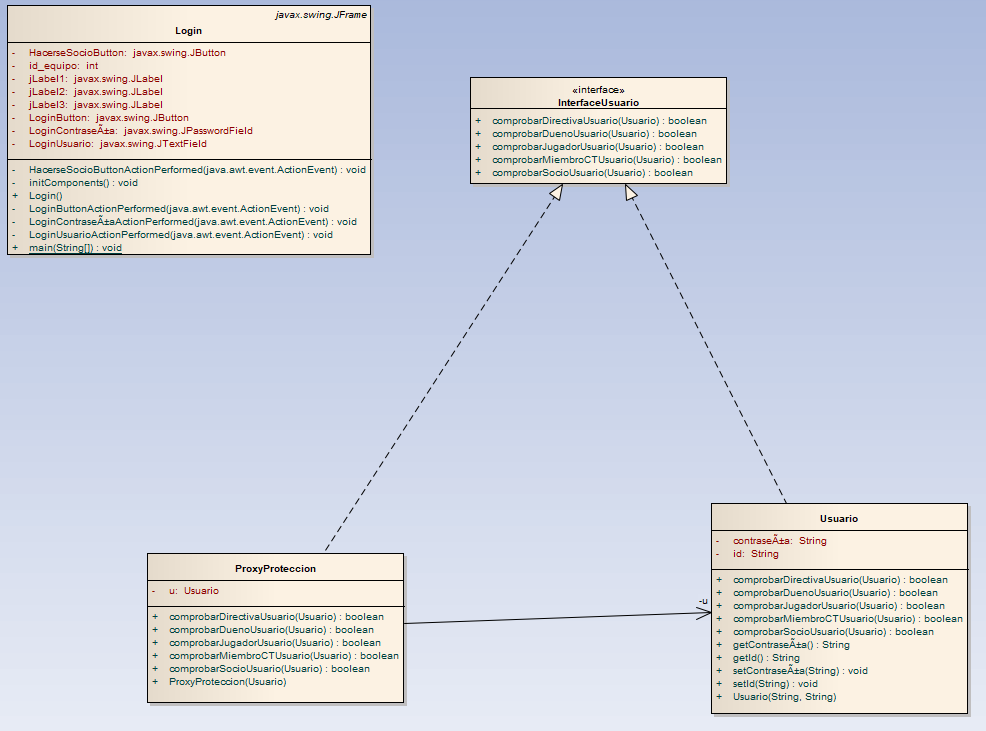
Fachada: FachadaIntegrante

Clases del subsistema: AltaJugador, BajaJugador, BajaMiembroCT, AltaMiembroCT, ModificarJugador y ModificarMiembroCT.

### Patrón Proxy

Se utiliza un proxy de protección para controlar los accesos a la aplicación únicamente a los usuarios a través de su usuario (DNI) y contraseña.

El UML es el siguiente:



Sujeto: InterfaceUsuario. Es la interfaz común entre el Proxy y el SujetoReal.

Proxy: ProxyProtección. Redirige las llamadas de los métodos al objeto real.

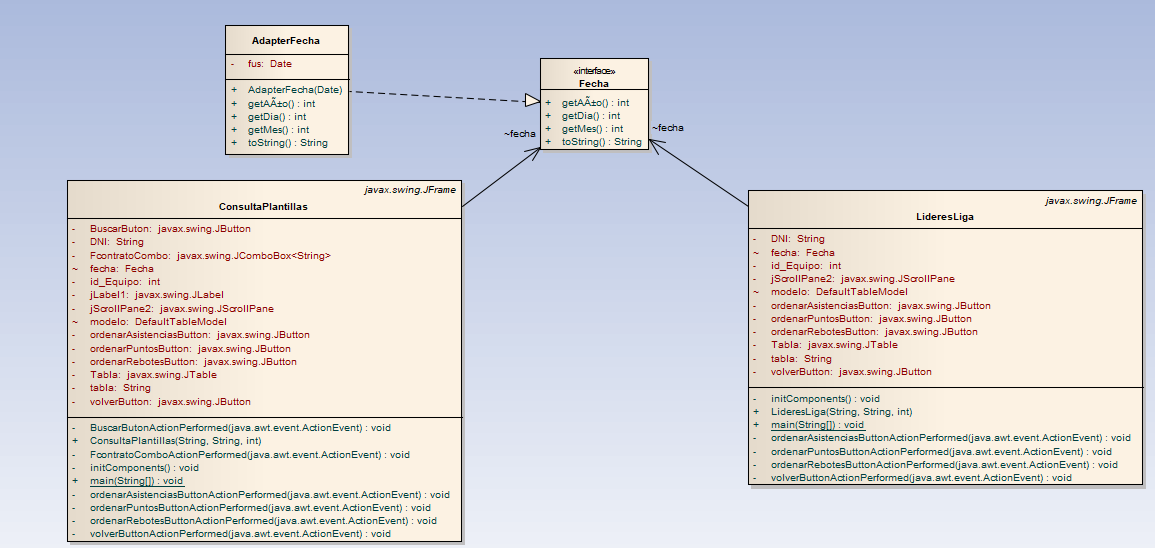
SujetoReal: Usuario. Implementa los servicios reales ofrecidos.

Cliente: Login. Utiliza la interfaz del proxy para usar la clase real (Usuario).

### Patrón Adapter

Este patron se utiliza para cambiar el formato de fecha US a nuestro formato, es decir, se adapta la fecha que tiene como formato el de la clase Date de Java al formato utilizado en España.

El UML es el siguiente:



Objetivo: en nuestro caso sería la interfaz Fecha que especifica el dominio a usar.

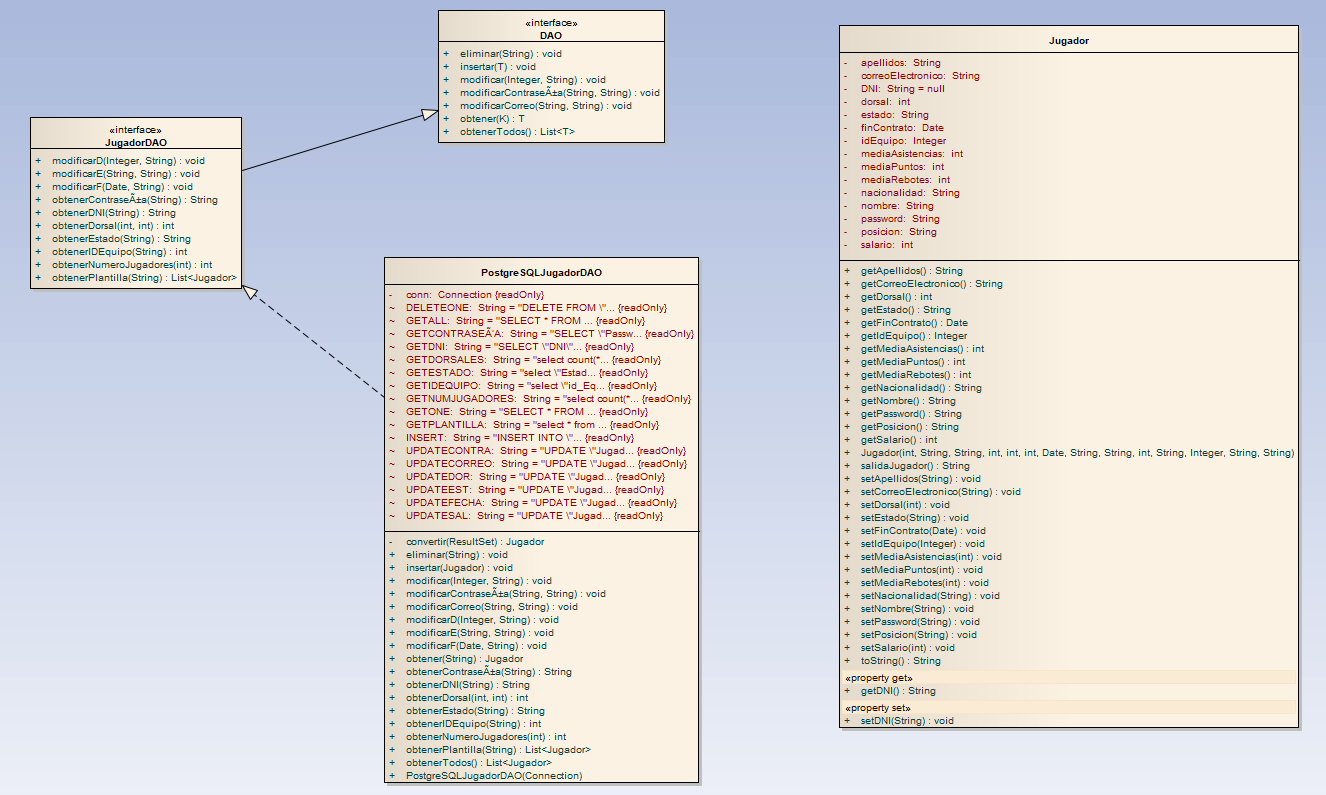
Adaptable: clase Date de Java que necesita ser adaptada y cambiar su formato

Adaptador: clase AdapterFecha que adapta el Adaptable a la interfaz Objetivo.

### Patrón DAO

Este patrón se utiliza para abstraer todos los accesos a la base de datos. En el diagrama UML que mostramos a continuación, únicamente mostramos el acceso a la tabla Jugador, pero lo hacemos con todas las tablas (MiembroCuerpoTecnico, Socio, Equipo, Directiva, Dueño y Jugador).

El UML es el siguiente:

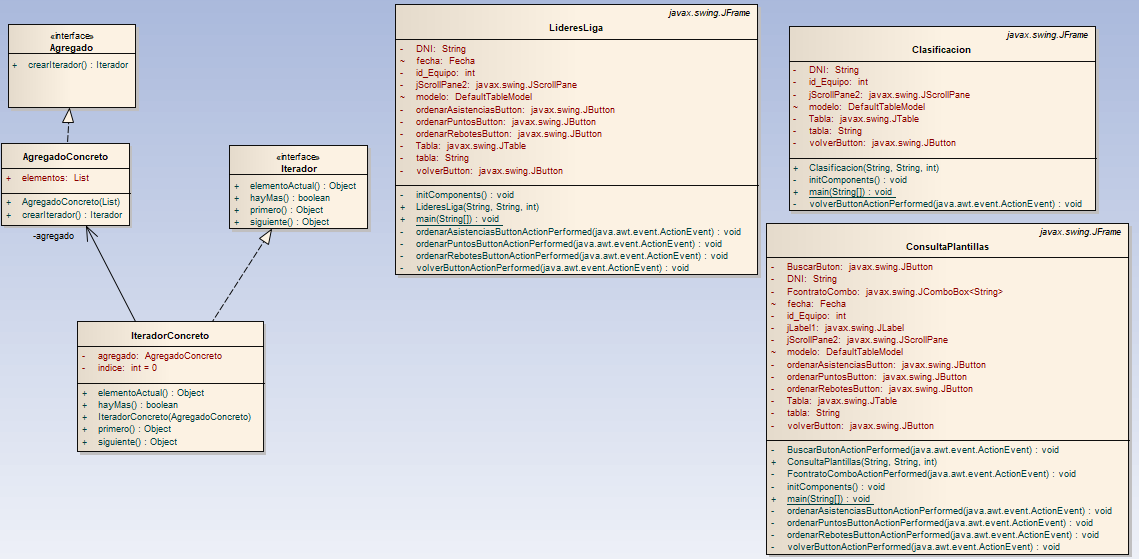


## Patrones de comportamiento

### Patrón Iterator

Se utiliza el patrón Iterator para recorrerse una lista de jugadores y equipos e ir añadiéndolos en la tabla que le corresponda, ya sea la de consulta plantillas, líderes de la liga y la clasificación general de equipos.

El diagrama UML es el siguiente:



Iterador: interfaz en la que se definen los métodos de iteración (primero(), siguiente(), hayMas(), elementoActual()).

Iterador Concreto: Implementación de los métodos de la interfaz Iterador que se han nombrado anteriormente y la creación de instancias se realiza por medio del Agregado Concreto

Agregado: se define el método para la creación del Iterador.

Agregado Concreto: implementa la interfaz Agregado, crea el Iterador y se define la lista que se recorrerá.

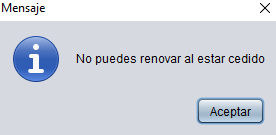
### Patrón State

Se utiliza el patrón Estado para cambiar el comportamiento de un objeto (Jugador) en tiempo de ejecución.

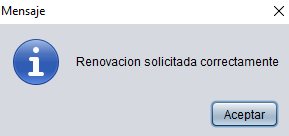
El Jugador podrá tener dos estados, Cedido y En propiedad y podrá solicitar renovación a través de este botón:



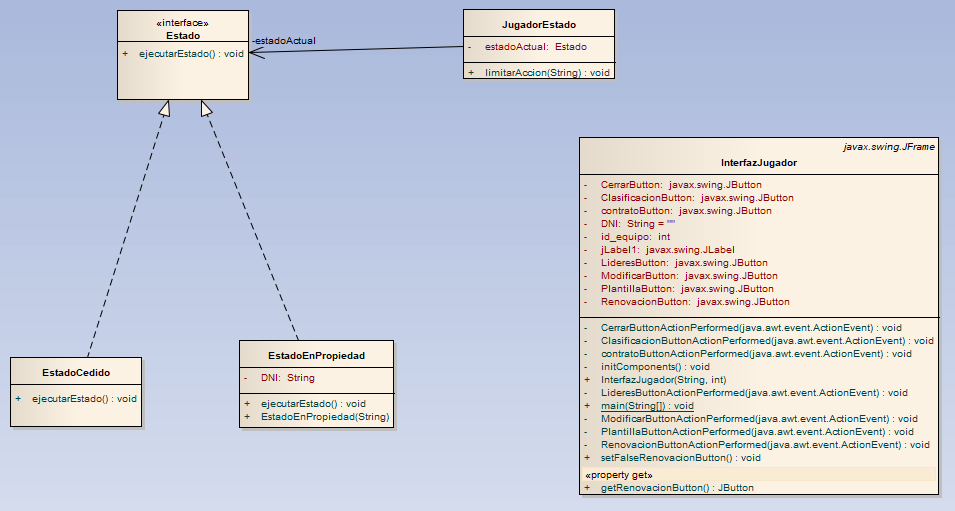
Si el estado es Cedido, no podrá solicitar renovación, ya que el Jugador no pertenece al club. Aparecerá esta advertencia:



Si el estado es En propiedad, el Jugador podrá solicitar la renovación al club y se enviará un correo al dueño del club:



El UML es el siguiente:



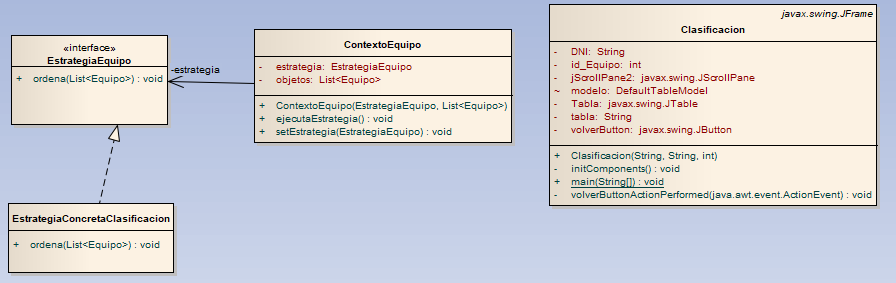
Contexto: JugadorEstado. Mantiene la referencia al estado actual.

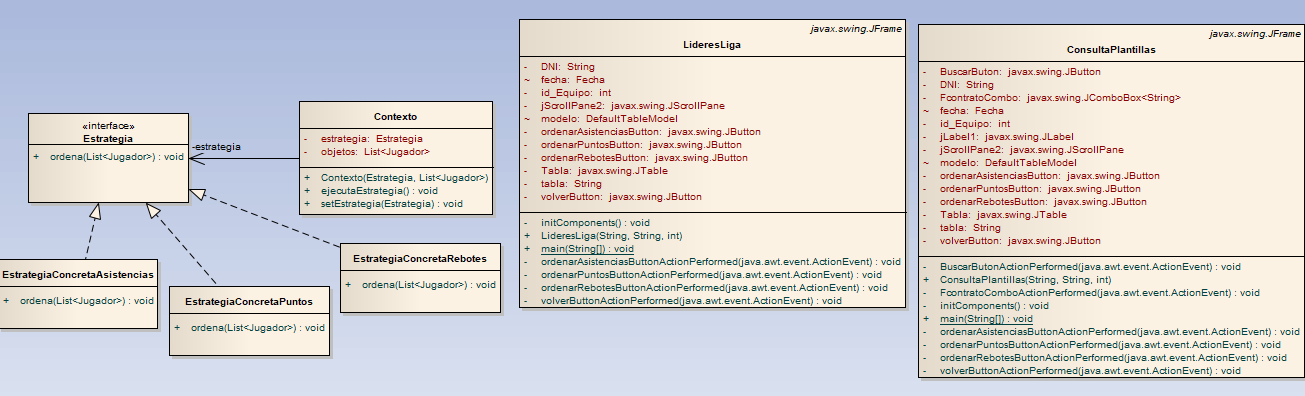
Estado: Estado. Define los métodos que dependen del estado del objeto.

EstadoConcreto: EstadoCedido y EstadoEnPropiedad. Implementan el comportamiento específico de cada estado, explicados previamente.

### Patrón Strategy

Este patrón se utiliza para realizar la ordenación de la lista de jugadores según sus rebotes, puntos y asistencias de orden mayor a menor. Por otro lado, también se usa para ordenar de mayor a menor según sus puntos, la lista de equipos de la liga.





Contexto: Contexto, ContextoEquipo, ambas clases se utilizan para ver que estrategia de ordenación se va a utilizar y ejecutarla a la lista de jugadores y lista de equipos respectivamente.

Estrategia: Estrategia y EstrategiaEquipos, son las interfaces en las que se definen los métodos de ordenación.

Estrategia Concreta: EstrategiaConcretaAsistencias, EstrategiaConcretaRebotes, EstrategiaConcretaPuntos, EstrategiaConcretaClasificacion, estas clases se utilizan para ordenar las listas de jugadores y equipos.

# MANUAL DE INSTALACIÓN

Para el correcto funcionamiento es necesario tener instalado en el equipo, en primer lugar java JDK y Netbeans. Para instalar el JDK habrá que introducirse en la siguiente dirección:

<https://www.oracle.com/java/technologies/javase/javase9-archive-downloads.html>

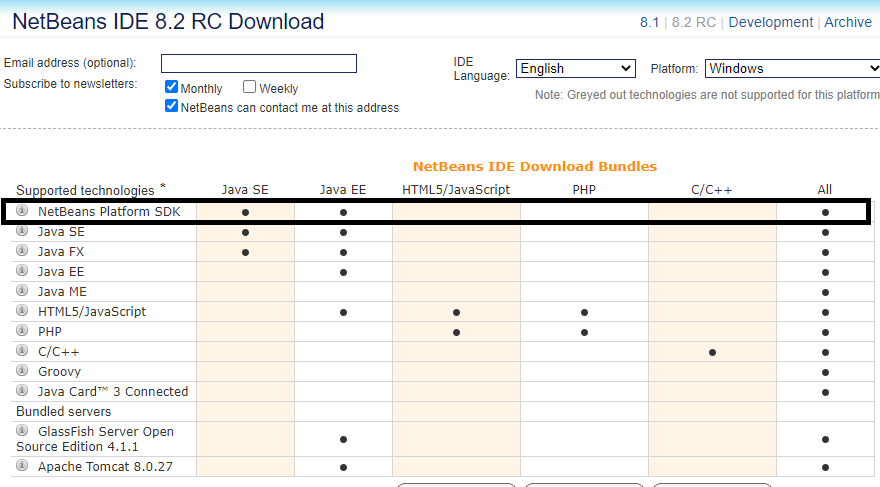
Aparecerá lo siguiente, donde se instalará dependiendo del sistema operativo de cada uno.



Después, para instalar NetBeans habrá que introducirse en la dirección:

<https://netbeans.org/downloads/8.2/rc/>

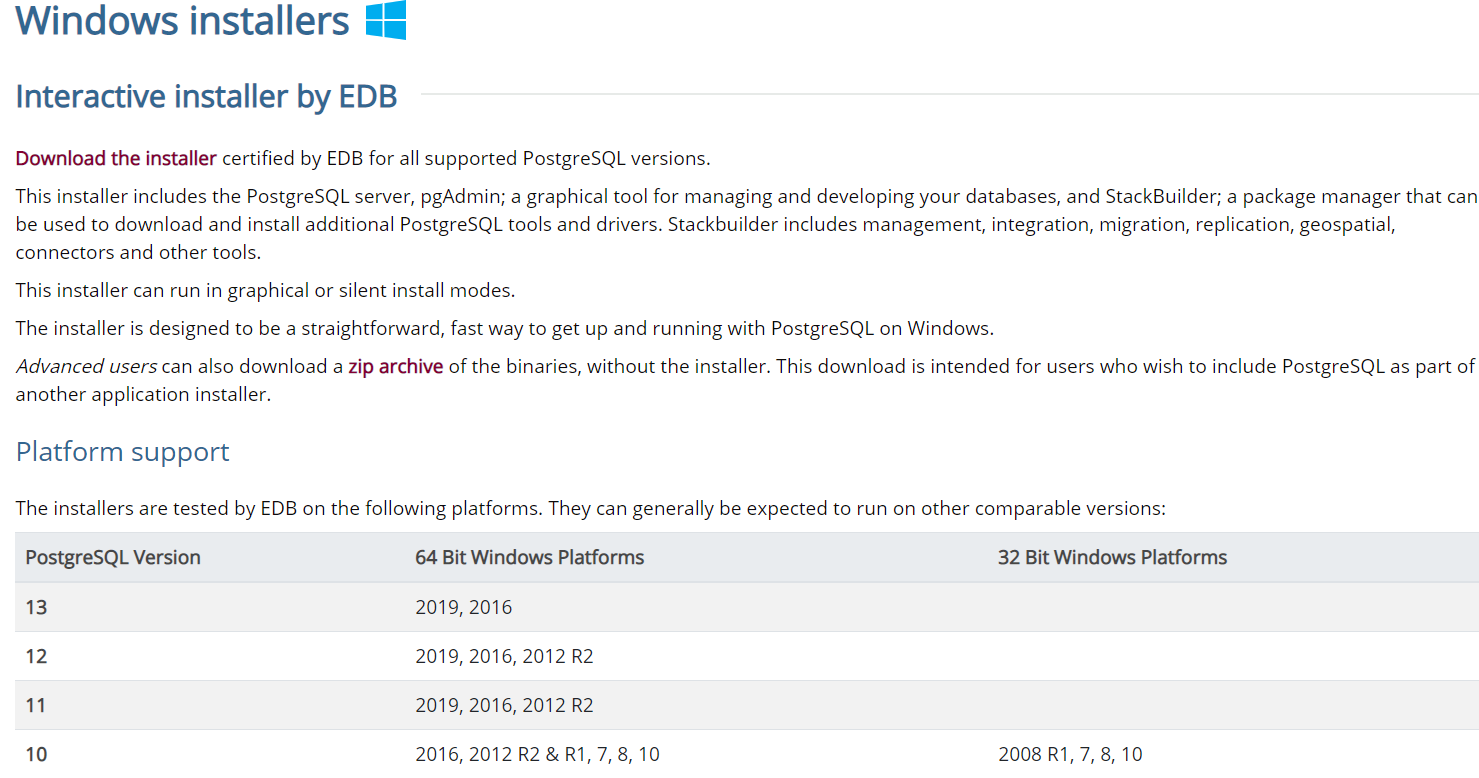
Aparecerá lo siguiente y habrá que instalarse lo remarcado:



Ahora, para instalar la base de datos (PostgreSQL), habrá que introducirse en el siguiente enlace:

<https://www.postgresql.org/download/windows/>

Aparecerá lo siguiente, donde seleccionamos la versión de PostgreSQL que queremos instalar (en nuestro caso hemos instalado la 12):

 Durante la instalación, también se instalará PgAdmin.

Para la conexión de PostgreSQL con NetBeans y poder enviar correos desde NetBeans, se utilizarán las siguientes librerías:

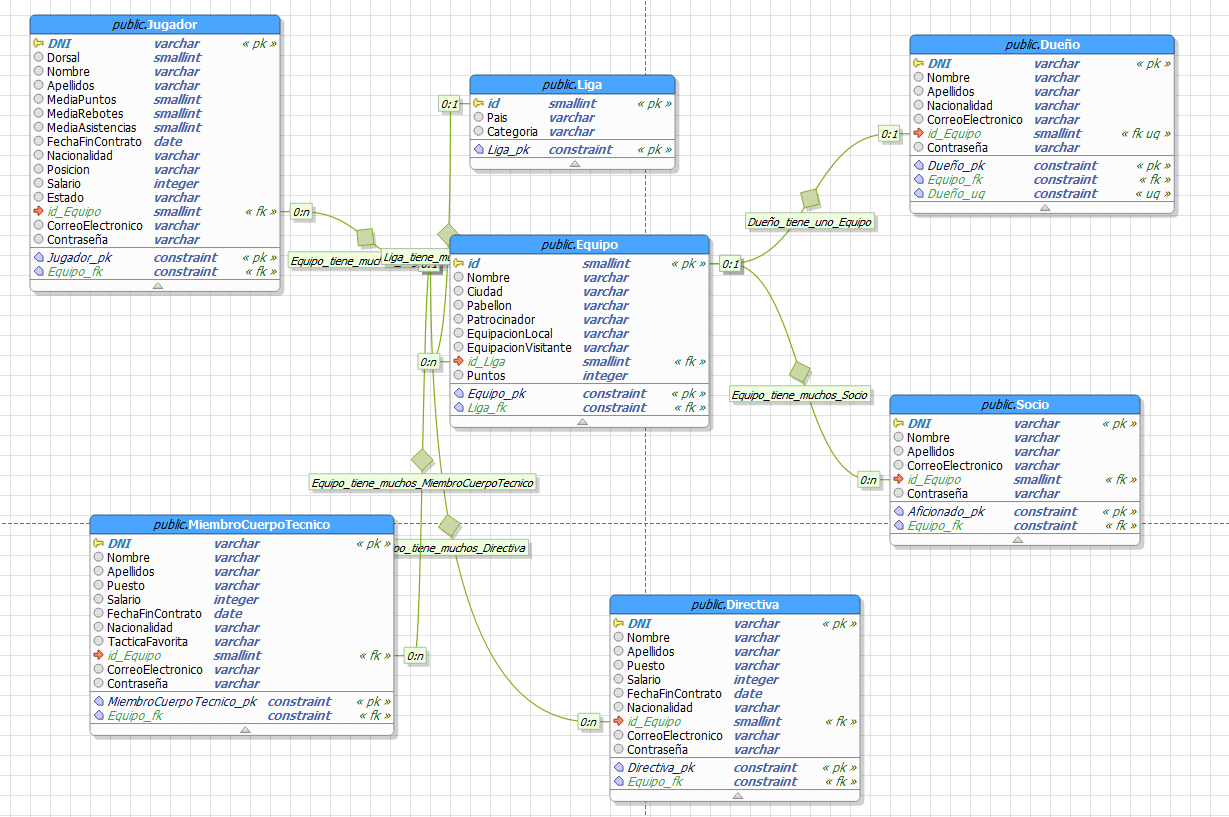


Con estas herramientas, se podrá ejecutar la aplicación.

## Base de datos

Para poner en funcionamiento la base de datos, habrá que ejecutar el script proporcionado llamado “sqlBD.sql”. Donde se crean las tablas.

El modelo e/r de la base de datos, realizado en PgModeler es el siguiente:



## Inserción de datos

Para crear registros en las tablas, habrá que abrir el script llamado “insertLigaDueño.sql” y ejecutar la línea donde se encuentra el insert de Liga.

Ahora, habrá que importar en la tabla "Equipo” el archivo “equipos.txt”. Después, se ejecutarán las líneas de los inserts de “Dueño” del script “insertLigaDueño.sql”.

Por último, se importarán en cualquier orden los archivos “cuerpotecnico.txt”, “directiva.txt”, “jugadores.txt” y “socios.txt” en sus respectivas tablas. Estos archivos contienen registros aleatorios que se usarán de modelo para probar la aplicación correctamente.

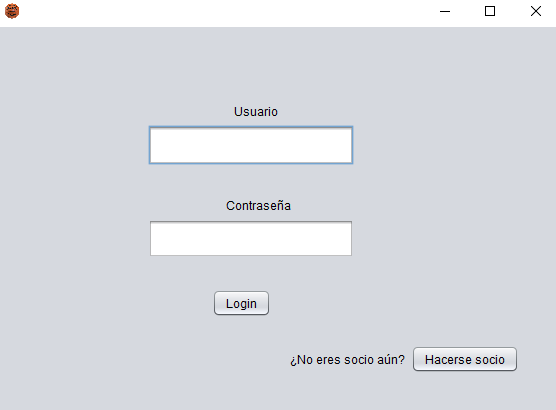
# MANUAL DE USUARIO

La aplicación se pondrá en funcionamiento arrancando el proyecto denominado PLPatrones.

Para el correcto funcionamiento de la aplicación, la clase Login (Login.java dentro del paquete Interfaces) debe ser la primera en ejecutarse.

## Login

Al iniciar la aplicación, aparecerá la ventana de login, donde cada usuario introducirá su usuario (DNI) y su contraseña. También, habrá un apartado para registrarte como socio:



Dependiendo del usuario que se introduzca, aparecerá su interfaz correspondiente. Recordemos que los usuarios son: dueño, directivo, jugador, miembro del cuerpo técnico y socio. Cada usuario tendrá su propia interfaz dependiendo de su tipo, sin embargo, habrá funciones comunes a todos los usuarios:

* **Consultar plantillas**: aquí se podrán observar los jugadores de cada equipo, pudiendo seleccionar cada equipo y, además, ordenar mediante botones a dichos jugadores por puntos, asistencias y rebotes. Vemos la plantilla de los lakers:



Se ordenan por puntos:



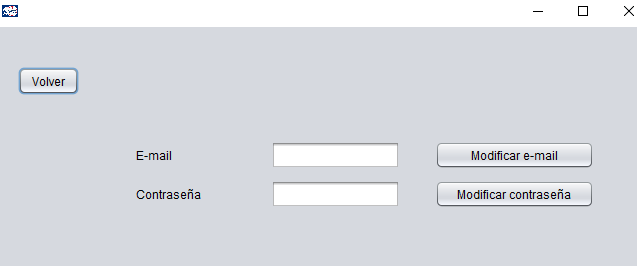
Ordenamos por asistencias la plantilla de los clippers:



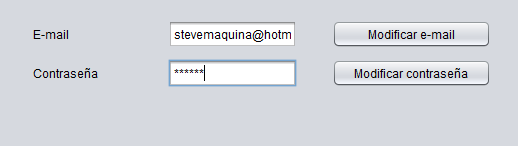
Ordenamos por rebotes a la plantilla de los bucks:



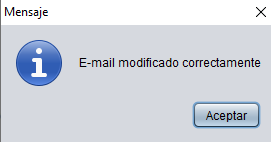
* **Modificar mis datos personales**: aquí podrás modificar tu correo electrónico y tu contraseña:



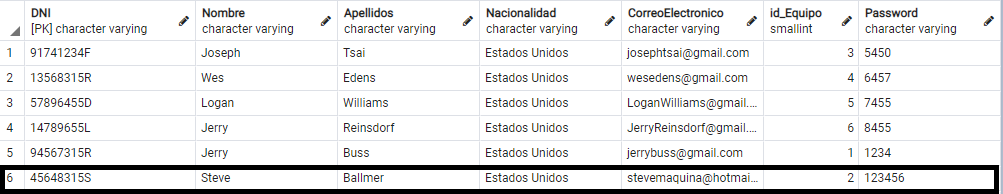
Modificamos el e-mail y la contraseña:



Aparecen mensajes de confirmación:



Comprobamos en la base de datos que se ha modificado correctamente:



* **Ver líderes de la liga**: aquí se podrán observar las estadísiticas (puntos, asistencias y rebotes) de todos los jugadores de la liga, pudiendo ordenarlos por puntos, asistencias o rebotes. Ordenados por puntos:

Ordenados por asistencias:

Ordenados por rebotes:

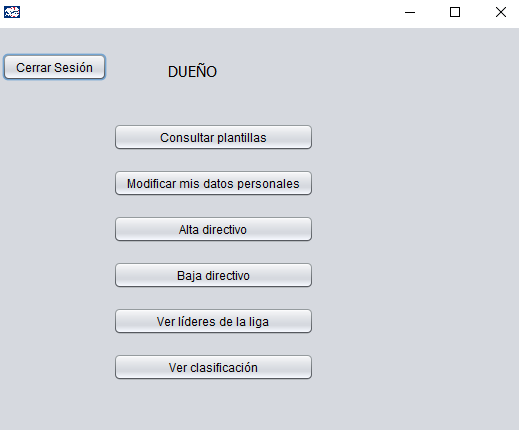


* Ver clasificación: se podrán observar los equipos de la liga, con algunas de sus caracerísticas, ordenados por puntos:



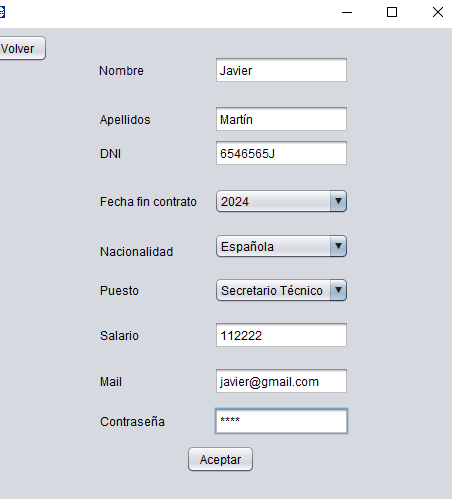
## Interfaz Dueño

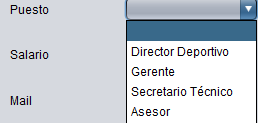
La interfaz del usuario Dueño será la siguiente:



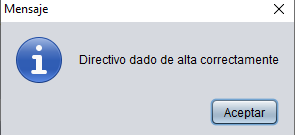
Aparte de las funciones comunes, se puede observar que también puede dar de alta y dar de baja a directivos:

* **Alta directivo**: podrá dar de alta a directivos en su equipo





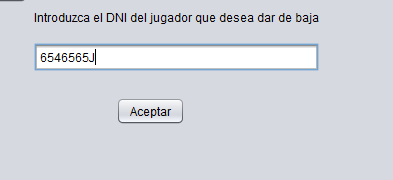


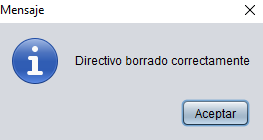


Vemos que en la base de datos se introduce correctamente:

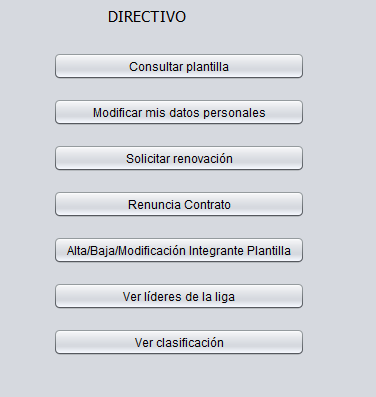


* **Baja directivo**: se da de baja a un directivo introduciendo su DNI. Borramos el directivo creado previamente.



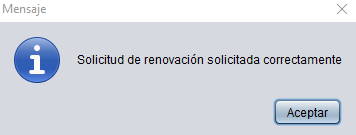


## Interfaz Directiva



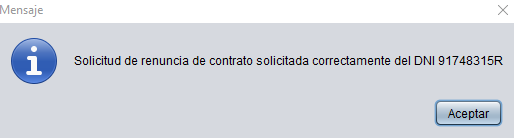
Se añaden diferentes funcionalidades como se puede comprobar:

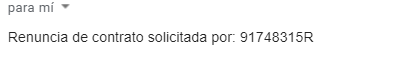
* Solicitar renovación: envía un e-mail al dueño solicitando su renovación:



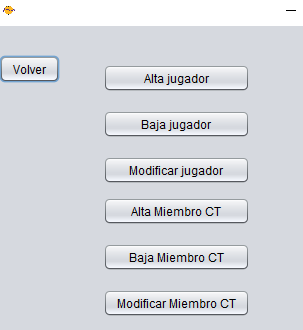


* Renuncia de contrato: también envía un correo, pero esta vez solicita renuncia de su contrato:

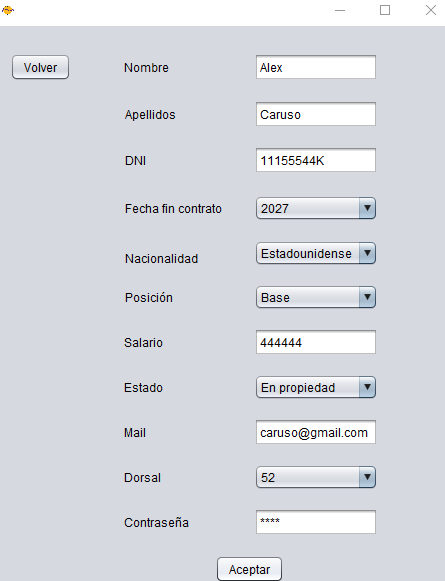


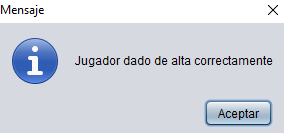


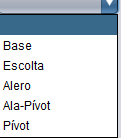
* Alta/baja/modificación integrante plantilla: esta función permite al directivo dar de alta, de baja o modificar a un integrante de la plantilla (jugador o miembro del cuerpo técnico):



* Alta jugador: da de alta a un jugador:

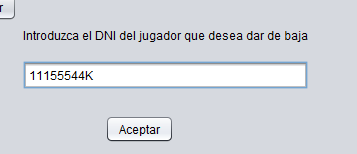




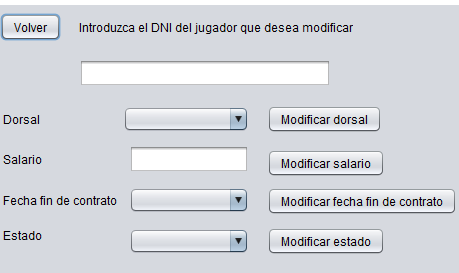


Controla excepciones como dar de alta a un jugador con el mismo DNI o mismo dorsal.ç

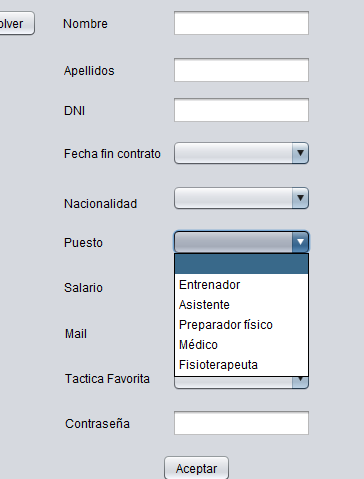
* Dar de baja jugador:

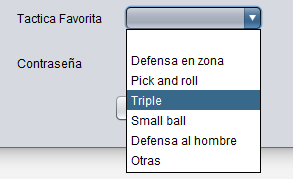


* Modificar jugador: se podrá modificar el dorsal, salario, fecha del fin del contrato o el estado:

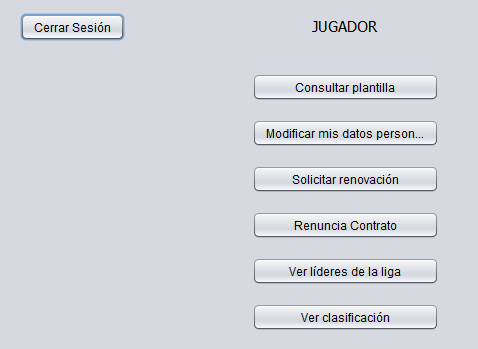


Del miembro del cuerpo técnico será prácticamente lo mismo, exceptuando pequeños matices como el puesto en vez de la posición o la táctica favorita:

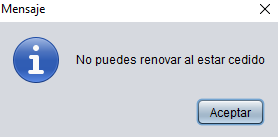




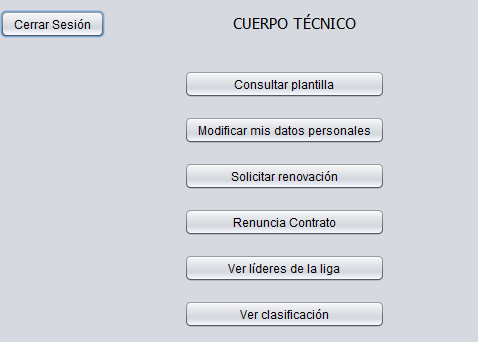
## Interfaz Jugador



Tendrá las funciones comunes más renuncia de contrato y solicitar renovación. Esta última se podrá solicitar solo si es un jugador en propiedad (los jugadores cedidos no pueden solicitar renovación):

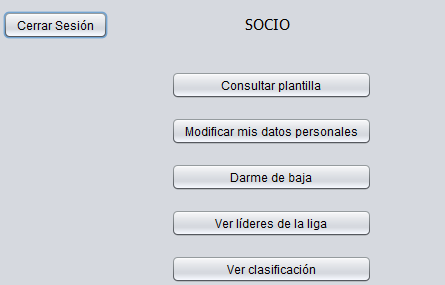


## Interfaz Miembro del cuerpo técnico

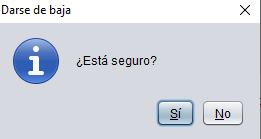


Igual que la de jugador, aunque ahora cualquier miembro del cuerpo técnico puede solicitar renovación.

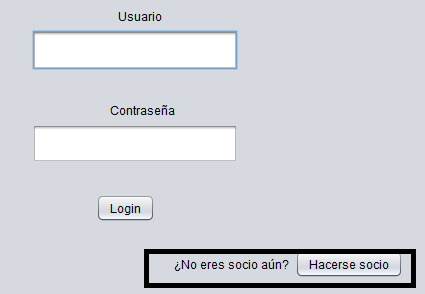
## Interfaz Socio



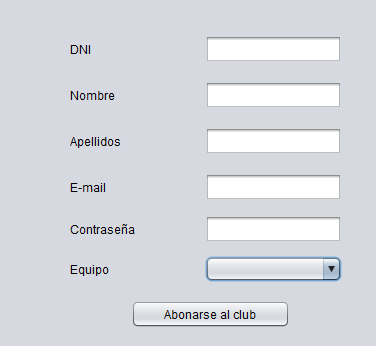
Tendrá las funciones comunes más darse de baja. Esta función elimina al socio directamente de la base de datos y cierra la sesión:



## Interfaz Registro Socio



Pulsando este botón aparece la siguiente interfaz:



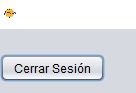
Introducirás tus datos personales y el equipo al que te quieres abonar.

## Botones volver y cerrar sesión

Todas las interfaces contendrán botón de volver a la interfaz anterior:



Cada interfaz principal, tendrá un botón de cerrar sesión que redirige a la interfaz del Login:



## Iconos

Habrá un icono genérico en la venta del Login:



Cada equipo dispondrá de su propio icono. Cada vez que un usuario de un equipo inicie sesión, las ventanas tendrán el icono del equipo del usuario registrado. Algunos de los iconos de los equipos son los siguientes:



