CASO PRÁCTICAS

Cuatro en raya

Interfaces Persona Computador

IPC – DSIC

UPV

Curso 2020-2021

Índice

1. Caso de Estudio 3
1.1. Registrarse en el juego 3
1.2. Autentificarse 4
1.3. Cerrar sesión 4
1.4. Jugar contra la máquina 4
1.5. Jugar contra otro jugador 4
1.6. Modificar perfil 5
1.7. Recordar datos de autenticación 5
1.8. Mostrar ranking 6
1.9. Ver las partidas realizadas en el sistema en un periodo de tiempo 6
1.10. Ver las partidas realizadas por un jugador en un periodo de tiempo 6
1.11. Ver las partidas ganadas por un jugador en un periodo de tiempo 6
1.12. Ver las partidas perdidas por un jugador en un periodo de tiempo 7
1.13. Ver el número de partidas realizadas en un periodo de tiempo 7
1.14. Ver el número de partidas ganadas/perdidas por un jugador en un periodo de tiempo 7
1.15. Cambiar el modo de visualización 8
2. Librería proporcionada para la persistencia de los datos 8
2.1. Agregar la librería Conecta4.jar a tu proyecto 8
2.2. API proporcionada por *Conecta4.jar* 9
2.2.1. Clases del modelo 10
Connect4 10
Player 12
Round 13
DayRank 13
Connect4DAOException 13
2.3. Uso de la librería desde el proyecto 14
Acceso a la librería 14
Utilización con componentes de la interfaz gráfica 14
Acceso a la información de los TreeMap 14
Carga de imágenes almacenadas en la base de datos en una ImageView 15
3. Ayudas a la programación 15
3.1. Carga de imágenes desde disco duro 15
3.2. Animar objetos 15
3.3. Manejo de fechas y del tiempo 16

Obtención de la semana del año a la que pertenece una fecha 16
Creación de una fecha o un campo de tiempo 16
Actualizando una fecha 16
3.4. Configurar DatePicker 16
4. Instrucciones de Entrega 17
4.1. Entrega I 17
4.2. Entrega II 18
5. Evaluación 18

Tablas

Tabla 1. API de la clase Connect4 10
Tabla 2. API de la clase Player 12
Tabla 3. API de la clase Round 13

Figuras

Figura 1. Descomprimir la librería en el directorio del proyecto 8
Figura 2. Pasos para añadir las librerías al proyecto 9
Figura 3. Descomprimir la carpeta avatars dentro de la carpeta src del proyecto 9
Figura 4. Métodos disponibles en una clase de la librería 10
Figura 5. DataPicker configurado para inhabilitar días en el pasado 17

1. Caso de Estudio

Desde IPCSoft quieren al lanzar al mercado el juego Conecta4, basado en el popular cuatro en raya que permita los jugadores jugar por parejas o contra la máquina. Además, todos los jugadores de la aplicación competirán en un ranking para ser el jugador con más puntos. Los jugadores podrán ganar puntos al jugar contra otros jugadores o contra la máquina (los puntos conseguidos en cada pueden variar). El tablero del juego tendrá 7 filas por 8 columnas, y deberá adaptarse a los cambios en la redimensión de la ventana por parte de los usuarios.

Para poder jugar, los usuarios tendrán que darse de alta en la aplicación para crear su perfil de jugador. Todos los jugadores podrán acceder a modificar los datos de su perfil, así como restablecer sus datos de acceso (a través de la cuenta de correo electrónico que proporcionarán en su registro). En concreto, los jugadores se acreditarán en el sistema mediante el nombre de usuario que ellos elijan (que no podrá repetirse en el sistema) y una contraseña.

Por otra parte, siempre será posible acceder al ranking con la clasificación de los jugadores. Además, los usuarios podrán acceder a diferentes gráficas con algunas estadísticas del juego:

- El número de jugadores por día que han participado en partidas entre contrincantes en un periodo de tiempo
- El número de partidas realizadas por día entre contrincantes en un periodo de tiempo
- El número de partidas ganadas o perdidas cada día por un jugador en un periodo de tiempo
- El número de contrincantes diferentes con los que ha jugado cada día en un periodo de tiempo

Seguidamente se detallan los escenarios de uso que se han obtenido tras el análisis de requisitos del sistema, que deben utilizarse para diseñar e implementar adecuadamente la aplicación requerida.

1.1. Registrarse en el juego

A Paco le encantan los juegos de mesa rápidos, en los que entretenerse un rato sin pensar demasiado. Echando un vistazo por la web se encuentra con la posibilidad de descargarse una aplicación que le permite jugar al conecta, tanto contra la máquina como contra otro jugador. Piensa que estaría bien probar el juego así que se descarga la aplicación y tras instalarla en su sistema, accede a la opción de registrarse.

El sistema le solicita que introduzca un nombre de usuario, que será utilizado para identificarle en el juego, una cuenta de correo electrónico válida, una contraseña, su fecha de nacimiento y si lo desea, una imagen para ser utilizada como avatar. Paco introduce como nombre de usuario "pgarcia". Como contraseña, elige "passConecta4!" y proporciona la cuenta de correo: "pgarcia@gmail.com". Finalmente, introduce una fecha de nacimiento unos años menor a su edad real. Como no le apetece buscar un

avatar en ese momento, se queda con el que se le ofrece por defecto. Tras introducir toda la información pedida, solicita finalizar el proceso.

El sistema comprueba los datos introducidos y le indica que ese nombre ya está siendo utilizado y debe seleccionar un nuevo valor que contenga entre 6 y 15 caracteres o dígitos si espacios, pudiendo usar guiones o sub-guiones. Paco decide seleccionar un *nickname* más gracioso e introduce "PlayfulPaco".

El sistema comprueba que el resto datos introducidos son válidos. La contraseña contiene entre 8 y 20 caracteres, incorpora al menos una letra en mayúsculas y minúsculas, así como algún dígito y algún carácter especial (!@#$%&*()-+=). El correo electrónico tiene un formato válido y el jugador tiene más de 12 años. El sistema registra al nuevo jugador y se lo notifica al usuario.

1.2. Autentificarse

Juan tiene un rato libe y piensa en desconectar un rato jugando al Conecta4. Abre la aplicación y selecciona la opción de autenticarse en el sistema para poder jugar y que se registren sus logros. El sistema le solicita que introduzca su nombre de usuario o *nickname* y su contraseña. Juan introduce su nickname y su contraseña. El sistema comprueba que existe un jugador con esos datos de autentificación y lo autentifica.

1.3. Cerrar sesión

Mario, Susana y Juan están jugando al Conecta4 por turnos, cambiándose para jugar todos contra todos. Susana y Juan han terminado su ronda de tres partidas, en las que ha resultado ganador Juan, así que Susana accede a la opción de cerrar la sesión para desconectarse como segundo jugador y que Mario pueda ocupar su plaza. El sistema informa a Susana que ha sido desconectada. Mario ya puede conectarse para jugar contra Juan.

1.4. Jugar contra la máquina

Tras un rato aburrida sin saber qué hacer, María decide jugar un rato al Conecta4, a ver si consigue ganar algunos puntos y superar a su hermano Mario. Sabe que jugando contra la máquina no puede conseguir tantos puntos como jugando contra su hermano, pero también es más sencillo ganar. Tras abrir la aplicación y autentificarse, selecciona la opción de jugar contra la máquina.

María es la primera en jugar. Selecciona la columna 4 para iniciar la partida. La máquina juega en la columna 1. Vuelve a ser el turno de María, por lo que decide colocar una ficha en la misma fila inicial, para lo que selecciona la columna 5. Los turnos continúan alternándose entre la máquina y María, y en pocas jugadas María consigue enlazar cuatro fichas seguidas en una misma fila. María suma más puntos, pero sigue sin alcanzar a su hermano. Decide jugar otra vez contra la máquina, puesto que parece que la máquina elige la posición donde jugar de forma aleatoria.

1.5. Jugar contra otro jugador

A Víctor y a Óscar les encanta jugar al ordenador. Desde hace unas semanas juegan al Conecta4 juntos. Sus partidas suelen estar muy reñidas, por lo que tienen un ranking

muy similar en el juego, aunque no consiguen alcanzar a sus padres. Víctor y Óscar deciden echar unas partidas, a ver quién queda por delante en el ranking hoy. Víctor es el primero en acreditarse y luego lo hace Óscar. Como los dos jugadores se han acreditado, pueden iniciar la partida uno contra otro. El primero en colocar una ficha es Víctor. Decide empezar por el centro del tablero, eligiendo la columna 5, por lo que su ficha se coloca en la fila 1 de esa columna. Óscar decide jugar su ficha en la columna 7, para evitar que su hermano gane al conectar 4 fichas en la primera fila, en la esquina derecha. Víctor decide seguir jugando sobre la columna 4, a ver si consigue conectar cuatro fichas seguidas en esa columna. Óscar decide jugar en la columna 2, aunque sabe que también debe prestar atención a las diagonales, puesto que también se puede ganar la partida de conectando 4 fichas en una diagonal. Víctor y Óscar continúan jugando alternativamente, hasta que Víctor consigue conectar cuatro fichas en una diagonal. Ha conseguido despistar a su hermano otra vez e incrementar su saldo de puntos.

1.6. Modificar perfil

Paco no eligió un avatar cuando se dio de alta, pero se ha cansado de aparecer en el ranking con el avatar por defecto, así que decide acceder a la opción de modificar los datos de su perfil. El sistema le ofrece la información que tiene actualmente: su nombre de usuario, su contraseña, su dirección de correo electrónica y su fecha de nacimiento, así como el avatar que tiene asignado. Paco se da cuenta que puede modificar cualquier dato excepto su nombre de usuario. Decide cambiar su avatar entre algunas imágenes que tiene en su máquina. Tras realizar el cambio, solicita que se actualice la información.

El sistema comprueba que todos los cambios introducidos cumplen con los requisitos establecidos. La contraseña contiene entre 8 y 20 caracteres, incorpora al menos una letra en mayúsculas y minúsculas, así como algún dígito y algún carácter especial (!@#$%&*()-+=). El correo electrónico tiene un formato válido y el jugador tiene más de 12 años. Por tanto, el sistema actualiza la información e informa al usuario.

1.7. Recordar datos de autenticación

Juan es un desastre recordando contraseñas. Iba a jugar con su amigo Jorge al Conecta4, pero al autenticarse en el sistema no recordaba su contraseña. Por suerte puede solicitar al sistema que se la recuerde. El sistema le pide que le indique su nombre de usuario y la cuenta de correo con la que se registró en el sistema. Juan introduce la cuenta que usa para registrarse en los juegos.

El sistema comprueba que la cuenta de correo registrada para el usuario indicado coincide y le manda un correo electrónico con un código de seguridad. El sistema solicita al usuario que introduzca el código de seguridad que le acaba de enviar.

Juan revisa su correo y encuentra el código. Proporciona dicho código al sistema. El sistema comprueba que coinciden y le muestra a Juan su contraseña.

NOTA: El envío del correo electrónico se simulará a través de informar al usuario por cualquier vía usando la interfaz.

1.8. Mostrar ranking

Víctor y Óscar han terminado una partida, y desean saber quién va el primero en el ranking. Óscar cree que por fin ha superado a su hermano mayor, ya que le ha ganado en las dos últimas partidas. Acceden a la opción de mostrar el ranking y el sistema es muestra la lista de jugadores ordenados de mayor a menor ranking. De cada jugador pueden ver su avatar, nombre de usuarios y puntuación. Como no se ven a primera vista, acceden a la opción de buscarse en la lista a través de su nombre de usuario. Óscar sigue sin superar a su hermano mayor, pero ambos están contentos porque están más arriba entre los mejores jugadores. **Ranquing y barra de búsqueda.**

1.9. Ver las partidas realizadas en el sistema en un periodo de tiempo

Es fin de semana y Juan quiere ver el movimiento en el juego en los últimos días, para ver a qué horas se han conectado sus amigos para jugar. Accede a la opción de ver las partidas realizadas en el sistema. El sistema le indica que seleccione la fecha de inicio y final del periodo de partidas que desea visualizar. Juan selecciona el sábado pasado como fecha de inicio y como final, la fecha de hoy. El sistema le ofrece como respuesta un listado con las partidas realizadas en esos días. En primer lugar, puede ver las más recientes. De cada partida, Juan puede observar el día y hora en el que se jugó, así como el nombre de usuario del ganador y perdedor de la partida.

1.10. Ver las partidas realizadas por un jugador en un periodo de tiempo

A Víctor le extraña que su hermano le lleve tanta diferencia en el ranking, así que decide mirar las últimas partidas que ha realizado. Accede a la opción de ver las partidas realizadas por un jugador en un periodo de tiempo. El sistema le pide que seleccione un jugador entre los que tiene registrados, así como una fecha de inicio y de fin del periodo a mostrar. Víctor selecciona el nombre de usuario de su hermano. Como fecha de inicio, elige el día anterior y como fecha de fin, elige hoy. El sistema le ofrece como respuesta un listado con las partidas realizadas en esos días. En primer lugar, puede ver las más recientes. De cada partida, Víctor puede observar el día y hora en el que se jugó, así como el nombre de usuario del ganador y perdedor de la partida. Víctor se da cuenta que solo ha jugado contra él. Por tanto, sabe que su hermano le ha superado porque se ha puesto a jugar contra la máquina. Piensa chivarse a su madre, ya que seguro que ha tenido que jugar durante mucho tiempo y su madre no les permite jugar tanto a juegos electrónicos. **no salen las partidas jugadas contra la máquina.**

1.11. Ver las partidas ganadas por un jugador en un periodo de tiempo

Jorge, Juan y Mario están en casa de Juan pasando la tarde. Han estado jugando al Conecta4, haciendo una pequeña competición. Tras ver cómo han quedado en el ranking del juego han visto que Miguel, otro de sus amigos, está por encima de ellos. Deciden ver contra quien está jugando Miguel. Acceden a la opción para ver las partidas ganadas por un jugador en un periodo de tiempo. El sistema les pide que seleccionen un jugador

entre los que tiene registrados, así como una fecha de inicio y de fin del periodo a mostrar. Eligen el nombre de usuario de Miguel, puesto que lo saben de otras ocasiones en las que han jugado juntos. Como fecha de inicio eligen el lunes pasado, y como fecha de fin hoy mismo. El sistema le ofrece un listado con el resultado. Como están ordenadas de más recientes a más antiguas pueden observar que todas las partidas las está jugando contra Pablo. Está claro que Pablo y Miguel han quedado y están en casa de alguno de los dos. Además, queda claro que Miguel le está dando una paliza al Conecta4.

1.12. Ver las partidas perdidas por un jugador en un periodo de tiempo

Jorge, Juan y Mario están discutiendo sobre quien es el que más veces ha perdido contra otro de ellos. Deciden entrar en la opción de ver las partidas perdidas por jugador. El sistema les pide que seleccionen un jugador entre los que tiene registrados, así como una fecha de inicio y de fin del periodo a mostrar. Empiezan por Juan, así que eligen su nombre de usuario. Como fecha de inicio eligen el sábado anterior, y como fecha de fin hoy mismo. El sistema les ofrece un listado con el resultado, donde pueden ver la fecha de la partida, así como el contrincante ganador de la misma. Tras ver los resultados queda claro que Mario es el que más veces le gana. Tras las risas, deciden ver quién es el azote de Mario, así que repiten la operación para ver las partidas perdidas por Mario en el mismo periodo, cambiando el dato del jugador de entre los que pide el sistema.

1.13. Ver el número de partidas realizadas en un periodo de tiempo

Últimamente el juego de Conecta4 se ha puesto muy de moda en el colegio. Por curiosidad, Mario decide ver la evolución de la popularidad del juego. Para ello decide acceder a la opción que le permite ver el número de partidas realizadas en un periodo de tiempo. El sistema le pide una fecha de inicio y de fin. Mario elige como fecha de inicio el primer día del mes pasado y como fecha de fin hoy mismo. El sistema le muestra la información pedida en una gráfica de líneas. Solo aparece la serie con el número de partidas realizadas por cada día desde la fecha de inicio indicada hasta la fecha de fin. Mario puede observar que a principios del mes pasado ya se jugaba mucho, pero se nota que en la última semana se han incrementado mucho las partidas. Mario decide cambiar la fecha de inicio por el 1 de enero de este año. Automáticamente el sistema cambia la información mostrada. Mario puede ver claramente que el juego se puso de moda a partir de marzo.

1.14. Ver el número de partidas ganadas/perdidas por un jugador en un periodo de tiempo

Jorge, Juan y Mario siguen discutiendo sobre quien es el mejor en el juego. La verdad es que van muy igualados en el ranking, así que deciden ver sus estadísticas de partidas ganadas y perdidas del último mes. Para ello acceden a la opción correspondiente en el sistema. El sistema les indica que deben proporcionar un nombre de usuario y una fecha de inicio y de fin. Juan selecciona su nombre de usuario. Como fecha de inicio del periodo, elige un mes atrás. Como fecha de fin, hoy mismo. El sistema le muestra dos gráficas. La primera es una gráfica de barras apiladas. En la primera serie puede observar las partidas perdidas y en la segunda serie las partidas ganadas. La segunda gráfica le

muestra en una gráfica de barras el número de oponentes diferentes a los que se enfrentó cada día. Los datos se muestran de más antiguos a más modernos. A la vista de los resultados queda claro que Juan ha mejorado un montón, porque al principio perdía casi todas las partidas.

1.15. Cambiar el modo de visualización

Está lloviendo, hace frio y Jose tiene poco que hacer. No puede salir a correr, que es lo que más le gusta los días que no tiene que trabajar. Decide encender el ordenador y jugar un rato al Conecta4. Tras abrir la aplicación se da cuenta que le molesta un poco a la vista el modo por defecto de la aplicación, que tiene un fondo más claro y brillante. Decide cambiar al modo de alto contraste, con fondo oscuro. Accede a la opción correspondiente y cambia el modo por defecto de toda la aplicación.

2. Librería proporcionada para la persistencia de los datos

La persistencia de la información se va a realizar a través de una base de datos *SQLite*[1], a través de una librería de acceso (*Connect4.jar*) que os permitirá almacenar y recuperar toda la información que se necesita persistir. En concreto la aplicación mantendrá un registro de los jugadores y de las partidas realizadas entre ellos en el sistema. La librería gestiona todo el acceso a la base de datos, creándola de forma automática si es necesario dentro del directorio de vuestro proyecto, en un fichero denominado "connect4.db".

Tal y como se explicará a continuación, se os proporciona un método capaz de volcar a un fichero de texto el contenido de la base de datos, por si necesitáis verificar su contenido durante el desarrollo del proyecto. No obstante, existen numerosas aplicaciones que os permiten abrir una base de datos *SQLite* y ver el contenido de sus tablas. Por ejemplo, podéis instalaros la aplicación gratuita *"DB Browser for SQLite"*[2].

2.1. Agregar la librería Connect4.jar a tu proyecto

En primer lugar, descargad y guardad dentro de la carpeta de vuestro proyecto el fichero *lib.zip,* que contiene las librerías necesarias. Descomprimid el contenido en ese directorio, de forma que se creará una nueva carpeta *lib*, con dos ficheros .jar: *Connect4.jar y sqlite-jdbc-3.30.1.jar*. En la Figura 1se muestra el resultado esperado.

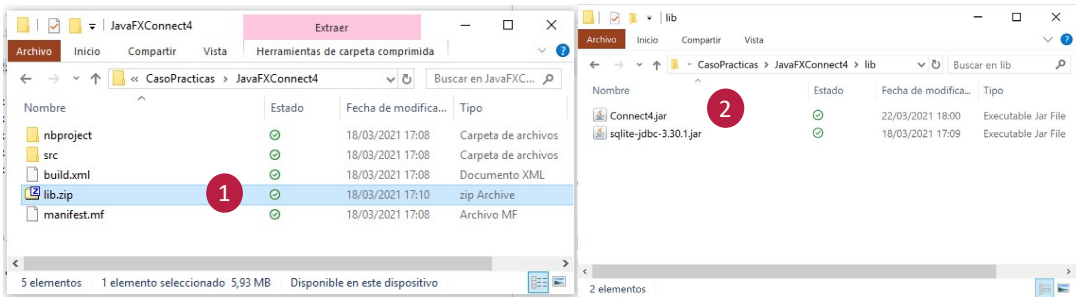

Figura 1. Descomprimir la librería en el directorio del proyecto

[1] https://www.sqlite.org/index.html
[2] https://sqlitebrowser.org/

A continuación, añadiremos estas librerías a vuestro proyecto. Los proyectos creados por Netbeans para aplicaciones FXML tienen una carpeta `Libraries` donde añadir las librerías externas que se desee. Para ello, basta situarse sobre esa carpeta, y desde el menú contextual (botón derecho del ratón), seleccionar la opción `Add JAR/Folder`, tal y como se muestra en la Figura 2.-1.

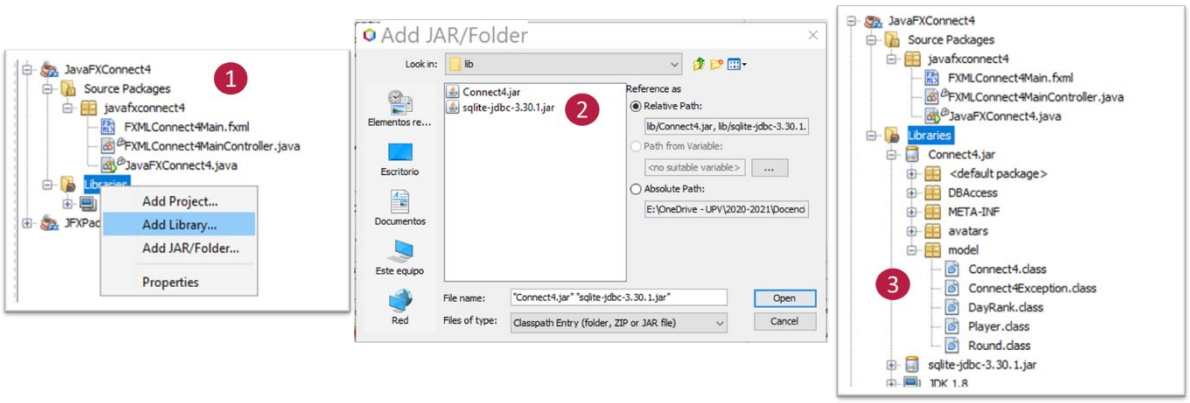

Figura 2. Pasos para añadir las librerías al proyecto

En el diálogo Add Jar/Folder(Figura 2-2), selecciona la carpeta lib que hemos creado anteriormente. Selecciona ambos ficheros .jar y pulsa sobre el botón *Open*. Netbeans añadirá las librerías al proyecto y como puedes observar en la Figura 2-3.

La librería necesita también que descarguéis el fichero avatars.zip y lo descomprimáis dentro de la carpeta src del proyecto, tal y como se muestra en la Figura 3.

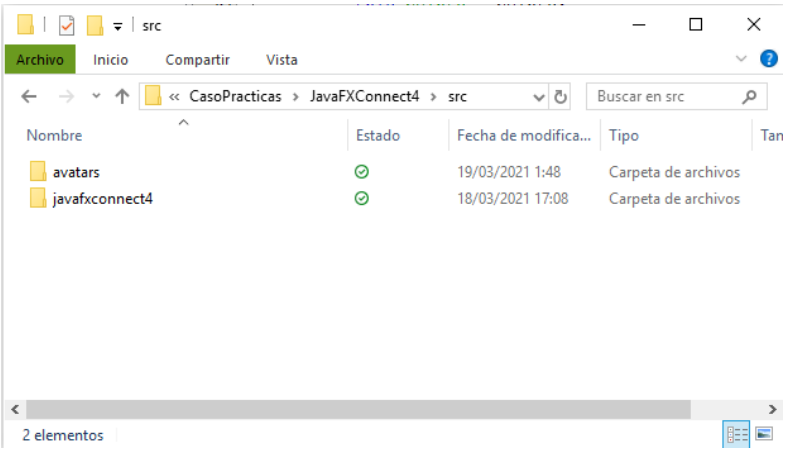

Figura 3. Descomprimir la carpeta avatars dentro de la carpeta src del proyecto

2.2. API proporcionada por *Conecta4.jar*

La librería proporciona diferentes clases, de las que solo necesitaréis usar aquellas que están dentro de la carpeta *model* (Figura 2-3). Si os situáis sobre cualquiera de las clases proporcionadas y hacéis doble clic, se desplegará en la pestaña *Navigator* todos los métodos públicos que proporciona, tal y como podéis apreciar en la Figura 4. A continuación, describiremos los aspectos más importantes de las clases proporcionadas.

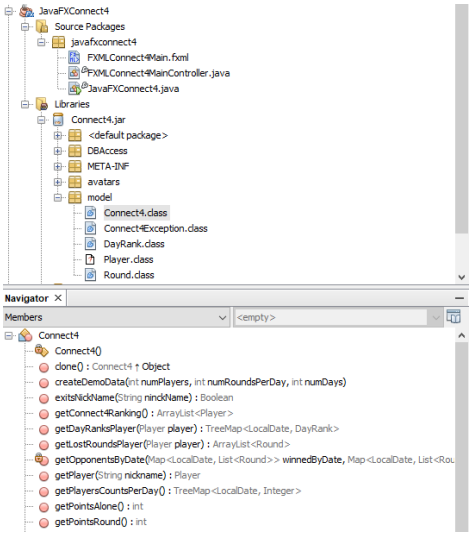

Figura 4. Métodos disponibles en una clase de la librería

2.2.1. Clases del modelo

Connect4

La clase *Connect4* permite la inserción y recuperación de toda la información que necesitaréis en vuestro proyecto. Esta clase se encarga de conectarse a la base de datos (BD) y cargar en memoria toda la información que contiene, así como de almacenar la nueva información y cambios que se realicen. Esta clase implementa el patrón *Singleton*, por lo que sólo puede instanciarse un objeto de esta clase en una aplicación. Para obtener este objeto debéis llamar al método estático getSingletonConnect4(). A partir de este objeto podéis acceder a los métodos que permiten acceder a la información del sistema, que podéis consultar en la Tabla 1.

Por otra parte, la clase Connect4 posee dos atributos a los que se puede acceder a través de sus *setters* y *getters*:

- int **pointsRound**: puntos ganados por un jugador cuando vence a otro jugador. Por defecto, su valor es de 50 puntos.
- int **pointsAlone:** puntos ganados por un jugador cuando vence a la máquina. Por defecto, su valor es de 5 puntos.

Tabla 1. API de la clase Connect4

`public static Connect4 getSingletonConnect4() throws Connect4DAOException`	Crea un objeto de la clase *Connect4*, si no había sido instanciado previamente. Si ya se había instanciado, devuelve el mismo objeto. Cuando lo crea por primera vez, comprueba si existe la BD y carga su información. Si no existe la BD, la crea. Si existe algún problema relacionado con la creación o carga de la información, lanzará una Connect4DAOException.
`public void removeAllData() throws Connect4DAOException`	Elimina todos los datos almacenados en la base de datos. Si existe algún problema relacionado con la conexión a la BD o eliminación de la información, lanzará una Connect4DAOException.
`public Player registerPlayer(String nickName, String email, String password, LocalDate`	Crea un nuevo jugador y lo registra en el sistema, devolviendo el objeto creado (*Player*). El jugador es almacenado con un avatar por defecto. Almacena la

`birthdate, int points) throws Connect4DAOException`	información en la BD. **Es el medio de crear un jugador** que pueda utilizarse en el sistema. Si existe algún problema relacionado con la conexión a la BD o el nickName ya estaba siendo usado, lanzará una Connect4DAOException.
`public Player registerPlayer(String nickName, String email, String password, Image avatar,LocalDate birthdate,int points)`	Crea un nuevo jugador con el avatar proporcionado y lo registra en el sistema, devolviendo el objeto creado (*Player*). Almacena la información en la BD. **Es el medio de crear un jugador** que pueda utilizarse en el sistema. Si existe algún problema relacionado con la conexión a la BD o el nickName ya estaba siendo usado, lanzará una Connect4DAOException.
`public Boolean exitsNickName(String ninckName)`	Devuelve True si el ninckName ya está siendo usado por algún jugador. False en caso contrario.
`public Player loginPlayer(String nickName, String password)`	Devuelve el jugador (Player) cuyo nickName y password coinciden con los datos proporcionados.
`public Player getPlayer(String nickname)`	Devuelve el jugador (Player) cuyo nickName coincida con el dato proporcionado.
`public Round regiterRound(LocalDateTime timeStamp, Player winner, Player loser) throws Connect4DAOException`	Registra una nueva partida en el sistema y lo devuelve (Round). Almacena la información en la BD**. Es el único medio de guardar una partida p**ara que su información sea accesible desde el sistema. Si existe algún problema relacionado con la conexión a la BD o la inserción de la información en la BD, lanza una Connect4DAOException.
`public ArrayList<Player> getConnect4Ranking()`	Devuelve un ArrayList con los jugadores del sistema ordenados de mayor a menor puntuación.
`public ArrayList<Round> getRoundsPlayer(Player player)`	Devuelve un ArrayList con todas las partidas que el jugador dado ha realizado. Este ArrayList estará ordenado por la fecha de las partidas, de más antiguas a más recientes.
`public ArrayList<Round> getWinnedRoundsPlayer(Player player)`	Devuelve un ArrayList con todas las partidas que el jugador dado ha ganado. Este ArrayList estará ordenado por la fecha de las partidas, de más antiguas a más recientes.
`public ArrayList<Round> getLostRoundsPlayer(Player player)`	Devuelve un ArrayList con todas las partidas que el jugador dado ha perdido. Este ArrayList estará ordenado por la fecha de las partidas, de más antiguas a más recientes.
`public TreeMap<LocalDate,DayRank> getDayRanksPlayer(Player player)` 1.14	Devuelve un TreeMap con el cómputo de partidas ganadas y perdidas en cada día, así como el número de contrincantes diferentes a los que se ha enfrentado un jugador. La clave para recuperar cada DayRank es la fecha de ese día. Los datos dentro del TreeMap están ordenados de más antiguos a más recientes.
`public TreeMap<LocalDate,List<Round>> getRoundsPerDay()` 1.9	Devuelve un TreeMap con las partidas realizadas en el sistema agrupadas por el día en el que se realizaron. Así, proporcionando como clave el día, devolverá una lista con las partidas realizadas ese día. Los datos dentro del TreeMap están ordenados de más antiguos a más recientes.
`public TreeMap<LocalDate,Integer> getRoundCountsPerDay()` 1.13	Devuelve un TreeMap con el número partidas realizadas en el sistema agrupadas por el día en el que se realizaron. Así, proporcionando como clave el día, devolverá el número de partidas realizadas ese día. Los datos dentro del TreeMap están ordenados de más antiguos a más recientes.
`public TreeMap<LocalDate,Integer> getPlayersCountsPerDay()`	Devuelve un TreeMap con el número de jugadores que han realizado partidas en el sistema agrupadas por el día en el que se realizaron. Así, proporcionando como clave el día, devolverá el número de jugadores de ese día. Los datos dentro del TreeMap están ordenados de más antiguos a más recientes.
`public void createDemoData(int numPlayers, int numRoundsPerDay, int numDays) throws Connect4DAOException`	Crea información en el sistema. Añade *numPlayers* jugadores. Además, crea *numRoundsPerDay* partidas desde el momento actual hasta *numDays* antes.

	Los datos de cada jugador siguen un patrón, predefinido con nickName y passwords que solo varían en el contador de creación. Además, la fecha de naciemiento es aetatoria, pero siempre mayor de 18 años y tienen la puntuación correspondiente a las partidas en las que se hayan incluido. Por ejemplo, los siguientes datos corresponden al primer jugador creado: *nickName= nickName1* *password= MyPassword!1* *email= nickName1@upv.es* *birthdate= 1963-03-18* *points= 50* *avatar =yes* Las partidas se crean emparejando de forma aleatoria a los jugadores recién creados.
`getConnect4DAO().toTextFile(String filePath)`	Para imprimir a un fichero el contenido actual de la base de datos debéis concatenar la llamada al getConnect4DA() con la llamada al método ToTextFile. Este método crea un fichero con el nombre correspondiente al path proporcionado y vuelva en formato de texto el contenido de la base de datos.

Player

La clase Player permite manejar la información de los jugadores del sistema, ofreciendo métodos *getters* y *setters* para acceder a sus atributos. Los métodos *setters* actualizan la información en la base de datos, **siempre y cuando** el objeto haya sido creado usando el método Connect4.*RegisterPlayer()* (véase la Tabla 1). Por tanto, **no debe utilizarse el constructor público** ofrecido por la clase si se desea que el jugador esté almacenado en la base de datos y sea manejado por el sistema (pueda participar en partidas, incorporarse en las estadísticas, etc.).

Los atributos de Player son:

- String **nickName**: nombre de usuario del jugador. No puede ser actualizado.
- String **email**: dirección de correo electrónico del jugador.
- String **password:** contraseña del jugador.
- Image **avatar**: imagen de avatar del jugador.
- LocalDate **birthdate**: fecha de nacimiento del jugador.
- int **points**: suma de los puntos ganados en las diferentes partidas jugadas en el sistema (tanto contra otros jugadores como contra la máquina).

Además de los setter y getters, la clase *Player* proporciona una serie de métodos útiles para validar información y actualizar la información de sus atributos que os mostramos en la Tabla 2.

Tabla 2. API de la clase Player

`public void plusPoints(int plusPoints) throws Connect4DAOException`	Incrementa los puntos actuales del jugador, actualizando también la información en la BD. Si existe algún problema relacionado con la conexión a la BD o la inserción de la información en la BD, lanza una Connect4DAOException.
`public Boolean checkCredentials(String nickName, String password)`	Devuelve true si el nombre de usuario y la contraseña proporcionados coinciden con la del usuario.
`public String toString()`	Extrae la información del jugador y la devuelve en una cadena de texto con el formato atributo = valor
`public static Boolean checkEmail (String email)`	Método estático que permite comprobar si el dato proporcionado corresponde con una cuenta de correo válida.

`public static Boolean checkNickName(String nickname)`	Método estático que permite comprobar si el dato proporcionado corresponde con un nombre de usuario válido. Un *nickname* es válido si tiene entre 6 y 15 caracteres y contiene letras mayúsculas, minúsculas o los guiones '–' y ' _ ' .
`public static Boolean checkPassword(String password)`	Método estático que permite comprobar si el dato proporcionado contiene una contraseña válida. Una contraseña es válida si: - contiene ente 8 y 20 caracteres - contiene al menos una letra mayúscula - contiene al menos una letra minúscula - contiene al menos un dígito - contiene un carácter especial del conjunto: !@#$%&*()-+= - no contiene ningún espacio en blanco

Round

La clase Round almacena la información sobre las partidas realizadas entre dos jugadores del sistema, ofreciendo métodos *getters* para acceder a sus atributos. Los atributos de un Round no pueden ser modificados tras haberse creado. Además, debe utilizarse el método Connect4.RegisterRound() para obtener el objeto (véase Tabla 1), puesto que es la única forma en la que la partida se almacena en el sistema y pueda recuperarse desde el API de Connect4. Los atributos de la clase Round son:

- LocalDateTime **timeStamp**: día y hora en el que se registrará la partida.
- Player **winner**: jugador ganador de la partida
- Player **loser**: jugador perdedor de la partida

Tabla 3. API de la clase Round

`public String toString()`	Extrae la información de la partida y la devuelve en una cadena de texto con el formato atributo = valor. También extrae a información del jugador ganador y del jugador perdedor, envolviéndola entre llaves.

DayRank

Esta clase recoge información estadística de las partidas realizadas por un jugador en un día concreto. Los objetos de esta clase son utilizados y creados por algunos de los métodos ofrecidos por la clase Connect4 (véase Tabla 1) y sus valores los calcula a partir de la información que se dispone del jugador en el sistema. Por tanto, siempre se ofrece esta información calculada para un jugador en concreto. Los atributos de la clase son:

- LocalDate **date:** día para el que se han calculado los datos recogidos en el objeto.
- int **winnedGames**: partidas ganadas por el jugador en el día *date*
- int **lostGames**: partidas perdidas por el jugador en el día *date*
- int **oponents**: número de oponentes diferentes contra los que jugó en día *date.*

Connect4DAOException

La librería define este nuevo tipo de excepción que puede ser lanzada por aquellos métodos de la librería que acceden a la base de datos para insertar, recuperar o actualizar información.

2.3. Uso de la librería desde el proyecto

Acceso a la librería

Para acceder a los métodos de la librería, es necesario instanciar primero un objeto de la clase *Connect4*, utilizando para ello el método estático *getSingletonConnect4*(). Después, puede obtenerse la información registrada o modificarla a través del API proporcionada. Por ejemplo, en el siguiente código se añade un nuevo jugador:

```
String nickName = "nickName";
String email = "email@domain.es";
String password = "miPassword";
LocalDate birthdate = LocalDate.now().minusYears(18);
int points = 10;
Connect4 connect4 = Connect4.getSingletonConnect4();
Player result = connect4.registerPlayer(nickName, email, password, birthdate, points);
```

Utilización con componentes de la interfaz gráfica

Ya se ha comentado que es posible obtener desde un objeto de tipo *Connect4* toda la información manejada por el sistema, tanto mediante *ArrayLists* como mediante *TreeMaps*. Todas estas listas y mapas pueden conectarse a los elementos gráficos de la interfaz, para lo que es necesario envolverlas en listas observables ***ObservableList o MapLists***.

A modo de ejemplo, se muestra el código necesario para conectar una *TableView* a la lista de jugadores ordenados por puntuación que proporciona la librería.

```
Connect4 connect4 = Connect4.getSingletonConnect4();
public ArrayList<Player> getConnect4Ranking();

ObservableList<Player> observablePlayers;
observablePlayers = FXCollections.observableList(connect4. getConnect4Ranking());

TableView<Player> tableViewPlayers;
tableViewPlayers.setItems(observablePlayers);
```

Acceso a la información de los TreeMap

Los métodos que devuelven TreeMap en la librería usan como clave un campo de tipo LocalDate, que se refiere a la fecha a la que corresponden el valor para esa entrada. Por ejemplo, en el siguiente código se obtiene las estadísticas de un jugador (*winner*) para hoy:

```
TreeMap<LocalDate,DayRank>dayRanksWinner = connect4.getDayRanksPlayer(winner);
LocalDate date = LocalDate.now();
if(dayRanksWinner. containsKey(date){ //Checks if the key entry exists
   DayRank dayrank = dayRanksWinner.get(date);
}
```

Por otra parte, los métodos keySet() devuelven una colección con todas las claves almacenadas den el TreeMap. De forma similar, el método values() devuelve en una colección todos los valores almacenados en el TreeMap (perdiendo la clave). Por ejemplo, el siguiente código guarda todos los datos estadísticos de un jugador en una lista observable:

```
TreeMap<LocalDate,DayRank>dayRanksWinner = connect4.getDayRanksPlayer(winner);
LocalDate date = LocalDate.now();
TreeMap<LocalDate,DayRank>dayRanksWinner = connect4.getDayRanksPlayer(winner);
ArrayList<DayRank> dayRanks = new ArrayList<>( dayRanksWinner.values());
ObservableList observableList = FXCollections.observableList(dayRanks);
```

Existen múltiples maneras de iterar un TreeMap[3], una forma muy rápida es utilizar lambdas y el método forEach de la propia estructura. Por ejemplo, el siguiente código recorre las partidas realizadas en el sistema, y por cada día muestra un mensaje:

```
TreeMap<LocalDate,List<Round>> roundsPerDay=connect4.getRoundsPerDay();
roundsPerDay.forEach((LocalDate date,List<Round>rounds)->{
    System.out.print("In "+date + " the player has played " + rounds.size() +" times\n");
  });
```

Carga de imágenes almacenadas en la base de datos en una ImageView

Todos los jugadores tienen almacenado una imagen de su avatar en su perfil. Para cargarla en una *Image View* solo es necesario recuperar dicho campo del objeto correspondiente. Por ejemplo:

```
   Connect4 connect4 = Connect4.getSingletonConnect4();
   Player player = connect4.loginPlayer(nickName, password);
   ImageView ivAvatar = new ImageView();
   ivAvatar.imageProperty().setValue( player.getAvatar());
```

3. Ayudas a la programación

3.1. Carga de imágenes desde disco duro

Existen diversas formas de cargar una imagen desde el disco duro y mostrarla en una *ImageView*:

1. La imagen está en un subdirectorio del directorio src del proyecto, por ejemplo, llamado images:

```
String url = File.separator+"images"+File.separator+"woman.PNG";
Image avatar = new Image(new FileInputStream(url));
myImageView.imageProperty().setValue(avatar);
```

2. La imagen está en cualquier parte del disco duro y tenemos el *path* completo de la misma:

```
String url = "c:"+File.separator+"images"+File.separator+"woman.PNG";
Image avatar = new Image(new FileInputStream(url));
myImageView.imageProperty().setValue(avatar);
```

3.2. Animar objetos

Para diseñar el tablero de juego tenéis varias opciones. Podéis usar tantos círculos como casillas, cambiándoles el color dependiendo de la ficha jugada y jugador. También podéis animar un objeto círculo de forma que se mueva por la pantalla. Por ejemplo, el siguiente código desplaza 50 posiciones un círculo en su eje y.

[3] Podéis consultar la siguiente dirección para las diferentes maneras de iterar un Map: https://www.geeksforgeeks.org/iterate-map-java/

```
        double currentYPostion = blueCircle.translateYProperty().getValue();
        blueCircle.translateYProperty().setValue(currentYPostion+50);
```

3.3. Manejo de fechas y del tiempo

En la aplicación se deben gestionar atributos de tipo *LocalDateTime, LocalDate* y de tipo *DateTime*. A continuación, os explicamos algunos métodos de utilidad.

Obtención de la semana del año a la que pertenece una fecha

El siguiente código obtiene la semana a la que pertenece el día de hoy, así como el número del día de la semana (1 para lunes, 2 para martes, etc.)

```
WeekFields weekFields = WeekFields.of(Locale.getDefault());
int currentWeek = LocalDate.now().get(weekFields.weekOfWeekBasedYear());
int numDayNow=LocalDate.now().get(weekFields.dayOfWeek());
```

Creación de una fecha o un campo de tiempo

Consulta el API de *LocalDate* y *LocalTime* para conocer todos los métodos que proporciona para crear un objeto de sus clases, pero algunos que te pueden resultar útiles son:

```
LocalDate sanJose = LocalDate.of(2019, 3,19);
LocalTime mascleta = LocalTime.of(14,0);
LocalDateTime sanJoseMascleta = LocalDateTime.of(2019,3,19,14,0);
LocalDate sanJose2 = sanJoseMascleta.toLocalDate();
LocalTime mascleta2 = sanJoseMascleta.toLocalTime();
```

Actualizando una fecha

Es posible incrementar o decrementar días, meses, años, minutos, horas, etc. a los campos de tipo *LocalTime*, *LocalDate*, o *LocalDateTime* usando su propia API. Por ejemplo, el siguiente código incrementa en siete días la fecha actual, guardando el resultado en una nueva variable. También incrementa en un mes la fecha actual, salvando la información en otra variable. Finalmente, incrementa el tiempo actual en 90 minutos, guardando el resultado en otra variable de tipo *LocalTime*.

```
LocalDateTime nextWeekDay= LocalDateTime.now().plusDays(7);
LocalDateTime nextMontDay = LocalDateTime.now().plusMonths(1);
LocalTime endedTime = LocalTime.now().plusMinutes(90);
```

3.4. Configurar DatePicker

Los componentes DatePicker permiten seleccionar al usuario una fecha, pudiendo acceder al valor seleccionado a través de su propiedad valueProperty(). Es posible configurar la forma en la que se visualiza por defecto cada día del calendario, siendo posible deshabilitar algunos días, cambiar el color de fondo, etc. La forma en la que se configura esta visualización es similar a la que empleamos para configurar una ListView o una TableView. La diferencia es que en este caso debemos extender la clase DateCell, y usar el método setDayCellFactory. Por ejemplo, el siguiente código configura un DatePicker para que se inhabiliten los días anteriores al 1 de Marzo de 2020 (ver Figura 5).

```
dpBookingDay.setDayCellFactory((DatePicker picker) -> {
            return new DateCell() {
                @Override
                public void updateItem(LocalDate date, boolean empty) {
                    super.updateItem(date, empty);
                    LocalDate today = LocalDate.now();
                    setDisable(empty || date.compareTo(today) < 0 );
                }
            };
        });
```

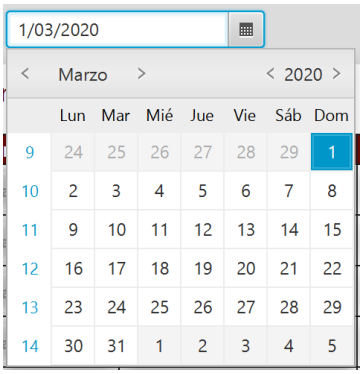

Figura 5. DataPicker configurado para inhabilitar días en el pasado

4. Instrucciones de Entrega

Este proyecto se realizará en dos entregas. En cada entrega se deberán completar los casos de uso que se piden para la misma. En las sesiones de prácticas iremos viendo los conceptos que necesitaréis conocer para completar los requisitos pedidos en cada entrega.

Todos los grupos tendrán la misma fecha entrega. Para cada una de las entregas, deberéis:

- Exportad el proyecto Netbeans a un fichero zip (opción `Fichero>Exportar Proyecto> A Zip`).
- Un único miembro del grupo sube el fichero zip a la tarea correspondiente, incluyendo en el campo de comentarios los nombres de los miembros del grupo.

4.1. Entrega 1

Para la primera entrega deberéis entregar un proyecto JavaFX que contemple los siguientes casos de uso:

- Autentificarse. Debe cumplirse el escenario de uso descrito en la sección 1.2.
- Cerrar sesión. Debe cumplirse el escenario de uso descrito en la sección 1.3.
- Jugar contra la máquina. Debe cumplirse el escenario de uso descrito en la sección 1.4. Además, si el jugador resulta ganador, sus puntos serán modificados sumándole la puntuación correspondiente a ganar contra la máquina.
- Jugar contra otro jugador. Debe cumplirse el escenario de uso descrito en la sección 1.5. Al finalizar la partida, esta deberá ser registrada adecuadamente en

el sistema. Además, el jugador ganador verá modificados sus puntos sumándole la puntuación correspondiente a ganar contra otro jugador.

- Recordar datos de autenticación. Debe cumplirse el escenario de uso descrito en la sección 1.7

Fecha de entrega: 30 de abril

4.2. Entrega II

Para la primera entrega deberéis completar el proyecto JavaFX anterior corrigiendo los errores detectados, contemplando el resto de casos de uso:

- Registrarse en el juego. Debe cumplirse el escenario de uso descrito en la sección 1.1
- Modificar perfil. Debe cumplirse el escenario de uso descrito en la sección 1.6.
- Mostrar ranking. Debe cumplirse el escenario de uso descrito en la sección1.8.
- Ver las partidas realizadas en el sistema en un periodo de tiempo. Debe cumplirse el escenario de uso descrito en la sección 1.9.
- Ver las partidas realizadas por un jugador en un periodo de tiempo. Debe cumplirse el escenario de uso descrito en la sección 1.10.
- Ver las partidas ganadas por un jugador en un periodo de tiempo. Debe cumplirse el escenario de uso descrito en la sección 1.11 .
- Ver las partidas perdidas por un jugador en un periodo de tiempo. Debe cumplirse el escenario de uso descrito en la sección 1.12.
- Cambiar el modo de visualización. Debe cumplirse el escenario de uso descrito en la sección1.15. Para implementar estar funcionalidad será necesario emplear un fichero CSS. En la práctica 5 veremos cómo puede hacerse.
- Ver el número de partidas realizadas en un periodo de tiempo. Debe cumplirse el escenario de uso descrito en la sección 1.13. En la práctica 6 veremos cómo podemos crear gráficas.
- Ver el número de partidas ganadas/perdidas por un jugador en un periodo de tiempo. Debe cumplirse el escenario de uso descrito en la sección 1.14. En la práctica 6 veremos cómo podemos crear gráficas.

Fecha de entrega: 21 de mayo

5. Evaluación

- Aquellos proyectos que no compilen o que no se muestren la pantalla principal al arrancar se calificarán con un cero.
- Se deberán incluir los diálogos de confirmación, errores, etc. que se considere necesario.

- Para evaluar el diseño de la interfaz de la aplicación se tendrán en consideración **las directrices estudiadas en clase de teoría**.
- Debe ser posible redimensionar la pantalla principal, cuyos controles se ajustarán adecuadamente para usar el espacio disponible (usa los contenedores vistos en clase: *VBox*, *HBox*, *BorderPane*, *GridPane*, etc.).
- Se aplicará la Normativa de Integridad Académica de la UPV y la Normativa de Honestidad Académica de la ETSInf. En la asignatura existen herramientas anti plagio.