ENTREGA 1. PRÁCTICAS

Club PaddleExperience

Interfaces Persona Computador

IPC – DSIC

UPV

Curso 2019-2020

Índice

1. Caso de Estudio

El club *PaddleExperience* desea ofrecer una nueva aplicación que permita a sus socios gestionar el alquiler de las pistas. La aplicación se adaptará al tamaño del dispositivo tamaño del dispositivo donde se ejecute.

El club dispone de 4 pistas, ofreciendo su alquiler en tramos de hora y media desde las 9 de la mañana hasta las 2100h inclusive (las instalaciones cierran a las 22:45h).

Para poder reservar, los usuarios tendrán que haberse registrado previamente, indicando sus datos de contacto Desde el club desean que la reserva sea un proceso sencillo e intuitivo, de forma que el usuario pueda reservar rápidamente sin necesidad de aportar demasiada información. Además, como saben que los usuarios tienen preferencias por unas pistas u otras, el proceso deberá permitirles ver la disponibilidad de cada una de las pistas del club.

Por otra parte, muchos usuarios no recuerdan qué pista tenían reservada, a veces porque no la reservaron ellos directamente, sino algún compañero. Por ello, el club desea que la aplicación informe a los miembros de quién reservó cada pista mediante un nombre de usuario en el sistema, a fin de preservar su privacidad.

Finalmente, las reservas solo podrán eliminarse con 24h de antelación, para evitar que las pistas se queden sin utilizar a última hora.

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. Registro de socio

Sonia ha visto que en el club *PaddleExperience* es posible reservar pistas en cualquier momento usando una aplicación gratuita. Como siempre se le olvida llamar en horas de oficina, le parece genial y decide registrarse en su aplicación. Tras descargarla, Sonia accede a la opción de registro, que está disponible nada más arrancar la aplicación. Desde la opción se abre un formulario donde Sonia puede introducir sus datos de contacto:

- Nombre
- Apellidos
- Teléfono
- Login (usado para acceder a la aplicación y para ser mostrado a los demás usuarios. No puede contener espacios. No puede repetirse)
- Password (cualquier combinación de letras y números con más de 6 caracteres)
- Número de la tarjeta de crédito (16 números) y código de seguridad SVC (3 números) (opcional)
- Imagen de perfil (opcional)

Tras introducir todos sus datos de contacto y elegir una raqueta de pádel como imagen de perfil, pulsa el botón aceptar del formulario. El sistema comprueba que todos los datos son correctos y no se ha introducido ningún dato con un formato no permitido. Como todos los datos son correctos, informa a Sonia de que ha sido añadida correctamente como usuaria y que debe autenticarse para poder reservar pistas.

1.2. Autenticarse

Juan acaba de hablan con su amigo José y han decidido reservar una pista para el jueves por la tarde, por lo que Juan accede a la aplicación del club *PaddleExperience* para ver si hay pista disponible. Nada más abrir la aplicación, accede a la opción de autenticarse, donde aparece un formulario donde introduce su nombre de usuario y contraseña. Tras pulsar el botón de acceder, el sistema comprueba si el usuario está registrado en el sistema.

Juan se ha confundido al introducir el *password*, por lo que el sistema no lo encuentra y le informa que no está registrado en el sistema, permitiéndole introducir los datos de nuevo. Juan vuelve a introducir sus datos, esta vez de forma correcta. El sistema lo autentifica, permitiéndole el acceso al resto de funcionalidades.

1.3. Reservar una pista

Mario, quiere reservar una pista. Tras autenticarse, Mario accede a la opción de reservar pista. Mario puede ver todas las pistas disponibles y reservadas para hoy. Sin embargo, la reserva la quiere para mañana, así que pulsa en la opción para cambiar de día. Tras cambiar al día siguiente, observa que la pista 3 está libre a las 18:30, por lo que la selecciona y accede a la opción de reservar.

Tras reservarla, el sistema muestra la pista 3 como reservada desde las 18:00 hasta las 19:30. Además, se indica que es el dueño de la reserva mostrando su *login*. Como Mario indicó la tarjeta de crédito cuando se registró, el sistema de gestión cobra la reserva[1], por lo que desde la aplicación de reservas se guarda que la reserva ya ha sido pagada.

1.4. Ver mis reservas

Miguel tiene pista reservada para esta tarde a las 15:30, pero no recuerda exactamente el número de la pista. Como va a ir justito de tiempo, prefiere comprobarlo antes de ir y no tener que preguntar en el club al llegar. Accede a la aplicación del club *PaddleExperience*, y tras autentificarse, accede a la opción *Mis Reservas.* En la pantalla le salen las 10 últimas pistas que ha reservado, de forma que en primer lugar puede ver la más reciente. Para cada reserva puede consultar para qué día es, a qué pista corresponde, el horario de inicio de la reserva y el horario en el que tiene que abandonar la pista, así como si la reserva ha sido pagada.

1.5. Eliminar una reserva

Marisa tenía una pista reservada para el jueves por la tarde, para ir a jugar con sus amigas al club *PaddleExperience*. Sin embargo, hoy martes le ha dicho su hijo que le han puesto un partido de fútbol el jueves por la tarde, y que le tiene que llevar porque es en otro colegio. Un poco molesta, Marisa avisa a sus compañeras de partida por teléfono que tienen que cancelar la partida.

[1] Se considera que la gestión y cobro de las reservas se realiza por una aplicación existente previamente.

Corriendo, Marisa accede a la aplicación del club para cancelar la reserva, de forma que no se la cobren porque aún está dentro del margen de 24h que le permite el club. Tras autentificarse, accede a la opción de sus reservas. En primer lugar, le aparece la reserva que tiene pendiente, así que la selecciona y pulsa en eliminar.

El sistema comprueba que la fecha de la reserva es posterior a la fecha actual por más de 24h, por lo que puede eliminar la reserva y poner la pista libre para ese horario.

1.6. Ver disponibilidad de las pistas

Juan y Marcos han quedado para jugar con sus compañeros habituales de partida, quienes se encargaron de reservar la pista para hoy. Juan y Marcos han quedado para ir juntos, pero ninguno de los dos recuerda en qué pista estaba la reserva. En un momento, Marcos accede a la aplicación del club, donde hay una opción que no requiere autenticarse donde es posible ver la disponibilidad de las pistas para hoy. En un momento, busca en el tramo de la tarde el *login* de su contrincante "junior33", viendo que les toca la pista 2, a las 17:00.

2. Persistencia de los datos

La persistencia de la información se realizará a través de un fichero XML, utilizando la librería JAXB de java. En concreto, se os ha proporcionado una librería de acceso (*clubDBAccess.jar*) capaz de almacenar y recuperar toda la información del club de pádel: socios, reservas de pistas, etc. Para que la librería funcione correctamente, el archivo XML *clubDB.xml* debe ubicarse en el home del usuario, que en sistemas Windows se encuentra en:

```
C:\users\userName
```

Donde `userName` es el *nick* del usuario con el que se ha realizado el inicio de sesión en Windows.

Si trabajáis en los equipos el laboratorio o accedéis desde el escritorio virtual, el home del usuario se ubica directamente en:

```
W:\
```

El archivo *clubDB.xml* no debe manipularse manualmente, sino a través del API que la librería *clubDBAccess.jar* proporciona. Como punto de partida, se os proporciona un archivo *clubDB.xml* con las cuatro pistas del club, algunos socios y reservas. Añadir la librería *clubDBAccess.jar* al proyecto.

En primer lugar, descarga y guarda en la carpeta de tu proyecto el fichero *clubDBAccess.jar,* que contiene la librería de acceso.

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 1.

Tras seleccionar la opción, aparecerá un diálogo donde se debe seleccionar el fichero jar que contiene la librería, que en este caso debería estar almacenado en la carpeta del proyecto (donde se había descargado). Tras aceptar, la librería se carga, mostrando todas las clases disponibles (Figura 2).

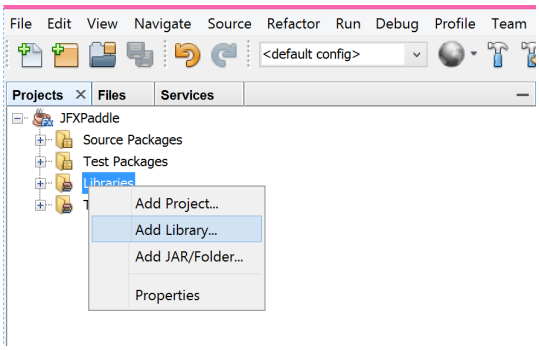

Figura 2. Seleccionar la opción de incluir la librería

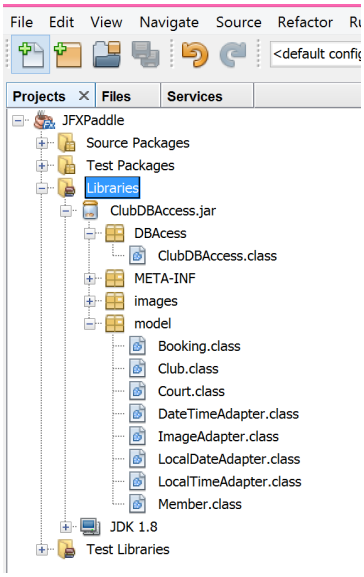

Figura 1. Librería ClubDBAccess.jar incorporada al proyecto

2.1. API proporcionada por *clubDBAccess.jar*

Tal y como se ha mencionado anteriormente, no hay que manipular directamente el fichero XML que contiene la información del club, sino acceder a él a través del API proporcionado. Así, la librería proporciona una clase de acceso denominada *ClubDBAccess*, que permite cargar la información del club desde el fichero, así como almacenarla en él. Esta clase implementa el patrón *Singleton*, por lo que sólo puede instanciarse un objeto de esta clase en una aplicación.

Tras obtener el objeto de la clase *ClubDBAccess*, es posible acceder a toda la información del cLub a través de diferentes métodos públicos, que pueden consultarse en la Tabla 1.

Tabla 1. API de la clase ClubDBAccess

public static ClubDBAccess getSingletonClubDBAccess()	Crea un objeto de la clase *ClubDBAccess*, si no había sido instanciado previamente. Si ya se había instanciado, devuelve el mismo objeto. Cuando lo crea por primera vez, comprueba si existe el archivo XML en el home del usuario y carga la información del club. Si no existe el

	fichero, crea un nuevo fichero XML vacío, creando un club con la información mínima necesaria.
public boolean saveDB()	Guarda el estado actual del club en el fichero XML, almacenando los miembros, las pistas y las reservas. Devuelve TRUE si los datos son grabados correctamente. FALSE en caso contrario.
public String getClubName()	Devuelve el nombre del club
public int getClubBookingDuration()	Devuelve los minutos que dura una reserva en el club.
public int getClubBookingSlots()	Devuelve el número de reservas que pueden realizarse en una pista cada día.
public ArrayList<Member> getMembers()	Devuelve un *ArrayList* con todos los miembros del club.
public ArrayList<Court> getCourts()	Devuelve un *ArrayList* con todas las pistas del club.
public ArrayList<Booking> getBookings()	Devuelve un *ArrayList* con todas las reservas realizadas. Las reservas se realizan para un día y hora concretos. Este *ArrayList* se devuelve ordenado por las reservas para los días más antiguos a los más modernos.
public ArrayList<Booking> getUserBookings(String login)	**Copia** todas las reservas realizadas por el usuario con el *login* proporcionado en un *ArrayList* y lo devuelve. Si se modifica este *ArrayList*, **la lista original del club** de reservas **NO CAMBIA**.
public ArrayList<Booking> getCourtBookings(String courtName, LocalDate madeForDay)	**Copia** todas las reservas realizadas para una pista y día concretos en un *ArrayList* y lo devuelve. Si se modifica este *ArrayList*, **la lista original del club** de reservas **NO CAMBIA**.
public ArrayList<Booking> getForDayBookings(LocalDate forDay)	**Copia** todas las reservas realizadas para día concreto en un *ArrayList* y lo devuelve. Si se modifica este *ArrayList*, la **lista original del club** de reservas **NO CAMBIA**.
public boolean existsLogin(String login)	Devuelve True si existe algún miembro del club que esté utilizando el login dado, Falso en otro caso.
public Member getMemberByCredentials(String login, String password)	Devuelve el objeto *Member* correspondiente al miembro del club cuyo *login* y *password* coinciden con los dados. Si no existe, devuelve *null*.
public Court getCourt(String name)	Devuelve el objeto *Court* correspondiente a la pista cuyo nombre sea igual al nombre dado.
public boolean hasCreditCard(String login)	Devuelve si el miembro del club correspondiente al *login* proporcionó los datos de su tarjeta de crédito en el registro. Si no existe o no proporcionó los datos, devuelve false.

2.2. Clases del modelo

Club

La clase ClubDB tiene como atributo un objeto de la clase Club, que se corresponde con el nodo raíz de la persistencia de información en el XML. La clase Club posee toda la información del sistema. Tiene una serie de atributos a los que se puede acceder a través de sus *setters* y *getters*. Sin embargo, os recomendamos usar siempre que sea

posible los métodos del API de acceso definidos en la Tabla 1 para acceder a la información.

En concreto, Club posee los siguientes atributos:

- String **name**: nombre del club
- int **bookingDuration**: duración en minutos de una reserva. 90 min por defecto, tal y como se indica en el enunciado.
- int **bookingSlots**; número de posibles reservas que se pueden realizar en un día y pista. 9 por defecto, tal y como se indica en el enunciado.
- **courts**: lista con las pistas del club. Por defecto se proporcionan 4 pistas.
- **members**: lista con los miembros del club.
- **Bookings**: lista con las reservas realizadas en el club.

Member

Almacena toda la información sobre un miembro del club, ofreciendo métodos *setter* y *getter* para acceder a los datos que se gestionan. Estos datos son:

- String **name**: Nombre del miembro.
- String **surname**: apellido del miembro.
- String **telephone**: cadena que contiene el teléfono del miembro.
- String **login**: credencial que usará el usuario para conectarse en el sistema.
- String **password**: contraseña que usará el usuario para conectarse en el sistema.
- String **creditCard**: cadena que contiene los 16 dígitos de la tarjeta de crédito del miembro. Puede estar vacía si no se proporcionó en el registro.
- String **svc**: cadena que contiene los 3 dígitos del código de verificación de la tarjeta.
- Image **image**: avatar con la imagen del miembro. Puede estar vacío si no se proporcionó en el registro usuario.

Booking

Almacena la información de una reserva, ofreciendo métodos *setter* y *getter* para acceder a los datos que se gestionan. Estos datos son:

- LocalDateTime **bookingDate**: fecha en la que el usuario realiza la reserva.
- LocalDate **madeForDay**: fecha para la que se hace la reserva, es decir, el día en el miembro del club quiere jugar.
- LocalTime **fromTime**: Hora de inicio de la reserve.
- Boolean **paid**: la reserva ha sido pagada con la tarjeta de crédito.
- Court **court**: pista del club para la que se ha reservado.
- Member **member**: miembro del club que ha reservado la pista.

Court

Almacena la información de una pista del club. En concreto solo se mantiene su nombre, que es accesible desde los correspondientes métodos *setter* y *getter*:

- String **name**: cadena que contiene el nombre del club.

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 *ClubDBAccess*, utilizando para ello el método estático *getSingletonDBAccess*(). Después, puede obtenerse la información registrada o modificarla a través del API proporcionada. Por ejemplo, en el siguiente código se añade un nuevo socio a la lista de miembros:

```
ClubDBAccess clubDBAcess;

clubDBAcess = ClubDBAccess.getSingletonClubDBAccess();

clubDBAcces.getMembers().add(

        new Member("John","Doe","601234567","jDoe","myPass1234","","",null));
```

Utilización con componentes de la interfaz gráfica

Ya se ha comentado que es posible obtener desde un objeto de tipo *ClubDBAccess* diferentes *ArrayLists* con los datos que debe manejar el sistema de reservas, como son miembros, las pistas del club, así como las reservas realizadas en ellas. Todas estas listas pueden conectarse a los elementos gráficos de la interfaz, para lo que es necesario envolverlas en listas observables ***`ObservableList`***, de forma que las nuevas inserciones y eliminaciones de elementos se transmitan a los *ArrayLists* originales del objeto *ClubDBAccess*. **En ningún caso** debe utilizarse una `ObservableArrayList`, puesto que los cambios no se reflejan en las listas originales.

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

```
ClubDBAccess clubDBAcess;

clubDBAcess = ClubDBAccess.getSingletonClubDBAccess();


ObservableList<Booking> observableBookings;
observableBookings = FXCollections.observableList(clubDBAcess. clubDBAcess.getBookings());

TableView<Booking> tableViewBookngs;
tableViewBookngs.setItems(observableBookings);
```

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

Los socios pueden tener almacenada una foto en su ficha. Para cargarla en una *Image View* solo es necesario recuperar dicho campo del objeto correspondiente. Por ejemplo:

```
   ClubDBAccess clubDBAcess = ClubDBAccess.getSingletonClubDBAccess();
   int index = 0;
   Member member = clubDBAcess.getMembers().get(index);
   ImageView imagePhoto = new ImageView();
   imagePhoto.imageProperty().setValue( member.getImage());
```

Persistencia de los cambios

Para que todos los cambios realizados por el programa en los datos se persistan, es decir, se almacenen en el fichero XML, es necesario llamar al método *saveDB()*. Este método cuesta mucho tiempo, por lo que es aconsejable añadir la llamada cuando se

cierre la aplicación. Así, el siguiente código añade un manejador para la ventana principal, que debería añadirse en el método *start* de la aplicación:

```
stage.setOnCloseRequest((WindowEvent event) ->{
      Alert alert = new Alert(AlertType.INFORMATION);
      alert.setTitle(clubDBAcess.getClubName());
      alert.setHeaderText("Saving data in DB");
      alert.setContentText("The application is saving the changes into the database. This action
can expend some minutes.");
      alert.show();
      clubDBAccess.saveDB();
});
```

3. Ayudas a la programación

3.1. Carga de imágenes desde disco duro

Los miembros del club pueden almacenar en su ficha de datos personales. 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. 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(2020, 3,19);
LocalTime mascleta = LocalTime.of(14,0);
```

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);
```

3.3. 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 3).

```
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 );
                }
            };
        });
```

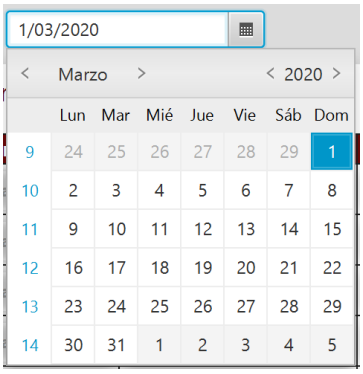

Figura 3. DataPicker configurado para inhabilitar días en el pasado

4. Instrucciones de Entrega

El proyecto debe implementar los escenarios de usos planteados en el punto Caso de Estudio (página 2). Debe diseñarse una interfaz usable, que permita realizar las funcionalidades reflejadas en los diferentes casos de uso.

Debido a las limitaciones del formato XML no se contemplan ninguna funcionalidad de modificación. Por otra parte, tampoco deben eliminarse ningún miembro o pista, puesto que podrían crearse inconsistenticas en los datos. Además, no se puede eliminar una reserva que esté situada en el pasado, puesto que se supone que ya se realizó.

Con respecto a la entrega:

- Exporta 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.
- La fecha de entrega para todos los grupos es el **21 de abril de 2020**

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.).