|  |
| --- |
| Aizpea Babaze Aizpurua |
| Aplicación de gestión de AAPPMM |
| Prácticas de empresa con APTES |
|  |
| **Aizpea Babaze Aizpurua** |
| **29/07/2016** |

|  |
| --- |
| Esta aplicación tiene como objetivo facilitar la gestión de de las asociaciones de personas mayores de Donostia - San Sebastián. Con una interfaz fácil y sencilla, la junta directiva de las asociaciones podrá gestionar la asociación a la vez que se inician en el mundo de la informática. |

Tabla de contenido

[Bases de la aplicación 3](#_Toc457558657)

[Base de datos 3](#_Toc457558658)

[Interface 3](#_Toc457558659)

[Operaciones 4](#_Toc457558660)

[Socios 4](#_Toc457558661)

[Actividades 5](#_Toc457558662)

[Asociación 6](#_Toc457558663)

[Terceros 7](#_Toc457558664)

[Listados 7](#_Toc457558665)

[Libro diario 7](#_Toc457558666)

[Calendario 7](#_Toc457558667)

# Bases de la aplicación

El código de la aplicación se encuentra en un repositorio de gitHub totalmente público **-** [**https://github.com/aepzia/gestionhator**](https://github.com/aepzia/gestionhator) **[[1]](#footnote-1) -** En ella, se encuentran todos los pasos que se han ido dando.

## Base de datos

La base de datos comenzó siendo en formato de Access debido a su uso en muchas asociaciones de personas mayores. Pero aquello suponía que las 19 asociaciones, y cualquier persona que quiera hacer uso de esta aplicación, necesiten licencia de Microsoft Access, la cual no es accesible para todo el mundo y supondría un gasto innecesario para el ayuntamiento de Donostia.

Finalmente, se decidió pasar a una base de datos “h2”. Para hacer uso de esta base de datos se necesitará la librería adecuada, en este caso se incluyo el fichero “h2-1.4.192.jar”. La base de datos se crea en *datuBasea.java*. Como se puede ver, primero se inicializa la conexión con la base de datos, que se debe realizar cada vez que se quiera hacer uso de la base de datos.

Class.*forName*("org.h2.Driver");

Connection con = DriverManager.*getConnection*("jdbc:h2:<ruta>", "", "" );

Statement stmt = con.createStatement();

Esta base de datos funciona con comandos sql que se ejecutan con strings mediante el variable stmt. ( <https://docs.oracle.com/javase/tutorial/jdbc/basics/processingsqlstatements.html> )

Recuerda cambiar la ruta de la base de datos de todo el programa, en caso de que sea necesario.

Existen 6 tablas en esta base de datos: asociación, socios, actividades, socio-actividades, monitores, cesionarios. Todas ellas contienen las características de su clase como por ejemplo los socios. La tabla de socio-actividades tiene tres columnas que llevan el identificador del socio (DNI), el identificador de la actividad y un booleano que guardara si esta actividad esta pagada por el socio o no. La tabla de actividades también guardara el identificador del monitor correspondiente.

## Interface

Esta aplicación usa la librería de *jaxaFx* que tiene ficheros de formato *fxml* que definen su aspecto. Para editar este tipo de archivos se puede utilizar bien *scene builder* para hacerlo de manera visual o bien editando el propio archivo *fxml* .

Por cada archivo *fxml* se deberá definir una clase controlador, que será la clase que manipulara esta interfaz. Para definir el controlador es suficiente con definir el atributo “fx:controller=<clase>” tal y como se ve en los archivos ya existentes de este tipo. En este archivo debemos de definir el id de los elementos que querremos usar en la clase controladora, usando el atributo “id:fx=<id>”. También podemos añadir la función que queremos que ejecute en función de un elemento ( ej. *Onclick=”#gorde”)* .

En la clase controladora, se deberá importar el archivo de la interface que queramos usar. Y es importante definir también todas las variables y funciones junto a “@FXML”, usando el identificador definido en el fichero que cargamos; sino, recibiremos un error de *nullPoiterException* al intentar acceder a él.

Para iniciar los variables definidos, se deben de iniciar en la función **@FXML protected void initialize()**. El fichero *fxml* se importara en la función **public void start(Stage primaryStage)** de esta manera:

setPrimaryStage(primaryStage);

page = (AnchorPane) FXMLLoader.*load*(getClass().getResource("../itxura/listaCesionarios.fxml"));

Scene scene = **new** Scene(page); scene.getStylesheets().add(getClass().getResource("../itxura/style.css").toExternalForm());

primaryStage.setScene(scene);

primaryStage.setTitle("Añadir un nuevo socio");

primaryStage.show();

Es importante mantener el *primaryStage* para pasar de una ventana a otra. Para ello, usaremos una variable global en la clase, que pasaremos a la función *start* de la siguiente ventana.

# Operaciones

## Socios

Aparecerá una lista - listaSociosGUI - de todos los socios que podrán ser ordenados por número de socio y en orden alfabético, y se podrá buscar al socio con cualquier dato que aparece en pantalla.

TODO: Poder ver los socios dados de alta o bien dados de baja. Y que aparezcan en otro color si el socio tiene algo pendiente de pago (actividades, cuota…). Otra opción que se me ocurrió, fue que hubiese una única lista, donde los socios que estén dados de baja (quienes tengan fecha de baja) aparezcan tachados.

En esta pantalla hay cuatro opciones:

* Dar socio de alta: Aparecerá una nueva pantalla - nuevoSocioGUI -, para dar de alta un nuevo socio. Si el socio ha sido bien guardado o si ha habido un error, apareceré un mensaje que lo refleje. TODO: No se debe permitir añadir un socio nuevo vacio.
* TODO: Ver pagos sin realizar: En este caso, se debería de ver los socios que tienen algo por pagar. Otra opción es tener la opción de ver la lista de morosos en esta misma ventana. También debe de haber una opción para poner como ‘pagada’ la actividad o cuota que el socio ha dejado de pagar, en caso de que el socio haya abonado el pago.

Las siguientes dos se activaran cuando un socio haya sido seleccionado:

* Dar socio de baja: Aparecerá una nueva pantalla – darBajaGUI – y después de rellenar los datos, se guardara la fecha de baja y el motivo de baja. TODO: Se tiene que comprobar que el socio no se había dado de baja antes, y al darse de baja deberá aparecer un mensaje de aviso.
* Ver socio: se abrirá una nueva ventana – verSocioGUI – en la que aparecerán los datos de los socios que podrán ser editados con el botón “editar” – editarSocioGUI – ver sus actividades o imprimir la ficha de socio.
  + Ver actividades de socio: - actividadesDeSocioGUI - Aparecerá la lista de actividades en la que el socio está matriculado ¿? valorar si interesa ver las actividades actuales o todas. Las actividades del socio que aparecen en rojo serán las que están pendiente de pago. En esta ventana habrá tres opciones:
    - Nueva actividad: ­- asignarActividadGUI – Aparecerá la lista de actividades en la que el socio no está matriculado, y seleccionando una se podrá asignar esta al socio. En este momento, aparecerá un mensaje en la pantalla, que preguntará si es socio ha pagado la actividad o no, es decir, si la actividad está pendiente de pago o no, y desaparecerá de la lista e imprimirá un recibo[[2]](#footnote-2).
    - Eliminar actividad: En esta acción simplemente se eliminará la actividad relacionada al socio en la tabla “socioActividad “y aparecerá un mensaje para saber el dinero que se le ha devuelto al socio, para el libro diario y se imprimirá un recivo. TODO: En vez de eliminar del todo esta actividad, es interesante guardar que el socio alguna vez ha sido matriculado y desapuntado después.
    - Pagar actividad: Si la actividad que se selecciona está pendiente de pago, se activara la opción de “pagar” que actualizara el booleano de la tabla de “socioActividad”. TODO: Al pagar la actividad, tendrá que aparecer un mensaje de aviso y actualizar la lista automáticamente para que aparezca en negro la actividad que antes aparecía en rojo e imprimirá un recibo.
  + Imprimir ficha: Esta vez, se abrirá un pdf en el que tendrá el carnet de socio en la parte superior y la ficha correspondiente a la asociación en la parte inferior. Este pdf se crea en la clase “ficha.java” y usa la librería “itexpdf”.

## Actividades

En esta pantalla – listaActividadesGUI – aparecerá la lista de todas las actividades, y será posible de buscarlos con cualquiera de los datos que aparezcan en pantalla. En caso de seleccionar una actividad, sus datos más importantes aparecerán en la casilla de la parte inferior.

TODO: ordenar las actividades con las opciones que se dan, e incluir más criterios si se cree necesario.

En este momento hay 3 opciones que deberán ser completadas. TODO: poner un nuevo botón para ver las actividades de pendiente de pago o tal y como decíamos en el socio, también puede ser un criterio más de ordenación.

* Nueva actividad: - nuevaActividadGUI - La interfaz cambia según el tipo de actividad a la que nos referimos ya que cada actividad necesita atributos diferentes. TODO: El hecho de cargar una interfaz diferente depende del tipo de actividad da problemas, los atributos diferentes de cada tipo dan *nullPointerException* a la hora de guardar la actividad en la base de datos.

Las siguientes dos se activarán cuando una actividad haya sido seleccionada:

* TODO: Eliminar actividad: Se deberá de eliminar la actividad seleccionada, pudiendo dar opción a poner más información. Las actividades eliminadas no serán eliminadas permanentemente, es recomendable guardarlos para la memoria.
* TODO: Ver actividad: Tal y como optamos a las actividades al ver un socio, esta vez deberemos poder optar a los socios de cada actividad y de la misma manera poder añadir o quitar un socio de la actividad y pagando o no la actividad. También debemos de tener la opción de editar la actividad si así lo deseamos.

TODO: esta aplicación debe tener una interfaz sencilla y fácil de entender por eso, los iconos elegidos deben de ser claros. Todavía no se ha encontrado una imagen adecuada que refleje el término “actividad” y es muy importante para el uso y la accesibilidad de la aplicación.

## Asociación

En este icono solo hay dos opciones, ver los datos – datosAcocioacionGUI – de la asociación y editar los datos de la asociación – editarAcocioacionGUI –. Esta opción tiene como objetivo generalizar la asociación, de manera que todos los datos que necesitemos de la asociación estén en ésta, pudiendo así funcionar para todas las asociaciones o ser modificada por los propios usuarios de la asociación en el caso de que hubiera cambiado el logo, dirección, …

Es importante el campo de numeración se socios, que afectara como su propio nombre indica a la numeración de los socios. Cuando la primera semana de las practicas visitamos las diferentes asociaciones se vio una diferencia a la hora de numerar los socios, normalmente las asociaciones pequeñas hacían una numeración liberatoria, es decir, que cuando un socio se daba de baja su número se guardaba para un futuro nuevo socio, así evitaban las largas listas de socios. Por otro lado, otras asociaciones le daban bastante importancia al número de socio que respetaba el orden de entrada, es decir, siempre que un nuevo socio se quería dar de alta, se le cedía un nuevo número, así el socio que tenía el número más pequeño seria quien se hubiera dado alta antes. TODO: Con el fin de generalizar esta aplicación, este atributo deberá influir en la oferta de los números para un socio nuevo. En caso de los números libres, la oferta puede ser numerosa, para que los nuevos socios puedan elegir entre los números que hay libres.

## Terceros

Existen dos tipos de terceros: monitores y cesionarios. Los monitores o los profesionales son aquellos que dinamizan actividades especificadas ¿incluir en el libro diario? como taichí, yoga, piscina … Los cesionarios en cambio, son trabajadores que desarrollan sus servicios en la sede de la asociación como por ejemplo el bar, el podólogo, la peluquería …

Los profesionales y los cesionarios funcionan de la misma manera, pueden ser trabajadores o empresas y hay que guardar los datos de ambos campos. En ambos casos tendremos la opción de añadir editar o eliminar un tercero TODO: y actualizar de la misma forma la base de datos. Incluir también, un campo opcional para otras observaciones se quiera guardar.

En el caso de los cesionarios tendremos tres campos más, que serán la documentación necesaria que deberá presentar el cesionario para poder trabajar en la asociación.

## Listados

Este campo esta comenzado, pero le falta mucho por terminar. La función de este – listadosGUI – es poder imprimir toda la información que se guarda en la base de datos. De momento, existe la opción de imprimir la lista de socios, en la que conseguiremos un pdf en la que aparecerá una tabla con los datos seleccionados. TODO: Poder optar a la lista en formato Excel también.

TODO: En este listado deberá de haber la opción de incluir varios filtros tales como la edad, la fecha de alta, el sexo, ….

TODO: De la misma manera deberá de haber opción a imprimir la lista de actividades, con sus respectivos filtros, los socios relacionados con las actividades, los monitores, los cesionarios…

TODO: También hay opción a incluir esta opción en la lista de actividades, actividades de socio, socios de actividades…

## Libro diario

En este campo, estarán accesibles todo tipo de gastos o ingresos que afectan a la asociación. La idea de esto es que los responsables de la asociación puedan ver los movimientos diarios e incluso de entre fechas y poder generar un pdf o un archivo de formato Excel con esos datos.

TODO: Guardar todos los gastos e ingresos

TODO: Generar un archivo pdf y Excel de libro diario.

## Calendario

En esta opción la idea era ver en formato calendario las distintas actividades de la asociación.

TODO: Cargar en formato calendario todas las acividades

1. Agradecería que quien siguiese el proyecto siguiese usando este repositorio, para ver el avance del proyecto y poder participar en él en caso de que fuese posible. Para poder subir el código al repositorio o preguntar cualquier cosa sobre el proyecto por favor, ponte en contacto conmigo vía email **-** [**aizpea\_babaze@hotmail.com**](mailto:aizpea_babaze@hotmail.com)- o telegram **- @aizpea -** [↑](#footnote-ref-1)
2. El ayuntamiento de Donostia – San Sebastián está interesado en utilizar una impresión térmica para el impreso de los recibos. [↑](#footnote-ref-2)