UNIVERSIDAD COMPLUTENSE
MADRID

INGENIERÍA DEL SOFTWARE
2º B

Swimming

Manager

GRUPO 6+2

JUAN LUIS ÁLVAREZ HERRADÓN

LARA CARRIÓN OVEJERO

CHARLES FLORES ESPINOZA

CLAUDIA GIL NAVARRO

VÍKTOR SHAMEL JACYNYCZ GARCÍA

JAIME MAYORDOMO MORENO

MARTA OLIVER VALIENTE

ÁNGEL LUIS ORTIZ FOLGADO

INDICE


SWIMMING
MANAGER

1. INTRODUCCIÓN

ACERCA DE SWIMMING MANAGER

Swimming Manager es un proyecto que nace con la finalidad de desarrollar un gestor de deportistas enfocado a competiciones de natación.

Éste está destinado principalmente a aquellos que se dedican a llevar un seguimiento de deportistas en pruebas de natación, tales como directores deportivos, entrenadores personales, clubes, federaciones y/o árbitros de natación.

Swimming Manager ofrece una gran variedad de funciones, entre las que cabe destacar las siguientes:

- ✓ Crear bases de datos de nadadores.
- ✓ Añadir información relevante sobre los deportistas, como son datos personales, marcas, club al que pertenece, etc.
- ✓ Clasificaciones por estilos, edad y marcas.
- ✓ Comparación de marcas entre deportistas.
- ✓ Listas de patrocinadores por clubes.

En cuanto a objetivos se refiere, *Swimming Manager* se enfoca en varios de ellos, detallados en el siguiente punto.

OBJETIVOS

Los objetivos de este gestor de nadadores pueden resumirse en:

- ✓ Facilitar la creación de una base de datos ordenada con los nombres y datos personales de todos los nadadores pertenecientes a un club o federación.
- ✓ Acceder fácilmente a una clasificación ordenada de los nadadores por pruebas y estilos.
- ✓ Facilitar la obtención de información referente a un deportista, club deportivo u organización.
- ✓ Agilizar trámites en lo referente a pruebas de un día o competiciones de alto nivel, así como obtener información detallada de las mismas.

ACERCA DEL PROYECTO

Este proyecto ha sido desarrollado en el contexto de la asignatura de Ingeniería del Software de 2º Curso del Grado en Ingeniería Informática de la Universidad Complutense de Madrid, bajo la supervisión del profesor Pablo Gervás.

El objetivo principal de nuestro proyecto es unir técnicas, recursos y personal en un solo entorno para conseguir resultados rápidos, ordenados y eficaces, enfocados principalmente a profesionales que trabajan en un escenario deportivo especializado en pruebas de natación.

Éste, al estar destinado principalmente a personas que tienen que llevar un seguimiento diario de las competiciones de natación y que suponemos van a dar un uso continuado e intensivo al programa, la interfaz está diseñada de forma sencilla e intuitiva, con el fin de facilitar el manejo del mismo y el seguimiento de deportistas al usuario.

El modelo de proceso de desarrollo utilizado será un desarrollo basado en componentes, reutilizando software para simplificar las pruebas y el mantenimiento, y de esta forma consiguiendo una mayor calidad al finalizar el proyecto.

El lenguaje utilizado para diseñar el programa será Java, porque nos resulta más apropiado debido a su orientación a objetos y la facilidad de uso de las interfaces, además de ser un lenguaje muy utilizado que ofrece gran cantidad de librerías.

De esta forma, utilizaremos dos entornos de programación:

- **Eclipse**, porque es el entorno que más ayudas ofrece al escribir código y es el que nos resulta más conocido.
- **NetBeans**, por la comodidad de su uso para diseñar interfaces gráficas de usuario, es ideal para ahorrar tiempo sustituyendo a Eclipse en esta labor.

GESTIÓN DE CONFIGURACIÓN

En un primer momento, comenzamos a utilizar **Dropbox** para gestionar nuestro proyecto compartiendo una misma carpeta entre los 8 componentes del grupo. Esto sirvió durante la fase de inicio, pero en la fase de elaboración comenzamos a portar la gestión a **Google Code**.

El motivo para cambiar ha resultado suficiente; Google Code permite una sincronización con NetBeans a través de una cuenta de GMail, generando así de manera automática las diferentes versiones de la aplicación mientras se está desarrollando por cualquier componente del grupo y permitiendo diferenciar los cambios entre ellas, línea a línea, incluso pudiendo recuperar aquella que se desee sin temor a haberla perdido. Además, cualquier interesado puede hacer un seguimiento completamente transparente del proyecto gracias al siguiente enlace público:

http://code.google.com/p/swimming-manager/

También se puede encontrar buscando "*Swimming Manager*" en la sección Project Hosting de Google Code. Desde esta página es posible descargar versiones ejecutables y observar el progreso que se va efectuando, así como comprobar los errores que quedan por resolver.

De esta forma, utilizamos Dropbox para gestionar la documentación y Google Code para publicar y gestionar la parte programable del proyecto Swimming Manager.

La experiencia con Dropbox ha sido buena para gestionar documentación en general. Creamos una carpeta compartida en la que incluimos un fichero de texto que usamos como historial, de forma que cada vez que un miembro del grupo realizara un cambio, lo detallaría en el historial. En cambio, para la gestión del código programable, resultó insuficiente, puesto que además de no poder recuperar versiones anteriores al cabo de un tiempo, era mucho más costoso subir los archivos que utilizando Google Code.

Por otro lado, la experiencia con Google Code ha sido la mayor sorpresa del proyecto, ya que con los rasgos comentados anteriormente, la sincronización y visualización de los cambios entre las diferentes versiones del programa se hacía sencilla y eficaz. Utilizamos una sección dedicada a reportar problemas que aunque no explotamos lo suficiente, nos fue bastante útil. En ocasiones se presentaba algún problema de sincronización, casi siempre por hacer alguna modificación local de un archivo que no podía subir correctamente, pero en general estamos muy satisfechos con la elección.

ENTORNO TECNOLÓGICO

El requisito principal de Swimming Manager, debido al lenguaje de programación utilizado, es tener Java 6 (o superior) instalado. Por ello, los requisitos software y hardware se heredan de éste:

EQUIPO FÍSICO	**REQUISITOS**	
	Mínimos	Recomendados
CPU	PC Pentium	PC-Pentium IV
Memoria RAM	64 MB	>128 MB
Espacio en disco	100 MB	1 GB
Tarjeta de Red	Ethernet	Ethernet 10/100
S.A.I. (Sistema de Alimentación Interrumpida)	No	Si

EQUIPO LÓGICO	**REQUISITOS**	
	Mínimos	Recomendados
Sistema Operativo	Windows XP/Vista/7	Windows XP/7
Software de Red	TCP / IP	
Software de Aplicación	No	No

2. REQUISITOS

1. Requisitos funcionales - ¿Qué hace el sistema?: Casos de uso.

2. Requisitos no funcionales - ¿Cómo lo hace el sistema?

REQUISITOS FUNCIONALES

Los requisitos funcionales del sistema son los siguientes:

- ✓ Dar de alta a un nadador[1].
- ✓ Dar de baja a un nadador.
- ✓ Añadir marca de un nadador[2].
- ✓ Eliminar marca de un nadador.
- ✓ Ver la información asociada a un nadador.
- ✓ Buscar nadadores introducidos en el programa por nombre, edad, sexo, nacionalidad, nº federativo, estilo, distancia, marca o récord.
- ✓ Comparar marcas entre nadadores.
- ✓ Guardar la información introducida en el programa en un archivo de texto.
- ✓ Cargar datos de un archivo de texto previamente generado con el programa.
- ✓ Mostrar una lista de todos los nadadores que hay actualmente dados de alta.

Todos ellos quedan detallados en el apartado de especificación del proyecto.

[1] Para dar de alta a un nuevo nadador, se solicitará al usuario la siguiente información: nombre, edad, sexo y nacionalidad.
[2] Las marcas se contabilizarán en minutos y segundos. Se almacenarán las 10 más recientes de cada nadador además de su mejor marca hasta la fecha.

REQUISITOS NO FUNCIONALES

RENDIMIENTO DEL SISTEMA

El sistema ha de cumplir unos requisitos no funcionales esenciales para su correcto funcionamiento:

- ✓ Debe visualizarse, ser compatible y funcionar correctamente en cualquier sistema operativo y versión, especialmente con Windows, Mac y Linux.
- ✓ Debe ejecutarse nada más abrirlo en un tiempo no superior a 5 segundos, no tardando más de 30 segundos en empezar a trabajar tras la carga de información almacenada.
- ✓ Debe realizar una búsqueda en un tiempo inferior a 5 segundos.
- ✓ Al dar de alta un nuevo deportista, el tiempo de espera no debe ser mayor de 3 minutos, ya que en este caso el sistema lanzará un mensaje de aviso pidiendo todos los datos nuevamente.
- ✓ Al dar de baja un deportista, el sistema no debe tardar más de 5 segundos en eliminar toda la información acerca de éste.
- ✓ Si ocurre un fallo en el sistema, el tiempo de restauración máximo aceptable será de 3 minutos hasta volver al punto en el que se encontraba el usuario antes de dicho fallo.

SEGURIDAD

En esta versión del proyecto, para poder acceder al sistema no es necesario introducir un nombre de usuario y contraseña, por lo que cualquier usuario podría tener acceso a datos y modificaciones. En versiones futuras, el sistema se comportará de la siguiente manera:

- ✓ Para poder acceder al sistema y a la información almacenada será necesario introducir un nombre de usuario y contraseña correctos que posteriormente serán verificados por el sistema.
- ✓ Cualquiera de los administradores podrá también acceder al sistema con su propio nombre de usuario y contraseña.

FIABILIDAD

El sistema debe ser fiable, ya que de otro modo puede ocasionar fallos que conlleven problemas difíciles de solucionar una vez desarrollado el software.

Los fallos de funcionamiento de un sistema pueden tener su origen en errores producidos al diseñar el software o problemas con el hardware.

Para aumentar la fiabilidad del sistema es conveniente prevenir los fallos y hacer que el sistema sea tolerante a ellos, siguiendo éste en funcionamiento aunque se produzcan errores, con el fin de evitar problemas tanto al usuario como a los desarrolladores.

MANTENIBILIDAD

La mantenibilidad es la facilidad con la que un sistema o componente software puede ser modificado para corregir fallos, mejorar su funcionamiento u otros atributos o adaptarse a cambios en el entorno.

El tipo de mantenimiento del proyecto hemos estimado que será mensual, previendo posibles fallos y reparando los ya existentes detectados en anteriores iteraciones.

También realizaremos las modificaciones oportunas si algún error es detectado por el usuario y lanzaremos una nueva versión previamente revisada.

PORTABILIDAD

Por su amplia compatibilidad con la mayoría de los sistemas operativos actuales y nuestro conocimiento acerca del mismo, hemos elegido diseñar nuestro sistema empleando el lenguaje de programación Java.

Para su desarrollo, utilizaremos Eclipse y NetBeans como entornos de programación; el primero, por su facilidad de uso y familiaridad con el entorno, y el segundo, por la comodidad que ofrece en la creación de interfaces gráficas.

3. ESPECIFICACIÓN DEL PROYECTO

ACTOR PRINCIPAL

En nuestro caso, el actor principal es un usuario que se encargue de llevar el seguimiento de los deportistas dados de alta en el programa. Sus objetivos serán cada uno de los **REQUISITOS FUNCIONALES** del sistema.

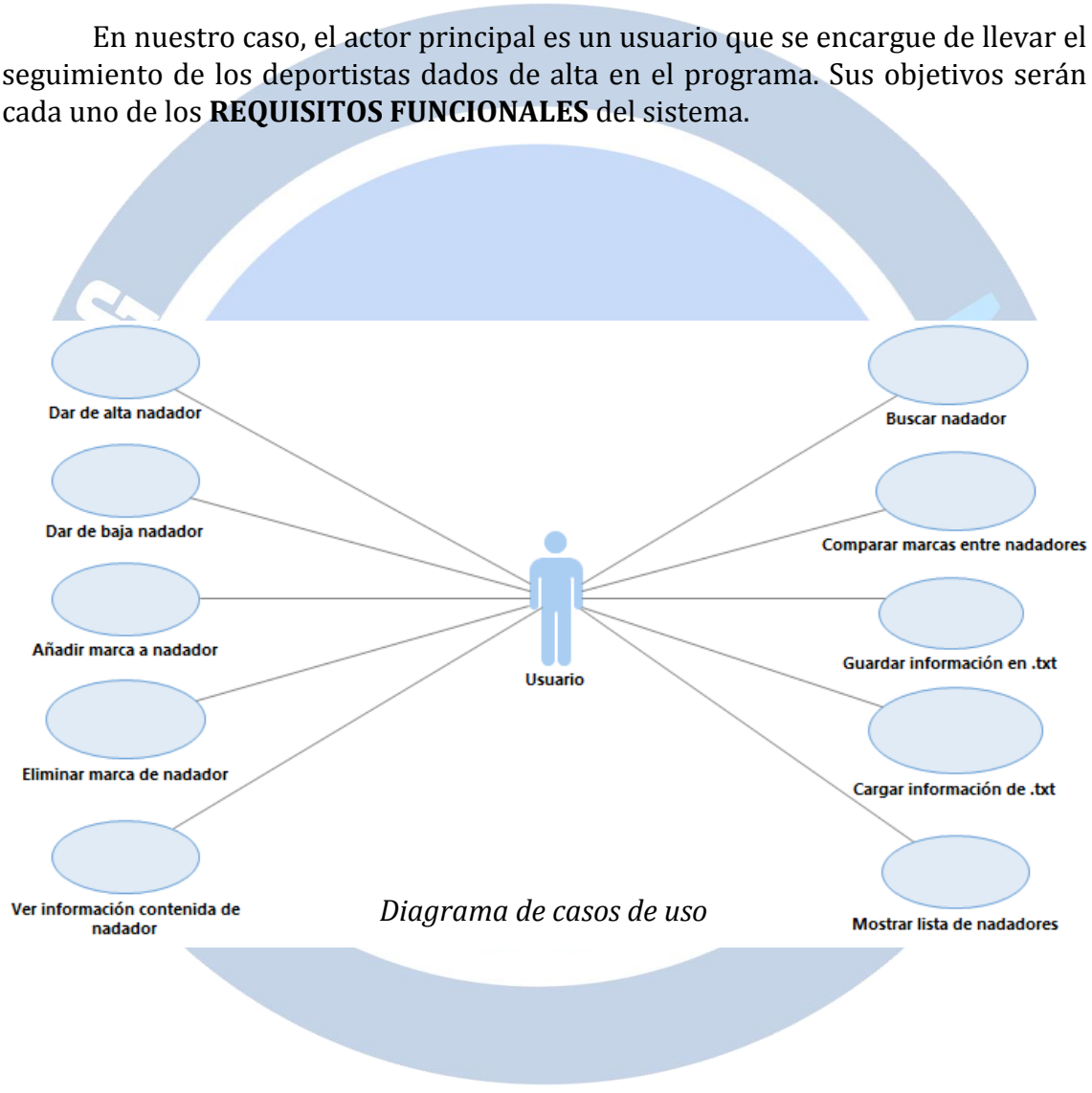


Diagrama de casos de uso

ACTOR SECUNDARIO

En este caso, el actor secundario es la base de datos que vamos a usar para almacenar los datos así como a la hora de hacer consultas.

CASOS DE USO

❖ **Dar de alta a un nadador**

- ***Objetivo***: Introducir en la base de datos del programa a un nuevo nadador aportando su nombre, edad, sexo y nacionalidad.
- ***Precondición***: El programa ha sido iniciado correctamente.
- ***Postcondición sin fallo***: La información del nuevo nadador es añadida al programa.
- ***Postcondición con fallo***: El nadador ya había sido introducido previamente. Se informa del error.
- ***Actores***: Usuario, base de datos.
- ***Secuencia normal***:

  1. Se pulsa *Editar=>Dar de alta* en la barra de opciones del programa.
  2. Se muestra un formulario con los campos *Nombre, Edad, Sexo y Nacionalidad* que el usuario debe completar.
  3. Una vez han sido rellenados todos los datos, se comprueba si ya existe algún nadador en el sistema con el mismo nombre. En caso afirmativo, se pasa a *(S1).* En caso contrario, se almacenan los datos en la base de datos.

- ***Secuencia alternativa***:

  S1. Se informa de que el nadador que el usuario intenta almacenar ya se encuentra dado de alta en el sistema.

❖ Dar de baja a un nadador

- ***Objetivo***: Eliminar de la base de datos del programa a un nadador seleccionado por el usuario.
- ***Precondición***: El programa ha sido iniciado correctamente.
- ***Postcondición sin fallo***: La información del nadador se elimina del programa.
- ***Postcondición con fallo***: No se ha introducido ningún nadador. Se informa del error.
- ***Actores***: Usuario, base de datos.
- ***Secuencia normal***:

1. Se pulsa *Editar=>Dar de baja* en la barra de opciones del programa.
2. Se solicita al usuario que elija el nadador que desea dar de baja de una lista con los nombres de los que actualmente se encuentran almacenados. En caso de no haber ninguno dado de alta, se pasa a *(S1).*
3. Una vez ha sido seleccionado el nadador, el programa lo elimina de la base de datos.

- ***Secuencia alternativa***:

S1. Se informa de que no hay ningún nadador dado de alta en el programa.

❖ Añadir marca a un nadador

- ***Objetivo***: Añadir una nueva marca para algún nadador ya dado de alta.
- ***Precondición***: El usuario ha introducido algún nadador previamente en el programa.
- ***Postcondición sin fallo***: Se añade la nueva marca al nadador seleccionado por el usuario.
- ***Postcondición con fallo***: No se ha introducido ningún nadador en el programa. Se informa del error.
- ***Actores***: Usuario, base de datos.
- ***Secuencia normal***:

  1. Se pulsa *Editar=>Nadador=>Añadir marca* en el menú de opciones del programa.
  2. Se solicita al usuario que elija al nadador de entre una lista con los nombres de los que actualmente se encuentran almacenados y rellene el campo *Marca* con el nuevo tiempo. En caso de no haber ninguno dado de alta, se pasa a *(S1).*
  3. Una vez ha sido seleccionado el nadador e introducida la nueva marca, se comprueba si ya se le habían asignado 10 distintas al deportista correspondiente. En caso afirmativo, se elimina la más antigua y se añade la actual. En caso contrario, simplemente se añade la nueva marca.
  4. Se compara la marca récord del nadador con la nueva que ha sido introducida. En caso de ser mejor que la que hay almacenada, ésta se actualiza con la nueva marca. En caso contrario, no se realiza ningún cambio en ese campo

- ***Secuencia alternativa***:

  S1. Se informa de que no hay ningún nadador dado de alta.

❖ Eliminar marca de un nadador

- ***Objetivo***: Eliminar una marca seleccionada de un nadador ya dado de alta.
- ***Precondición***: El usuario ha introducido algún nadador previamente en el programa.
- ***Postcondición sin fallo***: Se elimina la marca seleccionada por el usuario del nadador correspondiente.
- ***Postcondición con fallo***: No se ha introducido ningún nadador en el programa. Se informa del error.
- ***Actores***: Usuario, base de datos.
- ***Secuencia normal***:

  1. Se pulsa *Editar=>Nadador=>Eliminar marca* en el menú de opciones del programa.
  2. Se solicita al usuario que elija al nadador cuya marca desee eliminar de una lista con los nombres de los que actualmente se encuentran almacenados. En caso de no haber ningún nadador, se pasa a *(S1).* En caso de no haberse introducido ninguna marca para ese nadador, se pasa a *(S2).*
  3. Se muestra una lista de las marcas asignadas al nadador seleccionado y se pide al usuario que seleccione aquella que desea eliminar.
  4. Una vez el usuario ha seleccionado la marca deseada, el programa elimina dicha marca del nadador.

- ***Secuencias alternativas***:

  S1. Se informa de que no hay ningún nadador dado de alta.

  S2. Se informa de que aún no se le ha asignado ninguna marca al nadador seleccionado.

❖ Editar información asociada a un nadador

- ***Objetivo***: Editar la información de un nadador que había sido dado de alta previamente.
- ***Precondición***: El usuario ha introducido algún nadador previamente en el programa.
- ***Postcondición sin fallo***: Se guarda la información asociada al nadador que el usuario ha editado.
- ***Postcondición con fallo***: No se ha introducido ningún nadador en el programa. Se informa del error.
- ***Actores***: Usuario, base de datos.
- ***Secuencia normal***:

1. Se pulsa *Editar=>Nadador=>Información* en la barra de opciones del programa.
2. Se solicita al usuario que elija al nadador cuya información desee visualizar de una lista con los nombres de los que actualmente se encuentran almacenados. En caso de no haber ningún nadador, se pasa a *(S1).*
3. Una vez ha sido seleccionado el nadador, se muestran por pantalla sus datos en modo editable.

- ***Secuencia alternativa***:

S1. Se informa de que no hay ningún nadador dado de alta.

❖ Buscar nadadoresintroducidos en el programa por nombre, edad, sexo, nacionalidad, estilo, distancia, marca o récord

- ***Objetivo***: Buscar nadadores por medio del criterio seleccionado por el usuario (nombre, edad, sexo, nacionalidad, estilo, distancia, marca o récord).
- ***Precondición***: El usuario ha introducido a algún nadador previamente en el programa.
- ***Postcondición sin fallo***: Se muestra una lista con los nombres de los nadadores que el programa ha encontrado según el criterio que el usuario ha seleccionado.
- ***Postcondición con fallo***: No se ha encontrado ningún nadador que cumpla el criterio elegido. Se informa al usuario de ello.
- ***Actores***: Usuario, base de datos
- ***Secuencia normal***:

  1. Se pulsa *Buscar por…* y se selecciona la opción deseada en la barra de opciones del programa: *Nombre, Nacionalidad, Sexo, Estilo, Distancia, Edad, Marca, Récord.*
  2. El usuario rellena el campo seleccionado.
  3. Una vez ha sido introducido el campo correspondiente, el programa busca entre los nadadores almacenados aquellos que se correspondan con el criterio de búsqueda establecido. Se muestra una lista con los nombres de los nadadores encontrados o un mensaje indicando que no existe ninguno que cumpla lo establecido. En caso de no haberse almacenado ningún nadador, se pasa a *(S1).*

- ***Secuencia alternativa***:

  S1. Se informa de que no hay ningún nadador dado de alta en el sistema.

❖ Comparar marcas entre nadadores

- ***Objetivo***: Comparar la información de dos nadadores seleccionados por el usuario.
- ***Precondición***: El usuario ha dado de alta a dos o más nadadores en el sistema.
- ***Postcondición sin fallo***: Se muestra por pantalla los datos y marcas de los nadadores elegidos.
- ***Postcondición con fallo***: No hay dos o más nadadores dados de alta y no puede realizarse una comparación. Se informa del error.
- ***Actores***: Usuario, base de datos.
- ***Secuencia normal***:

1. Se pulsa *Comparar* en la barra de opciones del programa.
2. El usuario selecciona los dos nadadores que quiere comparar de una lista con los nombres de los que se encuentran actualmente dados de alta. En caso de no haber dos o más almacenados en el sistema, se pasa a *(S1).*
3. Una vez seleccionados, el programa muestra por pantalla los datos, marcas y récords de ambos deportistas.

- ***Secuencia alternativa***:

S1. Se informa de que no se han dado de alta a suficientes nadadores como para poder hacer una comparación.

❖ Guardar información introducida en el programa en un archivo texto

- ***Objetivo***: Almacenar la información que contenga el programa actualmente en un archivo de texto para su posterior uso en otra ocasión.
- ***Precondición***: El usuario ha dado de alta a algún nadador en el sistema y ha modificado la información contenida desde la última vez que se procedió a guardar.
- ***Postcondición sin fallo***: Se guarda la información contenida en el programa en un archivo de texto.
- ***Postcondición con fallo***: No hay ningún nadador dado de alta en el sistema. Se informa del error.
- ***Actores***: Usuario.
- ***Secuencia normal***:

1. Se pulsa *Archivo=>Guardar como* en la barra de opciones del programa.
2. Si no se ha guardado la información en un fichero desde el inicio de sesión en el programa, se solicitará que se introduzca un nombre y una ubicación del archivo en el que se quiere almacenar los datos. Si no, se procederá a reescribir el fichero seleccionado previamente. En caso de no haber ningún nadador dado de alta en el sistema, se pasa a *(S1)*.
3. Los datos de los nadadores y sus correspondientes marcas y récords son almacenados en el archivo de texto.

- ***Secuencia alternativa***:

S1. Se informa de que no hay ningún nadador dado de alta actualmente.

❖ Cargar datos de un archivo de texto generado previamente por el programa

- ***Objetivo***: Introducir datos en el programa por medio de un archivo de texto que ya habíamos generado en otra ocasión y así poder visualizarlo y/o modificarlo.
- ***Precondición***: Se ha guardado la información previamente en un archivo de texto mediante el programa.
- ***Postcondición sin fallo***: La información contenida en el archivo de texto se introduce en el programa
- ***Postcondición con fallo***: El archivo seleccionado no es válido. Se notifica al usuario del error.
- ***Actores***: Usuario.
- ***Secuencia normal***:

  1. Se pulsa *Archivo=>Abrir* en la barra de opciones del programa.
  2. El usuario selecciona el archivo de texto cuya información desea cargar en el programa.
  3. El programa lee el contenido del archivo de texto e intenta introducir la información en el sistema. En caso de producirse un error, se pasa a *(S1)*. En caso contrario, se notifica que el archivo se ha cargado correctamente.

- ***Secuencia alternativa***:

  S1. Se informa de que el archivo seleccionado no es válido y se pide al usuario que intente abrir uno que sí lo sea.

❖ Mostrar lista de los nadadores que hay actualmente dados de alta

- ***Objetivo***: Devolver a la vista principal los datos, marcas y récords de todos los nadadores que hay dados de alta en el programa hasta el momento.
- ***Precondición***: Se ha dado de alta a algún nadador en el sistema,
- ***Postcondición sin fallo***: Se muestra una lista de los nadadores dados de alta con toda la información disponible.
- ***Postcondición con fallo***: No se ha dado de alta a ningún nadador. Se notifica al usuario del error.
- ***Actores***: Usuario, base de datos
- ***Secuencia normal***:

  1. Se pulsa *Archivo=>Mostrar todo* en la barra de opciones del programa.
  2. Se intenta mostrar una lista con el nombre, edad, sexo y nacionalidad de cada nadador. En caso de no encontrarse ninguno dado de alta, se pasa a *(S2)*. En caso contrario, se muestra la información.

- ***Secuencia alternativa***:

S1. Se informa de que no hay ningún usuario dado de alta en el programa.

TARJETAS CRC

Nadador	
Responsibilities	**Collaborators**
Contiene toda la información sobre cada nadador.	Fecha Estilo Nacionalidad Titulo Marca

Marca	
Responsibilities	**Collaborators**
Contiene toda la información sobre una marca.	Fecha Tiempo Estilo

Medalla	
Responsibilities	**Collaborators**
Enumerado que contiene los distintos tipos de medalla: ORO, PLATA y BRONCE.	

Fecha	
Responsibilities	**Collaborators**
Contiene la información sobre una determinada fecha.	

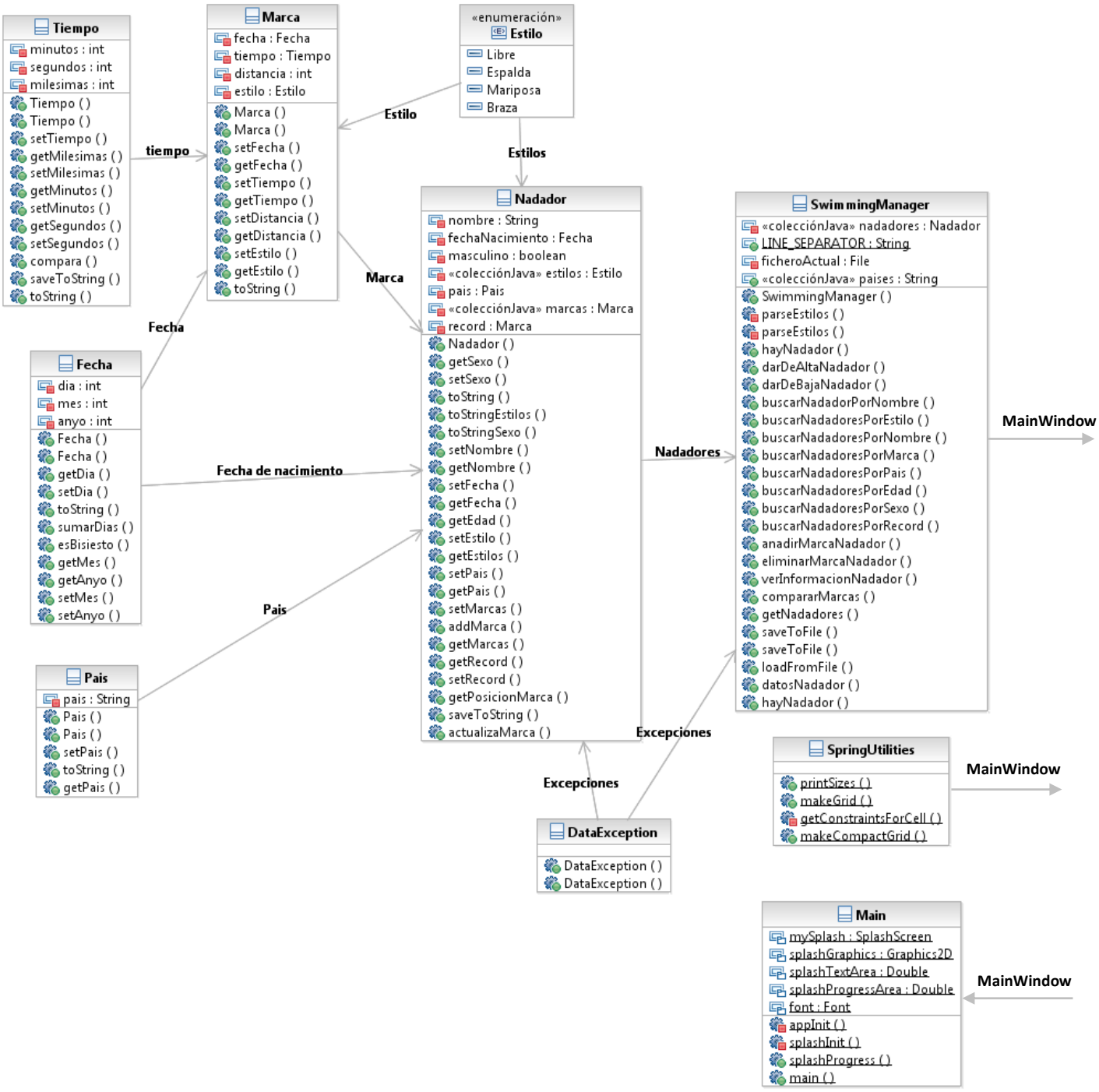
Estilo	
Responsibilities	**Collaborators**
Enumerado que contiene los distintos estilos: LIBRE, MARIPOSA, BRAZA, ESPALDA y CROLL.	

Nacionalidad	
Responsibilities	**Collaborators**
Contiene la información sobre la nacionalidad de un nadador.	

SwimmingManager	
Responsibilities	**Collaborators**
Poner en marcha la base de datos	
Gestión general de la base de datos (gestión de nadadores, gestión de marcas, etc.)	Nadador
Búsqueda y comparación	Nadador
Cargar y salvar la información relativa a la base de datos en/de un fichero de texto	

Tiempo	
Responsibilities	**Collaborators**
Contiene el tiempo de una marca.	

DIAGRAMA DE CLASES



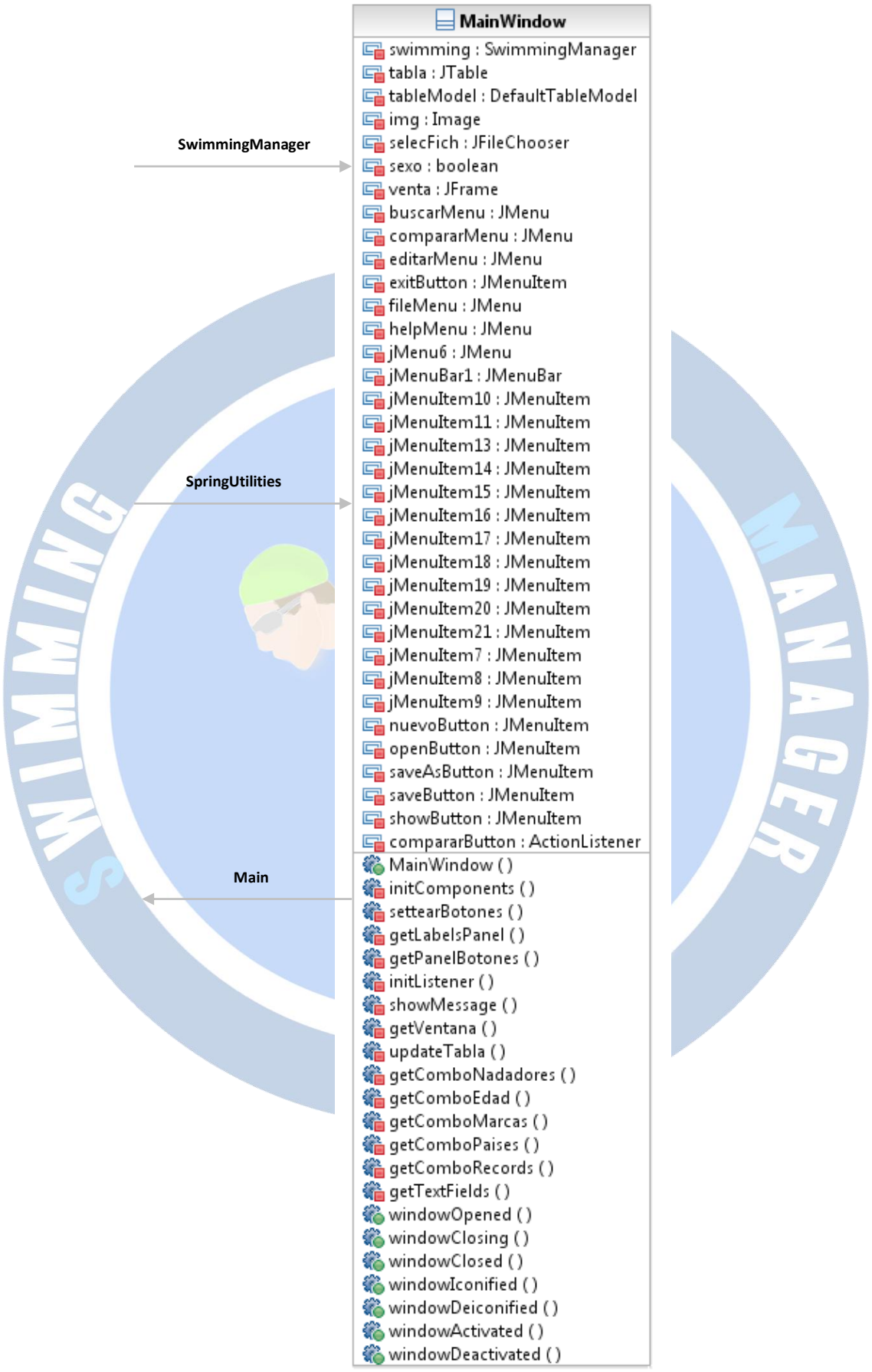
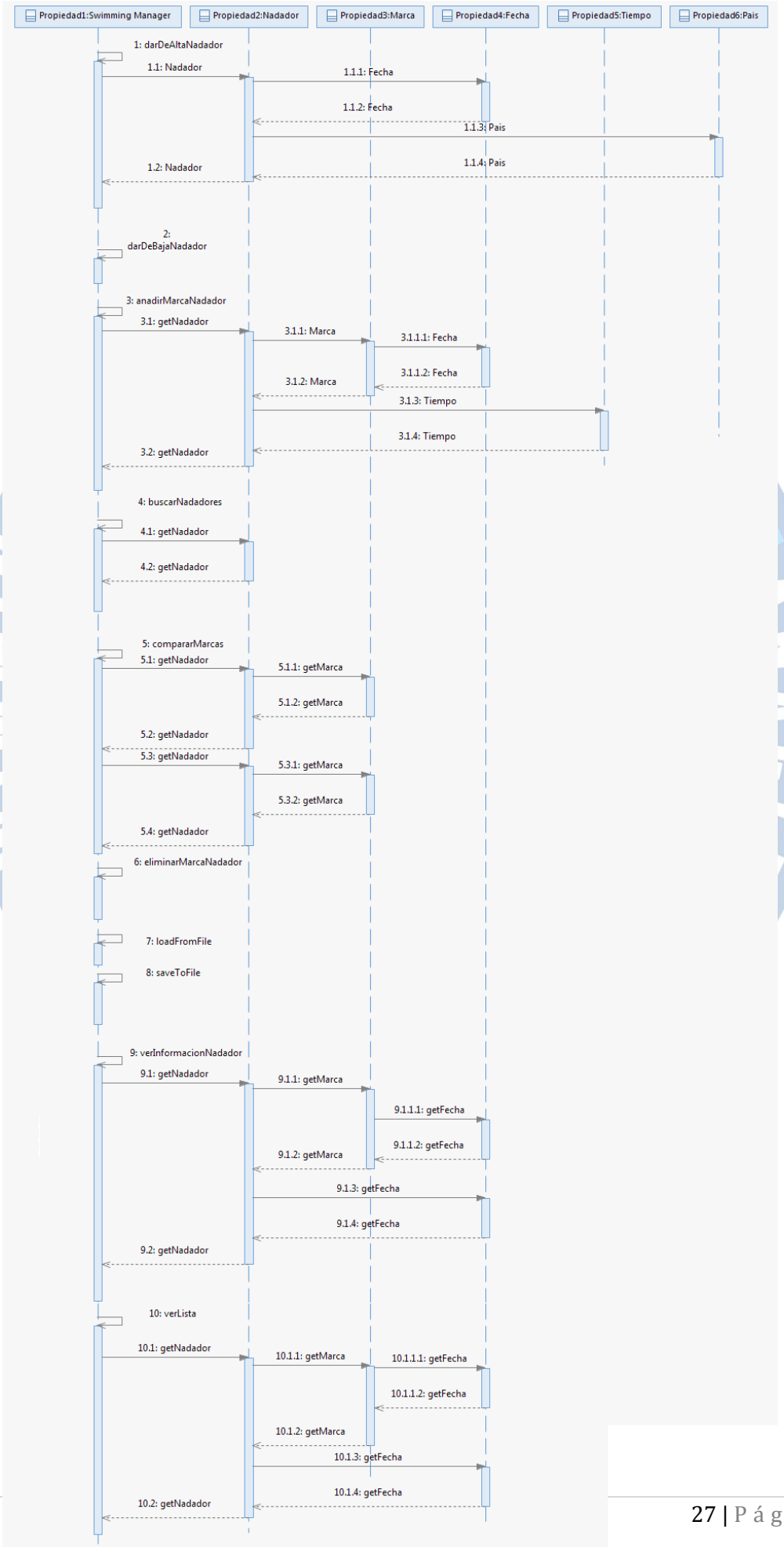
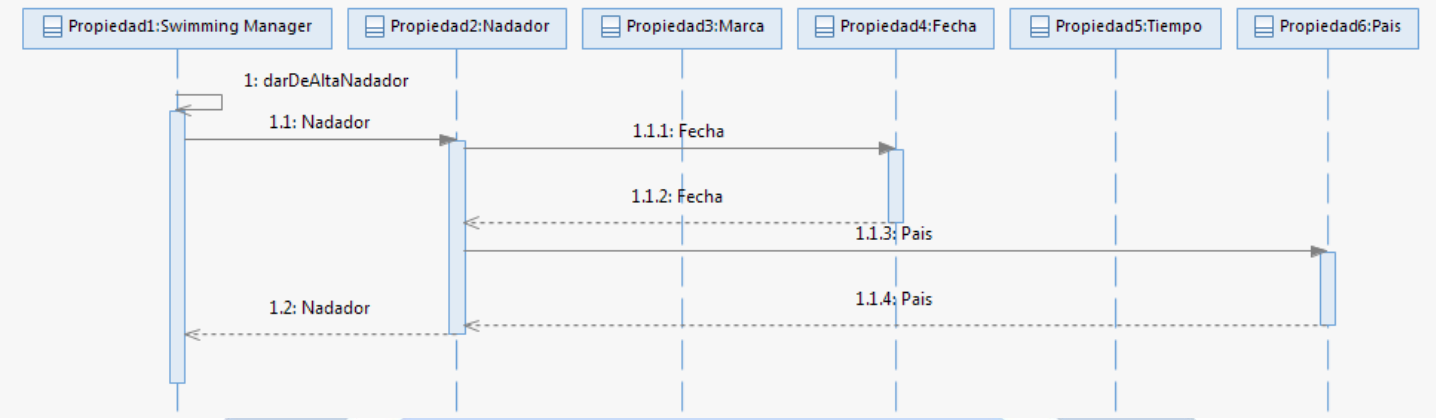

MainWindow
swimming : SwimmingManager
tabla : JTable
tableModel : DefaultTableModel
img : Image
selecFich : JFileChooser
sexo : boolean
venta : JFrame
buscarMenu : JMenu
compararMenu : JMenu
editarMenu : JMenu
exitButton : JMenuItem
fileMenu : JMenu
helpMenu : JMenu
jMenu6 : JMenu
jMenuBar1 : JMenuBar
jMenuItem10 : JMenuItem
jMenuItem11 : JMenuItem
jMenuItem13 : JMenuItem
jMenuItem14 : JMenuItem
jMenuItem15 : JMenuItem
jMenuItem16 : JMenuItem
jMenuItem17 : JMenuItem
jMenuItem18 : JMenuItem
jMenuItem19 : JMenuItem
jMenuItem20 : JMenuItem
jMenuItem21 : JMenuItem
jMenuItem7 : JMenuItem
jMenuItem8 : JMenuItem
jMenuItem9 : JMenuItem
nuevoButton : JMenuItem
openButton : JMenuItem
saveAsButton : JMenuItem
saveButton : JMenuItem
showButton : JMenuItem
compararButton : ActionListener
MainWindow ( )
initComponents ( )
settearBotones ( )
getLabelsPanel ( )
getPanelBotones ( )
initListener ( )
showMessage ( )
getVentana ( )
updateTabla ( )
getComboNadadores ( )
getComboEdad ( )
getComboMarcas ( )
getComboPaises ( )
getComboRecords ( )
getTextFields ( )
windowOpened ( )
windowClosing ( )
windowClosed ( )
windowIconified ( )
windowDeiconified ( )
windowActivated ( )
windowDeactivated ( )
SwimmingManager
SpringUtilities
Main

DIAGRAMA DE SECUENCIAS



A CONTINUACIÓN SE MUESTRAN LAS SECUENCIAS AMPLIADAS:

DAR DE ALTA A UN NADADOR

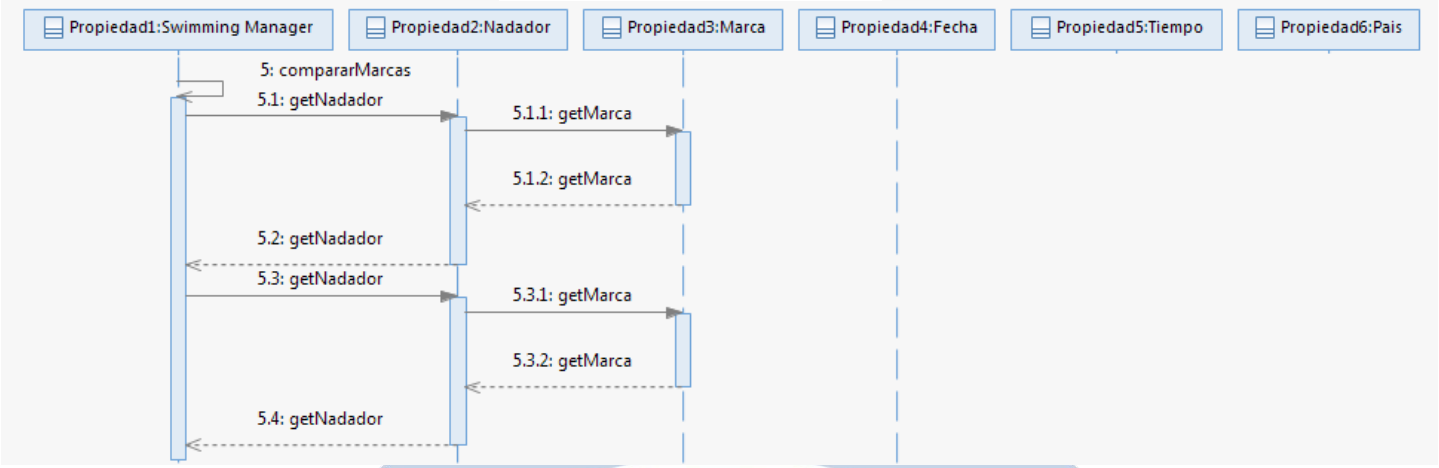


DAR DE BAJA A UN NADADOR

AÑADIR MARCA A UN NADADOR

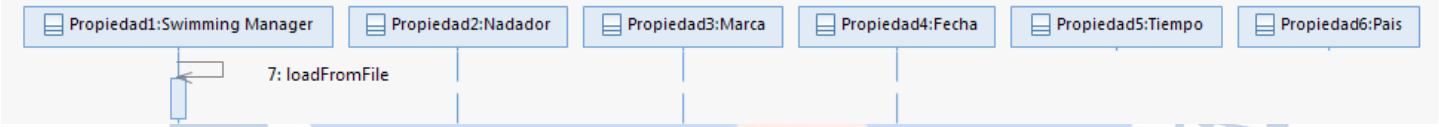
BUSCAR NADADORES

COMPARAR MARCAS

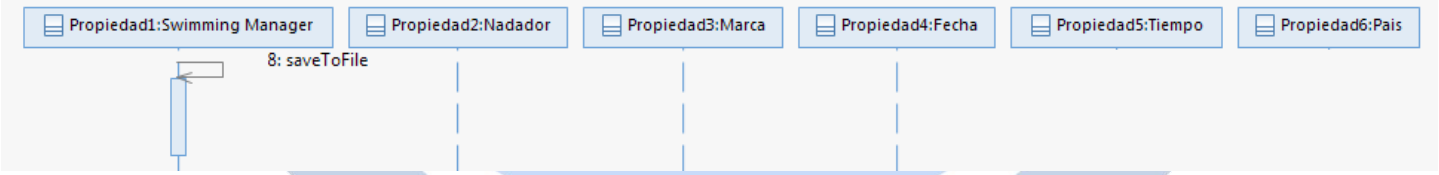


ELIMINAR MARCA

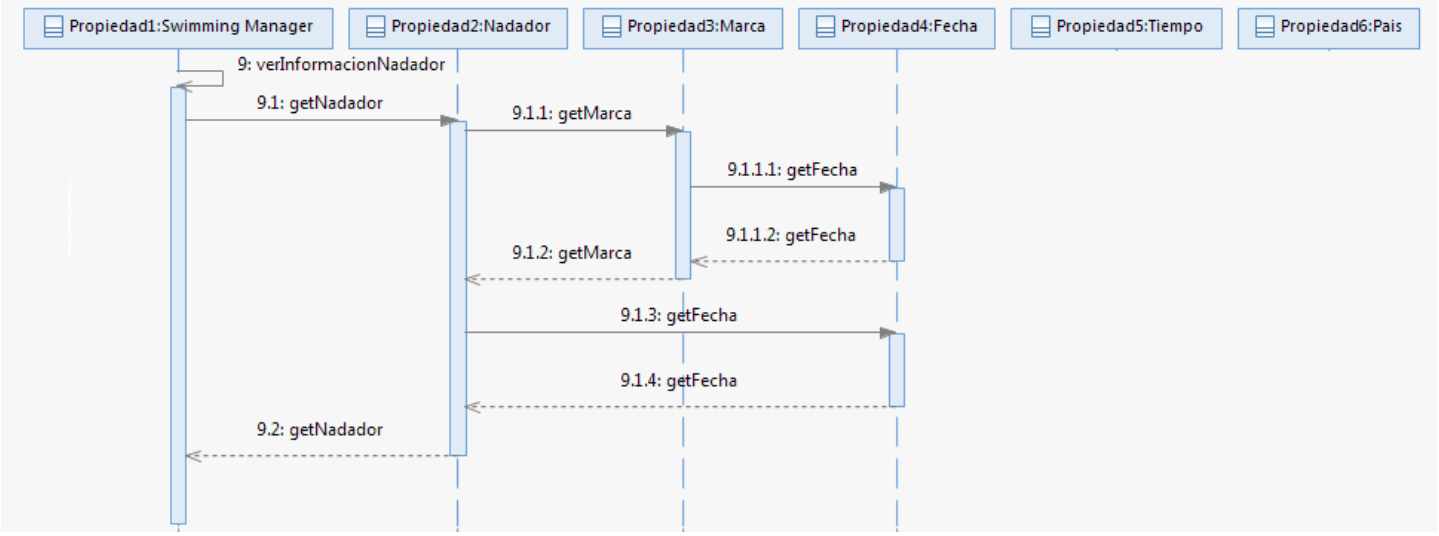
CARGAR DATOS



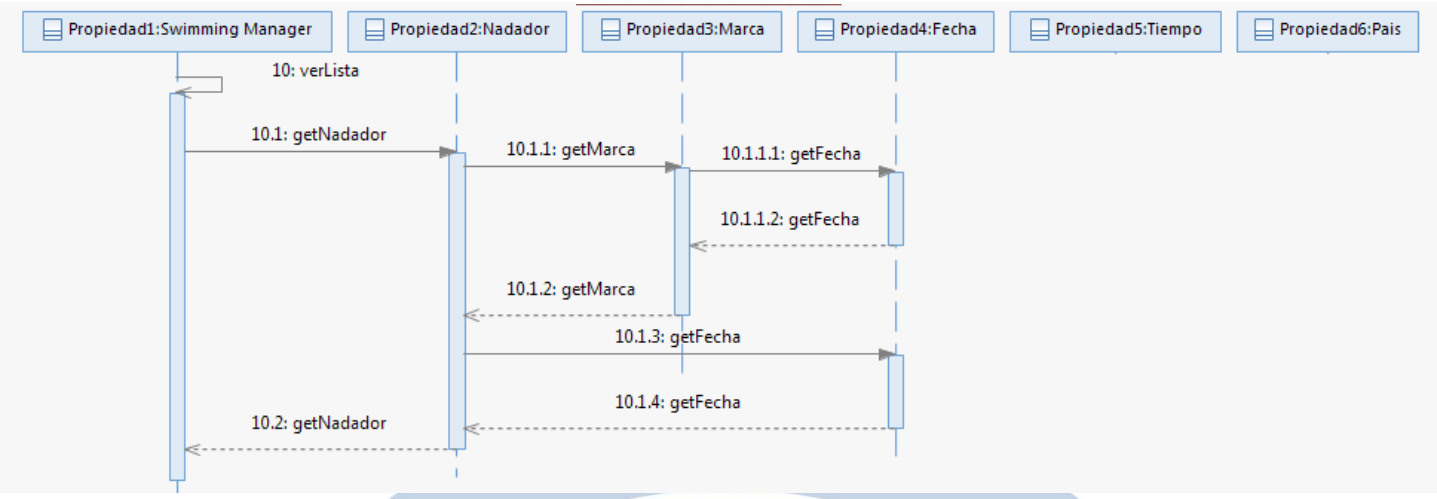
GUARDAR INFORMACIÓN



EDITAR INFORMACIÓN



MOSTRAR LISTA



IMPLEMENTACIÓN DE LA BASE DE DATOS

En este proyecto se necesita almacenar los datos de cada nadador (nombre, apellido, nacionalidad, respectivas marcas, etc.) de manera que se tenga acceso a ellos tanto para visualizarlos y modificarlos.

Para conseguirlo se usa una base de datos, en este proyecto se usa MySql.

Para incluirlo al proyecto se crean las tablas:

- Tabla 1 contiene (Datos del nadador):

1. El identificador del nadador que será único y será con el que se acceda a los datos de dicho nadador en el resto de las tablas. Es de tipo int.
2. Nombre y Apellido del nadador que ambos son de tipo string.

TABLA 1

IdNadador(int)	Nombre(string)	Apellido(string)

- Tabla 2 contiene (Datos de procedencia y fecha de nacimiento):
  1. El identificador del nadador.
  2. País de nacimiento de tipo string.
  3. Ciudad de tipo string también.
  4. Día de nacimiento de tipo int.
  5. Mes de nacimiento de tipo int.
  6. Año de nacimiento de tipo int.

TABLA 2

IdNadador (int)	País(string)	Ciudad (string)	Día(int)	Mes(int)	Año(int)

- Tabla 3 contiene(Información de los títulos de un nadador):
  1. Identificador del título de tipo int.
  2. Nombre del título de tipo string.
  3. Identificador del nadador que posee el título de tipo int.

TABLA 3

IdTítulo (int)	Nombre de Título (string)	Nombre de ganador [IdNadador(int)]

- Tabla 4 contiene (Los estilos):
  1. Identificador del estilo de tipo int.
  2. Nombre del estilo tipo string.

TABLA 4

IdEstilo(int)	Nombre de estilo(string)

- Tabla 5 contiene(Información sobre las marcas de cada nadador):
  1. Identificador de marca de tipo int.
  2. Minutos que tardó de tipo int.
  3. Segundos de tipo int.
  4. Milisegundos de tipo int.
  5. Distancia de tipo int.
  6. Día de tipo int.
  7. Mes de tipo int.
  8. Año de tipo int.
  9. Identificador de estilo de tipo int.
  10. Identificador de nadador de tipo int.

TABLA 5

IdMarca (int)	Min (int)	Seg (int)	Ms (int)	Distancia (int)	Día (int)	Mes (int)	Años (int)	IdEstilo (int)	IdNadador (int)

En la tabla 1 se almacena la información del nadador usando como identificador un integer, éste mismo identificador se usará para enlazarlo con el resto de tablas.

El atributo IdNadador se auto-incrementará (mediante instrucciones SQL) al añadir algún nuevo nadador. Se pondrán restricciones al tamaño de los campos de tipo String.

La tabla 2 contiene información complementaria del nadador (Nacionalidad, fecha de nacimiento), la cual se vincula con la primera tabla mediante el identificador del nadador (IdNadador). Al igual que en la primera tabla restringiremos el tamaño de los campos de tipo String.

Se accede a la información de cada nadador con su identificador.

En la tabla 3 encontramos una lista de títulos. La tabla contiene una IdTitulo(int) que se auto-incrementará al añadir un nuevo título.

Contiene un campo para el nombre de tipo String al que también se restringirá el tamaño.

Y por último contiene un IdNadador que servirá para saber cuál de todos los nadadores posee dicho título

En la tabla 4 tenemos la lista de estilos que hay en la disciplina de la natación. Ésta tabla tiene 2 campos: El primero, IdEstilo, se auto-incrementará y el segundo campo será el nombre, de tipo String al cual se restringirá su tamaño.

La tabla 5 será una lista de marcas (o tiempos que se consigan). Ésta tabla contiene todas las marcas de todos los nadadores (cada nadador posee un número limitado de marcas).

Su funcionamiento:

Al añadir un nuevo elemento:

- Comprueba si el IdNadador ya alcanzó su límite de marcas para almacenar:
  - Afirmativo: analiza si la nueva marca mejora alguna de las ya almacenadas(esto lo hace comparando que sean iguales el estilo y la distancia):

- Afirmativo: La nueva marca reemplaza a la marca que ha sido mejorada, esto se hará actualizando los campos de los tiempos (Min, Sec, Ms) y la fecha (Día, Mes, Año).
- Negativo: La nueva marca no mejora ninguna de las ya almacenadas, se ignora y no se almacena.

➢ Negativo: No se ha alcanzado aún el límite de marcas para almacenar, entonces simplemente añade la nueva marca.

4. TABLAS DE RIESGOS

TIPOS DE RIESGOS

ID	Riesgo	Tipo de Riesgo	Descripción
1	Falta de trabajo en algún miembro del equipo.	Proyecto.	Alguno de los miembros del grupo no es capaz de completar las tareas que tiene asignadas.
2	Incompatibilidad de hardware.	Proyecto.	El programa puede estar diseñado para un hardware específico que no corresponde con el que el usuario dispone.
3	Cambio en los Requerimientos.	Proyecto, Producto.	Puede haber un cambio en los requerimientos del producto.
4	Retraso en el cumplimiento de los plazos.	Proyecto.	Es posible que no se puedan cumplir los plazos requeridos para entregar el proyecto a tiempo.
5	Compatibilidad con el sistema operativo	Proyecto, Producto.	Es posible que el usuario no utilice el sistema operativo para el que el programa está preparado.
6	Competencia.	Negocio.	Puede haber mejores programas y más eficientes en el mercado.
7	Coordinación.	Proyecto.	Falta de coordinación a la hora de repartir las labores de cada miembro del grupo.
8	Falta de conocimiento.	Proyecto.	El nivel de conocimiento de bases de datos es insuficiente para continuar el proyecto.
9	Librerías.	Proyecto.	No se encuentran librerías suficientes para elaborar un diseño eficiente del producto.
10	Almacenamiento de datos.	Proyecto.	El método de almacenar los datos es ineficiente o no funciona.
11	Introducción de datos.	Proyecto.	El método de introducción de datos es poco intuitivo y tiene mal diseño.
12	Utilidad del programa.	Proyecto, Producto, Negocio.	El colectivo al que va dirigido el programa no le encuentre utilidad.
13	Acceso a la información.	Proyecto, Producto.	Poder acceder y manipular la información no esta restringido a ciertos usuarios.

PROBABILIDAD DE RIESGOS

ID	Riesgo	Probabilidad	Severidad
1	Falta de trabajo en algún miembro del equipo.	Frecuente	Seria
2	Incompatibilidad de hardware.	Improbable	Menor
3	Cambio en los Requerimientos.	Remota	Seria
4	Retraso en el cumplimiento de los plazos.	Probable	Crítica
5	Compatibilidad con el sistema operativo	Improbable	Catastrófica
6	Competencia.	Ocasional	Menor
7	Coordinación.	Frecuente	Seria
8	Falta de conocimiento.	Probable	Seria
9	Librerías.	Ocasional	Seria
10	Almacenamiento de datos.	Probable	Critica
11	Introducción de datos.	Frecuente	Seria
12	Utilidad del programa.	Ocasional	Catastrófica
13	Acceso a la información.	Frecuente	Seria

ESTRATEGIAS

ID	Riesgo	Superado	Descripción	Estrategia
1	Falta de trabajo de algún miembro del equipo.	No	La falta de coordinación entre algunos miembros del equipo ha dado lugar a problemas a la hora de llevar a cabo las tareas asignadas.	Reorganización del trabajo asignado a cada miembro del equipo para que se redistribuyan las tareas convenientemente.
2	Incompatibilidad de hardware.	Sí	Este riesgo no afectó a los requisitos del sistema.	Reducción de los requisitos para ejecutar el programa y maximizar la compatibilidad con él.
3	Cambio en los requisitos.	Sí	Se modificaron los requisitos con respecto a los inicialmente planteados.	Intentar aplicar el principio de la abstracción, en la medida de lo posible, de forma que cada una de las partes sea independiente a las otras y no altere en gran medida un cambio en alguna de las partes.
4	Retraso en el cumplimiento de los plazos.	Sí	Se llevó a cabo una redistribución del trabajo para cada una de las entregas.	Reorganizar el trabajo y el tiempo a dedicar por parte de cada miembro del grupo.
5	Compatibilidad con el sistema operativo.	Sí	Los requisitos mínimos del sistema están definidos en la introducción.	Alertar al cliente de los requisitos que el programa necesita.
6	Competencia.	No	Por falta de tiempo no se ha podido comparar nuestra aplicación con otras de gestión deportiva.	Realizar una búsqueda sobre otros programas disponibles que tengan relación con la aplicación en desarrollo para comparar funciones.

7	Coordinación.	No	Definimos un documento que asignaba a cada miembro una tarea, pero solo se actualizó hasta la primera entrega.	Elaborar un documento especificando la tarea de cada miembro del equipo, ciñéndose estrictamente a dicha labor.
8	Falta de conocimiento.	Sí	Cada miembro del equipo se encargó de informarse sobre cómo llevar a cabo su tarea, y dicha información se puso en común a través de la plataforma Dropbox.	Dedicar un tiempo a informarse y aprender a utilizar los recursos con los que vamos a contar a la hora de desarrollar el proyecto.
9	Librerías.	Sí	Se hicieron algunos cambios para trabajar con las librerías de que disponíamos.	Cambiar la especificación que habíamos hecho en un principio para adaptarla a la tecnología disponible.
10	Almacenamiento de datos.	Sí	Inicialmente pensamos en utilizar una base de datos, pero por falta de conocimiento y tiempo se decidió utilizar ficheros de texto.	Pensar varias formas para almacenar los datos, de manera que tengamos otra solución en caso de fallo.
11	Introducción de datos.	Sí		Pensar varias formas de realizar esta tarea de forma que tengamos estrategias alternativas en caso de problemas.
12	Utilidad del programa.	No	La falta de medios y de tiempo ha hecho imposible que podamos investigar sobre la demanda de podría tener este programa en el mercado.	Hacer un estudio de mercado, para ver si hay demanda de un programa con las características del que estamos desarrollando, así como si esa demanda está satisfecha o va a ser satisfecha a corto plazo, de modo que observemos si resulta rentable o no invertir tiempo y dinero en este proyecto.

13	Acceso a la información.	No	Por falta de tiempo no se ha podido restringir el uso del programa.	El uso de la información del programa se restringirá al usuario que lo haya adquirido, evitando así que terceros puedan manipular la información de la base de datos.

5. PLANIFICACIÓN

La planificación incluye asignar tareas y responsabilidades a cada miembro del equipo, así como hacer un seguimiento del progreso. La información que se incluye en este apartado ha sido extraída de las distintas reuniones celebradas a lo largo del tiempo.

La planificación se lleva a cabo en base a fases con una o más iteraciones en cada una de ellas, y está caracterizado por la realización en paralelo de todas las disciplinas de desarrollo, con lo cual la mayoría de los artefactos son generados al principio del proyecto pero van desarrollándose en mayor o menor grado de acuerdo a la fase e iteración del proyecto.

La fecha de entrega es finales de mayo del 2012.

FASE DE INICIO: ENTREGA DE DICIEMBRE

Se obtuvo una visión general de los requisitos principales y las restricciones más importantes, así como la valoración inicial de riesgos.

ARTEFACTOS	COMIENZO	APROBACIÓN
Especificación del sistema	Semana 21/11	Siguiente fase
Modelo de proceso	Semana 28/11	Semana 28/11
Requisitos:	Semana 12/12	
• *Funcionales*	12/12 – 16/12	Siguiente fase
• *No funcionales*	16/12 – 19/12	Siguiente fase
Diseño interfaz (*en papel*)	Semana 5/12	Revisar en cada iteración
Riesgos	Semana 1/12	
• *Tipos de riesgos*	1/12 – 3/13	Siguiente fase
• *Probabilidad*	3/12 – 6/12	Siguiente fase
• *Estrategia*	6/12 – 8/12	Siguiente fase

La fecha de aprobación indica cuándo el artefacto en cuestión tiene un estado de completitud suficiente para someterse a revisión y aprobación, pero esto no quita la posibilidad de su posterior refinamiento y cambios.

FASE DE ELABORACIÓN: ENTREGA DE MARZO

En esta fase terminamos de definir los requisitos, y se comenzó con la implementación de los elementos básicos de la arquitectura.

ARTEFACTOS	COMIENZO	APROBACIÓN
Especificación del sistema	Semana 23/01	Semana 30/01
Planificación de la estructura de datos	Semanas 30/01 – 20/02	
• *Identificación de clases*	Semana 30/01	Siguiente fase
• *Tarjetas CRC*	Semana 6/01	Siguiente fase
• *Diseño de interfaz*	Semana 13/02	Revisar en cada iteración
Riesgos	Semana 20/02	Revisar en cada iteración
Pruebas	Semana 20/02	Revisar en cada iteración
Gestión de cambios	Durante todo el proyecto	

FASE DE CONSTRUCCIÓN: ENTREGA DE ABRIL

En esta fase, los objetivos principales son la implementación iterativa de los requisitos del sistema y depurar los fallos encontrados hasta la fecha. El modelado es aditivo (se añaden más detalles a medida que la construcción progresa).

ARTEFACTOS	COMIENZO	APROBACIÓN
Planificación de la estructura de datos	Semana 5/03	Siguiente fase
Diseño UML	Semanas 12/03 – 26/03	
• *Implementación de clases*	Semana 12/03	Siguiente fase
• *Diagrama de secuencias*	Semana 19/03	Siguiente fase
Riesgos	Semana 26/03	Revisar en cada iteración
Pruebas	Semana 2/04	Revisar en cada iteración
Gestión de cambios	Durante todo el proyecto	
Prototipo semifuncional	Semana 2/04	Revisar en cada iteración

FASE DE TRANSICIÓN: ENTREGA DE MAYO

Esta fase fue la más complicada, ya que tuvimos que redefinir los objetivos del proyecto por falta de tiempo. También se modificó la estructura de datos y se terminó de adaptar la implementación. Se continuó con las pruebas.

ARTEFACTOS	COMIENZO	APROBACIÓN
Planificación de la estructura de datos	Semanas 9/04 – 23/04	
• *Modificación de clases*	Semana 9/04	Semana 30/04
• *Diseño interfaz*	Semana 16/04	Semana 7/05
Requisitos	Semanas 16/04 – 30/04	
• *Modificación Casos de Uso*	16/04 – 30/04	Semana 30/04
Diseño UML	Semanas 23/04 – 30/04	
• *Implementación de clases modificadas*	Semana 23/04	Semana 30/04
• *Diagrama de secuencias*	Semana 30/04	Semana 7/05
Riesgos	Semana 23/04	Semana 14/05
Pruebas	Semana 30/04	Semana 14/05
Gestión de cambios	Durante todo el proyecto	

5.1 PLAN DE ITERACIONES

FASE DE INICIO: ENTREGA DE DICIEMBRE

En esta fase se desarrollan los requisitos del producto desde la perspectiva del usuario. Se identifican los principales casos de uso y se hace un breve esbozo de lo que será la memoria. Se define sobre papel y a grandes trazos el diseño de la interfaz.

A finales de Noviembre, una vez definido el equipo de trabajo e identificada la idea que queríamos desarrollar, todos los miembros del grupo nos dedicamos a detallar cuáles iban a ser las características principales del sistema.

Claudia y Ángel identificaron los 10 riesgos más importantes. Se dividieron en 3 tablas, clasificándolos por el tipo de riesgo que era, la probabilidad de que ese riesgo se diera en algún momento del desarrollo del proyecto, y una última tabla en la que se establecía la estrategia a seguir en caso de que el riesgo en cuestión surgiera. Estos riesgos se actualizan semanalmente.

Juan Luis diseñó en papel la interfaz del programa. Tras una reunión con todo el equipo, se definió la forma más cómoda (en cuanto a diseño) de manejar un programa con las características ya definidas previamente. La dificultad a la hora de definir la interfaz se resolvió respondiendo a preguntas como *¿a qué estaban acostumbrados los usuarios a la hora de manejar programas? ¿Qué resultaba más sencillo?* Y partiendo de la base de que el usuario/cliente no tenía porqué saber utilizar ningún programa.

Lara realizó la introducción de la memoria, con los objetivos y una breve descripción de la aplicación, y la portada. Detalló un índice aproximado de los contenidos que deberían figurar en la memoria, y pensó en el *logo* que identificaría a la aplicación. También se encargó de redactar los principales requisitos del sistema, así como algunas restricciones que podían surgir en el proceso de desarrollo.

Al mismo tiempo que se establecían los requisitos de la aplicación, el equipo tuvo que informarse sobre cómo definir los casos de uso que Ángel fue identificando y añadiendo a la memoria (solo los identificables a estas alturas) . También definimos el modelo de proceso que íbamos a usar.

Marta desarrolló los casos de uso y actores identificados, y concretamos un plan de proyecto, planificando las fases, las iteraciones y una aproximación de la duración de las mismas.

Esta fase no tiene iteraciones reales, ya que no se produce software. Por último, se decidió que en principio no se definiría un glosario con términos utilizados a menos que diera tiempo en la última fase del desarrollo.

FASE DE ELABORACIÓN: ENTREGA DE MARZO

En esta fase se analizan los requisitos y se desarrolla un prototipo de arquitectura (incluyendo las partes más relevantes y/o críticas del sistema). Al final de esta fase, todos los casos de uso correspondientes a requisitos que serán implementados en la fase de Construcción deben estar analizados y diseñados.

Lo primero que tratamos al empezar en esta fase fue resolver los fallos que había en la especificación del sistema, así como incluir algunos datos que faltaban.

Al mismo tiempo ya se había decidido que utilizaríamos la plataforma Dropbox para poner el material, tanto de información como sobre el proyecto en sí, obtenido individualmente en común a todos los miembros del equipo.

Juan Luis se encargó de esa gestión de configuración. Se decidió usar Google Code. Aquí fue donde más problemas surgieron, ya que prácticamente ningún miembro del equipo estaba familiarizado con Google Code y su uso. Empleamos bastante tiempo en informarnos sobre cómo manejarlo.

Al mismo tiempo, se fue añadiendo todo lo relacionado con la gestión en la memoria del proyecto.

En esta fase se inició la planificación de la estructura de datos, teniendo que comenzar con el diseño de las tarjetas CRC y aprender a manejar el programa IBM Rational Software Arquitect. Llegados a este punto, destinamos mucho tiempo a aprender a utilizar el programa, ya que nunca habíamos trabajado con UML.

Lara, Jaime, Marta y Claudia fueron los encargados de prepararse en todo lo relacionado con UML, los diagramas de secuencias que se incluirían en la memoria y la representación de los casos de uso (actores, etc). Viktor y Charles trabajaron en la creación del prototipo de la interfaz.

Se revisó la lista de riesgos iniciales y se añadieron algunos nuevos a las tablas. Ángel se encargó de empezar con el apartado de pruebas con la interfaz, ya que al final de esta fase contábamos con un prototipo arquitectónico ejecutable.

En nuestro caso particular, esta fase cuenta con dos iteraciones. La primera tuvo como objetivo terminar de definir los requisitos y ajustar algunos factores nuevos: nueva tecnología, como el uso de programas como el IBM o Google Code; y nuevo lenguaje de programación, ya que acabábamos de empezar a programar en Java y decidimos que ese sería nuestro lenguaje. En la segunda iteración se empezó con el prototipo del sistema, y se añadieron todos los cambios hasta la fecha en la memoria. También se ajustó la planificación para asegurarnos de cumplir los plazos establecidos y los objetivos.

FASE DE CONSTRUCCIÓN: ENTREGA DE ABRIL

Durante la fase de construcción se terminan de analizar y diseñar todos los casos de uso. Al principio casi todo el equipo se centró en depurar los fallos encontrados hasta la fecha, aunque lo que más tiempo nos quitó fue la modificación de la estructura de datos del sistema debido a fallos que aparecieron y no teníamos previstos, y a restricciones. También influyó en esta modificación el hecho de que planeamos cada fase, cada iteración y sus objetivos pensando que dispondríamos de más tiempo.

En esta fase nos centramos en la modificación de los casos de uso, lo que implicó después una revisión de los diagramas de secuencias y la implementación de las clases. Todo esto surgió al ir desarrollando el prototipo del sistema y ver que no disponíamos de tiempo suficiente para implementar todas las clases definidas inicialmente.

Esta fase cuenta con dos iteraciones, una en la que se modificaron parte de los requisitos de la aplicación, y la segunda centrada en descubrir los errores de nuestro sistema tras pasar las pruebas correspondientes.

A la hora de programar, a cada miembro del equipo se le asignaron unas clases y métodos específicos, todo ellos gestionado a través de Google Code y sincronizado por Gmail a través de NetBeans. Esto permitía que todos los cambios quedaran registrados, y que el trabajo individual resultara relativamente sencillo.

Se actualizaron los riesgos, y volvimos a tener que utilizar el programa IBM para rehacer el diagrama de secuencias. Al mismo tiempo, se fueron actualizando todos los cambios en la memoria, y se decidió descartar definitivamente la propuesta de incluir un glosario de términos en la documentación.

Se terminó de describir la gestión de configuración en la memoria del proyecto, y se añadió a la misma un apartado que incluía el entorno tecnológico del sistema: en una tabla se definieron los requisitos en cuanto a equipo físico, y en otra en cuanto a equipo lógico.

En cuanto a la planificación, se empezó anotando fechas y tareas de forma muy general en la memoria.

FASE DE TRANSICIÓN: ENTREGA DE MAYO

El hito que marca el fin de esta fase incluye la entrega de toda la documentación del proyecto.

Al principio de la fase nos centramos en actualizar el diagrama de casos de uso con las nuevas clases añadidas tras la fase anterior, y en resolver los errores cometidos en la entrega anterior (ya que faltaba el atributo sexo en la clase nadador). Tuvimos que modificar los casos de uso quitando atributos obsoletos (como *ID Nadador*), y terminar de adaptar la implementación a fin de tenerla lista para la última entrega.

Nos encargamos de depurar los errores del código utilizando Google Code y volviendo a dividir el trabajo que debía realizar cada uno. Necesitamos varias reuniones con todo el equipo para llegar a un acuerdo sobre cómo debíamos trabajar y cuánto tiempo al día requería el proyecto, así como resolver dudas y plantear posibles soluciones.

Las tablas de riesgos se modificaron para ajustarse a las últimas semanas de cambios, y finalmente se aprobó añadir una columna más en cada tabla, *ID*, donde estuviera el número de riesgo y resultara más fácil identificarlo y comprarlo con el resto de tablas. En la tabla de estrategias a seguir en caso de que se diese el problema descrito en cualquier riesgo, se añadió otra columna más, *Superado*, en la que se indicaba si el riesgo había sido o no superado a lo largo del desarrollo del proyecto, y una breve descripción de porqué.

Se continuó con las pruebas sobre la aplicación y se actualizó la memoria. Se creó la base de datos y se usó la interfaz Java Connection para conectarla.

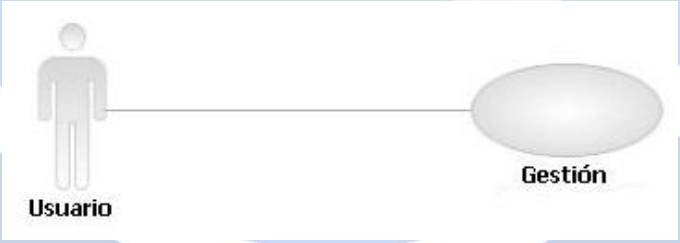
Inicialmente esta fase iba a contar con una única iteración, pero con los errores descubiertos tras algunas pruebas todo el equipo pensó en añadir otra iteración más y depurar así los fallos.

5.2 ALCANCE DEL PROYECTO

El alcance del proyecto es la suma total de todos los artefactos y sus requisitos o características. Este apartado abarca el sistema en su totalidad, por lo tanto todas las funcionalidades incluidas en este documento serán las funcionalidades que la aplicación "Swimming Manager" ofrecerá una vez finalizado.

En la gestión del proyecto, la herramienta primaria que utilizamos para describir el alcance del trabajo es la estructura de descomposición del trabajo. El detalle de las iteraciones individuales se describe en los planes de cada iteración, en el apartado de Planificación.

Nuestra aplicación no llega a gestionar un control de usuario, aunque en un primer momento pensamos en restringir su uso, pero por falta de tiempo cualquier usuario podrá entrar en la gestión de los nadadores y en sus distintas sub-gestiones: nadadores (dar de alta, dar de baja, editar, eliminar), marcas (añadir, editar). Los distintos perfiles de usuario no están definidos en la aplicación.



En la primera iteración que llevamos a cabo, todo el equipo coincidía en que, con esfuerzo y organización, el sistema que ya habíamos definido algunos días antes podía llegar a implementarse en su totalidad, es decir, se podría tener una buena documentación del proyecto y una aplicación real que funcionara (con el código en java sin errores y depurado).

Tras las primeras semanas de trabajo (elaboración de listas de riesgos, identificación de los casos de uso, etc.) pensamos que el objetivo que nos habíamos planteado anteriormente iba a superarnos, ya que no contábamos con el nivel requerido para desarrollar la aplicación como decidimos en un principio (nivel de programación en java alto, manejo de bases de datos). Determinamos, por tanto, que debíamos reducir los casos de uso definidos hasta el momento, modificando con ello todo lo ya elaborado hasta la fecha: diagrama de secuencias, tarjetas CRC, identificación de las clases, etc. Decidimos que tampoco utilizaríamos bases de datos para almacenar y gestionar toda la información sobre los nadadores, sino que cargaríamos y guardaríamos dicha información en ficheros de texto, con los que ya estábamos familiarizados. También establecimos que en la parte de programación nos daría tiempo únicamente a tener preparados y funcionando los casos de uso definidos tras esta modificación.

Sin embargo, durante la Fase de Construcción, nos dimos cuenta de que podíamos extender y mejorar todas las características de la aplicación (salvo la parte de utilizar bases de datos, que quedó descartada por falta de tiempo y de conocimiento), lo que nos llevó a tener que reorganizarnos otra vez y volver a cambiar la estructura del sistema para incluir, sobre todo, más código java y más clases funcionando. Prácticamente llegamos al punto de haber desarrollado lo definido al principio de la planificación.