

Documentación Protectora – Práctica IPO I

Alberto Vázquez Martínez

Alberto.Vazquez1@alu.uclm.es

Ángel Villafranca Iniesta

Angel.Villafranca@alu.uclm.es

Grupo C1

Curso 2021/22 – Interacción Persona-Ordenador I

Universidad de Castilla-La Mancha

Escuela Superior de Informática



Abstracto

Este documento mostrará la diferentes etapas y decisiones que se han realizado para desarrollar una aplicación para la gestión de una protectora de animales utilizando la tecnología WPF para así lograr un prototipo de una aplicación interactiva de escritorio enfocada a los trabajadores de dicha protectora.

Contenido

1. Introducción.....	3
2. Análisis de requisitos.....	3
2.1. Casos de uso y actores.....	3
3. Bocetos de aplicación.....	4
3.1. Panel de acceso.....	4
3.2. Ventana de gestión principal.....	5
3.3. Añadir/editar ventanas.....	6
4. Tecnología y recursos utilizados.....	6
5. Justificación del diseño de la GUI.....	7
5.1. Leyes de Gestalt.....	7
5.1. Uso de TabControl.....	8
5.2. Patrón master-detail.....	9
5.3. Listas de tablas.....	9
5.4. Diseño de botones.....	10
5.5. Colores de la aplicación.....	11
6. Manual de usuario.....	11

1. Introducción

El objetivo de esta práctica es el diseño y desarrollo de una aplicación para la gestión de una protectora de animales, más concretamente para la gestión de perros, socios y voluntarios de esta. Esta aplicación estará enfocada para el uso de los gestores o trabajadores de la protectora y no para cualquier usuario.

Para llevar a cabo esta tarea se utilizará la tecnología de WPF para desarrollar y diseñar la interfaz de usuario.

2. Análisis de requisitos

Los requisitos funcionales que podemos extraer el documento *pdf* proporcionado son los siguientes:

- RF1: Acceso a la aplicación con credenciales
- RF2: Salir en cualquier momento
- RF3: Mostrar información de usuario
- RF4: Mostrar ayuda
- RF5: Gestión de listado de perros.
- RF6: Gestión de voluntarios de la protectora.
- RF7: Gestión de información de socios.

2.1. Casos de uso y actores

De estos los requisitos funcionales mencionados, extraemos diferentes casos de uso para cada uno de ellos:

- CDU1: Acceso/autenticar mediante usuario y contraseña (panel de *login*)
- CDU2: Botón de *logout* de la aplicación.
- CDU3: Mostrar información de usuario
- CDU4: Mostrar ayuda
- CDU5: Añadir perros
- CDU6: Eliminar perros
- CDU7: Editar perros
- CDU8: Listar perros
- CDU9: Añadir voluntarios
- CDU10: Eliminar voluntarios
- CDU11: Editar voluntarios
- CDU12: Listar voluntarios
- CDU13: Añadir socios
- CDU14: Eliminar socios
- CDU15: Editar socios
- CDU16: Listar socios

En este caso el único actor que se identifica es el gestor o trabajador de la protectora. El diagrama de casos de uso resultante es el siguiente:

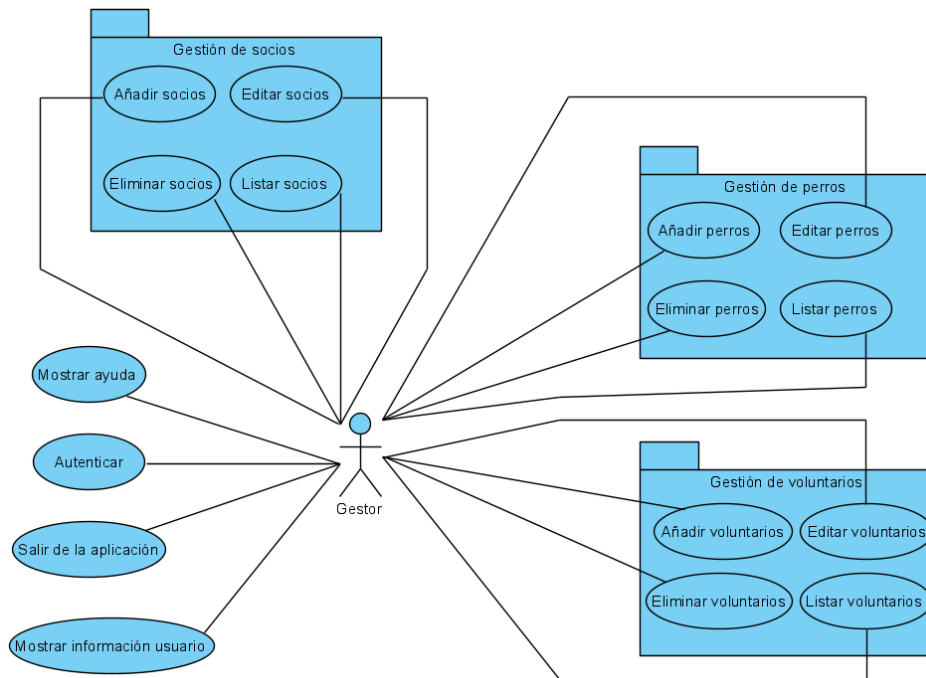


Fig. 1 - Diagrama de casos de uso

3. Bocetos de aplicación

3.1. Panel de acceso

Debido a que nuestra aplicación necesita un panel de *login*, la primera idea de un panel de *login* que se nos ocurre es la siguiente:



Fig. 2 - Primera aproximación panel *login*

En este *login* como podemos ver tenemos dos campos correspondientes al campo de usuario y al campo de la contraseña. También se añadiría el correspondiente botón de acceso para validar las credenciales. Además, añadir un icono que haga referencia a un usuario ayuda al entendimiento de la finalidad de esta ventana, indicando el acceso a usuarios a la aplicación.

3.2. Ventana de gestión principal

La idea principal es la tener en una ventana principal todas las opciones requeridas por la aplicación y que conforme el usuario utilice la aplicación se vayan desplegando nuevas ventanas para los diferentes casos de uso.

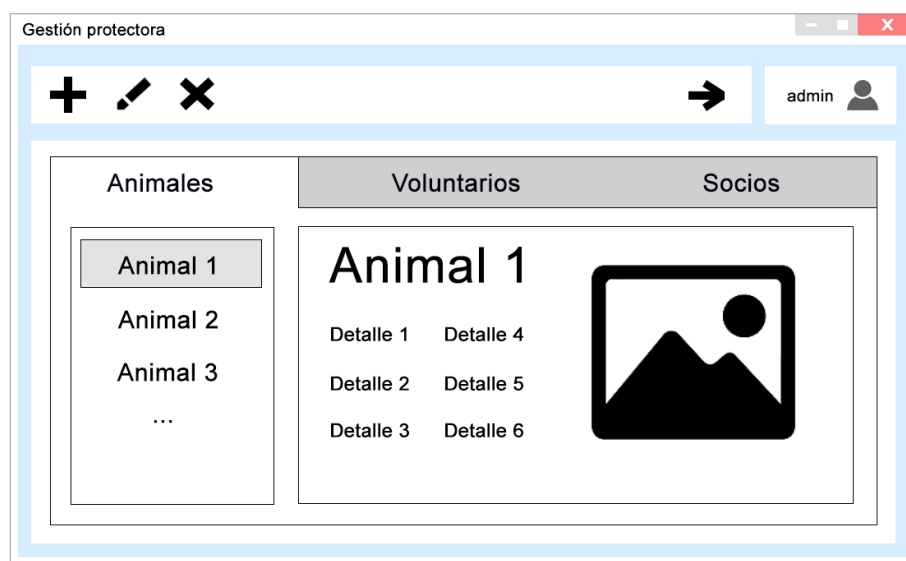


Fig. 3 - Primera aproximación ventana gestión principal

En esta ventana tendríamos acceso a todas las secciones en las que dividirá la aplicación las cuales serían: gestión de animales, gestión de voluntarios y gestión de socios, donde tendríamos unos botones en la parte superior que controlarían la lógica de añadir, editar, eliminar y salir de la aplicación.

También se podría utilizar un patrón *master-detail*, donde en una parte se mostraría la lista de elementos y al otro lado información más detallada sobre el elemento que el usuario haya seleccionado. Dentro de las secciones de detalles, se mostraría características de los animales, socios o voluntarios, imágenes y/o vídeos sobre los mismos.

Otra forma de mostrar la información sería a través de tablas con diferentes columnas, y que cada celda sea editable. Así, no sería necesario utilizar los botones superiores para editar la información.

3.3. Añadir/editar ventanas

Este tipo de ventanas contendrían diferentes cuadros de texto donde el usuario, en este caso el gestor, introduciría o modificaría los campos solicitados por la aplicación. Se incluirían diferentes tipos de formas para que el usuario pueda introducir información como *radio buttons*, menús desplegables, selección de imágenes, etc.



Fig. 4 - Primera aproximación ventana para añadir/editar

4. Tecnología y recursos utilizados

Debido a que no se usará ningún tipo de base de datos para el almacenamiento de la información, se utilizarán archivos XML que contendrán datos previos para realizar un uso de la aplicación. Tendremos los siguiente archivos XML asociados al proyecto:

- Animals.xml: contendrá registros de animales.
- Sponsors.xml: contendrá registros de las personas que apadrinan animales.
- Volunteers.xml: contendrá registros de los voluntarios de la protectora.
- Partners.xml: contendrá registros de los socios de la protectora.

Todos estos archivos estarán almacenados en un directorio llamado *Data* que estará asociado al proyecto WPF. Para mostrar imágenes y vídeos sobre los animales, socios, voluntarios, etc, se creará un directorio llamado *Images*, que contendrá este tipo de archivos.

Como se ha comentado anteriormente, la tecnología utilizada para el desarrollo de la aplicación será WPF donde se diseñará y desarrollará el software, utilizando XAML para la parte de diseño y C# para la parte de la lógica del programa.

Para los bocetos del diseño de la interfaz de la aplicación se ha utilizado Adobe Photoshop CS6 debido a su versatilidad y libertad a la hora de crear cualquier tipo de elemento dentro de la interfaz y así sea lo más fiel posible a la idea que tenemos.

5. Justificación del diseño de la GUI

5.1. Leyes de Gestalt

Para el diseño de la interfaz gráfica se ha tenido en cuenta las leyes de Gestalt. Algunos ejemplos del cumplimiento de estas leyes son los siguientes:

A la hora de listar los animales, cuando se muestran información detallada sobre los mismos, se cumple la ley de proximidad y similitud para que así, los ítems cercanos se perciban como agrupados.



Fig. 5 – Ejemplo ley Gestalt 1

A la hora de desarrollar los menús de añadir/editar elementos, se cumplen las leyes de proximidad y de región común.

Añadir animal

Macho

PPP
Si
No

Esterilizado
Si
No

Soc perros
Si
No

Soc niños
Si
No

Proximidad y región común para distinguir entre tres tipos de secciones en el formulario

Fig. 6 – Ejemplo ley Gestalt 2

En el menú de gestión de animales (perros), también vemos que se cumple la ley de Gestalt de cierre, para así separar la parte de la lista de animales con la de los detalles del animal seleccionado.



Fig. 7 – Ejemplo ley Gestalt 3

5.1. Uso de TabControl

Hemos decidido optar por utilizar la utilidad del *TabControl* que proporciona XAML, de tal manera que podamos tener diferentes pestañas dentro de una misma parte de la interfaz gráfica.

Las pestañas estarán divididas para cada uno de los requisitos funcionales R5, R6 Y R7, los cuales se corresponden a:

- Gestión de animales
- Gestión de voluntarios
- Gestión de socios

Además, se ha añadido una pestaña para la ayuda del usuario (R4), donde se encontrará una pequeña descripción del funcionamiento de cada una de las pestañas.



Fig. 8 – Uso de TabControl

5.2. Patrón master-detail

En nuestro caso se ha utilizado el patrón *master-detail* para la pestaña de la gestión de animales de la protectora. En este patrón identificamos dos partes, en la parte izquierda estará la parte del *master*, donde tendremos una *listbox* con los diferentes animales que estén registrados actualmente, y en la parte derecha tendremos la parte del *detail*, la cual variará dependiendo del animal seleccionado y donde se mostrarán las diferentes características del animal, así como unos botones que nos permitirán acceder a más información y datos sobre el animal seleccionado.

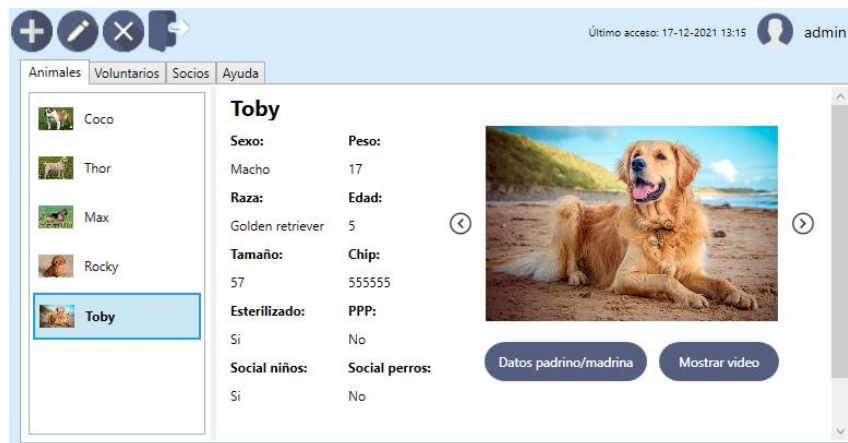


Fig. 9 – Patrón master-detail en la pestaña de gestión de animales

5.3. Listas de tablas

Otra forma de mostrar la información que hemos utilizado son los *DataGrid* donde podemos establecer las columnas y crear *DataGridTextBoxColumn* que contendrán los diferentes datos extraídos de los archivos XML. De esta forma mostramos mucha información sobre elementos que no contienen muchos atributos en poco espacio. La ventaja de este tipo de forma de mostrar información es que se pueden editar los campos desde la propia tabla haciendo doble clic en el campo.






Dni	Nombre	Apellido	Edad	Fotografia	Teléfono	Email	Disp. horaria	Zona	Estudios
11111111	Marcelo	Lopez	68		654876321	mlopez@mail.com	8:00-14:00	Valdepeñas	Grado en veterinaria
22222222	Maria	Jimenez	34		683632323	mjimenez@mail.com	16:00-20:00	Ciudad Real	Grado en veterinaria
33333333	Roberto	Alarcon	43		670342745	ralarcon@mail.com	8:00-14:00	Almagro	Auxiliar veterinario
44444444	Manuel	Cantero	28		645321323	mcantero@mail.com	8:00-14:00	Daimiel	Grado en veterinaria
55555555	Laura	Cabellos	38		693313343	lcabellos@mail.com	16:00-20:00	Ciudad Real	Auxiliar veterinario

Fig. 10 – Lista de voluntarios de la protectora

5.4. Diseño de botones

Para el diseño de los botones se ha optado por utilizar botones circulares o en botones que tienen una forma rectangular utilizar bordes redondeados ya que según los expertos los rectángulos con bordes redondeados son más fáciles de ver que los que tienen los bordes “afilados”, ya que los redondos toman menos esfuerzo cognitivo para procesar visualmente, debido a que la fóvea procesa más rápido círculos. Puedes encontrar más información en [este artículo](#).

Para llevar esto a cabo, se ha creado un diccionario de recursos (*Styles.xaml*) que contenga ese estilo con una *key* asociada llamada *FocusVisual*, por tanto, cada vez que creamos un botón, le asociaremos dicho estilo.

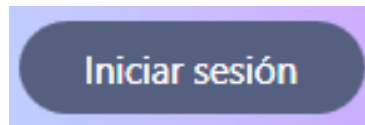


Fig. 11 – Ejemplo de botón de inicio de sesión

En otros botones, simplemente se han asociado unas imágenes descriptivas para que la funcionalidad se muestre de manera implícita. Por ejemplo, en los botones para añadir, editar, eliminar y salir utilizar formas como el signo de un más para representar “añadir”, la forma de un lápiz que representa “editar”, la forma de una cruz que representa “eliminar” y la forma de una puerta abierta que representa “salir”.

Además, cada uno de los botones tiene un *tooltip*, lo cual hace que, si el usuario coloca el cursor del ratón sobre los iconos, se mostrará un mensaje indicando la funcionalidad de este.



Fig. 12 – Botones añadir, editar, eliminar y salir




A parte del diseño gráfico de los botones mencionados, estos tienen una particularidad y es que son botones dinámicos, es decir, cambian su funcionalidad dependiendo de la pestaña en la que se encuentre el usuario. Por tanto, estos botones siempre se mostrarán en la ventana principal de gestión.



Fig. 13 – Botones dinámicos

5.5. Colores de la aplicación

Los colores seleccionados en esta aplicación son diferentes tonalidades de azules. Esto es debido a que el color azul es el más usado para el diseño de interfaces gráficas como por ejemplo usan las aplicaciones de Facebook, Twitter, Safari, etc, aunque también se ha utilizado tonalidades de morado. En [este recurso](#) podemos encontrar más información sobre la importancia y preferencia de colores en una interfaz gráfica de usuario. Por lo tanto, las tonalidades escogidas son las siguientes:

-  • Azul cielo: #FFD8ECFF
-  • Azul marino: #FF556080
-  • Morado claro: #FFB08CBC

En el caso de que en alguno de los formularios de la aplicación existan errores, entonces se utilizarán tonos rojizos para indicarle el fallo al usuario.

También cabe destacar que hay partes de la aplicación que se han dejado con un color blanco debido a que un exceso de estos colores en la interfaz no sería muy estético y el blanco siempre será el color utilizado como alternativa para estas situaciones.

6. Manual de usuario

El manual de usuario se encuentra en los siguientes recursos:

- Ver online desde YouTube: <https://www.youtube.com/watch?v=kXt1KngolSw>
- Descargar vídeo (220 MB):
https://mega.nz/file/2epjSChA#myEHxVJkipYOpayfrWV_llo9v78NaeverxF8Re5wPJc

Para ejecutar el programa, se puede utilizar el ejecutable encontrado en el siguiente directorio: */bin/x64/Debug/net5.0-windows/Protectora.exe*.

Puede que para ejecutar la aplicación desde el ejecutable sea necesario desactivar el antivirus de Windows y el SmartScreen.

En caso de que no funcione por algún motivo, entonces, abre el archivo de Visual Studio: *Protectora.sln*.

Las credenciales para acceder son las siguientes:

- Usuario: admin
- Contraseña: password

Descarga el proyecto WPF desde este enlace (375 MB):

https://mega.nz/file/rPx30KRL#h9BZ1L_VWBzPq5Ckx9EegoS4OqdGxb_MeHgxrgtIIGI