ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

Titulación:

INGENIERO TÉCNICO DE TELECOMUNICACIÓN, ESPECIALIDAD SONIDO E IMAGEN

Título del proyecto:

DISEÑO Y DESARROLLO DE COMPLEMENTOS QUE FACILITEN EL USO DE UN INTERFAZ TÁCTIL PARA DISCAPACITADOS CON PROBLEMAS DE COMUNICACIÓN.

Elisabeth Esandia Melero
Dr. Carlos Fernández Valdivielso
Pamplona, 29 de Julio de 2011

Agradecimientos.

Y la niña creció, después de todos los juegos, la inocencia, la compañía, las grandes caricias de sus padres y el amor incomparable de su hermano.

Y se forjó un escudo, grande pero débil, para ser mayor, pues no quería serlo y había que disimularlo.

Detrás de aquella armadura seguía siendo una niña, con sus juegos, su inocencia, su compañía, las grandes caricias de sus padres y el amor incomparable de su hermano.

Sólo algunos que la rodearon supieron quitárselo una y otra vez para hacerla sentir como antes... natural.

Ahora era una adulta, con su trabajo, sus conocimientos, su independencia, el gran apoyo de sus padres y el amor siempre incomparable de su hermano.

Y siempre, siempre, siempre... se sentía como una niña, rodeada de gente especial.

Gracias a Vosotros.

Índice.

Introducción.

Día a día el ser humano intenta mejorar su forma de vivir, su forma de trabajar, su forma de descansar. Trata de evolucionar constantemente para hacer de este mundo algo mejor y más justo, intentando otorgar a todas las gentes de las mismas oportunidades y recursos. De lo mejor para todos.

Como gran desarrollo mundial, ha de destacarse la rapidez con la que los avances tecnológicos se han hecho hueco dentro de la propia evolución del hombre. Transporte, cirugía, información, ocio, … pero sobretodo, han tenido lugar en el ámbito de la comunicación.

Se ha conseguido que la mayoría de las personas pueda conversar, tratar con alguien de palabra o por escrito con gran facilidad y velocidad, cambiando el correo ordinario por el electrónico o el teléfono fijo por el móvil. Simplemente se ha logrado una mayor eficacia a la hora de que el ser humano esté siempre en contacto con sus semejantes.

Figura 1. Mundo de comunicaciones

Aunque es cierto que sólo la mayoría. Un pequeño porcentaje de la población que no tiene las mismas capacidades que el resto, no ha podido hacerse con las nuevas mejoras. A este grupo se le es referido como *personas discapacitadas*.

Es un hecho que las TIC (Tecnologías de Información y Comunicación) han entrado en el campo de la discapacidad, pero no lo han hecho en profundidad puesto que aún están desarrollándose, y sobretodo no están cubriendo actualmente todas las áreas demandadas por éste grupo, siendo más efectivas en temas de audición y visión.

El presente proyecto surge como complemento a un trabajo anterior, centrado en el tema comunicativo.

En él se trata de buscar y desarrollar una solución para este tipo de personas que mejore su calidad de vida y la de los que le rodean, permitiéndoles darse a entender allá donde vayan, donde quiera que estén.

Forma parte de un trabajo común en el que tres proyectandos del ámbito ingenieril han realizado sus tareas independientes y luego se han integrado de forma unificada.

También hay que destacar la formación de un equipo de trabajo multidisciplinar, compuesto principalmente por colaboradoras de instituciones especializadas en discapacidad (Anfas e Isterria), familiares directos de personas con dificultades y los técnicos encargados de la implementación como proyectandos de esta tarea.

Todas estas personas han aportado sus conocimientos pedagógicos, psicopedagógicos, vivencias y experiencias personales, las cuales han marcado las pautas a seguir en el proyecto conjunto.

En resumen, el proyecto compuesto se basa en el desarrollo de una aplicación para un dispositivo de interfaz táctil e inalámbrico que abarque las carencias demandadas por el colectivo de la discapacidad orientado a los problemas de comunicación.

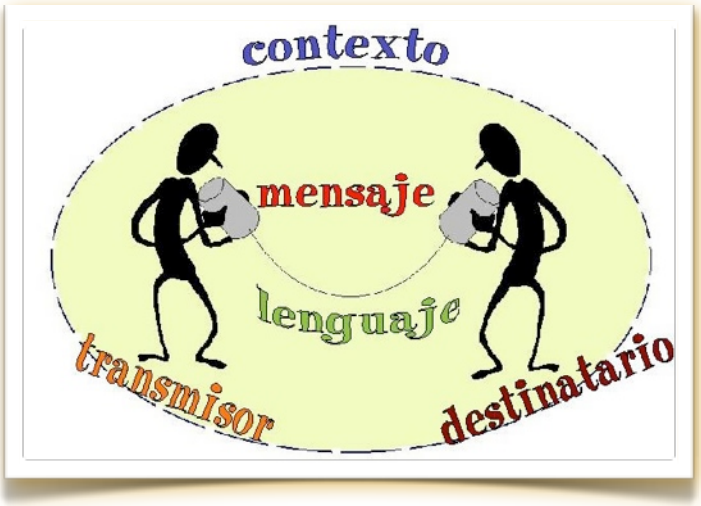

Figura 2. Método de comunicación.

Una vez cubierta la necesidad principal de *"Descubrir, manifestar o hacer saber a alguien algo"*, se ha complementado la educación y proceso de aprendizaje de estas personas con la creación de más herramientas integradas en el mismo sistema.

Se puede decir entonces que éste es un trabajo de integración de capacidades y coordinación en el que diferentes personas han participado activamente al desarrollar partes del proyecto bien diferenciadas. Es aquí donde entra a formar parte directa el presente proyecto.

A lo largo de estas páginas se pueden ver diferentes capítulos explícitamente desarrollados que pretenden explicar todo el trabajo llevado a cabo a lo largo de estos 10 meses.

En el primer capítulo se habla de los antecedentes que motivaron el desarrollo de este proyecto. De las aplicaciones existentes y del trabajo previamente realizado por otros proyectandos.

En el capítulo 2 se desarrolla lo que fue la preparación inicial. Se conocen en profundidad las tareas que se llevan a cabo y se hace una breve introducción sobre los conceptos que se deben tener claros a la hora de crear una aplicación, que en este caso se desconocían.

Posteriormente se procede a explicar cómo fue la etapa del estudio de la programación. No sólo es importante tener claros unos conceptos iniciales, sino que es necesario saber con qué elementos se cuentan y cómo se pueden manejar para poder trabajar con fluidez.

En el cuarto capítulo se detalla el desarrollo de las mini-aplicaciones realizadas. Se explica el diseño, implementación y funcionamiento de las aplicaciones reloj, álbum y agenda, y se va guiando al lector por el progreso del proyecto así como las decisiones tomadas en todo momento.

A continuación se indican las conclusiones finales a las que se llega tras el desarrollo completo del proyecto, tanto profesionales como generales.

En el capítulo 7 se comentan las líneas futuras que pretende el proyecto conjunto, la ampliación de la aplicación, la mejora de las opciones presentes y el objetivo del desarrollo como empresa del trabajo realizado en equipo.

Por último se reserva el espacio final de este documento para resumir el coste de elaboración de este trabajo.

Anexos a estas páginas se encuentra la bibliografía utilizada, así como todo el código implementado en este proyecto.

MOTIVACIÓN

Si se profundiza en el mundo de la discapacidad, normalmente se encuentran diversos patrones de dificultad que se repiten y se pueden mejorar estimulándolos de manera eficaz y en los que también se pueden buscar soluciones tecnológicas.

Destaca por ejemplo la obsesión por el orden y la rutina, y la dificultad de soportar cambios en las tareas, quehaceres o situaciones sin razón aparente. Este tipo de personas no entiende porqué un juego acaba, o porqué un trabajo empieza en un determinado momento. Por ello, es importante contar con un recurso poderoso con el que poder guiar a la persona con discapacidad y enseñarle ese cambio, esa rutina, el porqué de cada situación.

Se habla de una planificación de eventos, de una disposición de horas diarias, semanales o mensuales mediante el mismo sistema que se usa para que se dé la comunicación. Incluso de un elemento indicador que ayude a finalizar la tarea que se esté realizando.

Asimismo, se puede decir que un problema persistente en este tipo de personas es que al no haber comunicación, hay *"ausencia de aprendizaje"*, y si se comienza por algo, es por aprender el lenguaje con el que se va a comunicar. Se pueden enseñar elementos, verbos, lugares, que tengan lugar en la conversación, y así facilitar el uso del dispositivo principal de comunicación, así como la asimilación de la lengua.

De igual manera es necesario contar con una biblioteca completa de imágenes que conformen el lenguaje de comunicación.

Se analiza entonces la posibilidad de incorporar al proyecto inicial, un reloj, una agenda, un diario y un álbum de fotos donde mostrar los elementos de la comunicación.

Para finalizar, es importante decorar todo esto en un mismo entorno, y aplicar el diseño adecuado para que visualmente el proyecto dentro del dispositivo táctil sea atractivo, fácil de manejar y sobretodo intuitivo. Es por ello que se buscan unos gráficos simples sin recargar y una navegación fácil, sin complicaciones.

Por estas razones y por muchas más, es por lo que se decide seguir adelante expandiendo el proyecto origen. Aún queda mucho por hacer y mucho por conseguir.

El mero hecho de hacer posible que las personas con dificultades puedan mejorar su forma de vivir es motivación suficiente como para desarrollar este proyecto.

OBJETIVOS

Principalmente los objetivos que componen este proyecto son:

- Mejorar la calidad de vida de las personas discapacitadas. Hacer todo lo posible por cubrir las demandas que éste sector solicita complementando un proyecto anterior.

- Establecer un elemento señalizador que facilite el entendimiento del concepto tiempo y que controle el inicio y fin de las actividades deseadas. A su vez debe ser visualmente sencillo y atractivo.

- Crear un álbum de fotos con el que posibilitar el aprendizaje de los pictogramas iniciales y/o nuevos introducidos, para facilitar la introducción al comunicador de personas con problemas.

- Definir una agenda estructurada mensual, semanal y diariamente que, mediante pictogramas, ayude a la organización temporal del usuario en cuestión. Así mismo, diseñar un diario que el usuario sea capaz de rellenar con pictos y mediante el cual favorecer la comunicación en todo caso al poder mostrar lo realizado diariamente a cualquier otra persona.

- Crear las miniaplicaciones con las herramientas suficientes, que sean manejables y funcionales para personas con algún tipo de discapacidad en la comunicación y más problemas derivados de ella.

- Unificar el proyecto dentro de un mismo entorno gráfico, diseñando una interfaz común, sencilla e intuitiva.

- Realizar una librería completa con los elementos que el usuario identificará como parte de su sistema de comunicación. Serán los pictogramas que formen parte de la base de datos, corazón de la aplicación.

Capitulo 1: Antecedentes

Hasta ahora el método de comunicación de las personas con algún tipo de discapacidad es simple pero rudimentario: Los dibujos (pictogramas) dispuestos en cartulinas representan elementos, personas, lugares o verbos. Se guardan dentro de una bolsa, y la persona en cuestión los busca, los elige y los distribuye de forma ordenada sobre una superficie horizontal, o vertical enganchándolos con velcro.

Así por ejemplo, un niño con discapacidad en el habla, que quiera comunicarse con su educador en el colegio o con sus padres en casa, tiene que llevarse *"la bolsa de los pictos"* a donde quiera que vaya y buscar entre las más de 1500 imágenes diferentes con las que cuenta un sistema SPC (Sistema Pictográfico de Comunicación).

De esta forma, el usuario consigue expresar lo que quiere decir, pero de una forma lenta y costosa.

El presente proyecto es un complemento de un trabajo anterior, en el que ya se estudió el sistema pictórico y a través del cual se informatiza para hacerlo más práctico y manejable.

Una fase previa de análisis es la de la observación de las aplicaciones ya existentes y la posibilidad de la creación de una nueva en un dispositivo actual.

1.1. APLICACIONES EXISTENTES:

Hoy en día hay pocas aplicaciones informáticas que dedican su funcionamiento a la ayuda de personas con dificultades de comunicación.

Algunas resultan sencillas de utilizar pero incompletas, otras menos accesibles y/o en otro idioma que no es el español, y otras simplemente caras para lo que ofrecen.

Aquí se presenta una tabla con un resumen de las aplicaciones más completas existentes en el mercado:

Aplicación	 **Voice4you**	 **Iprompts**	 **Proloquo2go**	 **Iconverse**	 **Ablah**
Pictogramas y/o fotos	SI	SI	SI	SI	SI
Audio	SI	NO	NO	NO	NO
Forma frases	NO	NO	SI	NO	NO
Cuenta atrás	NO	SI	NO	NO	NO

Organización de tareas	NO	SI	NO	NO	NO
Añadir pictos o texto	SI	SI	NO	SI	NO
Disponible para iPad	NO	SI	SI	NO	SI
Disponible para iPhone	SI	SI	SI	SI	SI
Idioma	Inglés	Inglés	Inglés	Inglés	Español Inglés
Precio	29.99$	49.99$	189$	9.99$	29.99€

Tabla 1. Aplicaciones existentes.

En conclusión se puede decir que hay aplicaciones que funcionalmente cuentan con lo solicitado por la gente discapacitada, pero principalmente no se encuentran en el idioma deseado, además de que todavía son muy limitadas.

Con relación a los objetivos que el presente proyecto pretende cumplir, la única aplicación existente anteriormente que se asemejaría a lo deseado podría ser **Iprompts**.

Dispone de una cuenta atrás animada bastante rudimentaria que da una idea de lo que puede ser el concepto de tiempo mediante una barra vertical de color que se va vaciando y por supuesto el tiempo restante en números al lado.

*Figura 3. Cuenta atrás de la aplicación **Iprompts**.*

Ésta aplicación también permite crear secuencias de actividades, que de alguna manera podrían representar una organización de eventos diaria.

Las fotografías pueden adquirirse desde la propia librería que ofrece la aplicación, incorporar imágenes personales o bajadas de la web.

Figura 4. Organización de eventos de la aplicación ***Iprompts****.*

Es el educador quien construye estas secuencias pero no están asociadas a ninguna fecha en concreto, sino que están más enfocadas a la realización de rutinas diarias.

Como se observa, la aplicación es muy completa en sí misma, y además está también creada para iPad, pero en ningún momento dispone de un álbum de fotos desarrollado.

Así pues se concluye que los antecedentes asociados a los objetivos de este proyecto no están lo suficientemente preparados.

La animación del reloj es muy pobre y no es explicativa. Y por su parte la organización de tareas es muy simple y no está asociada con eventos ni fechas concretas como las que se podría encontrar una agenda.

Eso sí, en cualquier lugar de Internet que se busque se encuentran animaciones de relojes, galerías varias y agendas tecnológicamente preparadísimas, pero ninguna con las características que aquí se necesitan, sencillas y educativas, enfocadas a un público con discapacidad.

1.2. PROYECTO ANTERIOR:

Como ya se ha comentado anteriormente, este proyecto parte como complemento a uno anterior. En este apartado se explicará detenidamente en qué consistía lo ya proyectado y el planteamiento en paralelo de las diferentes ramas de proyección.

Se encuentra en un punto de análisis y toma de decisiones avanzado. Tras estudiar los diferentes dispositivos posibles para implementar en ellos una aplicación comunicadora, se ha decidido contar con el iPad, por su gran sensibilidad táctil, su excelente calidad de imagen, su tamaño adecuado, y su clara ventaja calidad – precio.

Figura 5. Dispositivo iPad de Apple.

También se ha diseñado una estructura inicial, con el planteamiento de las aplicaciones futuras ya decidido y parcialmente implementado:

- **Comunicador**: La aplicación principal del proyecto conjunto. Debe de permitir la formación de frases por parte del usuario.

En este momento no se contaba con una estructura definida, ya que era muy complicado definir el comienzo de la frase con un determinado elemento y su navegación y filtrado posterior.

A partir de aquí se desarrolla plenamente este apartado con la creación de una base de datos completa y un complicado diseño de programación.

- **Quiero - Estoy:** Posee la misma base de ejecución que el Comunicador sólo que de una forma abreviada y directa.

En él sólo habrá un sujeto y se hablará siempre en primera persona mediante la utilización de los dos únicos verbos *quiero y estoy*. De esta forma el usuario cumplirá con sus necesidades más básicas de una forma rápida y eficaz.

Aquí se encuentra una versión muy primaria, que sirve como ejemplo de lo que va a ser finalmente, puesto que todavía no dispone de la base se datos.

- **Cuenta atrás:** Finalmente denominado *Reloj*.

Es una aplicación que permite elegir el tiempo deseado para realizar una determinada actividad (trabajar, comer, jugar…) y marca, mediante una rudimentaria cuenta atrás, el tiempo que falta para que la actividad finalice.

En este apartado es donde se desarrolla la animación final que el usuario verá y entenderá para el aprendizaje del concepto tiempo.

- **Pizarra:** Como elemento que mejore las habilidades motoras del usuario. Permite pintar con los dedos, trazar líneas de diferentes grosores, así como hacer uso de distintos colores.

Esta aplicación se encuentra completamente desarrollada, salvo pequeños fallos a corregir o mejorar en detalle.

Figura 6. Versión inicial de la aplicación a desarrollar.

- **Agenda:** Tanto educadores como familiares sugirieron en seguida la creación de una agenda personalizada con pictogramas, editable por el/la monitor/a, papá o mamá que estuviera con el usuario final para que éste reconociera las actividades planteadas para el día, semana y/o mes.

Sólo se cuenta con esta necesidad, pero en este momento no hay nada diseñado ni desarrollado. Como objetivo dentro de este proyecto se encuentra la definición de esta aplicación.

- **Diario:** Al igual que la agenda, era necesario un elemento que el/la niñ@ editara, de tal forma que hubiera una intencionalidad a la hora de mostrar a terceras personas lo que se había realizado durante el día.

Contaría con la formación de frases mediante pictogramas como base de esta comunicación siempre buscada.

Este apartado también se encuentra como objetivo a diseñar dentro del presente proyecto.

- **Álbum:** Con el objeto de sacar el máximo provecho del dispositivo elegido ya que posee una gran resolución de pantalla, está planteada la realización de un álbum de fotos que muestre tanto las fotografías personales que se encuentren en el iPad, como los dibujos realizados mediante la aplicación *Pizarra*, como los pictogramas de los que se disponga en la base de datos.

Además, mediante la observación de los pictos del programa, el usuario puede iniciarse al uso del comunicador, aprendiendo las imágenes con las que va a trabajar para construir las frases.

Ésta es la última aplicación planteada por el equipo multidisciplinar, y que faltaría por desarrollar, también objeto de este trabajo.

Como último punto a comentar dentro del proyecto inicial, está que el diseño gráfico del programa en general no tiene un hilo conductor común, fruto del trabajo interdisciplinar, en el que se han aportado conocimientos de todo tipo, siempre buscando la mejora del programa. Por ello se le da un aspecto nuevo, integrado y moderno.

Capitulo 2: Preparación inicial

Una vez aceptado el proyecto y conocidos los antecedentes y el dispositivo elegido en el que ya se estaba creando la aplicación del "*COMUNICADOR*", el siguiente paso a tener en cuenta es saber en qué consiste la programación de una aplicación y cuales son los conceptos generales a saber antes de familiarizarse con el entorno de trabajo.

2.1. CONCEPTOS BÁSICOS:

2.1.1. Interfaz de Programación de Aplicaciones: API

Una API (Application Programming Interface en inglés) constituye un conjunto de rutinas, procedimientos, protocolos, funciones y herramientas que una determinada biblioteca pone a disposición para que sean utilizados por otro software como una capa de abstracción.

En otras palabras, es una interfaz que permite la comunicación entre distintos componentes software. Representa, entonces, un método para alcanzar abstracción a la hora de programar, en particular en la relación entre niveles superiores e inferiores del software.

De esta forma, se puede decir que una de las funciones centrales de una API es la de ofrecer un grupo de funciones generales, como sería la de dibujar uno o más iconos en la pantalla. Esto tiene como resultado principal que los programadores se vean beneficiados gracias a estas API, en la medida en que al utilizar sus funcionalidades se evitan la necesidad de programar todo desde el inicio.

Sin embargo, es cierto también que una API con un alto nivel tiende a perder flexibilidad en su uso. Así, cuando se elige usar una API es necesario que lograr un determinado equilibrio, contemplando tanto su potencia como su simplicidad y su grado de flexibilidad.

En el caso de Apple, el API utilizado para la funcionalidad táctil, o lo que es lo mismo, el sistema operativo iOS, se llama Cocoa Touch y está construido sobre el paradigma "*Modelo – Vista – Controlador*" que más adelante se detalla.

Las librerías de Cocoa Touch que manejan las aplicaciones de iOS están construidas con un enfoque especial basado en el contacto y la optimización de la interfaz. Éste aporta las herramientas básicas que se necesitan para implementar gráficamente las aplicaciones orientadas a eventos en iOS.

UIKit, la librería por excelencia, proporciona los controles especiales de interfaz de usuario, botones e vistas en pantalla completa. También se consigue controlar las aplicaciones mediante el acelerómero y el gesto multi-touch.

2.1.2. Librerías / Frameworks:

Es una estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, con base en la cual otro proyecto de software puede ser organizado y desarrollado.

Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros programas para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Son diseñados con la intención de facilitar el desarrollo de software, permitiendo a los diseñadores y programadores pasar más tiempo identificando requerimientos de software que tratando con los tediosos detalles de bajo nivel de proveer un sistema funcional.

En el caso de Cocoa Touch, se dispone de cuatro módulos integrados, compuestos por las principales librerías que se citan a continuación:

Cocoa Touch Layer	Media Layer	Core Services Layer	Core OS Layer
AddressBookUI	AssetsLibrary	AddressBook	Acelérate
EventKitUI	AudioToolbox	CFNetwork	ExternalAccessory
GameKit	AudioUnit	CoreData	Security
iAd	AVFoundation	CoreFoundation	System
MapKit	CoreAudio	CoreLocation	
MessageUI	CoreGraphics	CoreMedia	
UIKit	CoreMIDI	CoreMotion	
	CoreText	CoreTelephony	
	CoreVideo	EventKit	
	ImageIO	Foundation	
	MediaPlayer	MobileCoreServices	
	OpenAL	QuickLook	
	OpenGLES	StoreKit	
	QuartzCore	SystemConfiguration	
		UIAutomation	

Tabla 2. Librerías de la Interfaz de Programación de Aplicaciones Cocoa Touch.

2.1.3. Entorno de Desarrollo Integrado: IDE

Un entorno de desarrollo integrado (en inglés Integrated Development Environment o IDE) es un programa compuesto por una serie de herramientas que utilizan los programadores para desarrollar código.

Esta herramienta puede estar pensada para su utilización con un único lenguaje de programación o bien puede dar cabida a varios de estos.

Las herramientas que normalmente componen un entorno de desarrollo integrado son las siguientes: un editor de texto, un compilador, un intérprete, unas herramientas para la automatización, un depurador, un sistema de ayuda para la construcción de interfaces gráficas de usuario y, opcionalmente, un sistema de control de versiones.

El entorno de desarrollo con el que Apple cuenta a la hora de desarrollar aplicaciones se llama Xcode.

Está diseñado para aprovechar al máximo las nuevas tecnologías Apple. Posee una interfaz unificada (última versión) para intercambiar el modo de programación sin problemas, para ver el código fuente, la depuración e incluso el diseño de la interfaz de usuario en una misma ventana.

A su vez, es muy cómodo a la hora de programar, puesto que alerta en tiempo real de posibles errores tipográficos o de formato de instancias o métodos. También se puede iniciar la depuración y examinar el valor de las diferentes variables en tiempo de ejecución pasando el puntero del ratón por encima.

Al trabajar estrechamente con el portal de desarrolladores web, se puede firmar y enviar de forma segura y con un solo clic la aplicación al Apple Store.

2.1.4. Lenguaje de Programación Orientado a Objetos:

La programación Orientada a Objetos (POO) es una forma especial de programar, más cercana a como se expresan las cosas en la vida real que otros tipos de programación.

Con la POO es necesario pensar las cosas de una manera distinta, para escribir los programas en términos de clases, objetos o instancias, propiedades y métodos.

Los objetos son entidades que tienen un determinado comportamiento (método) e identidad):

- El ***estado*** está compuesto de datos, será uno o varios atributos a los que se habrán asignado unos valores concretos (datos).
- El ***comportamiento*** está definido por los métodos o mensajes a los que sabe responder dicho objeto, es decir, qué operaciones se pueden realizar con él.
- La ***identidad*** es una propiedad de un objeto que lo diferencia del resto, dicho con otras palabras, es su identificador (concepto análogo al de identificador de una variable o una constante).

Un objeto contiene toda la información que permite definirlo e identificarlo frente a otros objetos pertenecientes a otras clases e incluso frente a objetos de una misma clase, al poder tener valores bien diferenciados en sus atributos.

A su vez, los objetos disponen de mecanismos de interacción llamados métodos, que favorecen la comunicación entre ellos. Esta comunicación favorece a su vez el cambio de estado en los propios objetos. Esta característica lleva a tratarlos como unidades indivisibles, en las que no se separa el estado y el comportamiento.

Objective C es el lenguaje que se emplea para el desarrollo de aplicaciones tanto para Mac OS X como para iOS.

Es un lenguaje de programación simple diseñado para permitir una sofisticada programación orientada a objetos. Está creado como un superconjunto de C pero que implementa un modelo de objetos parecido al de Smalltalk, uno de los primeros lenguajes de programación orientado a objetos.

Originalmente fue creado por Brad Cox y la corporación StepStone en 1980. En 1988 fue adoptado como lenguaje de programación de NEXTSTEP y en 1992 fue liberado bajo licencia GPL para el compilador GCC. Actualmente se usa como lenguaje principal de programación en Mac OS X y GNUstep.

Capitulo 3: Estudio de la programación

Tras conocer de qué se está hablando es hora de dar el siguiente paso y comenzar a familiarizarse con el entorno de trabajo.

Será necesario disponer de un ordenador Apple con sistema operativo Mac OS X en el que descargar e instalar el kit de desarrollo y conexión a Internet para registrarse como desarrollador oficial.

3.1. PREPARACIÓN DEL ENTORNO:

Cuando se quiere desarrollar aplicaciones para iOS hay que tener en cuenta principalmente dos aspectos.

Por un lado se requiere un ordenador Mac basado en Intel en el cual se encuentre instalado un sistema operativo Leopard o superior.

Por otro, se cuenta con una serie de herramientas y recursos instalados en él, que son proporcionados por Apple a través de la web: http://developer.apple.com.

Entre las opciones que ofrece dicha web, está el registro en el programa oficial de desarrolladores de Apple en el cual es necesario inscribirse para descargar el kit de desarrollo y después poder colocar la aplicación implementada en el Apple Store.

Además, la página facilita un centro de desarrollo, para acceder a contenidos, foros, documentación específica, recursos y herramientas.

El SDK (Software Development Kit) es el kit de desarrollo para iOS, que consiste en un conjunto de utilidades que permiten la creación y ejecución de los programas para el dispositivo deseado.

3.1.1. Registro como desarrollador:

Se debe acceder a la página de desarrollador de iOS para poder registrarse: http://developer.apple.com/programs/register/

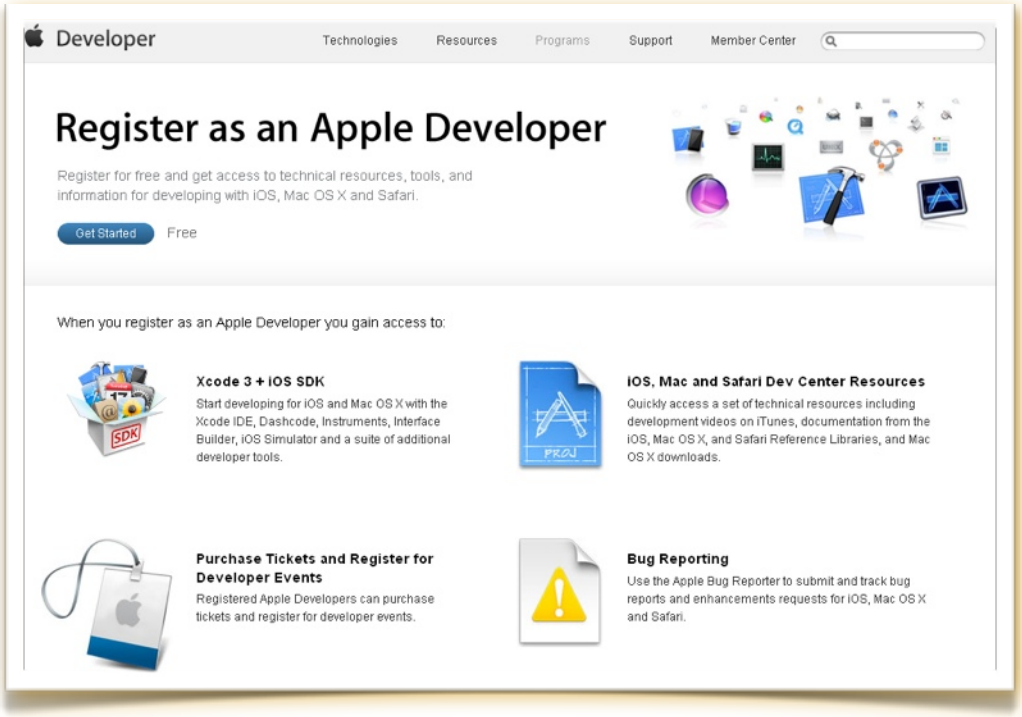

Figura 7. Página web del inicio de registro como Desarrollador de Apple.

Entre la información que se solicita a través de los formularios de registro, se encuentra cuál es el mercado principal, qué tipo de aplicaciones se van a desarrollar, si van a ser gratuitas o no, si son específicas para una empresa, si se lleva tiempo desarrollando para esta plataforma y si se trabaja también para otros dispositivos con otros sistemas operativos.

Todo esto es requerido para completar la información de perfil de cara a su uso posterior en la Apple Store.

Una opción extra es la de pagar 99$ y obtener así la licencia adecuada que permite simular legalmente la aplicación creada en un dispositivo real. Mientras no se tome esta alternativa, todas las pruebas se realizan en el entorno de desarrollo mediante el simulador específico de iOS.

Para tener acceso al SDK y al resto de recursos, es necesario verificar el mail que se solicita en el registro.

3.1.2. Descarga e instalación del kit de desarrollo:

Una vez dentro de la cuenta de desarrollador, se descarga la versión correspondiente al sistema operativo del que se disponga. Hay que tener en cuenta que el paquete entero ocupa alrededor de 3 Gb de peso, tamaño considerable para una descarga con una conexión a Internet lenta.

Al hacer doble clic sobre el icono obtenido se procede a la instalación del kit.

Las herramientas básicas que componen el SDK y que se detallan en el siguiente apartado son: Xcode, Interface Builder y el Simulador iOS. Se pueden encontrar dentro del disco duro interno, en la carpeta *Developer/applications*.

3.2. ENTORNO DE TRABAJO:

3.2.1. Xcode:

Es el entorno del que parten todos los proyectos para iOS y a partir del cual se puede acceder al Interface Builder y al Simulador para que corra la aplicación.

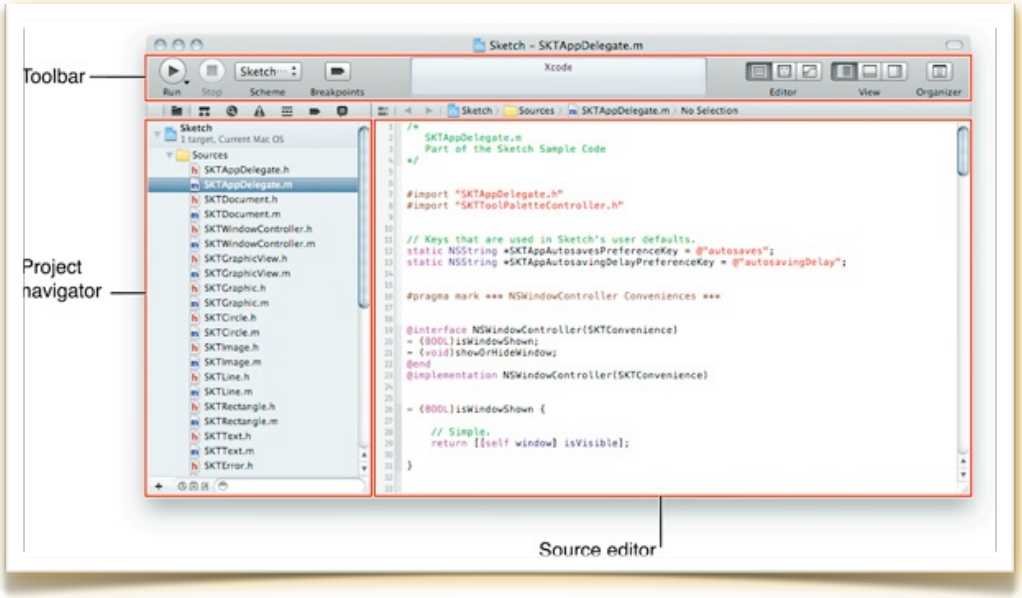

Figura 8. Estructura del entorno de trabajo Xcode.

Además de ser el punto de unión de las demás aplicaciones sirve como editor de código y es el lugar donde se programa la interacción con el usuario.

Actualmente, en la última actualización del kit de desarrollo, la aplicación Interface Builder viene integrada completamente en Xcode, y no a parte, como ocurría en las

anteriores. En el presente proyecto se han utilizado ambas versiones pero sólo se cita la última de la que se ha hecho uso por cuestiones gráficas.

Al abrir un nuevo proyecto, el SDK proporciona una gran variedad de plantillas para realizar diferentes tipos de trabajo. La plantilla elegida para este proyecto ha sido View Based Application.

La ventana del espacio de trabajo es el centro de control para uno o más proyectos.

Cualquier proyecto contiene los elementos necesarios para diseñar y construir el producto deseado. Aquí se incluyen archivos de código, diseños de interfaz, sonidos, imágenes y links hacia las librerías y bibliotecas soportadas.

- **Navegador de Proyecto:** Lugar en el que se organizan los componentes principales del proyecto. Carpetas de recursos, clases ordenadas jerárquicamente, alertas, breakpoints, etc.

Es importante anotar que las carpetas aquí presentes no están representadas necesariamente como carpetas dentro de la ubicación donde está guardado el proyecto. No existe una correspondencia salvo en la carpeta "***Classes***".

- **Editor de Código:** Donde se escribe código directamente. Incluye autocompletado y corrección de código y depuración en tiempo real.

- **Toolbar:** Provee de un acceso rápido a la configuración del proyecto y la gestión de las herramientas que se usan frecuentemente. Por ejemplo análisis de código, correr la aplicación, ver el progreso de varias actividades y configurar el espacio de trabajo al gusto.

3.2.2. Interface Builder:

Es un conjunto de herramientas que permiten diseñar gráficamente la interfaz de la aplicación y conectar los elementos y controles de dicha interfaz con el código que se programa en Xcode.

Los ficheros con extensión *.xib* son los que se abren con esta aplicación.

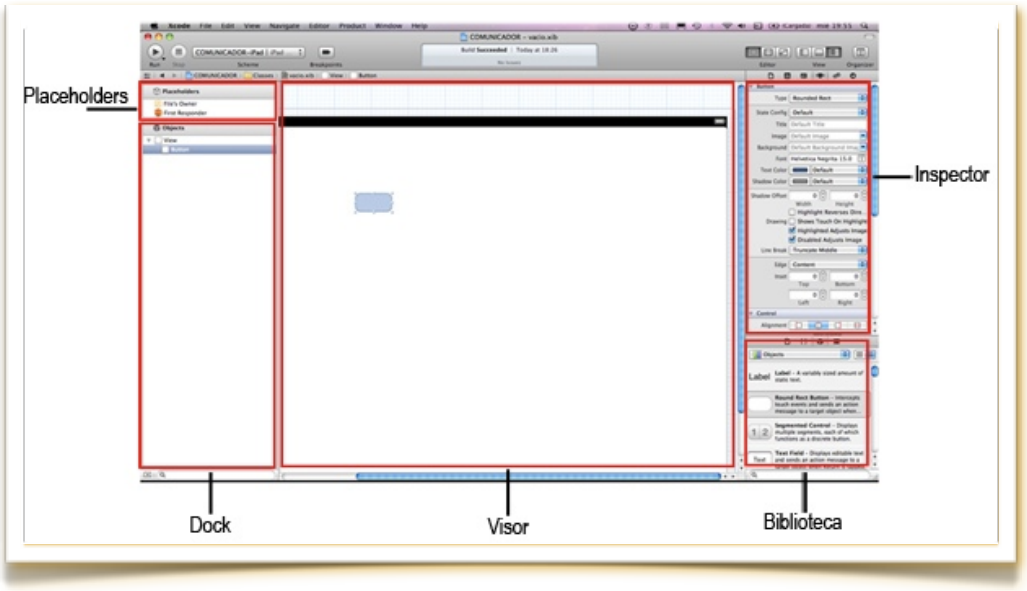

Figura 9. Estructura del entorno de trabajo Interface Builder.

Se observa en la parte central un visor de los elementos que serán visualizados directamente en el dispositivo. A su derecha abajo, una biblioteca de controles que se pueden arrastrar sobre la vista principal, y arriba, un inspector o panel de propiedades mediante el cual estos elementos puedan ser configurados tras ser seleccionados.

En la parte izquierda, sin embargo, y con la posibilidad de verse en iconos o en lista, se encuentra el *dock* de la interfaz, donde estarán todos los objetos que la están componiendo, y los *placeholders* elementos interesantes a mencionar para el conexionado del "***Modelo – Vista – Controlador***":

- **File's Owner:** Representa el objeto que cargarán el archivo *.xib* en el disco. El que une las propiedades y métodos de cada clase con los objetos referenciados aquí, en el Interface Builder. Es el propietario del fichero cargado de ahí su nombre.

- **First Responder:** Representa el elemento que tiene el foco (la actividad) en un momento dado. Por eso varía dependiendo de la interacción del usuario con la aplicación. Por código se podría saber qué elemento y en qué momento ha recibido una acción por parte del usuario, ya que aparecería aquí referenciada.

3.2.3. Simulador iOS:

Dispositivo clave a la hora de probar el funcionamiento de la aplicación que se está creando. Éste emulador se lanza automáticamente una vez se pulsa el botón *Run* de Xcode.

Simula a la perfección lo que supone utilizar un dispositivo real, bien sea iPhone como iPad, con la interactividad adecuada.

Además, mediante los menús, se pueden realizar cambios para ejecutar diferentes acciones con el aparato y ver el comportamiento que la aplicación tiene ante las mismas: mover, rotar, agitar, llamar…

3.3. INICIACIÓN AL LENGUAJE:

3.3.1. Clases:

Las clases son un modelo o patrón que marcan una serie de propiedades y comportamientos.

Los objetos son instancias reales de ese modelo patrón, y poseen valores para dichas propiedades, además de ejecutar los comportamientos que la clase define.

Todas las clases, en Objective C, se estructuran en dos ficheros:

- **Header.h:** Define las propiedades y métodos que va a tener la clase, pero no les da un comportamiento.

- **Implementación.m:** En él se introduce el código real que va a definir los métodos por los cuales las clases se comportarán de una cierta manera.

PROPIEDADES

Las clases admiten distintas propiedades variables para cada una de sus instancias, de tal forma que cada una se define con parámetros diferentes.

```
@interface Vehiculo : NSObject {

    int velocidad;
    int velocidadMaxima;
    NSString *marca;
    NSString *modelo;
    NSString *matricula;
}
```

En Objective C no existen propiedades de clases concretamente, sólo de sus instancias.

Además, éstas propiedades no son accesibles externamente y por tanto se tendrán que declarar métodos de tipo *getter y setter* para habilitar su acceso.

La forma de hacerlo para cada una de las instancias es mediante las expresiones ***@property*** (en el header) y ***@synteshize***. (en la implementación).

```
@property(nonatomic) int velocidad;
@property(nonatomic) int velocidadMaxima;
@property(nonatomic,retain) NSString *marca;
@property(nonatomic,retain) NSString *modelo;
@property(nonatomic,retain) NSString *matricula;
```

De este modo se puede tanto acceder a la propiedad de la instancia externamente como modificar su valor.

MÉTODOS:

Son los elementos con los que se da funcionalidad a cada clase y son utilizados para asegurar que externamente se pueda emplear dicha clase por otras clases u otros elementos del lenguaje.

Se declaran inicialmente en el fichero Header, a continuación de las propiedades.

Se diferencian los métodos relativos a:

- **Clases**: No requieren de una instancia para ser llamados. Son utilizados en determinadas clases que tienen que dotar de funcionalidad al lenguaje pero que no tienen funcionalidad propia. Por ejemplo, la clase *matemáticas* tendría métodos que realizan operaciones relativas a potencias, logaritmos, etc.

- **Instancias**: Cada una de las instancias podría llamar al método y el contenido del mismo haría referencia a dicha instancia. Son los más comunes.

```
-(void)acelerar:(int)incremento;
{
    if (velocidad+incremento>velocidadMaxima)
    {
        velocidad=velocidadMaxima;
    }
    else
    {
        velocidad=velocidad+incremento;
    }
}
```

De la misma forma que los métodos se nombran en el fichero header, hay que "rellenarlos" en la parte de implementación.

Por otra parte, continuamente se da el caso encontrarse en una clase y necesitar llamar al método de otra para hacer una operación o una acción determinada. Para poder tener acceso a dichas funciones, es necesario importar una dentro de la otra mediante la expresión ***# import "ClaseAImportar.h".***

Se crea una instancia de la clase y se inicializa, reservando en memoria, el espacio correspondiente.

3.4. INTERACCIÓN BÁSICA: Patrón MVC

Modelo – Vista – Controlador es un patrón de diseño que derivó de *Smalltalk* en los ochenta. Propone tres tipos de objetos en una aplicación separados por límites abstractos y comunicados entre ellos por esos límites, como se ilustra en la figura.

Es el patrón de diseño que se encuentra por detrás del diseño de muchos programas orientados a objetos. Ésta configuración ayuda al desarrollo de sistemas sostenibles, extensibles y comprensibles.

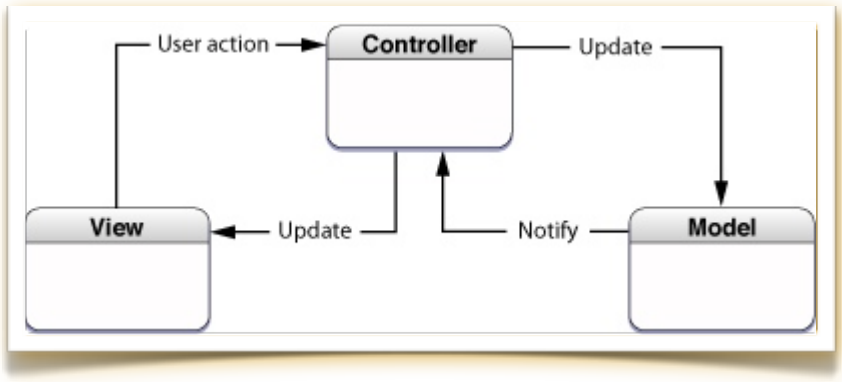

Figura 10. Diagrama del Patrón MVC

- **MODELO**: Representa el elemento que tiene el foco (la actividad) en un momento dado. Por eso varía dependiendo de la interacción del usuario con la aplicación.

Por código se podría saber qué elemento y en qué momento ha recibido una acción por parte del usuario, ya que aquí se gestionan y almacenan todos los datos de la aplicación.

- **VISTA**: Representa la parte visible en la interfaz de usuario.

Cada elemento aquí desconoce la procedencia de la información que muestra. La librería *Application Kit* proporciona los elementos necesarios para la creación de la interfaz deseada: ventanas, campos de texto, vistas con scroll, botones, buscadores, etc.

Ésta se encuentra en los archivos cuya extensión es *.xib*.

- **CONTROLADOR**: Es la parte encargada de lleva la lógica de la aplicación.

Es decir, cuando el usuario ejecuta una interacción con la aplicación, la vista la recibe (por ejemplo, a través de un botón) y se notifica al controlador de que se ha recibido dicha pulsación.

En función del botón que se haya pulsado, tendrá que realizar una serie de acciones que podrían desembocar por ejemplo en el guardado de datos en el modelo.

En definitiva, actúa como mediador entre los objetos del modelo y la vista. Comunica toda la información necesaria entre ellos.

Será la clase determinada como *viewController(h y m)*.

En general la capa *modelo* no siempre es utilizada, sobretodo si no se necesita almacenar los datos de forma persistente. Por otro lado, las capas *vista* y *controlador* siempre están disponibles en una aplicación para iOS.

3.5. APLICACIONES MULTIVISTA:

Como aplicación multivista se entiende aquella con un *Controlador Principal* que gestiona diferentes *vistas* y que cada una de ellas tiene a su vez sus propios *Controladores de Vistas.*

De esta manera se pueden crear mini-aplicaciones dentro de una aplicación general.

Habitualmente se hace uso de determinados controles para navegar por las distintas *pantallas o vistas*, como al *Navigation Bar, Tool Bar o Tab Bar,* con iconos que realizan diferentes tareas. Pero también se puede crear un control propio, una extensión de *viewController* que ejerza la función de gestión de vistas.

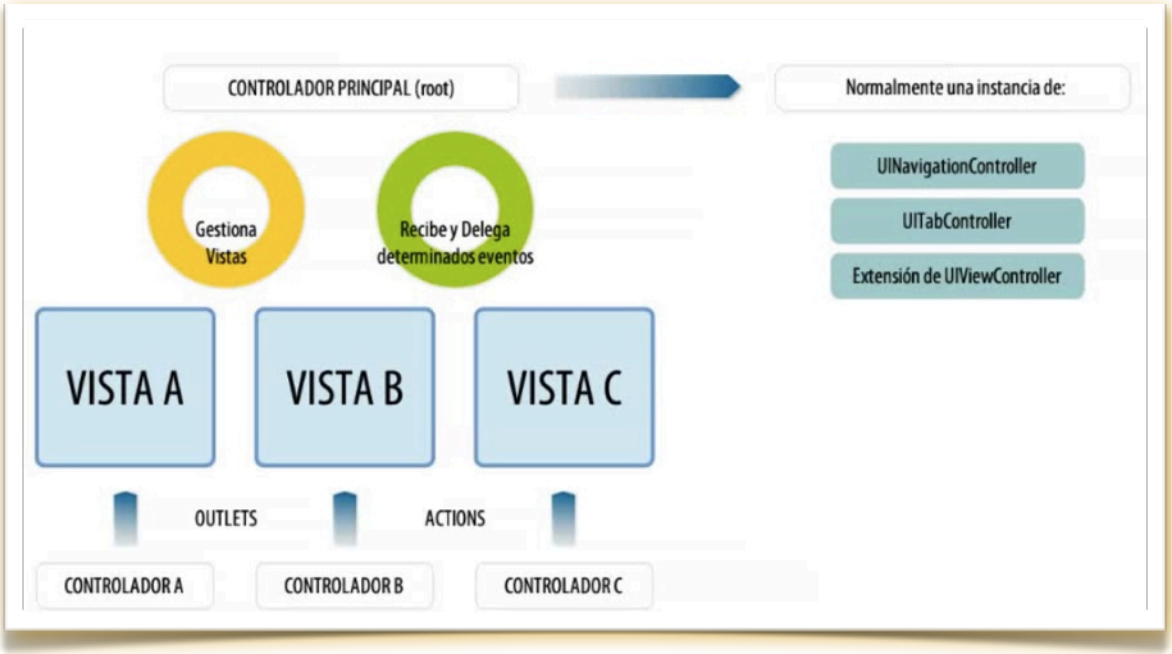

Figura 11. Esquema de una aplicación multivista.

Como se puede intuir, el proyecto conjunto es un perfecto ejemplo de lo que es una aplicación multivista, ya que posee mini-aplicaciones bien diferenciadas que pueden estar dirigidas por un controlador principal.

A su vez, cada una de ellas tendrá control sobre cada una de sus vistas y pantallas.

Capitulo 4: Diseño y Desarrollo

Tras estudiar a conciencia el lenguaje de programación y tener claros los objetivos del proyecto, se inicia el diseño y desarrollo de las partes correspondientes comenzando por un planteamiento previo de la estructura del programa.

4.1. PLANTEAMIENTO INICIAL:

Es importante comenzar con un diseño de la estructura del programa ya que como se ha visto anteriormente, el planteamiento de una aplicación multivista sería el correcto para seguir avanzando.

En este momento del desarrollo, el comunicador se encuentra dispuesto de la siguiente manera esquemática:

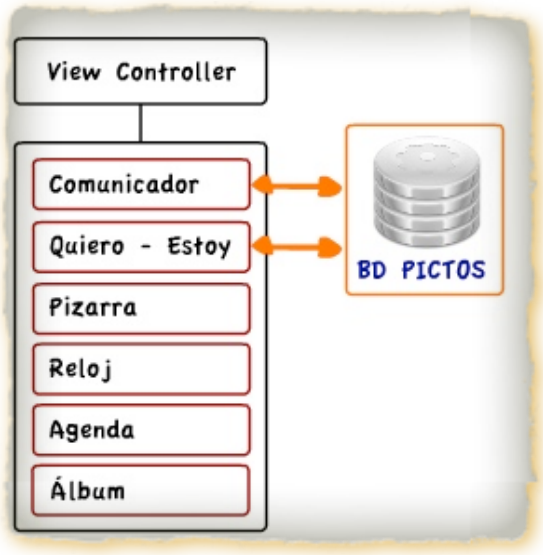

Figura 12. Esquema del comunicador inicial

Este proyecto se basa en la importancia que tiene el desarrollo de un software que cubra las necesidades de ciertas personas, orientado por gente especializada en el tema.

De igual forma, y contando con que este proyecto es ampliable en todos los aspectos y probablemente sea tocado por una variedad de manos, es igual de importante diseñar una forma de implementación adecuada, pensando en los futuros programadores de la aplicación y en la mejora de la misma.

Actualmente y con el proyecto ya iniciado, no se ha seguido ningún patrón de programación que establezca un orden definido, sí que es cierto que lo que está hecho funciona perfectamente, pero puede llevar a complicaciones, retrasos y dificultad a la hora de entrar en código.

Un modelo estructurado en módulos (clases controladoras) permite tener un mayor orden en la lógica del programa y además supone una gran ventaja para modificar o expandir la/s aplicación/es.

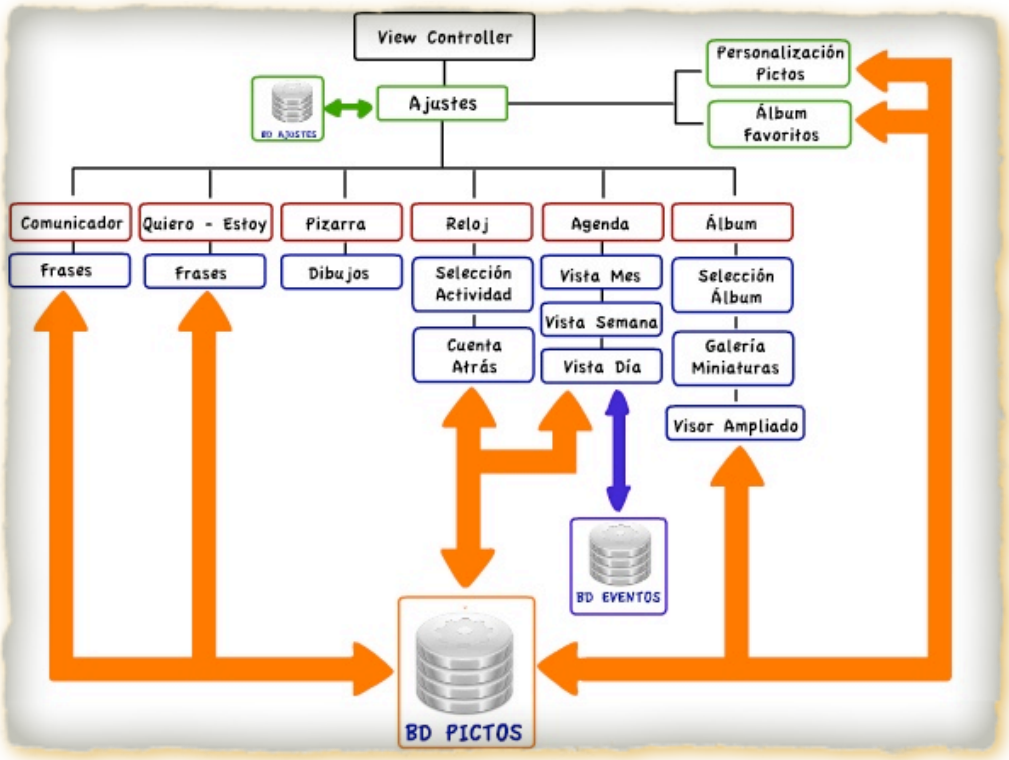

Figura 13. Esquema comunicador final.

De esta manera se pueden agregar más módulos programados independientemente e integrarlos en la aplicación principal de manera mucho más sencilla.

Además de hacerse con la estructura adecuada para la implementación de código, hay que tener en cuenta que será necesaria la búsqueda de las librerías que proporcionen las funciones adecuadas para las acciones deseadas en cada aplicación.

Esto lleva una serie de tiempo y no siempre es tan fácil encontrar la más adecuada para el objetivo que se busca.

4.2. ANIMACIÓN RELOJ:

El reloj es un apartado creado con el fin de que la persona discapacitada tome conciencia del concepto de *tiempo*.

A él se accede a través de la parte inferior izquierda de la pantalla principal de la aplicación.

Figura 14. Pantalla inicial de la aplicación que da acceso al Reloj.

Contiene dos objetos de selección a elegir entre una acción determinada y un tiempo de duración. La actividad se debe realizar mientras el tiempo está en activo, cuando este acaba, la tarea que se estaba llevando a cabo finaliza también.

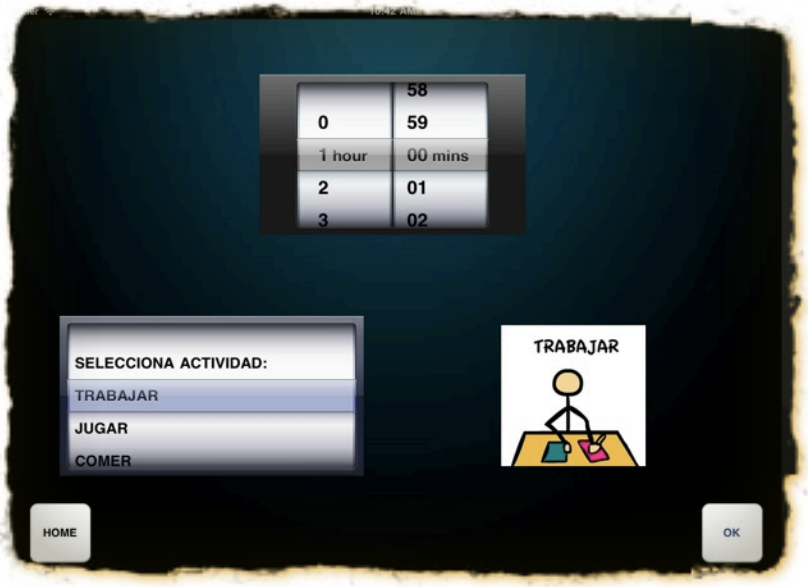

Figura 15. Acceso al reloj, primera pantalla de selección de actividad y tiempo.

Inicialmente la estructura con la que cuenta el reloj es sencilla. Una cuenta atrás con el tiempo numéricamente y la actividad a realizar con el picto asociado acompañan a un reloj de arena que se transforma lentamente en la pantalla.

Esta transformación se da de forma interrumpida ya que consta de 10 imágenes con diferentes niveles de arena que van alternándose dependiendo del tiempo que quede para la finalización del crono.

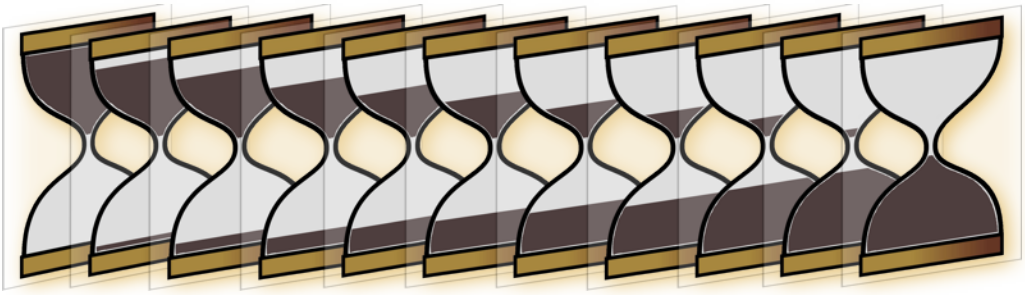

Figura 16. Imágenes del reloj inicial.

Visualmente no se encuentra ninguna continuidad ni suavidad en la animación por lo que se trata de mejorarlo.

También, hay que tener en cuenta que la aplicación del reloj posee un botón oculto para posibilitar la interrupción de la cuenta atrás si el cuidador o la persona que esté con el usuario lo cree conveniente.

Figura 17. Imágenes del reloj en el momento "stop".

4.2.1. Reloj de arena:

ESTRUCTURA DE DISEÑO

Se comienza con el diseño del objeto en sí mismo, el reloj de arena. Es prácticamente igual que el anterior pero algo más realista.

Figura 18. Comparación entre los relojes de arena inicial - final.

A continuación se distribuyen las capas que lo componen, asignando los movimientos correspondientes a cada una de ellas para que la animación tenga el efecto deseado y no haya solapamientos entre ellas:

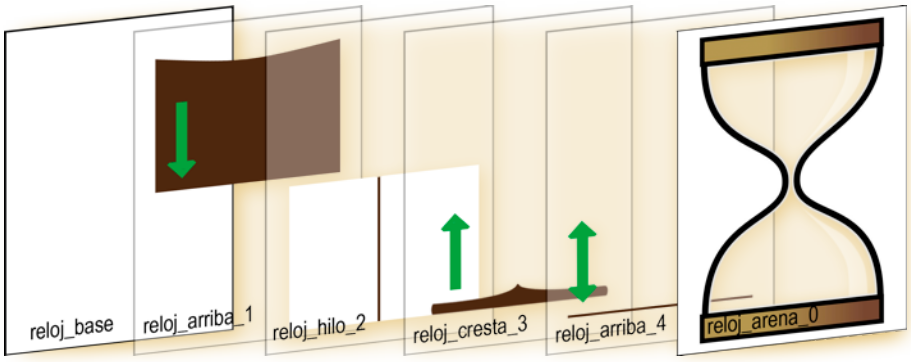

Figura 19. Capas que componen el reloj con sus movimientos.

Habrá que tener en cuenta que algunas capas serán opacas y otras transparentes por lo que el formato de cada una de estas imágenes debe de ser *.png*

Además se observa que las imágenes en general mantienen un movimiento constante, exceptuando la arena de la parte de abajo. Ésta consiste en una capa no que se mueve, sino que se expande hacia arriba. Habrá que tenerlo en cuenta en su animación.

MANUAL DE CONFIGURACIÓN

Una vez el diseño está terminado se procede a su implementación.

Se recurre a la librería ***Core Animation***, perteneciente al grupo ***Core Graphics***, para el uso de las sencillas funciones que darán lugar al movimiento buscado.

Esta librería sitúa sus objetos en la capa *modelo*, de esta forma cada uno de ellos mantiene su información referida a geometría y tiempo, así como lo que muestran visualmente en cada momento.

Es necesario incluirla dentro del proyecto, así que se añade dentro de la carpeta *Frameworks*:

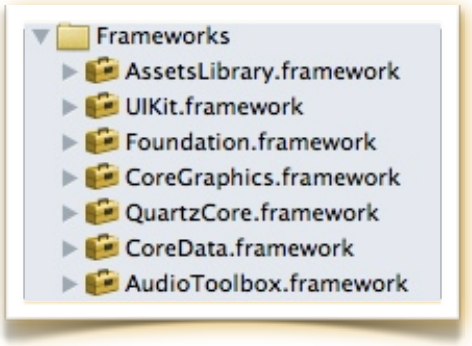

Figura 20. Librerías incluídas dentro de la aplicación.

Como se observa en la captura anterior, a lo largo del proyecto se incluyen más librerías necesarias para otro tipo de funciones.

Cada objeto se incluye dentro de una capa y esto permite trabajar con las capas reales de forma sencilla, al tomar cada una de ellas un valor central, una posición, unos límites y hasta una referencia con la capa que le precede.

Así que en primer lugar se debe crear en el fichero.h los objetos a ser utilizados:

```
    // Creo las capas del reloj

CALayer * reloj_arena_arriba_4;
CALayer * reloj_arena_hilo_3;
CALayer * reloj_arena_cresta_2;
CALayer * reloj_arena_abajo_1;
CALayer * reloj_arena_0;
CALayer * reloj_base;
```

Posteriormente en el fichero.m se especifican los valores y propiedades que cada capa adquiere y la imagen que debe mostrar.

```
// Coloco las capas del reloj:

reloj_base=[CALayer layer];
reloj_base=reloj.layer;
reloj_base.bounds=CGRectMake(594.0f, 181.0f, 310.0f, 438.0f);
reloj_base.position=CGPointMake(594.0f, 181.0f);
reloj_base.anchorPoint=CGPointMake(0.0f, 0.0f);
reloj_base.contents= (id)[[UIImage imageNamed:@"reloj_base.png"] CGImage];

reloj_arena_0=[CALayer layer];
reloj_arena_0.bounds=CGRectMake(594.0f, 181.0f, 310.0f, 438.0f);
reloj_arena_0.position=CGPointMake(594.0f, 181.0f);
reloj_arena_0.anchorPoint=CGPointMake(0.0f, 0.0f);
reloj_arena_0.contents= (id)[[UIImage imageNamed:@"reloj_arena_0.png"] CGImage];

reloj_arena_abajo_1=[CALayer layer];
reloj_arena_abajo_1.bounds=CGRectMake(594.0f, 620.0f, 310.0f,98.0f);
reloj_arena_abajo_1.position=CGPointMake(594.0f, 620.0f);
reloj_arena_abajo_1.anchorPoint=CGPointMake(0.0f, 1.0f);
reloj_arena_abajo_1.contents= (id)[[UIImage imageNamed:@"reloj_arena_abajo_1.png"] CGImage];

reloj_arena_cresta_2=[CALayer layer];
reloj_arena_cresta_2.hidden=YES;
reloj_arena_cresta_2.bounds=CGRectMake(594.0f, 181.0f, 310.0f, 438.0f);
reloj_arena_cresta_2.position=CGPointMake(594.0f, 200.0f);
reloj_arena_cresta_2.anchorPoint=CGPointMake(0.0f, 0.0f);
reloj_arena_cresta_2.contents= (id)[[UIImage imageNamed:@"reloj_arena_cresta_2.png"] CGImage];

reloj_arena_hilo_3=[CALayer layer];
reloj_arena_hilo_3.hidden=YES;
reloj_arena_hilo_3.bounds=CGRectMake(594.0f, 181.0f, 310.0f, 438.0f);
reloj_arena_hilo_3.position=CGPointMake(594.0f, 181.0f);
reloj_arena_hilo_3.anchorPoint=CGPointMake(0.0f, 0.0f);
reloj_arena_hilo_3.contents= (id)[[UIImage imageNamed:@"reloj_arena_hilo_3.png"] CGImage];

reloj_arena_arriba_4=[CALayer layer];
reloj_arena_arriba_4.bounds=CGRectMake(594.0f, 181.0f, 310.0f, 438.0f);
reloj_arena_arriba_4.position=CGPointMake(594.0f, 181.0f);
reloj_arena_arriba_4.anchorPoint=CGPointMake(0.0f, 0.0f);
reloj_arena_arriba_4.contents= (id)[[UIImage imageNamed:@"reloj_arena_arriba_4.png"] CGImage];
```

Cada capa se añade una encima de otra de tal forma que se mantiene el orden de superposición en que se han ido agregando:

```
[reloj_base addSublayer:reloj_arena_arriba_4];
[reloj_base addSublayer:reloj_arena_hilo_3];
[reloj_base addSublayer:reloj_arena_cresta_2];
[reloj_base addSublayer:reloj_arena_abajo_1];
[reloj_base addSublayer:reloj_arena_0];
```

Por otra parte y teniendo en cuenta los movimientos, se elige la ***Explicit Animation***, Animación Explícita. Con ella se crea una animación para cada objeto o capa independiente dentro del *fichero.h*:

```
        // y sus animaciones:

CABasicAnimation *arena_abajo;
CABasicAnimation *arena_cresta_arriba;
CABasicAnimation *arena_arriba;
```

Como se observa, en cada creación de movimiento se especifica qué es lo que se va a variar. Tanto en la a*rena que baja* como en la *cresta que sube*, los valores a modificar son los de posición, con lo que se mueve la capa completa. Sin embargo, en la *arena que sube* lo que se transforma son los límites de la capa, (una capa uniforme), haciendo su altura mayor con el paso del tiempo.

```
        // La arena que baja
arena_abajo=[CABasicAnimation animationWithKeyPath:@"position"];
arena_abajo.fromValue=[NSValue valueWithCGPoint: CGPointMake(594.0, 181.0)];
arena_abajo.toValue=[NSValue valueWithCGPoint: CGPointMake(594.0, 330.0)];
arena_abajo.duration=cuenta_atras;

        // La arena que sube
arena_arriba=[CABasicAnimation animationWithKeyPath:@"bounds"];
arena_arriba.fromValue=[NSNumber valueWithCGRect: CGRectMake(594.0f, 620.0f, 310.0f, 98.0f)];
arena_arriba.toValue=[NSNumber valueWithCGRect: CGRectMake(594.0f, 620.0f, 310.0f, 410.0f)];
arena_arriba.duration=cuenta_atras;

        // La cresta que sube encima de la arena
arena_cresta_arriba=[CABasicAnimation animationWithKeyPath:@"position"];
arena_cresta_arriba.fromValue=[NSValue valueWithCGPoint: CGPointMake(594.0, 200.0)];
arena_cresta_arriba.toValue=[NSValue valueWithCGPoint: CGPointMake(594.0, 55.0)];
arena_cresta_arriba.duration=cuenta_atras;
```

En todas ellas se establecen los valores de inicio y final. Con este tipo de funcionamiento, la animación nunca comenzará hasta que sea añadida al objeto correspondiente en el instante que se desee.

```
// Añadimos la animación a la capa correspondiente y así la inicializamos:

[reloj_arena_arriba_4 addAnimation:arena_abajo forKey:@"arena_abajo"];
[reloj_arena_abajo_1 addAnimation:arena_arriba forKey:@"arena_arriba"];
[reloj_arena_cresta_2 addAnimation:arena_cresta_arriba forKey:@"arena_cresta_arriba"];
```

Por último se muestran las imágenes del hilo (arena cayendo) y la cresta, que estaban ocultas, en el instante exacto en el que comienza la cuenta atrás.

```
reloj_arena_hilo_3.hidden=NO;
reloj_arena_cresta_2.hidden=NO;
```

El resultado final es el siguiente:

Figura 21. Muestra del funcionamiento de la cuenta atrás del reloj de arena.

4.2.2. Caracol:

Tras la realización del reloj de arena animado surge la idea de crear un elemento distinto que muestre igualmente el concepto de *tiempo* y que sea quizás más atractivo e interesante.

La idea nació por el hecho de que un símbolo tan común como es el reloj de arena para expresar el concepto de *tiempo*, podía no entenderlo el niño según los conocimientos de las pedagogas.

Aparece entonces la idea de un caracol animado, que recorra un trecho hasta llegar a una meta, por ejemplo una hoja, y que tarde un tiempo determinado, el correspondiente a la cuenta atrás.

Al finalizar el timer, el caracol debe realizar una acción que demuestre que efectivamente se ha acabado con la actividad, en este caso se llega a la conclusión de que el animal debe comerse la hoja.

Con esto se consiguen dos acciones: por un lado se ve un animal conocido moviéndose dentro de un espacio para conseguir algo, y como segunda parte, el animal ha conseguido lo que esperaba tras el tiempo transcurrido.

ESTRUCTURA DE DISEÑO

Al igual que con el reloj de arena, hay que comenzar diseñando el dibujo que va a animarse y las capas que lo van a componer.

Hay que tener en cuenta que las capas *CALayer* se colocan siempre sobre un *UIImageView*, por lo que se utilizan 3 *UIImageView* para los 3 elementos principales.

Figura 22. Archivo .xib que muestra los tres UIImageView donde está contenido el caracol.

La hierba y la hoja. que permanecen quietas sobre el primero.

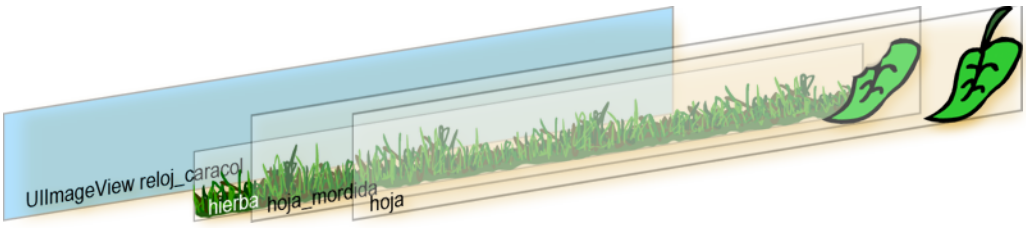

Figura 23. Capas dentro del primer UIImageView "reloj_caracol".

La baba, por su parte, hace uso de la máscara para ir mostrándose sobre el segundo.

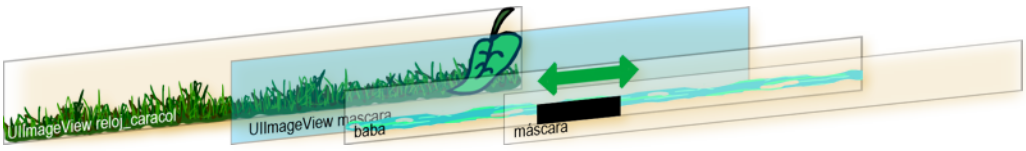

Figura 24. Capas dentro del segundo UIImageView "mascara".

El animal, por último, se mueve sobre el tercer *UIImageView*.

Los movimientos son sencillos, pero hay que diferenciar dos momentos concretos. Se hablará de la animación de la cuenta atrás, durante la cual el caracol se traslada de izquierda a derecha, dejando su característico rastro en el camino, y a continuación, cuando se da el fin del reloj, hay una animación posterior, la que muestra el caracol comiéndose la hoja al llegar a su meta.

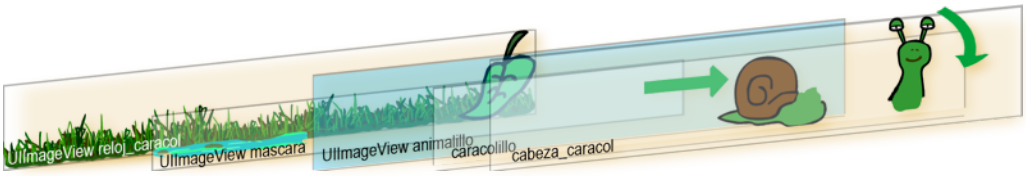

Figura 25. Capas dentro del tercer UIImageView "animalillo".

Como se observa, se utiliza una capa máscara que mostrará poco a poco la baba del caracol conforme se dé el movimiento de éste. La cabeza del animal también es independiente, para poder realizar la inclinación hacia la hoja cuando llega el momento de comérsela.

Esto es así porque una vez que se incluyen las *CALayer* dentro de un mismo *UIImageView* ordenadamente, las que están por encima afectan a las de debajo, por lo que si la capa máscara no se utiliza de forma independiente en otro *UIImageView*, ésta ejercería su efecto también sobre la hierba y la hoja y no es lo que se desea.

MANUAL DE CONFIGURACIÓN

Se comienza entonces con la implementación de todos los objetos citados anteriormente. Además, ha quedado claro que sólo se utilizan 3 movimientos en toda la animación, por lo que son necesarias 3 animaciones, una para cada objeto móvil:

```
/// UIImageViews que sostienen las capas del caracol

IBOutlet UIImageView *reloj_caracol;
IBOutlet UIImageView *mascara;
IBOutlet UIImageView *animalillo;
```

```
//Creo las capas del caracol

CALayer *hoja;
CALayer *babamask;
CALayer *hierba;
CALayer *caracol;
CALayer *cabeza;
CALayer *baba;

//animaciones del caracol
CABasicAnimation *cabezamove;
CABasicAnimation *caracolmove;
CABasicAnimation *maskmove;
```

Se van colocando las capas y los objetos en su lugar correspondiente, con los valores que necesitan los parámetros asociados, en el fichero.m. Por ejemplo, para el objeto caracol se implementa:

```
caracol=[CALayer layer];
caracol=animalillo.layer;
caracol.bounds=CGRectMake(0.0f, 0.0f, 200.0f, 174.0f);
caracol.anchorPoint=CGPointMake(2.3f, 0.25f);
caracol.contents= (id)[[UIImage imageNamed:@"caracolillo.png"] CGImage];

cabeza=[CALayer layer];
cabeza.bounds=CGRectMake(0.0f, 0.0f, 200.0f, 174.0f);
cabeza.position=CGPointMake(0.0f, 174.0f);
cabeza.anchorPoint=CGPointMake(0.0f, 1.0f);
cabeza.contents= (id)[[UIImage imageNamed:@"cabeza_caracol.png"] CGImage];

[caracol addSublayer:cabeza];
```

Una *sublayer* dentro de una *layer* seguirá los mismos movimientos que se le hayan mandado a la capa en la que está contenida. Por eso, aunque la cabeza del caracol sea una imagen diferente del cuerpo, al estar contenida dentro de su capa, seguirá su mismo movimiento.

De igual forma se crean las animaciones con sus inicios, finales y su duración. Como se ha comentado antes, se diferencian dos momentos: aquí se presenta la animación durante la cuenta atrás.

En este caso, la variación del caracol se da en su *AnchorPoint* que es su posición referida a la vista en la que está contenida. Por su parte, la máscara, al igual que ocurría con el reloj de arena, varía los límites de sí misma, aumentando su anchura con el paso del tiempo.

```
//Creamos la animacion del caracol
caracolmove=[CABasicAnimation animationWithKeyPath:@"anchorPoint"];
caracolmove.fromValue=[NSValue valueWithCGPoint: CGPointMake(2.3, 0.25)];
caracolmove.toValue=[NSValue valueWithCGPoint: CGPointMake(-0.8, 0.25)];
caracolmove.duration=cuenta_atras;
caracolmove.fillMode = kCAFillModeForwards; //estas dos líneas hacen que la animación se mantenga en su posición
caracolmove.removedOnCompletion = NO;


maskmove=[CABasicAnimation animationWithKeyPath:@"bounds"];
maskmove.fromValue=[NSNumber valueWithCGRect: CGRectMake(0.0f, 0.0f, 150.0f, 95.0f)];
maskmove.toValue=[NSNumber valueWithCGRect: CGRectMake(0.0f, 0.0f, 700.0f, 95.0f)];
maskmove.duration=cuenta_atras;
maskmove.fillMode = kCAFillModeForwards;
maskmove.removedOnCompletion=NO;


// Añadimos la animación a la capa correspondiente y así la inicializamos:

babamask.speed=1.0; //Activamos la velocidad en caso de que haya sido desactivada.
caracol.speed=1.0;

[caracol addAnimation:caracolmove forKey:@"caracolmove"];
[babamask addAnimation:maskmove forKey:@"maskmove"];
```

También se añaden las animaciones a sus objetos para que se inicien en el momento del *Start*.

Asimismo, se puede ver que se activa la velocidad de las animaciones en este momento, ya que en algún momento anterior se han parado, teniendo en cuenta el stop oculto comentado anteriormente.

Como segundo momento importante, cabe destacar el fin de la cuenta atrás, o cuando el crono llega a cero. En este momento se quiere que el caracol se anime, dando a entender que ha llegado a su fin la actividad que se estaba realizando. El animal se come la hoja. Así pues, se crea la animación correspondiente y se añade:

```
//el caracol se come la hoja

cabezamove=[CABasicAnimation animationWithKeyPath:@"transform.rotation.z"];
cabezamove.fromValue = [NSNumber numberWithFloat:0];
cabezamove.toValue = [NSNumber numberWithFloat:0.05];
cabezamove.duration=1;
cabezamove.autoreverses=YES;
cabezamove.repeatCount=2;

[cabeza addAnimation:cabezamove forKey:@"cabezamove"];
hoja.contents= (id)[[UIImage imageNamed:@"hoja_mordida.png"] CGImage];
```

En este momento del proyecto se ve la necesidad de añadir un efecto sonoro que estimule de algún modo al niño o niña y que enfatice el fin de la actividad. Éste no debe de ser estridente para no sobresaltar al usuario, y además debe de ser cómodo de oír y de entender.

Así que como punto final de esta aplicación, se añade el sonido de un mordisco, el perfecto para señalar que el animal se ha comido la hoja, y se ha terminado la actividad a realizar.

Para ello, es preciso importar al archivo de implementación, la librería que enlazará las funciones de audio:

```
#import <AudioToolbox/AudioServices.h>
```

Posteriormente se programa mediante código la reproducción del sonido en el momento deseado:

```
//Hacemos que suene el mordisco
SystemSoundID mordisco;
AudioServicesCreateSystemSoundID((CFURLRef)[NSURL fileURLWithPath:[[NSBundle mainBundle]
                                                    pathForResource:@"mordisco"
                                                             ofType:@"wav"]], &mordisco);
AudioServicesPlaySystemSound (mordisco);
```

En el resultado final también se puede apreciar un cambio en la estructuración de los elementos dentro de la pantalla, en comparación con el reloj de arena, con el fin de que la distribución sea más ordenada y más comprensible:

Figura 26. Animación del caracol durante la cuenta atrás.

4.3. ÁLBUM DE FOTOS:

La siguiente acción a realizar que se plantea es la creación de otra aplicación: el álbum de fotos.

Aprovechando la resolución del dispositivo táctil se ve que es imprescindible contar con un visor de imágenes, que muestre las fotografías personales que hay en el iPad y los dibujos que se hayan realizado en la pizarra del mismo comunicador.

Por otra parte, la necesidad de que el niño en cuestión aprenda el *lenguaje* con el que va a trabajar, está latente en todo momento. Por tanto también se alojarán en este álbum todos los pictogramas que componen la base de datos y que puedan ser mostrados uno a uno en el visor.

El acceso a esta aplicación se da a través de la parte inferior derecha de la pantalla principal.

Figura 27. Pantalla inicial de la aplicación que da acceso al Álbum.

Se plantea entonces un diseño inicial.

4.3.1. Planteamiento inicial:

El álbum debe contar de tres galerías de imágenes, o tres categorías diferentes, que conformen las fotografías contenidas en el iPad, los dibujos realizados en la pizarra, y los pictogramas asociados a la base de datos, de la siguiente manera:

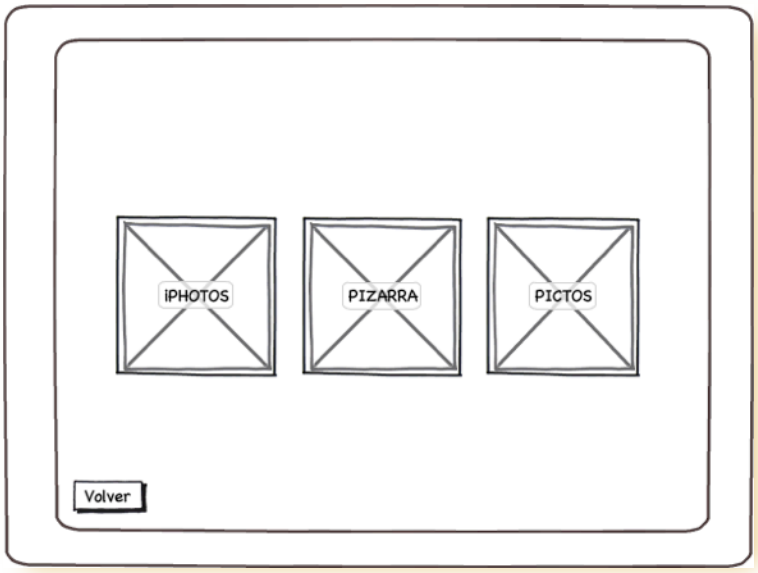

Figura 28. Esquema de la selección de los tres álbumes.

La idea es que al pulsar cualquiera de estos tres álbumes la siguiente pantalla cargue las imágenes contenidas en cada uno de ellos como si fuera una galería de miniaturas, en total 15, y de haber más, que exista una posible navegación mediante dos botones flechas.

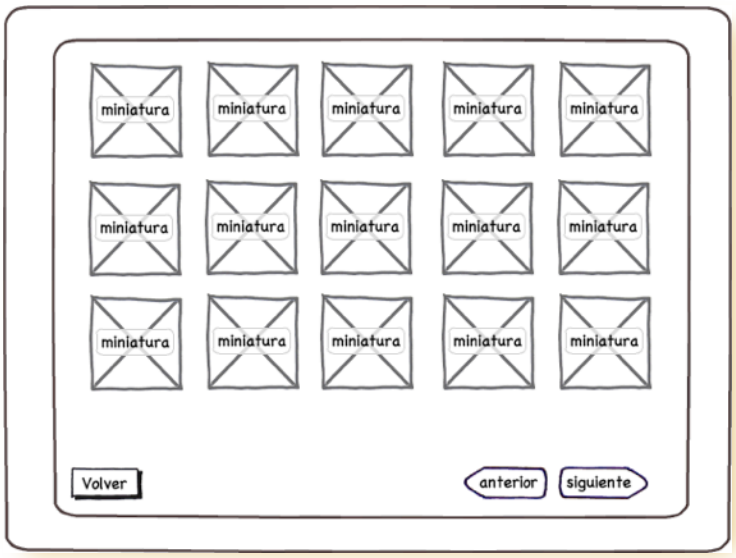

Figura 29. Esquema de la Galería de Miniaturas.

El siguiente paso es que al hacer clic en cualquiera de las imágenes miniatura aparezca un visor maximizado de la foto en cuestión. Éste debe de ser táctil y navegable, de tal forma que al *arrastrar* con los dedos se pueda acceder a la anterior o posterior imagen.

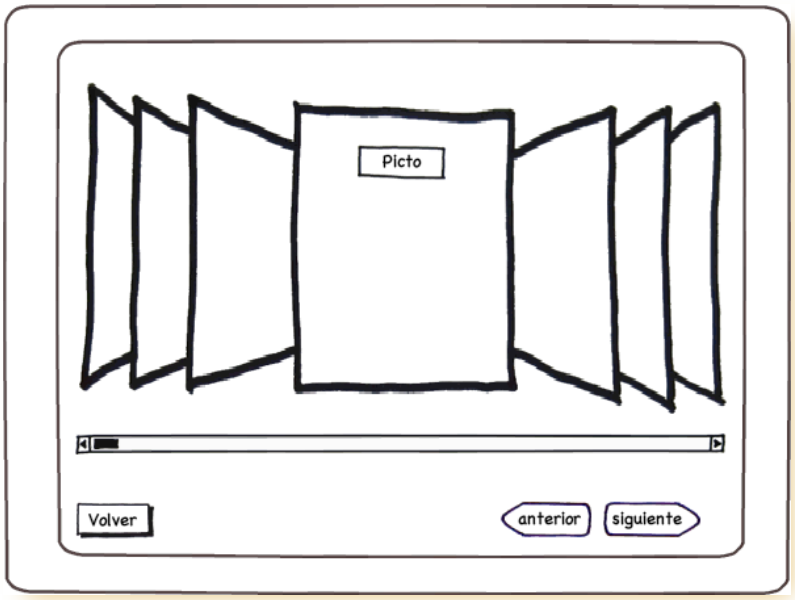

Figura 30. Esquema del Visor de Imágenes.

Ésta nueva función estimularía en gran medida a la persona que haga de su uso, existiendo una intencionalidad en todo caso del *pase de página* para observar el objeto próximo.

Aún así, puede existir dificultad para ciertos sujetos a la hora de utilizar esta opción, y por eso se decide añadir igualmente que en la galería de miniaturas, unas flechas en forma de botones que faciliten la navegación.

En cualquier caso el *arrastre táctil* serviría para mejorar las habilidades motoras de cualquier persona con dificultad.

4.3.2. Estructura de diseño:

Una vez se tienen claros cuales son los objetivos de esta nueva aplicación, es necesario elaborar una estructura de diseño.

Si se analiza en profundidad, se observa que el álbum es claramente en sí mismo lo que se denomina una aplicación multivista. Así que al igual que se planteó la organización de la aplicación general, habrá que hacerlo con ésta.

Se define entonces un diagrama con las funciones principales de este álbum – galería:

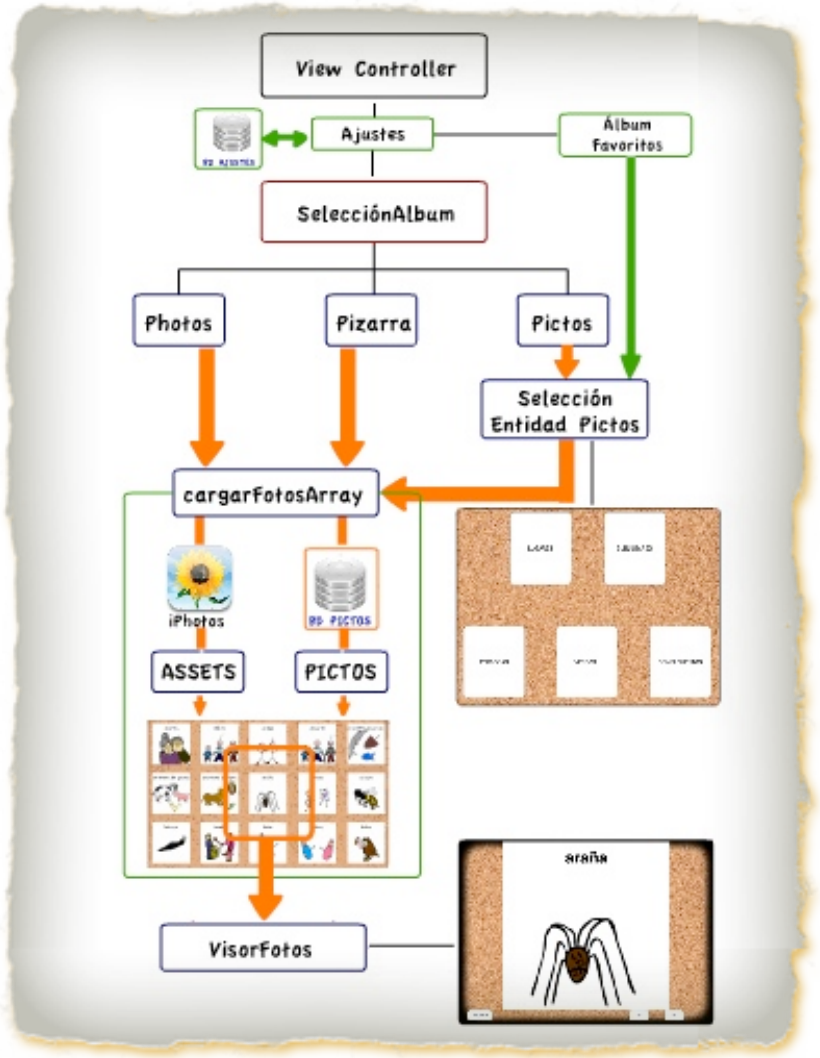

Figura 31. Diagrama de funcionamiento del Álbum.

En un principio las cuatro funciones principales que aquí aparecen son las correspondientes a cada pantalla independiente. Por ello se crean cuatro controladores, o clases diferentes, que controlen cada una de las vistas: *seleccionAlbum, seleccionEntidadPictos, cargarFotos y VisorFotos.*

Como sus propios nombres indican, estas cuatro clases se encargan de las tareas principales del álbum: elección de la categoría de fotos a ver, selección de la agrupación de imágenes si se trata de pictogramas, carga de las mismas en la galería de miniaturas y selección de la imagen deseada para verla ampliada en el visor navegable.

A continuación se detalla cada una de ellas en el *Manual de Configuración*.

4.3.3. Manual de Configuración:

Se cuenta con un *controlador principal* que se encarga de dar paso a la aplicación del álbum. A partir de aquí, tres controladores independientes hacen las funciones principales y se intercambian información internamente (de forma transparente para el usuario) para mostrar las imágenes deseadas en cada momento.

SELECCIÓN DEL ÁLBUM

Es el más sencillo. Tiene tres botones y sus únicas funciones son las de llamar a la siguiente clase *cargarFotos* con la pulsación de cada botón, y avisarle qué imágenes debe sacar.

Los álbumes se llamarán a partir de aquí *Photos*, *Pizarra* y *Pictos*.

En primer lugar se crean las acciones que van a realizar los botones y a continuación se asocia a cada uno de ellos creado en *InterfaceBuilder* su método correspondiente:

```
-(IBAction)volverPpal:(id)sender;

-(IBAction)onBotonPhotos:(id)sender;
-(IBAction)onBotonPizarra:(id)sender;
-(IBAction)onBotonPictos:(id)sender;
```

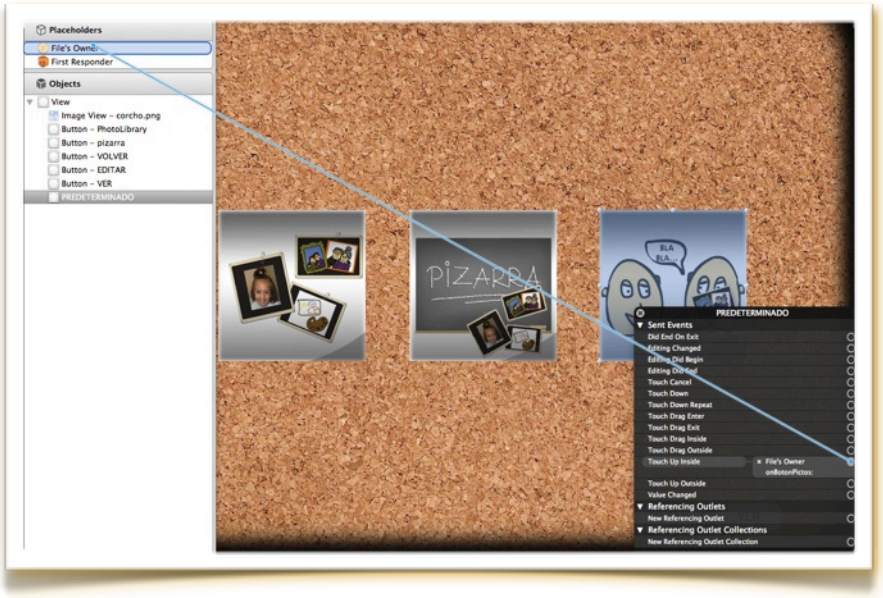

Figura 32. Asociación de botones a sus acciones.

Posteriormente habrá que implementar los métodos de cada una de estas funciones de pulsación, así que se piensa qué es lo que se desea ejecutar con cada una de ellas.

Como objetivo principal está el cargar las fotos en la pantalla contigua, por lo que es necesario importar la clase a la que se llama, crear una instancia de la misma, y llamarla desde el método en cuestión.

Para ello, en *seleccionAlbum.h* se escribe algo así:

```
@class cargarFotos;

@interface seleccionAlbum : UIViewController {

    cargarFotos * cargarFotosVista; //creamos una instancia de cargarFotos de la que utilizaremos su vista
```

Y por su parte en la parte de implementación *seleccionAlbum.m* se añade, por ejemplo:

```
-(IBAction)onBotonPhotos:(id)sender
{
    cargarFotosVista=[[cargarFotos alloc] initWithNibName:@"cargarFotos" bundle:[NSBundle mainBundle]];
    [cargarFotosVista cargarDatosEnArray:0];

    [self.view addSubview:cargarFotosVista.view];
}
```

Para cada pulsación del álbum elegido se escribe el código correspondiente, llamando a la siguiente clase indicando qué imágenes debe cargar.

Por último, para volver atrás, al menú inicial o a la ventana anterior, es necesario asociar una sencilla acción al botón volver que será la misma para todos los momentos en los que se desee volver hacia atrás, y no será mencionada de nuevo más adelante.

```
-(IBAction)volverPpal:(id)sender
{
    [self.view removeFromSuperview];
}
```

En definitiva el resultado inicial es el siguiente:

Figura 33. Resultado de la pantalla principal del Álbum: Selección Álbum.

Posteriormente y debido al desarrollo del proyecto, se comenta que estos botones se cambian, pero no sus acciones asociadas ni los eventos que lanzan.

SELECCIÓN ENTIDAD PICTOS

Hay que señalar que hay una segunda ventana que se encargará de otra selección de álbumes, y esta es la correspondiente a la elección de la visión de pictogramas.

Para la muestra y el aprendizaje de los pictos es necesario organizarlos y distribuirlos igualmente en categorías.

Se decidió que al igual que la base de datos, los pictogramas estarían categorizados en *lugares, subyugares, personas, verbos y complementos*.

Así pues, se crea una pantalla que así los filtre.

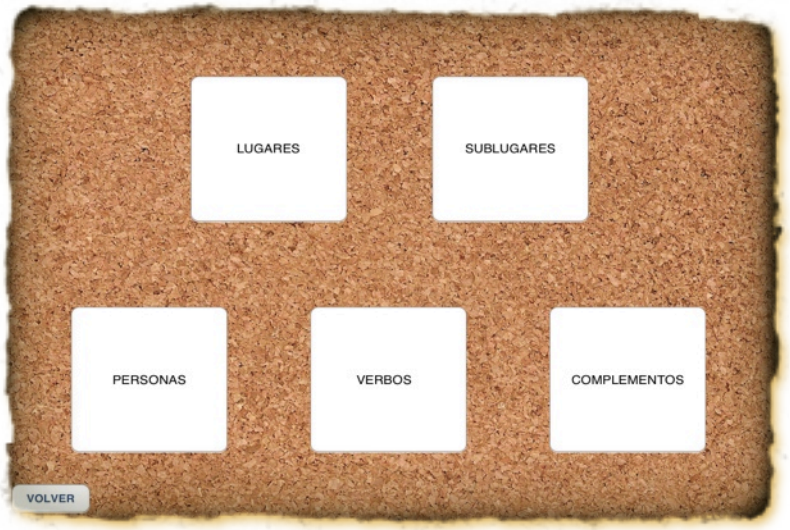

Figura 34. Selección de la categoría de pictos dentro del Álbum: Selección Entidad Pictos.

La programación de esta clase es tan sencilla como la anterior, simplemente hay que asociar a cada botón una acción común que permita la llamada pantalla siguiente.

Figura 35. Asociación de botones a sus acciones.

Además, a cada botón se le da un número identificador, el cual se envía a la clase siguiente para que sepa qué categoría se ha elegido, y por tanto las imágenes que debe cargar.

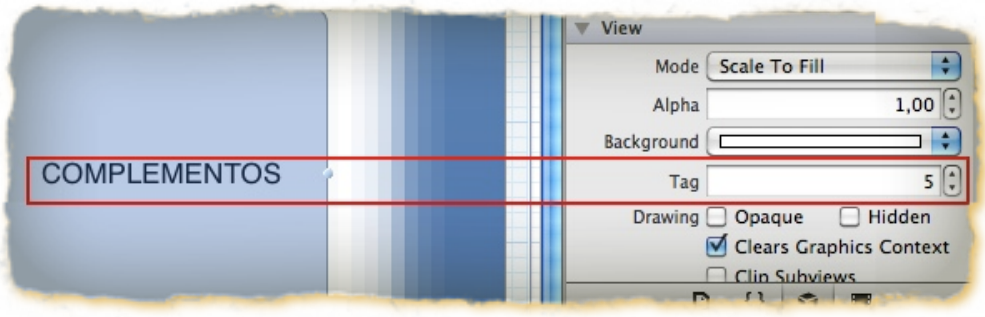

Figura 36. Asignación del número de identificación de los botones: tag.

El código que se inserta en la acción *onEntidadPicto* corresponde a las líneas:

```
-(IBAction)onEntidadPicto:(id)sender
{
    NSInteger ButtonTagInteger= [sender tag];

    [cargarFotosVista cargarDatosEnArray:ButtonTagInteger];
    [self.view addSubview:cargarFotosVista.view];
}
```

El siguiente paso es cargar las fotos en la galería de miniaturas.

CARGAR FOTOS

Permite que el usuario eche un vistazo de manera rápida a las fotos o pictogramas que puede ver a través de las páginas de navegación como si de un álbum analógico se tratase.

Posibilita la interacción del niño que, mediante una intencionalidad, elige el elemento que desea ver ampliado.

Como se ha visto en esquemas anteriores, su vista posee 18 botones, de los cuales 15 son las miniaturas de las imágenes de la galería, y los otros tres tienen la función de navegación. Además, contiene 15 etiquetas, que varían su texto dependiendo del picto que se muestre en cada botón. Un último elemento es un indicador de actividad.

Figura 37. Diagrama de funcionamiento de la clase CargarFotos.

Entre sus tareas, se encuentran destacadas las de:

- **Cargar fotos:** acceder al lugar donde se encuentren las imágenes a mostrar.
- **Representar las miniaturas:** Mostrar las fotos y el texto en cada botón y actualizarlos con el paso de página.
- **Navegar entre imágenes:** Pasar a la página siguiente o la anterior si hay más de 15 miniaturas.
- **Acceder al visor:** Al pulsar cualquier botón miniatura se tiene que incorporar la siguiente clase: el visor de imágenes.
- **Volver al menú:** Retroceder a la pantalla anterior.

Carga de fotos

En primer lugar hay que tener en cuenta que la carga de fotos se realiza principalmente desde dos lugares diferenciados.

La primera, toma las imágenes desde el propio dispositivo, desde el iPhoto, donde están guardadas todas las fotografías del iPad distribuidas en los álbumes que el usuario haya deseado y además los dibujos de la pizarra, puesto que se guardan en el mismo lugar.

La segunda, por otra parte, coge y representa todos los pictogramas que haya alojados dentro del programa, simplemente buscando cada uno por su nombre.

Como se cita entonces arriba, es necesario entrar en una aplicación externa para coger las fotografías del álbum *Photos* y *Pizarra*.

La librería de Xcode que posibilita el acceso a los objetos de su álbum iPhoto y que permite la representación de todos a la vez en la aplicación creada se llama ***AssetsLibrary***.

Cada elemento que se coja mediante las funciones de esta librería se denomina *asset*, y es distinto a una imagen común, puesto que posee no sólo información de color sino también dimensiones, *thumbnails*, álbum al que pertenece, etc, de la imagen que representa. Por eso ha de ser tratado de una forma especial.

En este punto del análisis del proyecto, se observa que va a resultar excesivamente complicado separar las imágenes guardadas como dibujos de la pizarra del resto, ya que se encuentran normalmente alojadas dentro del mismo lugar *Fotos guardadas* en el iPad. Se dice "*normalmente*" puesto que depende totalmente del usuario, si ha creado uno o más álbumes, y por lo tanto es incontrolable mediante la aplicación.

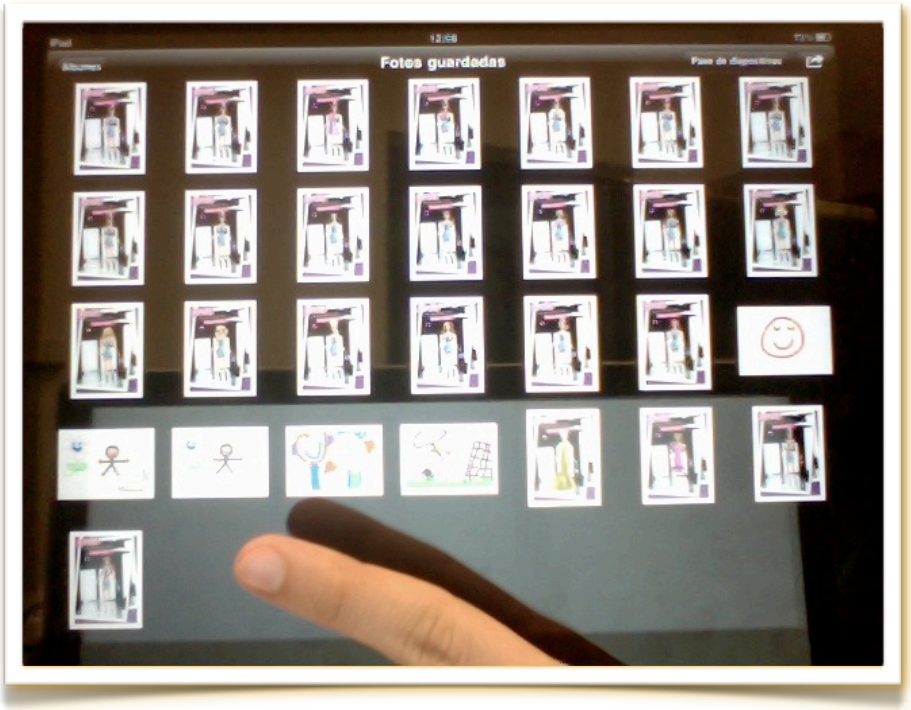

Figura 38. Fotografía de un iPad que muestra el alojamiento en el iPhotos de fotografías personales y dibujos de la Pizarra.

Así que se decide llevar a cabo la supresión de uno de los álbumes, el de *Pizarra*, y la incorporación del mismo al general de *Photos*. El cambio no es de gran importancia ya que lo que realmente se quiere es distinguir el álbum de los pictogramas del resto y esto sí que es controlable.

Una vez solucionada la procedencia de las imágenes, el siguiente paso es cogerlas.

Como se ha comentado, hay que tratar de forma diferente *assets* de *imágenes*, por ello se crea una variable que establezca esta distinción que se llama *métodoCarga* y cuyo valor será 0 (cero) o 1 (uno) respectivamente.

Así pues se cargan todas las imágenes procedentes de iPhoto mediante el siguiente código en un *NSMutableArray* llamado *assets*. A continuación se llama a la función que las representa *reloadRenderPics*.

```
[library enumerateGroupsWithTypes:ALAssetsGroupAll
                       usingBlock:
                    // Group enumerator Block
                        ^(ALAssetsGroup *group, BOOL *stop)
                    {
                           if(group != nil)
                           {
                               [group enumerateAssetsUsingBlock:

                                // Asset enumerate Block
                                ^(ALAsset *result, NSUInteger index, BOOL *stop)
                               {
                                    if(result != NULL)
                                    {
                                        [assets addObject:result];
                                    };
                                }
                                 ];
                           };
                        [self.activity stopAnimating];
                        [self.activity setHidden:YES];
                        metodoCarga=0;
                        [self reloadRenderPics];
                    }
                     failureBlock:
                        ^(NSError *error) {
                            NSLog(@"Failure");}
 ];
[library release];
```

Por otra parte, para representar las imágenes normales (los pictos) procedentes de la base de datos, se debe hacer una llamada a la misma y obtener todos los elementos que cumplan una cierta condición. Por ejemplo, que pertenezcan al grupo *Personas*: Estos se guardan igualmente que en el caso anterior, en un *array* distinto (puesto que son objetos distintos), llamado *objectsCogidos*.

```
////////////////////////////////////////////////////////////////
// LLAMADA A LA BD ENTIDAD ORIGINAL
////////////////////////////////////////////////////////////////
NSLog(@"Se está cargando el álbum PERSONAS");
entityDesc = [NSEntityDescription entityForName:@"Personas" inManagedObjectContext:context];
request = [[NSFetchRequest alloc] init];
[request setEntity:entityDesc];
objects = [context executeFetchRequest:request error:&error];

if ([objects count] == 0)
{
    [alertaSimple show];
}
else
{
    [self.activity stopAnimating];
    [self.activity setHidden:YES];
    metodoCarga=1;
    [self.activity stopAnimating];
    [self.activity setHidden:YES];
    metodoCarga=1;
    [objectsCogidos setArray:objects];

    [self reloadRenderPics];
    [self.activity stopAnimating];
    [request release];

}
```

Un pequeño detalle de esta carga de fotos, es que se hace uso de un elemento llamado *UIActivityIndicatorView*, encargado de aparecer mientras dure la carga de imágenes en el *NSMutableArray* correspondiente.

Con él, en el caso de que la cantidad de elementos a cargar sea importante, la espera se hará entendible, al indicar que el programa continúa trabajando y no permitir que una pantalla vacía aparezca durante los segundos que pueda permanecer así.

Figura 39. Elemento UIActivityIndicatorView.

Además, si por lo que sea no existen objetos en la base de datos con las características solicitadas, el programa lanzará una *UIAlertView* con el aviso:

```
alertaSimple = [UIAlertView alloc];
[alertaSimple initWithTitle:@"ALERTA"
                    message:@"No hay pictos asociados"
                   delegate:self
          cancelButtonTitle:@"Aceptar"
          otherButtonTitles: nil];
```

Figura 40. Alerta lanzada cuando no se encuentran pictos asociados en la base de datos.

Representación de miniaturas

La representación de los objetos cogidos en los botones que hacen de miniaturas también es diferenciada para los casos de *assets* e imágenes.

Para mostrar los primeros, se ve que supone un problema el sacar cada imagen y que ésta se ajuste automáticamente al botón, puesto que requiere el uso de gran cantidad de memoria del dispositivo, al mostrar el contenido completo de 15 imágenes a la vez, pero ajustadas al tamaño de la miniatura.

Se recurre pues a la utilización de su propiedad *thumbnail*, que como se ha comentado anteriormente, forma parte de la información que cada *asset* contiene. Ésta miniatura ya no ocupa tanto, de hecho es de menor tamaño que el botón, por lo que se escala ligeramente para que encaje a la perfección en su interior.

```
[[thumbnail_table objectAtIndex:i] setImage:[UIImage imageWithCGImage:[[assets objectAtIndex:(i+indice)] thumbnail]
                                    scale:0.5 orientation:UIImageOrientationUp] forState:UIControlStateNormal];
```

Con esto se pierde calidad en la visualización de la representación, pero se considera más que suficiente al tratarse de una única muestra que sirve para posteriormente y tras ser pulsada, dar acceso al visor en completo de la misma.

Por otra parte, los objetos recogidos de la llamada a la base de datos poseen la información de unas determinadas columnas llamadas *keys*.

TABLE ZPERSONAS Buscar Mostrar Todo

rowid	Z_PK	Z_ENT	Z_OPT	ZNOMBRE	ZNOMBRE_MAY	ZNOMBRE_ING	ZNOMBRE_MAY_ING
1	1	1	5	abuelos	ABUELOS	grandparents	GRANDPARENTS
2	2	1	1	adulto	ADULTO	adult	ADULT
3	3	1	1	amigo	AMIGO	friend	FRIEND
4	4	1	1	anciano	ANCIANO	elderly	ELDERLY
5	5	1	1	animales acuáti...	ANIMALES ACU...	aquatic animals	AQUATIC ANIMALS
6	6	1	1	animales de gr...	ANIMALES DE G...	farm animals	FARM ANIMALS
7	7	1	1	animales salvajes	ANIMALES SALV...	wild animals	WILD ANIMALS
8	8	1	1	araña	ARAÑÑA	spider	SPIDER
9	9	1	1	artista	ARTISTA	artist	ARTIST
10	10	1	1	avispa	AVISPA	wasp	WASP
11	11	1	1	babosa	BABOSA	slug	SLUG
12	12	1	1	banda	BANDA	band	BAND

Figura 41. Ejemplo de vista de las columnas que conforman la base e datos.

Las imágenes se encuentran guardadas dentro de la aplicación y su búsqueda y representación atiende a su nombre, que está contenido dentro de la columna *nombre*.

```
[[thumbnail_table objectAtIndex:i] setImage:[UIImage imageNamed:[[objectsCogidos objectAtIndex:(i+indice)]
                                                    valueForKey:@"nombre"]]forState:UIControlStateNormal];
```

De igual manera, los *label text* que están encima de cada botón, toman el valor que corresponda en la base de datos, en este caso dependiendo de los ajustes que haya habido y que estén conservados dentro de la variable *Idioma*:

```
if (ajuste_tipo_letra==20)
{
    Idioma=@"nombre";
}
else if (ajuste_tipo_letra==30)
{
    Idioma=@"nombre_may";
}
else if (ajuste_tipo_letra==50)
{
    Idioma=@"nombre_ing";
}
else if (ajuste_tipo_letra==60)
{
    Idioma=@"nombre_may_ing";
}
```

ZNOMBRE	ZNOMBRE_MAY	ZNOMBRE_ING	ZNOMBRE_MAY_ING
abuelos	ABUELOS	grandparents	GRANDPARENTS
adulto	ADULTO	adult	ADULT
amigo	AMIGO	friend	FRIEND
anciano	ANCIANO	elderly	ELDERLY
animales acuáti...	ANIMALES ACU...	aquatic animals	AQUATIC ANIMALS
animales de gr...	ANIMALES DE G...	farm animals	FARM ANIMALS
animales salvajes	ANIMALES SALV...	wild animals	WILD ANIMALS
araña	ARAÑÑA	spider	SPIDER
artista	ARTISTA	artist	ARTIST
avispa	AVISPA	wasp	WASP

Figura 42. Llamada por código a a base de datos.

```
[[label_table objectAtIndex:i] setText:[[objectsCogidos objectAtIndex:i+indice]valueForKey:Idioma]];
```

Como es lógico, para ambos casos se utiliza un bucle *for* que muestra cada elemento con ayuda de un puntero.

```
for (int i=0; i<[thumbnail_table count]; i++)
```

Los botones dentro de la vista son controlados mediante programación, y se ocultan o se muestran según el número de objetos que se hayan obtenido en las variables *assets* y *objectsCogidos*.

Paso de página

Para controlar la recarga de las imágenes con el paso de página existe un índice que varía según el momento y que recorre cada elemento dentro de los *NSMutableArray*.

```
indice=num_pag_view*[thumbnail_table count];
```

Como siempre hay 15 botones en pantalla, el valor que este *índice* tiene continuamente varía entre: 0, 15, 30, 45, 60, … Dependiendo de la página en la que se halle.

Esto se controla mediante los botones *anterior* y *siguiente* asociados a las siguientes acciones:

Figura 43. Asociación de los botones de navegación a sus acciones.

A través de estas acciones se reconoce si es la última o la primera página, lo cual provoca que se oculten e inhabiliten dichos botones asociados.

```
-(IBAction)pulsarPagAnterior:(id)sender
{
    [BotonSiguiente setEnabled:YES];
    [BotonSiguiente setHidden:NO];

    if (num_pag_view>0) {
        num_pag_view -=1;
    }
    if(num_pag_view==0)
    {
        [BotonAnterior setEnabled:NO];
        [BotonAnterior setHidden:YES];
    }
    [self reloadRenderPics];
}
```

```
(IBAction)pulsarPagSiguiente:(id)sende

    num_pag_view +=1;

    [self reloadRenderPics];

    [BotonAnterior setEnabled:YES];
    [BotonAnterior setHidden:NO];
```

Una vez cambiada la página actual se envía la orden de recarga de imágenes y se tiene ya el ciclo completo.

Acceso al Visor

El último paso dentro de esta clase es el incorporar la acción de llamamiento a la nueva clase. Como ya se ha comentado, al pulsar cualquiera de los botones miniaturas se pretende que se lance una nueva función que mantenga la foto seleccionada en un visor ampliado y siga posibilitando la navegación entre las imágenes del álbum.

Pues bien, esto se realizará mediante una sola acción asociada a todos los botones.

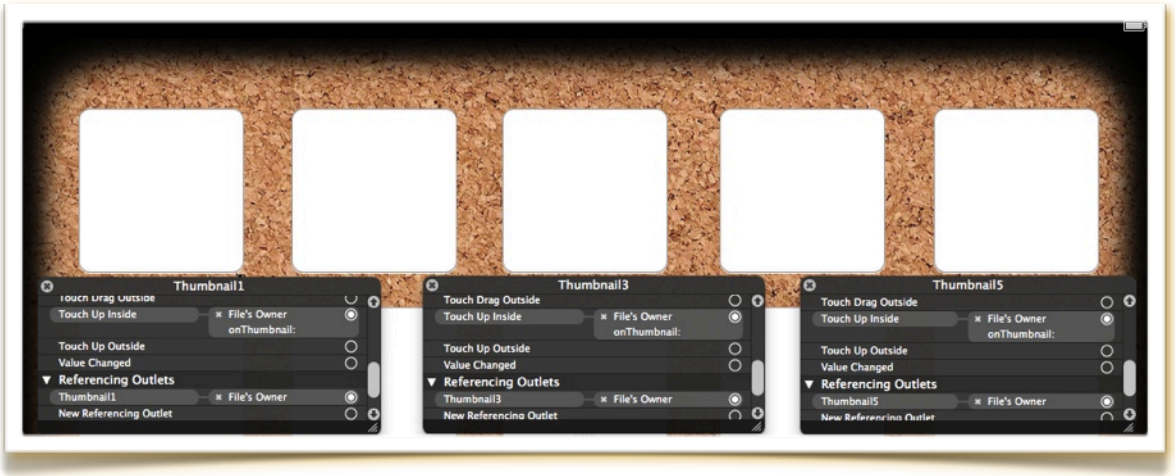

Figura 44. Asociación de cada thumbnail a su acción común.

Además, cada *thumbnail* posee su propio número de identificación entre 0 – 14 que facilitará el llamamiento.

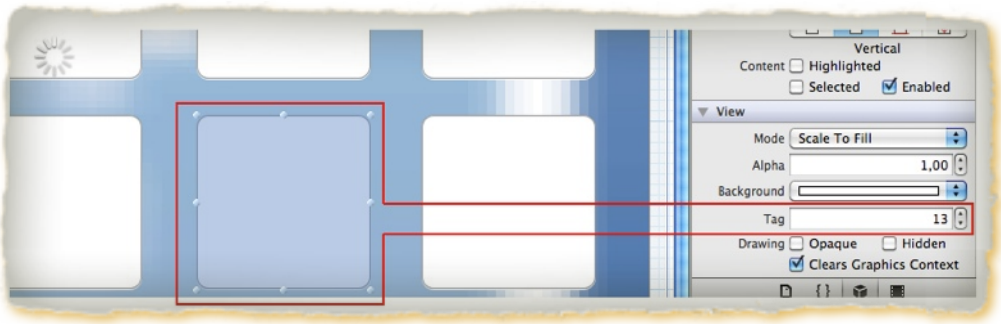

Figura 45. Asignación del tag a cada botón miniatura.

Para dar paso al visor, es necesario enviarle cierta información. Entre ella se encuentra una combinación entre *tag* del botón pulsado y la página actual, que corresponde a la imagen seleccionada en *cargarFotos*, y el *array* de objetos a mostrar.

```
-(IBAction)onThumbnail:(id)sender
{
    [self.view addSubview:VisorFotosVista.view];

    NSInteger ButtonTagInteger= [sender tag];
    int imagenSeleccionada=((num_pag_view*[thumbnail_table count])+ButtonTagInteger);

    NSMutableArray*nuevasImagenes=[[NSMutableArray alloc]init];

    if (metodoCarga==0)
    {
        nuevasImagenes=assets;
    }
    else
    {
        nuevasImagenes=objectsCogidos;
    }

    [VisorFotosVista cargarImagen:imagenSeleccionada cargarArray:nuevasImagenes metodo:metodoCarga];
}
```

Siempre hay que acordarse de importar la siguiente clase a ésta mediante el código adecuado, sino no habrá acceso posible.

VISOR DE FOTOS

Mediante esta clase la persona puede disfrutar de la visión de forma ampliada de los pictogramas que componen el comunicador, así como de las fotos que estén contenidas en el iPad o los dibujos que haya realizado en la aplicación de *Pizarra*.

Se trata de un visor central que permite pasar a la siguiente imagen mediante el arrastre con los dedos de forma horizontal sobre la pantalla del dispositivo. Como ya se ha comentado anteriormente, esta funcionalidad permite al usuario estimular su capacidad motora.

Además, para aquellos que por diversas causas no puedan interactuar con este elemento de arrastre, se han diseñado igualmente dos componentes de navegación.

Las tareas principales de esta clase son entonces:

- **Dimensionado del *scroll*:** detección de las imágenes a mostrar y ajuste del tamaño del scroll a las mismas.

- **Estructura interna:** incluir el *array* de imágenes dentro del visor.

- **Elemento seleccionado:** mostrar la primera fotografía seleccionada en la clase anterior en primer lugar.

- **Método delegación y actualización:** en el momento del arrastre ejecutar la función que muestre la imagen que corresponde a continuación.

- **Navegación externa:** mediante las flechas anterior, siguiente y vuelta atrás.

El elemento principal que se utiliza en esta clase es un *UIScrollView*. La idea del funcionamiento de este objeto es que tiene un tamaño determinado, el cual puede albergar cierta cantidad de vistas, que incluyen ciertas imágenes.

Posee un protocolo de delegación, lo que supone que hace caso a una serie de funciones internas que en todo momento de la ejecución del programa se están llevando a cabo.

Por ejemplo, responde al método de ***scrollViewDidScroll***, lo que indica que en cualquier momento que se comience el arrastre táctil del visor se estará lanzando esta función.

***Dimensionado del scroll*:**

El movimiento que puede realizar este elemento es tanto vertical como horizontal, pero en este caso se desea que sea exclusivamente horizontal.

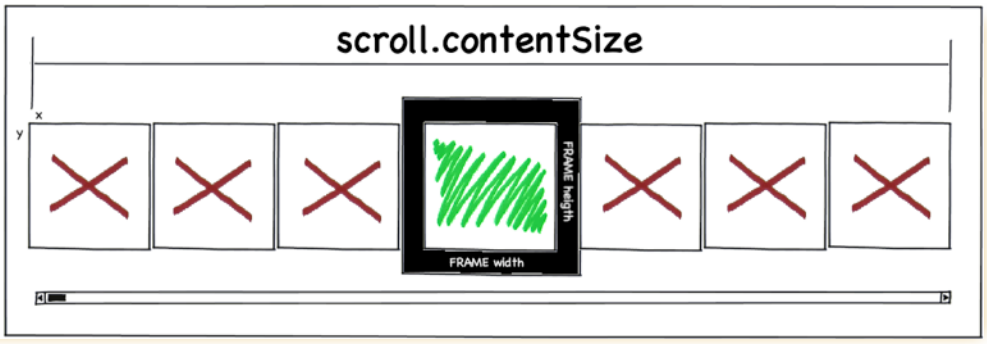

Figura 46. Esquema de un UIScrollView.

En todo momento se encuentra mostrando una foto dentro de su *frame*, aquella que se le indique, como si tuviera un visor propio.

En esta aplicación se presenta un pequeño problema a la hora de mostrar las diferentes imágenes. Por un lado, está diseñada para trabajar en formato horizontal, ya que el iPad dispone de forma rectangular. Esto provoca que las fotos que se obtienen con él, así como los dibujos que se realicen mediante la pizarra, tengan formato apaisado.

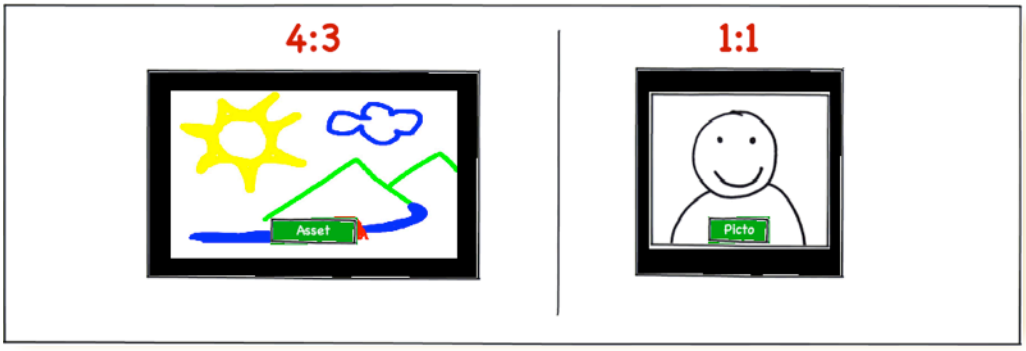

Figura 47. Formatos diferentes de imágenes a mostrar en el visor.

Al contrario ocurre con los pictogramas, que están creados con dimensiones cuadradas y que, como se puede suponer, se deforman a tamaño completo con un visor rectangular.

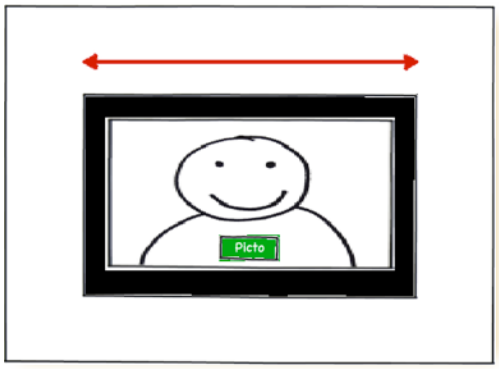

Figura 48. Deformación de los pictogramas en un visor apaisado.

Éste resultado no sería para nada el deseado, ya que lo que se pretende justamente es permitir el aprendizaje de los pictogramas para el uso del comunicador y no tendría sentido su enseñanza con los pictos deformados.

Se decide pues implementar un *scroll* común, que varía sus dimensiones dependiendo del tipo de imagen que muestra, o lo que es lo mismo, del *metodoCarga* que está activado. La altura en ambos casos, es siempre la misma.

En esta programación se definen unos parámetros constantes al inicio del código, para que resulte más fácil modificarlo, en el caso que se quiera, posteriormente.

```
#define ScrollX 8
#define ScrollY 8
#define ScrollWidth 1012
#define ScrollHeight 680

#define ScrollXPic 172
#define ScrollWidthPic 680
```

En primer lugar entonces, es necesario indicarle qué dimensiones va a alcanzar, en ambos casos, así como el tamaño del *frame* que va a mostrar.

```
// Cambiamos el tamaño del visor dependiendo de si estamos mostrando pictos o no.
if (metodoCarga==1)//mostramos pictos.
{
    scroll.frame = CGRectMake(ScrollXPic, ScrollY, ScrollWidthPic, ScrollHeight);
    scroll.contentSize = CGSizeMake(NumImages*ScrollWidthPic, ScrollHeight); //tamaño total del scroll
}
else if(metodoCarga==0) // Mostramos assets
{
    scroll.frame = CGRectMake(ScrollX, ScrollY, ScrollWidth, ScrollHeight);
    scroll.contentSize = CGSizeMake(NumImages*ScrollWidth, ScrollHeight); //tamaño total del scroll
}
```

El esquema del resultado final sería entonces algo así:

Estructura interna

La forma de mostrar imágenes dentro del *scroll* es incluyendo en su interior tantas vistas como elementos vaya a enseñar, y representar cada uno de ellos dentro de cada vista.

Así pues la primera idea fue la de crear esta cantidad de vistas dentro del *scroll* y cargar internamente todas las imágenes de tal forma que quedase esquematizado de la siguiente manera:

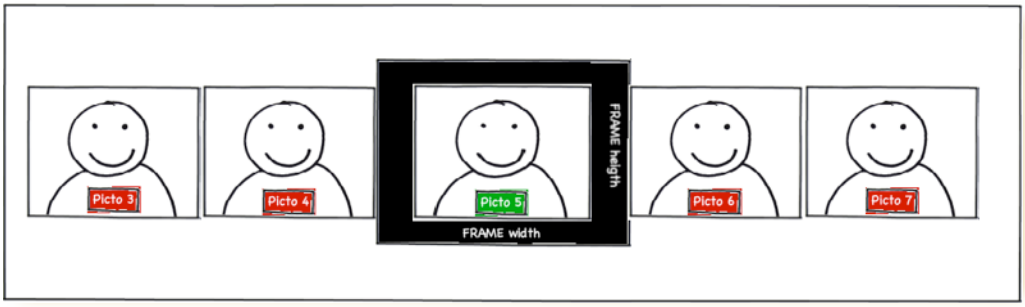

Figura 49. Esquema de la estructura interna inicial del UIScrollView.

Después de escribir el código adecuado, se prueba en un dispositivo real con una gran cantidad de imágenes y el miedo esperado ocurre: se consume una gran cantidad de memoria para la muestra de muchas fotografías. Así pues se ha de cambiar el diseño del *scroll*.

La alternativa surge con la idea de utilizar únicamente dos vistas dentro del *scroll* que, controladas por el método adecuado, vayan cargando la imagen que corresponda en cada momento, de la siguiente manera:

```
view1 = [[UIImageView alloc] init];
[scroll addSubview:view1];

view2 = [[UIImageView alloc] init];
[scroll addSubview:view2];
```

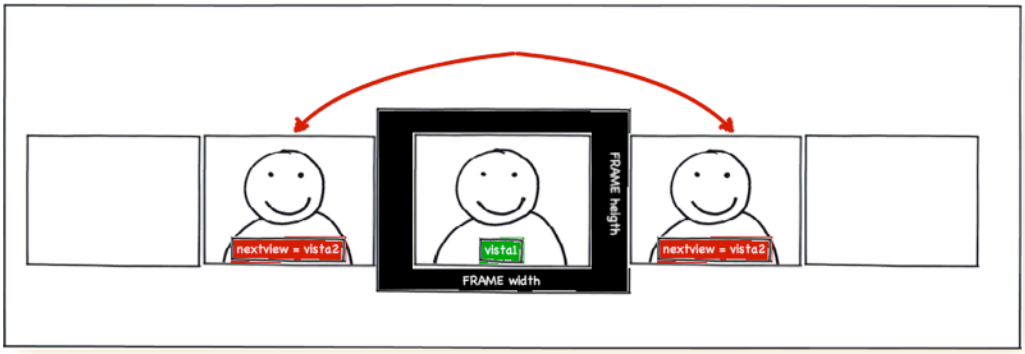

Figura 50. Esquema de funcionamiento final del UIScrollView.

La labor de la función que carga los elementos correctos según se realice el movimiento táctil se ve en el apartado contiguo de esta memoria.

Se comienza cargando en esta clase todos los elementos que nos da la anterior:

```
-(void)cargarImagen:(int)imagen_inic cargarArray:(NSMutableArray*)nuevasImagenes metodo:(BOOL)metodoDeCarga
{
    [imageSet autorelease];

    imageSet = [nuevasImagenes retain]; //imageSet:array de assets(imagenes).

    metodoCarga = metodoDeCarga;

    NumImages= [imageSet count]; // nº de imagenes dentro del array

    imagen_inicial = imagen_inic; //imagen que se ha clickado y hay que colocar la primera.
```

El siguiente paso es colocar la primera imagen en su lugar, la que ha sido seleccionada en la clase *cargarFotos*.

Elemento seleccionado

De igual forma que en la clase anterior hay que representar de distinto modo un *asset* de una *imagen* procedente de la base de datos.

Si el elemento es un pictograma, hay que tener en cuenta también que el texto que incluye debe estar en el idioma que anteriormente haya sido ajustado.

```
if (metodoDeCarga==1) // NSMutableArray = Assets
{
    view1.frame = CGRectMake((ScrollWidthPic*imagen_inicial), ScrollY, ScrollWidthPic, ScrollHeight); //
        Establecemos la posicion correspondiente a la imagen elegida dentro del scroll
    view2.frame = CGRectMake((ScrollWidthPic*(imagen_inicial+1)), ScrollY, ScrollWidthPic, ScrollHeight);

    view1.image = [UIImage imageNamed:[[imageSet objectAtIndex:(imagen_inicial)] valueForKey:@"nombre"]];
    view2.image = [UIImage imageNamed:[[imageSet objectAtIndex:(imagen_inicial+1)] valueForKey:@"nombre"]];

    [pictosText1 setText:[[imageSet objectAtIndex:(imagen_inicial)] valueForKey:Idioma]];
    [pictosText2 setText:[[imageSet objectAtIndex:(imagen_inicial+1)] valueForKey:Idioma]];

    [view1 addSubview:pictosText1];
    [view2 addSubview:pictosText2];

}
else //NSMutableArray = imagenes normales

{
    view1.frame = CGRectMake((ScrollWidth*imagen_inicial), ScrollY, ScrollWidth, ScrollHeight); //Establecemos
        la posicion correspondiente a la imagen elegida dentro del scroll
    view2.frame = CGRectMake((ScrollWidth*(imagen_inicial+1)), ScrollY, ScrollWidth, ScrollHeight);

    view1.image = [UIImage imageWithCGImage:[[[imageSet objectAtIndex:imagen_inicial]defaultRepresentation]
        fullScreenImage]];//Imagen elegida y siguiente
    view2.image = [UIImage imageWithCGImage:[[[imageSet objectAtIndex:imagen_inicial+1]defaultRepresentation]
        fullScreenImage]];

}
```

Anteriormente se han añadido al *scroll* las dos vistas que lo forman, pero no se ha establecido en qué punto de toda su longitud debe comenzar.

Es necesario indicar entonces cuál es el lugar exacto (*offset*) donde se encuentra el visor del *scroll* y por tanto de comienzo del mismo.

```
if (metodoCarga==1)
{
    [scroll setContentOffset:CGPointMake((ScrollWidthPic*imagen_inicial), ScrollY)]; //Cargar imagen
        inicial en el scroll
}
else
{
    [scroll setContentOffset:CGPointMake((ScrollWidth*imagen_inicial), ScrollY)]; //Cargar imagen
        inicial en el scroll
}
```

Una vez colocadas las vistas en su lugar y establecido el punto de partida, se procede a la función del movimiento del visor.

Método delegación y actualización

Como ya se ha comentado, el elemento scroll posee un método intrínseco que cumple durante la ejecución del programa y se llama ***scrollViewDidScroll*** y responde a cada movimiento táctil que se realice sobre él.

Lo deseado es que cuando se comience a mover éste elemento, se visualice la imagen contigua, bien sea la imagen de su derecha si el arrastre se hace a la izquierda, o la fotografía de su izquierda si el arrastre se realiza hacia la derecha, tal y como se ha mostrado en los esquemas anteriores.

En todo caso hay que indicarle qué debe mostrar durante este evento. Para ello aquí se crea un método llamado *update*.

```
# pragma mark -
# pragma mark UIScrollView Delegate

- (void) scrollViewDidScroll:(UIScrollView *) scrollView
{
    [self update];
}
```

La idea general de esta función es averiguar cual es la página del *scroll* que se está mostrando en cada momento para averiguar con el movimiento actual cuál es la página siguiente.

```
float currPos = scroll.contentOffset.x; //Posición del scroll en cada momento

int selectedPage = roundf(currPos / pageWidth); //Página que se ve en cada momento (empieza en la cero=0)

int nextpage = truePosition > currPos ? selectedPage-1 : selectedPage+1; //Nos movemos hacia la izq.
    o la dcha.
```

A continuación se asigna a una vista llamada *nextView* la imagen siguiente, dependiendo de si corresponde o no a un pictograma, y si necesita rellenarse con un campo de texto.

```
if (metodoCarga==1)
{
    nextView.frame = CGRectMake(nextpage*ScrollWidthPic, ScrollY, ScrollWidthPic, ScrollHeight);

    nextView.image = [UIImage imageNamed:[[imageSet objectAtIndex:(nextpage)] valueForKey:
        @"nombre"]];
    [pictosTextNext setText:[[imageSet objectAtIndex:(nextpage)] valueForKey:Idioma]];
}
else
{
    nextView.frame = CGRectMake(nextpage*ScrollWidth, ScrollY, ScrollWidth, ScrollHeight);

    nextView.image = [UIImage imageWithCGImage:[[[imageSet objectAtIndex:nextpage]
        defaultRepresentation]fullScreenImage]];
}
```

Por último se coloca esta "*imagen siguiente*" dentro de la vista creada anteriormente que le corresponda, teniendo en cuenta cual de ellas está activa en ese instante.

```
if (view1Active)// if ZONA==IMPAR --> nextView=view1; else -->nextView=view2
{
    nextView=view1;
    pictosTextNext=pictosText1;
}
else
{
    nextView=view2;
    pictosTextNext=pictosText2;
}
```

Navegación externa

Como último recurso queda establecer la navegación de las flechas, que es tan sencillo como hacer uso de una función del *scroll* que posibilita la muestra automática de aquel lugar dentro del mismo que se desee una vez que se le dan las coordenadas.

Para ello es necesario saber la posición actual en la que se encuentra el visor cuando se le da, por ejemplo, a la flecha *siguiente*.

```
int selectedPage = roundf(currPos / ScrollWidth);

[scroll scrollRectToVisible:CGRectMake(ScrollWidth*(selectedPage+1), ScrollY, ScrollWidth,
    ScrollHeight) animated:YES];
```

Como se observa, el mismo método da la opción de que el paso se dé de forma animada, y así no se pierda la continuidad del *paso de página*.

El resultado esperado y actual es el siguiente:

Para pictogramas:

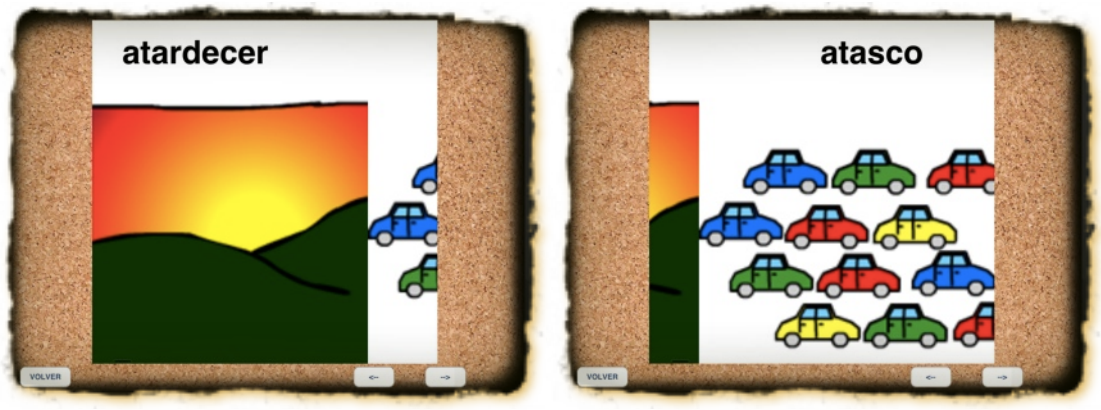

Figura 51. Visualización del visor de pictogramas.

Para *assets* que coinciden con dos dibujos de la pizarra:

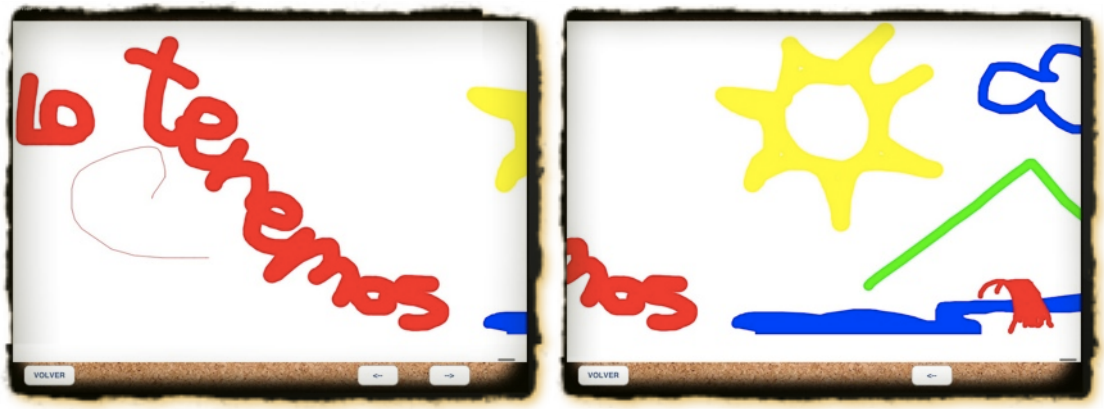

Figura 52. Visualización del visor de assets. (iPhoto y Pizarra).

En ambos casos existe un detalle que facilita la navegación del usuario, y es un pequeño señalizador del lugar actual en el que se encuentra el visor.

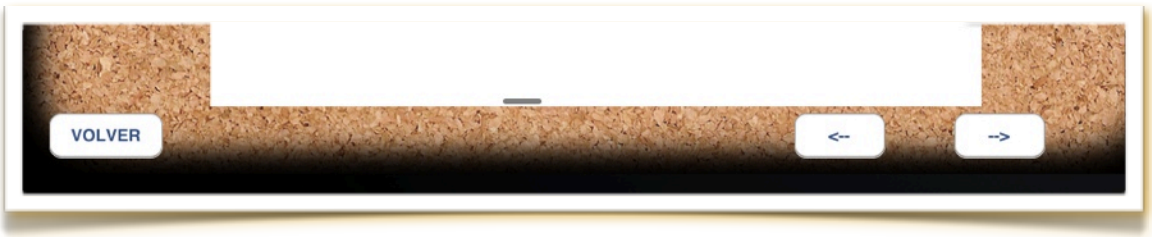

Figura 53. Detalle del elemento señalizador de la posición del scroll.

Por otra parte, se observa que los pictogramas ampliados se ven pixelados, la razón de esto es porque éstos no se crearon con las dimensiones adecuadas. Éste hecho se explica más adelante en el apartado *4.6.2.Pictogramas.*

4.4. OTRAS FUNCIONALIDADES

Como objetivo dentro de este proyecto se encuentra también la definición de una agenda y un diario estructurados que mediante pictogramas informen al usuario y su entorno de los eventos que éste tiene dentro de su horario.

4.4.1. Agenda:

El monitor, papá o mamá que se encuentre con la persona discapacitada debe poder editar esta agenda como si de la suya propia se tratara, para después mostrársela al usuario, el cual con ella puede aprender mejor el concepto de tiempo y la limitación de horarios y eventos.

PLANTEAMIENTO INICIAL:

La agenda debe contener tres vistas: mensual, semanal y diaria y la posibilidad de navegar a través de ellas con facilidad.

Debe tener un menú desplegable en cada una de ellas para poder editar los eventos del día seleccionado y/o agregar más compromisos.

En todas las vistas los fondos deben de representar la estación del año que corresponda y existir el picto que así lo exprese. Además, todos los fines de semana deben de estar marcados en color rojo para que el niño entienda cuales son los días festivos. Éstos incluirán las fiestas que el monitor haya configurado como tales, bien sean vacaciones o días puntuales, para que el usuario deduzca que no hay colegio ese día.

Todo este planteamiento facilita en gran medida la compresión del concepto de año, mes, semana y día, que muchas personas con capacidad desconocen.

Una agenda ayuda a que se distribuya el tiempo y las acciones diarias de forma ordenada. Sea más llevadero iniciar y finalizar actividades, y de cualquier modo permite la comunicación entre compañeros, o monitor y usuario, tan siempre buscada.

Vista Mensual:

Se establece que la versión de la vista mensual ha de ser como un calendario normal navegable entre los meses del año.

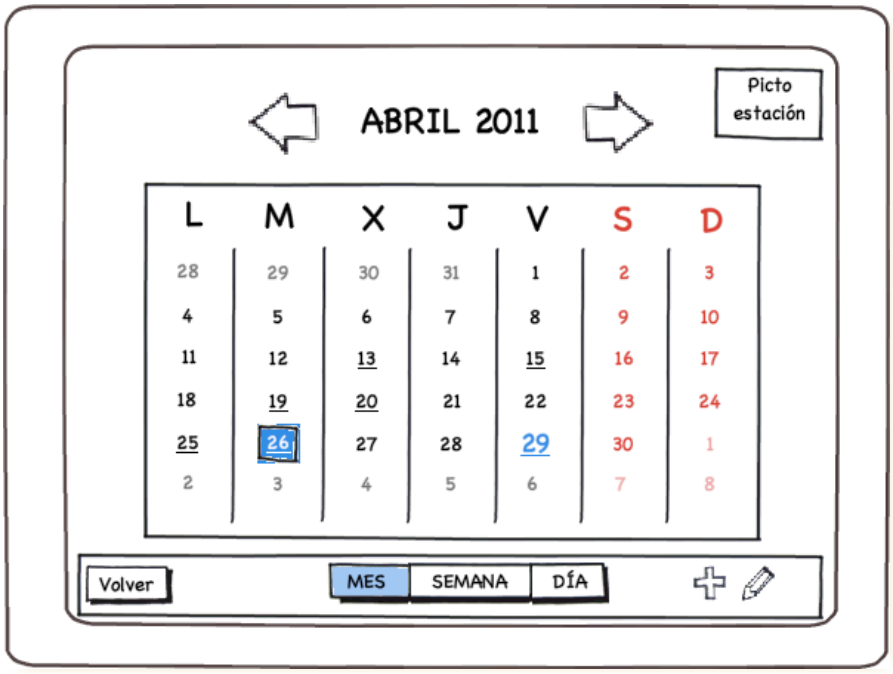

Figura 54. Diseño de la vista mensual.

En ella se ha de marcar de forma especial los días que contengan algún evento así como el día actual. Además, los días deben de ser seleccionables para posibilitar la edición o el agregado de acontecimientos.

En el caso de que cualquier día esté seleccionado y se pase a la vista semanal, ésta debe mostrar la semana correspondiente al día elegido en la pantalla anterior.

Vista Semanal:

La versión de la vista semanal debe de ser en forma horizontal, y con los eventos en vertical debajo de cada día. Cada evento será señalizado con un solo picto, y no habrá más de cuatro pictos/eventos colgando de cada día.

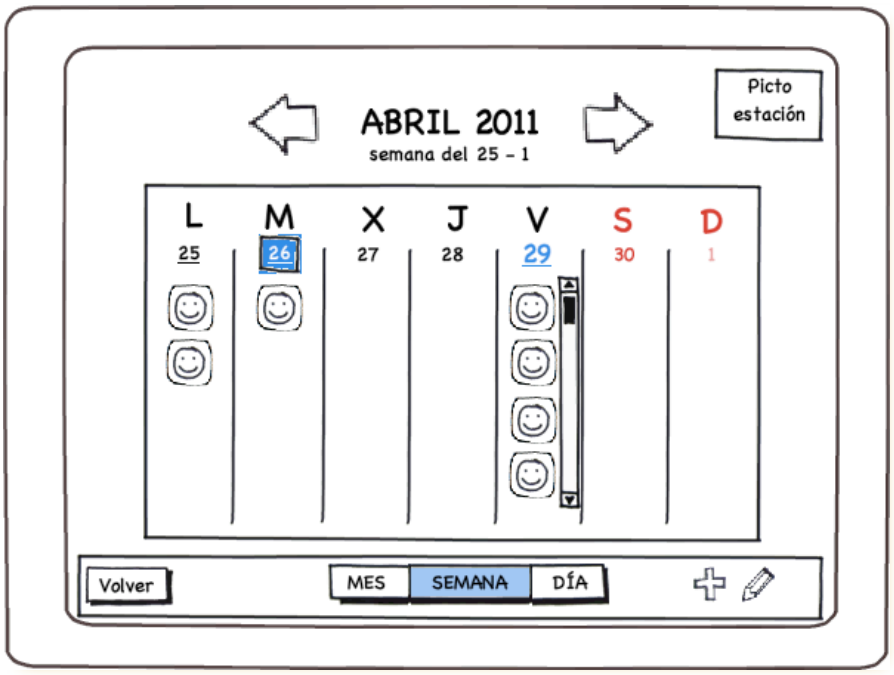

Figura 55. Diseño de la vista semanal.

En un futuro estos eventos deben de ser *scrollizables* y de este modo contener más de cuatro pictos por día.

Se debe de destacar el día seleccionado o el actual según corresponda. Además es necesario que esté activado para poder editar o agregar acciones.

También ha de existir un texto señalizando el mes de esa semana y los días que se estén mostrando. Igualmente ha de ser navegable.

Vista Diaria:

La vista diaria será la más sencilla, con los pictogramas que representen los compromisos diarios y un texto que represente la fecha actual.

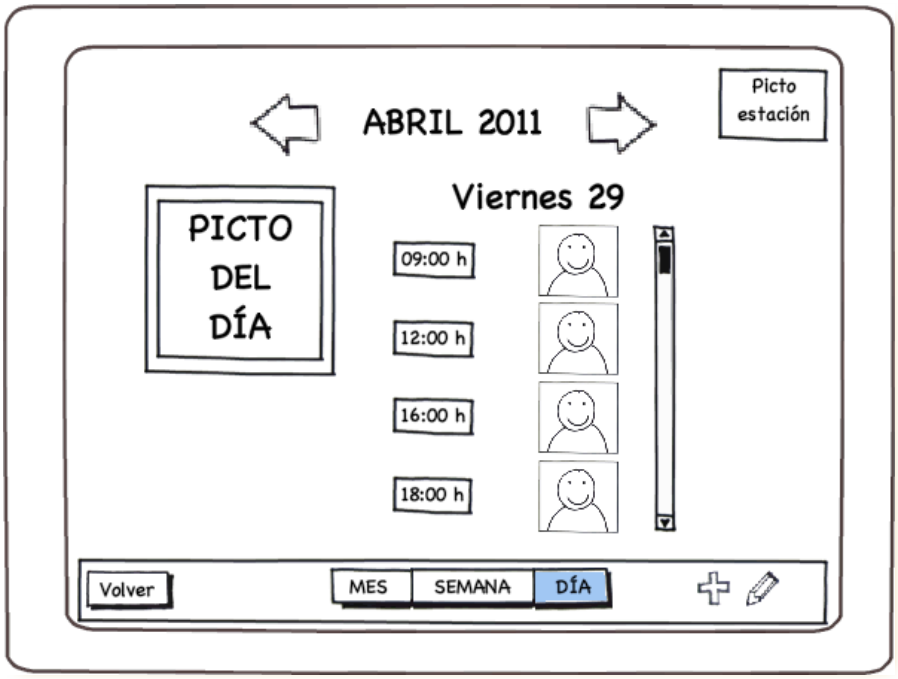

Figura 56. Diseño de la vista diaria.

El picto del día estará recuadrado en rojo si es festivo.

Las imágenes correspondientes a los sucesos serán cuatro, igualmente con posibilidad de tener un *scroll* en el futuro. También habrá etiquetas de texto en las que se represente la hora de cada evento.

Por último mencionar que el menú de edición y agregado sólo será posibilitado para el día en el que se encuentre la vista.

Vista Añadir Evento:

Ha de ser posible su aparición en todas las vistas de la agenda y sólo se mostrará si en cualquiera de las vistas actuales existe un día del mes o semana seleccionado.

Figura 57. Diseño de la vista añadir.

En ella se da la posibilidad de realizar la búsqueda del evento escribiendo el nombre que corresponde al picto asociado.

Debe haber tres elementos de selección: la hora, la duración y si es festivo o no.

Por supuesto, habrá un botón de guardado que registre todos los datos introducidos en algún lugar de forma permanente.

Vista Editar Evento:

Ha de ser posible su aparición en todas las vistas de la agenda y sólo se mostrará si en cualquiera de las vistas actuales existe un día seleccionado en el que haya eventos.

Figura 58. Diseño de la vista de edición.

En ella deben aparecer los pictos de los acontecimientos de ese día ordenados según la hora a la que sean. Junto a ellos, un botón de borrado.

En un futuro contendrá un *scroll* para poder incluir más de cuatro imágenes.

Existirá también un botón de guardado al igual que en la vista de *añadir*.

ESTRUCTURA DE DISEÑO:

La estructura que se plantea es simple. Se observa que aquí también existe una aplicación multivista, y por tanto habrá que crear controladores para cada una de las vistas, así como uno principal que permita la navegación entre ellas.

El menú desplegable de edición y agregado de eventos también debe de ser una clase con vista independiente.

El elemento que posibilita la navegación sencilla entre vistas se llama ***TabBar***. Dentro de su programación hay que señalizar qué clases tiene que controlar.

Hay que tener en cuenta que esta miniaplicación debe estar en contacto en todo momento con una base de datos donde se registren de forma persistente los datos de los eventos.

El diagrama de funcionamiento de la misma vendría a ser el siguiente:

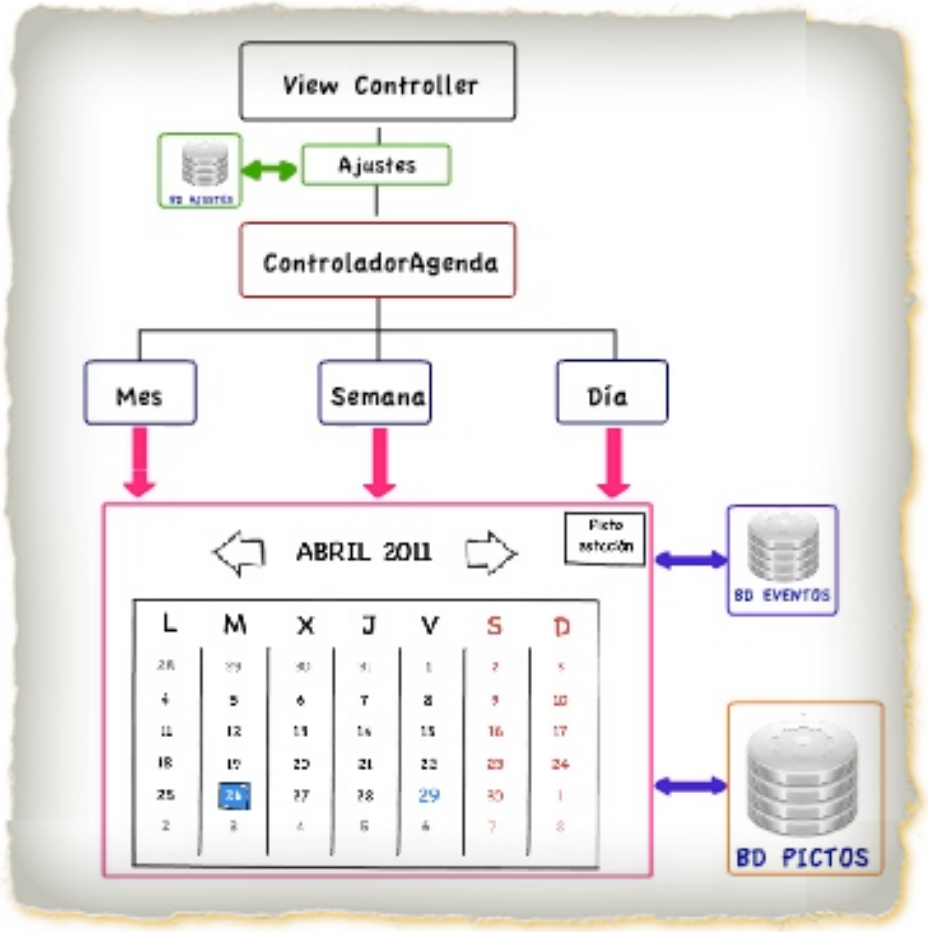

Figura 59. Diagrama de funcionamiento de la Agenda.

Una vez se tiene claro todo el diseño se procede a la implementación del mismo.

INICIO DEL DESARROLLO:

Se comienza el desarrollo creando las clases necesarias que controlen las vistas que se van a mostrar: *controlAgenda, vistaMes, vistaSemana y vistaDia*.

Cada una de ellas es un *UIViewController* y la que lleva el control principal es especial por delegar su funcionamiento según el protocolo *UITabBarDelegate*.

Control Agenda:

Como ya se ha comentado, *controlAgenda* es la encargada de manejar las distintas subclases que se muestran en la navegación.

Contiene el elemento central *Tab Bar* que controla las vistas de las distintas clases de las que ya se ha hablado.

Entonces, el primer paso es incluir a todas ellas en el código.

```
@interface controlAgenda : UIViewController <UITabBarDelegate>{

    IBOutlet UITabBar *controlVistasAgenda;
    UIViewController *vistaMesController;
    UIViewController *vistaSemanaController;
    UIViewController *vistaDiaController;
    UIViewController *vistaActualController;

    IBOutlet UIButton * BotonAgregarEvento;
    IBOutlet UIButton * BotonEditarEvento;

}
```

A continuación, hay que indicar mediante programación cuántos botones tiene el *TabBar* y cuales son las clases a las que hacen referencia.

```
- (void)activateTab:(int)index {
    switch (index) {
        case 1:
            if (vistaActualController != nil)
                [vistaActualController.view removeFromSuperview];
            vistaActualController = vistaMesController;
            if (vistaMesController == nil) {
                self.vistaMesController =
                [[vistaMes alloc] initWithNibName:@"vistaMes" bundle:nil];
            }
            [self.view insertSubview:vistaMesController.view belowSubview:controlVistasAgenda];
            break;
        case 2:
            if (vistaActualController != nil)
                [vistaActualController.view removeFromSuperview];
            vistaActualController = vistaSemanaController;
            if (vistaSemanaController == nil) {
                self.vistaSemanaController =
                [[vistaSemana alloc] initWithNibName:@"vistaSemana" bundle:nil];
            }
            [self.view insertSubview:vistaSemanaController.view belowSubview:controlVistasAgenda];

            break;
        case 3:
            if (vistaActualController != nil)
                [vistaActualController.view removeFromSuperview];
            vistaActualController = vistaDiaController;
            if (vistaDiaController == nil) {
                self.vistaDiaController =
                [[vistaDia alloc] initWithNibName:@"vistaDia" bundle:nil];
            }
            [self.view insertSubview:vistaDiaController.view belowSubview:controlVistasAgenda];
            break;
        default:
            break;
    }
}
```

Posteriormente se incluyen los botones de *editar*, *añadir* y *volver* desde el panel de gráficos de *Interface Builder*.

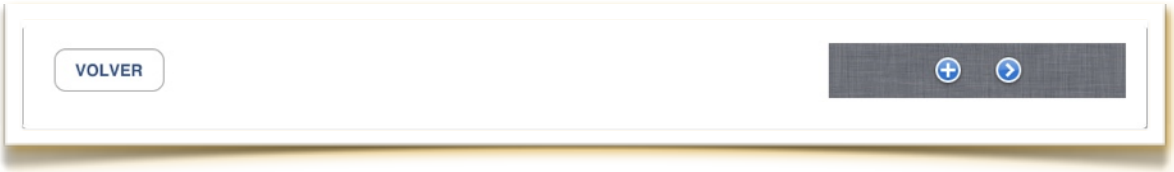

Figura 60. Detalle de la vista común de la Agenda.

El resultado gráfico inicial de esta prueba de programación navegable es el siguiente:

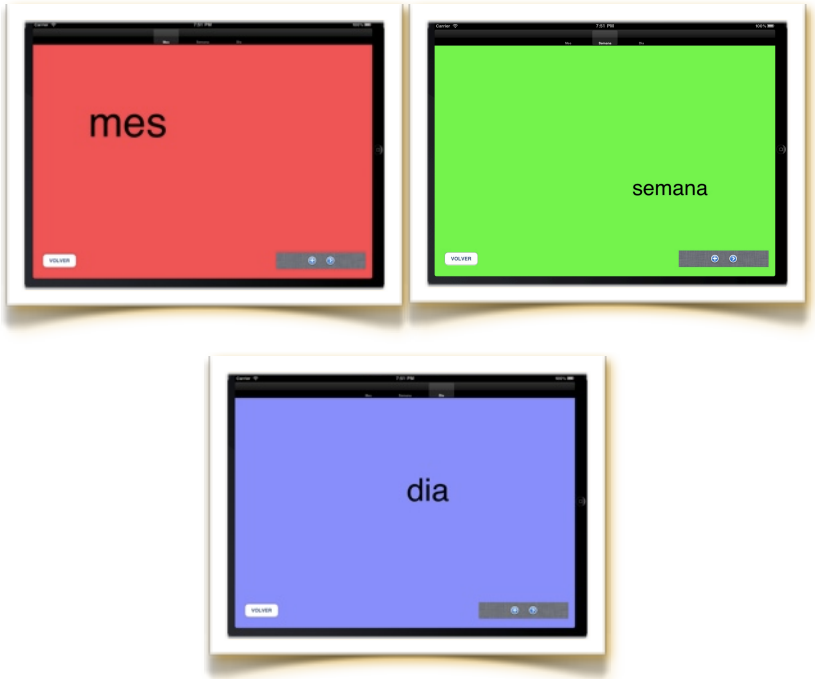

Figura 61. Intercambio de vistas mediante el Tab Bar de la Agenda.

Para observar el detalle superior del *TabBar* se muestra la siguiente imagen:

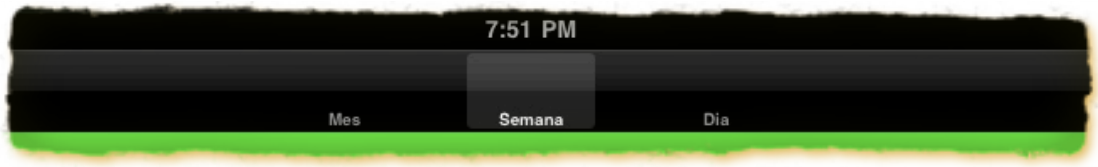

Figura 62. Detalle de los botones del Tab Bar de la Agenda.

Como se observa, el funcionamiento es correcto. El elemento de control navegable es capaz de mostrar cada una de las vistas de las clases independientes, y ahora sólo queda programar cada una de ellas.

Al comenzar por la clase correspondiente a la vista mensual, se comprueba que es realmente complicado programar un calendario completo, sencillo y decente desde cero, por lo que se recurre a la búsqueda de alguno de código abierto que se pueda usar en esta aplicación.

Se encuentra un proyecto llamado ***Kal***. Este archivo *open – source* implementa la vista mensual de un calendario para una aplicación para iPhone.

Permite que el usuario pulse un día y que aparezca la información asociada a ese día en una tabla inferior.

Para poder utilizarlo es necesario "*decirle*" qué días deben estar marcados con un punto, aquellos que tengan algún evento.

Por otro lado hay que proveerle de la base de datos que contiene dichos eventos, y que el calendario mostrará por debajo.

Kal posee un protocolo de delegación propio que satisface estas posibilidades.

Figura 63. Vista ejemplo del calendario Kal.

Este proyecto se puede incluir en cualquier aplicación para iOS, pero es necesaria una meticulosa y cuidadosa integración del mismo.

En el ANEXO (Pag. 133) se adjunta la información necesaria para su unificación.

Por otra parte, hay que tener en cuenta que el código está preparado para iPhone, por lo que hay que re-estructuralo para que pueda ser útil en un iPad.

Después hay que modificar la representación de la información asociada a cada día, pues en este proyecto ya se ha explicado anteriormente cómo se desea que sea.

Además hay que creas una base de datos diferente a la de los pictogramas por razones evidentemente organizativas.

Es hasta aquí, donde queda definida la estructuración de la Agenda. La implementación de la misma y su desarrollo queda pendiente para un próximo proyecto.

Así queda presente la investigación realizada para el desarrollo de la misma y el planteamiento inicial, así como su estructura de diseño y su implementación inicial.

4.4.2. Diario:

El objetivo del diario es el de permitir al usuario mostrar las cosas que haya hecho día a día a quien quiera.

Es importante destacar que aquí es el niño o la niña con discapacidad quien decide qué hechos, de los que ha realizado, quiere enseñar y a quien.

En todo caso esta aplicación pretende desarrollar la capacidad de toma de decisiones de la persona y la de mejorar su autoestima, al verse realizado pudiendo mostrar aquello que ella quiera.

Además, y siguiendo el patrón de metas de metas de esta aplicación, el diario busca mantener en todo momento la comunicación entre la persona discapacitada y los que le rodean.

PLANTEAMIENTO INICIAL:

El desarrollo de esta aplicación iría asociado con la del *Comunicador* principal y el *Quiero y Estoy*.

La intención es que al final de la forma de frases en estas aplicaciones, haya un botón al lado que sea el del *Diario*, que ejecute directamente el guardado de la frase que se haya formado en la base de datos.

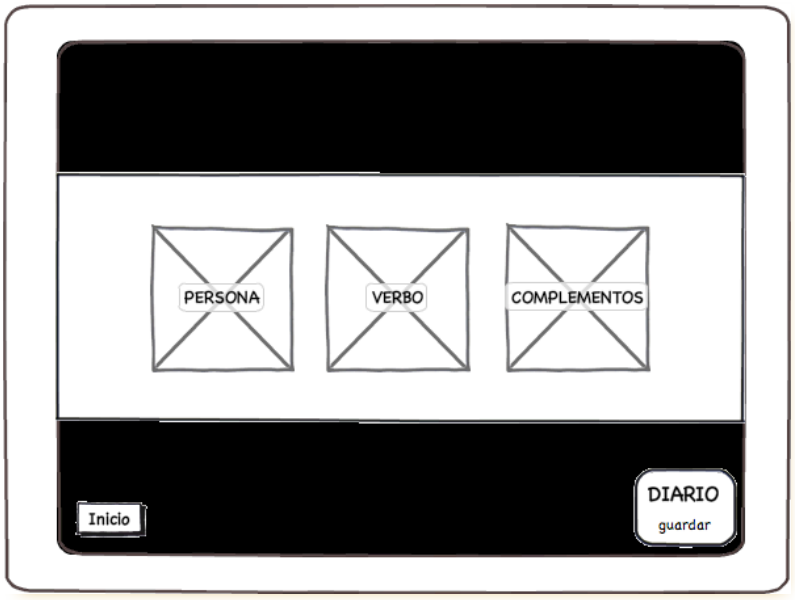

Figura 64. Diseño del guardado de frases en el Diario.

Por otra parte, en la propia aplicación *Diario* se debe observar el día actual con las frases en forma de pictogramas que se hayan guardado.

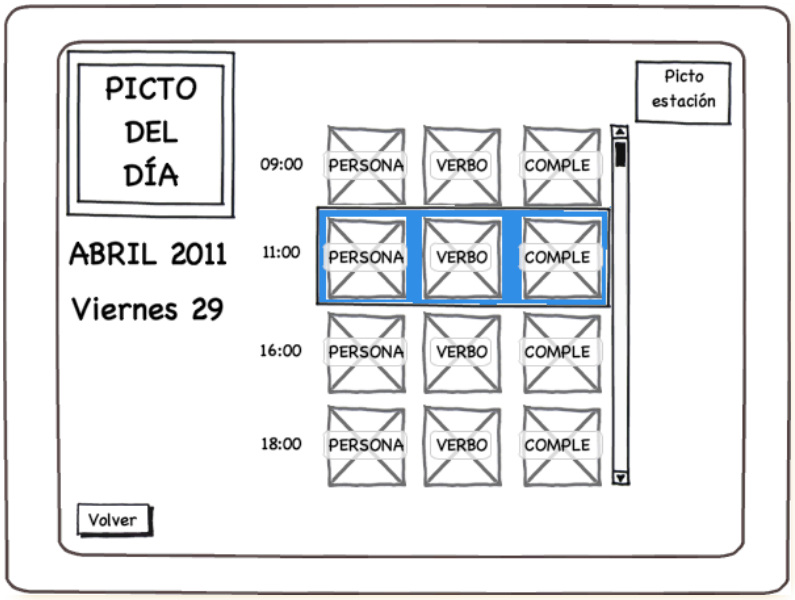

Figura 65. Diseño de la vista de Diario.

Al pulsar cada una de ellas se han de hacer grandes, difuminado el resto del fondo.

Por último y al igual que ocurría en la agenda, el fondo de esta aplicación debe mantener el aspecto de la estación del año que corresponda.

INICIO DEL DESARROLLO:

Como ya se ha comentado en el apartado anterior, el desarrollo de esta parte de la aplicación queda relegado a un trabajo posterior.

4.5. AJUSTES E INTEGRACIÓN

Para completar este proyecto, se propone realizar un apartado de ajustes de la aplicación que sea configurable por la persona que vaya a estar al lado de usuario discapacitado.

Será una clase independiente en sí misma, y su vista se lanzará con la pulsación de un botón oculto en la parte superior izquierda de la pantalla de la aplicación, mientras se está viendo la ventana inicial.

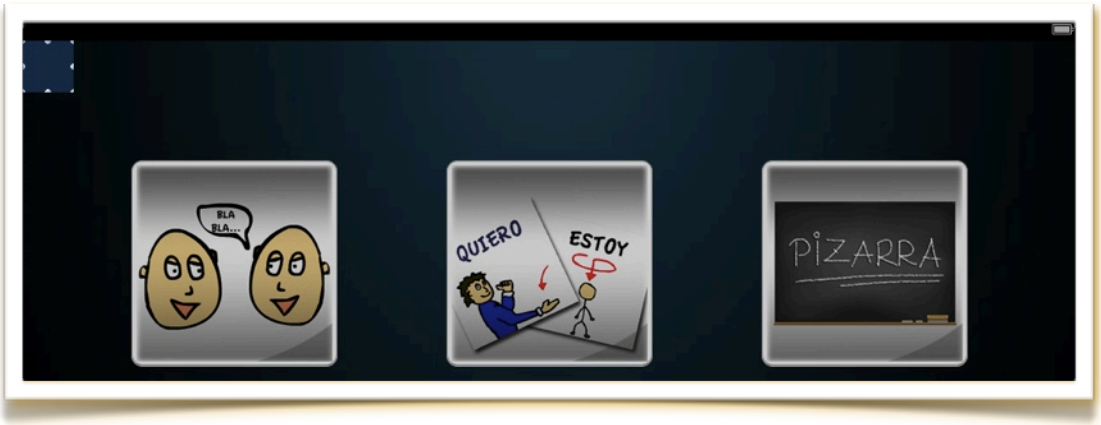

Figura 66. Detalle del acceso a la pantalla Ajustes.

En general, todos los ajustes han sido realizados en conjunto por el grupo de trabajo común.

4.5.1. Ajustes generales:

La idea general es la de poder editar diferentes parámetros dentro de la aplicación general.

Se plantea la modificación de los siguientes aspectos:

- **Respuesta negativa:**

Dentro de la aplicación *Comunicador y Quiero – Estoy* existe la posibilidad de que al finalizar la formación de la frase el monitor, papá o mamá conteste con un ***NO*** rotundo.

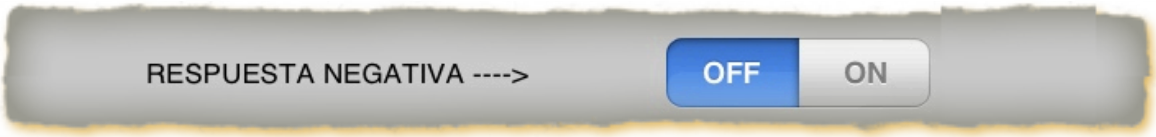

Figura 67. Detalle del ajuste Respuesta Negativa.

Se pretende dar la posibilidad de activar o desactivar esta opción.

- **Tiempo de pulsación:**

Para algunas personas con discapacidad resulta complicado distinguir si el pulsado que han realizado sobre un botón lleva tras de sí una intencionalidad o simplemente ha sido fruto de la suerte al toquetear toda la pantalla del iPad.

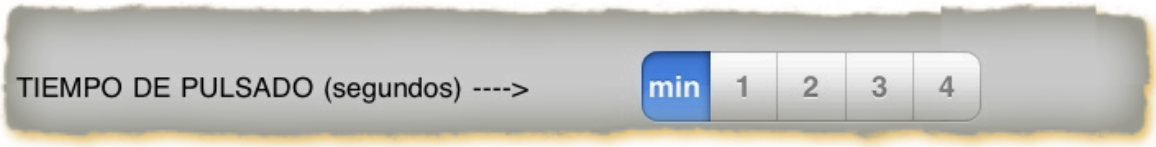

Figura 68. Detalle del ajuste Tiempo de Pulsado.

Por eso se propone la existencia de la duración de un tiempo de pulsación: 1, 2, 3 o 4 segundos necesarios para mantener el dedo en el lugar intencionado y con ello pasar al siguiente evento dentro de la aplicación.

- **Tipo de letra:**

Dentro de los pictogramas es también importante saber que no todos los usuarios saben distinguir entre mayúsculas y minúsculas.

Figura 69. Detalle del ajuste Tipo de Letra.

Aquí se pretende elegir cuál de ellos es el más adecuado para el niño y cambiar uno u otro dependiendo del aprendizaje que éste conlleve.

- **Idioma:**

Una parte fundamental para la comunicación es entender el lenguaje utilizado.

Figura 70. Detalle del ajuste Idioma.

Si se quiere desarrollar éste proyecto en todo su esplendor es necesario establecer mínimamente dos idiomas para que una mayor parte del público pueda utilizarlo.

- **Personalización de pictogramas:**

Una parte que resultó interesante a desarrollar fue la posibilidad de que cada usuario agregase los pictogramas que se desearan.

En este proyecto todos los pictos han sido diseñados por los proyectandos y muchas de las personas que ya utilizaban anteriormente este sistema de modo analógico para la comunicación están habituados a unos dibujos diferentes, pertenecientes a otros estándares.

Figura 71. Detalle del ajuste Personalizar Pictogramas.

Por ello se creyó necesaria la opción de que cada cual pudiera insertar en esta aplicación aquellos pictogramas que quisiera.

Esta personalización consta de una clase independiente por lo que contiene diferentes tareas a realizar en sí misma.

Figura 72. Visualización de la pantalla de Personalización.

Además, también es importante dar la posibilidad de personalizar pictos ya existentes, como por ejemplo *papá* y/o *mamá*, en los que sustituir la imagen del dibujo predefinido por una fotografía real que esté guardada en el dispositivo iPad.

Este ajuste ha sido completamente desarrollado por otro proyectando dentro del equipo conjunto.

- **<u>Álbum de favoritos:</u>**

Con la cantidad de pictogramas que hay en la base de datos es complicado aprender todos de golpe.

En un principio, el álbum anteriormente explicado se diseña con la intención de facilitar el aprendizaje de todas las imágenes y que luego exista una comunicación mejor, más fluida.

Pues bien, se creyó conveniente dar la posibilidad al usuario de establecer qué pictos iban a aprenderse, configurándolos a su gusto en un *álbum de favoritos*.

Figura 73. Detalle del ajuste Editar Álbum Favorito.

Cada categoría que los agrupa puede configurarse al gusto y en todo momento estará disponible según la persona lo haya decidido.

Como en el ajuste anterior, éste se compone por una clase en sí mismo con diferentes tareas a realizar.

Figura 74. Visualización de la pantalla de Edición del Álbum Favorito.

La idea es disponer de dos apartados diferenciados, arriba y abajo, correspondientes a la base de datos completa, y a la categoría personalizada respectivamente.

En cada una de ellas se seleccionan las imágenes que se quieren agregar a la categoría personalizada, o las que se quieren quitar de la misma. Finalmente se deben guardar los cambios.

Éste apartado está realizado completamente por el presente proyectando y por eso se detalla su programación más adelante.

Tras definir con exactitud los ajustes que se desean posibilitar, es importante hacer que todas las clases que definen la aplicación completa sigan los patrones ajustados. Es necesario llevar a cabo la integración final.

4.5.2. Integración final:

Para que todos los ajustes establecidos conformen unas acciones comunes que permanezcan en el tiempo se requiere del uso de una pequeña base de datos, que guarde la información de cada parámetro y desde la que se lea para ajustar cada clase dentro de la aplicación.

Es importante también que en el momento de cierre de la aplicación estos valores se guarden y se carguen de nuevo nada más lanzar el proyecto. Por ejemplo, en el caso de idioma, se ve claramente la necesidad de que esto sea así para no tener que ajustar el mismo cada vez que se desee abrir el proyecto.

Para todo esto se recurre al uso de un archivo del tipo ***.plist***, que almacena toda esta información de manera persistente.

Las aplicaciones para iOS se ejecutan en un directorio de seguridad *sandbox* que es único en cada aplicación dentro del ordenador: */Users/**Usuario**/Library/Application Support/iPhone Simulator/**Versión utilizada**/Application/**Nombre de la aplicación**/*

Es en este lugar donde se alojará el fichero de la base de datos, desde el cual se puede leer y escribir.

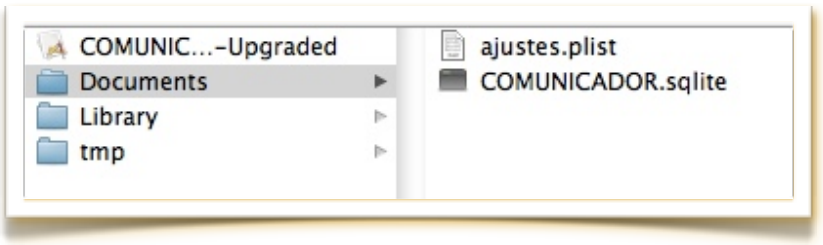

Figura 75. Directorio donde se guardan los ficheros persistentes de la aplicación.

Como es lógico, es aquí donde también se aloja la base de datos completa de los pictogramas en formato *.sqlite*.

La razón por la que se elige este tipo de archivo es porque es el más sencillo de utilizar tanto mediante código como visualmente.

Hay que saber que los puntos clave para gestionar información persistente en una aplicación para iOS son:

- Elegir el método a utilizar entre: Property Listi (*.plist*), Gestión de archivos convencional, BBDD Sqlite 3 o la herramienta por defecto de Apple Core Data.
- Poder trabajar sobre la carpeta en la que es permitido (*sandbox*).
- Gestionar la información de manera completa, esto es, atendiendo a las tres tareas:

    - **Cargar y manejar información:** Leer datos e interpretarlos.
    - **Actualizar la información:** mostrar los cambios realizados visualmente.
    - **Guardar la información:** Al cerrar la aplicación guardar los datos de forma estable.

MANUAL DE CONFIGURACIÓN

En primer lugar se deben crear todos los *IBOutlets*, los elementos que figuran en la vista de ajustes y que son modificables mediante programación. Se observan dos diferencias principales: los elementos que controlan los ajustes en sí mismos *UISegmentedControl* y los que se modifican una vez se hayan establecido los valores deseados: los textos de etiquetas y botones.

```
int selectedUnit_resp_negativa;
IBOutlet UISegmentedControl *segmentedControl_resp_negativa;

int selectedUnit3;
IBOutlet UISegmentedControl *segmentedControl_tiempo_pulsacion;

int selectedUnit_minusc_mayusc;
IBOutlet UISegmentedControl *segmentedControl_minusc_mayusc;

int selectedUnit_cast_ing;
IBOutlet UISegmentedControl *segmentedControl_cast_ing;

IBOutlet UILabel * label_resp_negativa;
IBOutlet UILabel * label_tiempo_pulsacion;
IBOutlet UILabel * label_tamano_letra;
IBOutlet UILabel * label_idioma;
IBOutlet UILabel * label_pictos_personalizar;
IBOutlet UILabel * label_album_favorito_personalizar;

IBOutlet UIButton * but_pictos_personalizar;
IBOutlet UIButton * but_album_favorito_personalizar;
IBOutlet UIButton * but_guardar_ajustes;
IBOutlet UIButton * but_inicio;
```

Cada objeto se enlaza con su correspondiente referencia en *Interface Builder* tal y como se ha hecho en anteriores ocasiones.

Figura 76. Asociación de los botones de ajuste a sus acciones correspondientes.

Todos se controlan mediante métodos concretos y están especificados en el ANEXO (Pág.133) correspondiente de código.

```
- (IBAction)segmentedControlValueChangedRespNegativa;
- (void)cambioRespuestaNegativa;

- (IBAction)segmentedControlValueChanged3;
- (void)cambioTiempoPulsacion;

- (IBAction)segmentedControlValueChanged4;
- (void)cambioIdioma;

- (IBAction)segmentedControlValueChanged6;
- (void)cambioMayusculasMinusculas;
```

Por otra parte, se decide aunar dos variables en un solo parámetro. Las que controlan el idioma y el formato de letra, en una llamada *tipo_letra*, que mediante los valores 20, 30, 50 y 60, determina los cambios necesarios de la siguiente manera:

```
if (selectedUnit_cast_ing == 0 && selectedUnit_minusc_mayusc==0)
{
    tipo_letra=20; // castellano - minuscula
    [self.view setNeedsDisplay];
}
else if (selectedUnit_cast_ing == 0 && selectedUnit_minusc_mayusc==1)
{
    tipo_letra=30; // CASTELLANO - MAYUSCULA
    [self.view setNeedsDisplay];
}
else if (selectedUnit_cast_ing == 1 && selectedUnit_minusc_mayusc==0)
{
    tipo_letra=50; //ingles  - minuscula
    [self.view setNeedsDisplay];
}
else if (selectedUnit_cast_ing == 1 && selectedUnit_minusc_mayusc==1)
{
    tipo_letra=60; // INGLES - MAYUSCULA
    [self.view setNeedsDisplay];
}
```

Así pues, solamente quedan tres pasos: leer desde la base y en el caso de que no exista crearla, guardar datos en ella y actualizar las vistas mostrando las modificaciones detectadas.

Leer o crear

Nada más lanzar la clase *Ajustes* es necesario acceder directamente al *.plist* o en el caso de su inexistencia, crearlo con unos valores por defecto.

Al igual que se ha hecho en otro apartado del proyecto, aquí se utiliza la expresión *# define* para otorgar a unos nombres unos valores determinados, y así poder hacer uso de su expresión en el resto de código.

```
# define KEY_TIPO_LETRA @"tipo_letra"
# define KEY_RESP_NEGATIVA @"resp_negativa"
# define KEY_TIEMPO_PULSACION @"tiempo_pulsacion"

# define ficheroPersistente @"ajustes.plist"
```

Se observa que se realiza esta acción para las variables comunes a ajustar, y también para el nombre del fichero persistente.

Para ver si éste último se ha creado anteriormente, se comprueba en la ruta del directorio *sandbox* si aparece.

```
- (NSString *)rutaFichero
{
    NSArray *rutas = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES);
    NSLog(@"Directorios: %@", rutas);

    NSString *directorio = [rutas objectAtIndex:0];

    return [directorio stringByAppendingPathComponent:ficheroPersistente];
}
```

En el caso de que no sea así, se crea un *array* con los valores por defecto de los parámetros y se continúa el proceso en el método *guardarAjustes*.

```
- (void)viewDidLoad
{
    NSString *fichero= [self rutaFichero];
    if ([[NSFileManager defaultManager] fileExistsAtPath:fichero])
    {
        NSArray *listado =[[NSArray alloc] initWithContentsOfFile:fichero];
        ajustes=[[NSMutableArray alloc]initWithArray:listado];
        [self mostrarAjustes];
    }
    else
    {
        ajustes=[[NSMutableArray alloc]init];

        resp_negativa=1;
        tiempo_pulsacion=0.5;
        tipo_letra=20;

        [self guardarAjustes];
    }

    [super viewDidLoad];
    // Do any additional setup after loading the view from its nib.
}
```

Por último se comprueba si en el directorio externo se encuentra el archivo creado y se abre para ver su contenido.

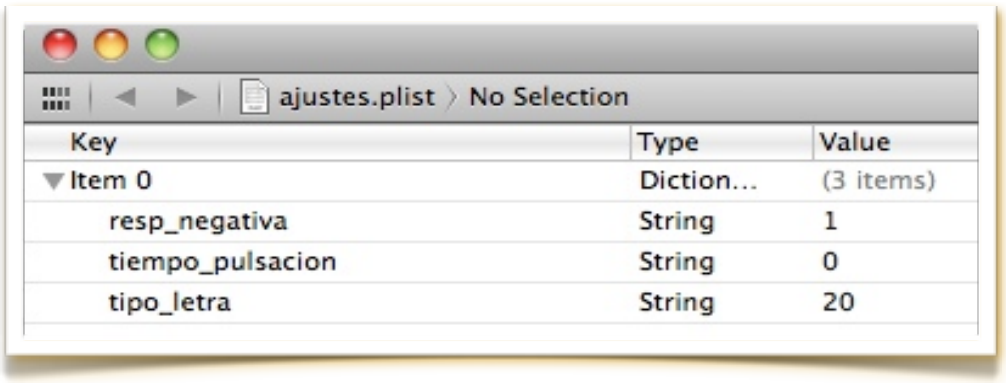

Figura 77. El archivo Ajustes.plist contiene las variables de ajuste.

Escribir y guardar

Lo que se ha realizado en *guardarAjustes* ha sido una conversión de las variables *integer* a tipo *string* porque *Property List* no permite almacenar valores enteros.

A continuación, lo que ha hecho este método ha sido crear un elemento tipo *NSDictionary* y guardar las *keys* que se han creado en él.

```
resp_negativa_string=[NSString stringWithFormat:@"%d",resp_negativa];
tiempo_pulsacion_string=[NSString stringWithFormat:@"%d",tiempo_pulsacion];
tipo_letra_string=[NSString stringWithFormat:@"%d",tipo_letra];

NSDictionary *item =[NSDictionary dictionaryWithObjectsAndKeys:tipo_letra_string, KEY_TIPO_LETRA,
    tiempo_pulsacion_string, KEY_TIEMPO_PULSACION, resp_negativa_string, KEY_RESP_NEGATIVA, nil];
[self.ajustes removeAllObjects];
[self.ajustes addObject:item];

[ajustes writeToFile:[self rutaFichero] atomically:YES];

[self mostrarAjustes];
```

Posteriormente se escriben los cambios directamente sobre el fichero Ajustes.plist y se procede a la actualización de la vista con la información modificada.

Actualizar cambios

En primero lugar se vuelven a transformar los valores cogidos como *string* en enteros para poder utilizarlos en los elementos correspondientes.

```
NSDictionary *item =[ajustes objectAtIndex:0];

//se convierten los valores a números
resp_negativa= [[item objectForKey:KEY_RESP_NEGATIVA] intValue];
tiempo_pulsacion = [[item objectForKey:KEY_TIEMPO_PULSACION] intValue];
tipo_letra = [[item objectForKey:KEY_TIPO_LETRA] intValue];

int idioma;
int tamano_letra;

//se colocan los segmentos seleccionados correspondientes.
[segmentedControl_resp_negativa setSelectedSegmentIndex:resp_negativa];
[segmentedControl_tiempo_pulsacion setSelectedSegmentIndex:tiempo_pulsacion];
[segmentedControl_cast_ing setSelectedSegmentIndex:idioma];
[segmentedControl_minusc_mayusc setSelectedSegmentIndex:tamano_letra];
```

Después se cambian a mano los textos de etiquetas y botones, dependiendo del idioma que se haya establecido.

```
if (tipo_letra<40)
{
    idioma=0;
    label_idioma.text=@"ACTIVAR 'DECIR NO'";
    label_tiempo_pulsacion.text=@"TIEMPO DE PULSACIÓN";
    label_tamano_letra.text=@"FORMATO DE LETRA";
    label_idioma.text=@"IDIOMA";
    label_pictos_personalizar.text=@"INTRODUCE TUS PICTOGRAMAS O MODIFICA LOS EXISTENTES";
    label_album_favorito_personalizar.text=@"ELIGE QUÉ IMÁGENES APARECEN EN TU ÁLBUM DE FAVORITOS";

    [segmentedControl_minusc_mayusc setTitle:@"minúsculas" forSegmentAtIndex:0];
    [segmentedControl_minusc_mayusc setTitle:@"MAYÚSCULAS" forSegmentAtIndex:1];
    [segmentedControl_cast_ing setTitle:@"Castellano" forSegmentAtIndex:0];
    [segmentedControl_cast_ing setTitle:@"Inglés" forSegmentAtIndex:1];

    [but_pictos_personalizar setTitle:@"Personalizar pictos" forState:normal];
    [but_album_favorito_personalizar setTitle:@"Personalizar álbum" forState:normal];
    [but_guardar_ajustes setTitle:@"GUARDAR" forState:normal];
    [but_inicio setTitle:@"INICIO" forState:normal];
}
else
{
    idioma=1;
    label_idioma.text=@"SET ACTIVE 'SAY NO'";
    label_tiempo_pulsacion.text=@"TOUCHING TIME";
    label_tamano_letra.text=@"FONT TYPE";
    label_idioma.text=@"LANGUAGE";
    label_pictos_personalizar.text=@"ADD OR EDIT YOUR OWN PICTOS";
    label_album_favorito_personalizar.text=@"CHOOSE YOUR FAVOURITE ALBUM PICTURES";

    [segmentedControl_minusc_mayusc setTitle:@"small letter" forSegmentAtIndex:0];
    [segmentedControl_minusc_mayusc setTitle:@"CAPITAL LETTER" forSegmentAtIndex:1];
    [segmentedControl_cast_ing setTitle:@"Spanish" forSegmentAtIndex:0];
    [segmentedControl_cast_ing setTitle:@"English" forSegmentAtIndex:1];

    [but_pictos_personalizar setTitle:@"Pictos personalize" forState:normal];
    [but_album_favorito_personalizar setTitle:@"Album personalize" forState:normal];
    [but_guardar_ajustes setTitle:@"SAVE" forState:normal];
    [but_inicio setTitle:@"HOME" forState:normal];
}
```

Hay que destacar que estos son los ajustes que modifican directamente todas las clases dentro de la aplicación general, por lo que dentro de cada una de ellas hay que hacer una llamada al controlador de ajustes para realizar los cambios necesarios antes de entrar en ninguna mini – aplicación.

```
- (void)Ajustar
{
    [ControladorAjustes viewDidLoad];
    ajuste_resp_negativa=ControladorAjustes.resp_negativa;
    ajuste_tiempo_pulsacion=ControladorAjustes.tiempo_pulsacion;
    ajuste_tipo_letra=ControladorAjustes.tipo_letra;

    if (ajuste_tipo_letra==20){
        Idioma=@"nombre";
    }
    else if (ajuste_tipo_letra==30){
        Idioma=@"nombre_may";
    }
    else if (ajuste_tipo_letra==50){
        Idioma=@"nombre_ing";
    }
    else if (ajuste_tipo_letra==60){
        Idioma=@"nombre_may_ing";
    }
}
```

Posteriormente, y con la referencia del idioma, es necesario cargar a mano todos los textos integrados en botones, etiquetas, pictogramas o alertas.

La integración final es un éxito y el resultado está a la vista en todas las partes del proyecto conjunto.

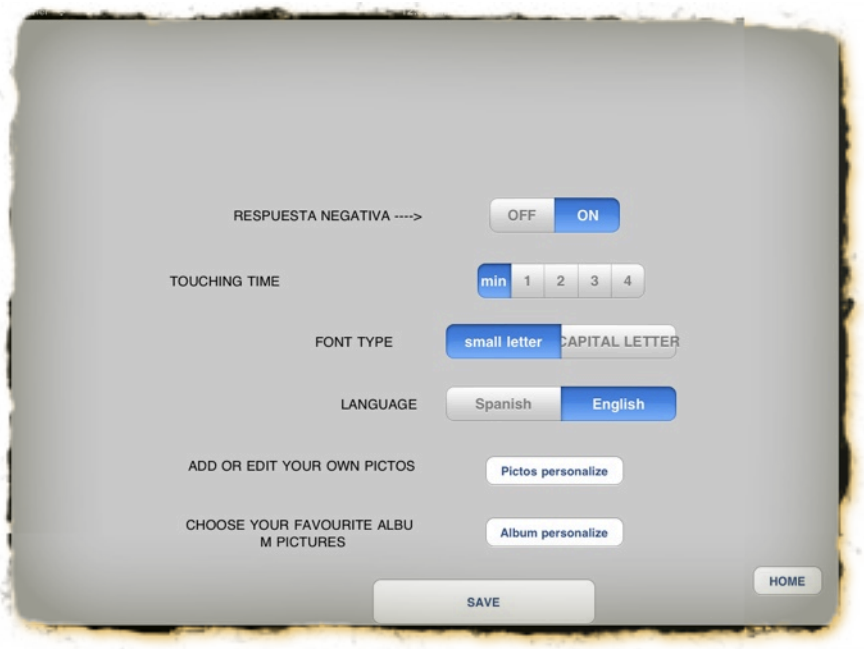

Figura 78. Pantalla de ajustes con el Idioma puesto en Inglés.

Figura 79. Pantallas de todas las mini-aplicaciones con el ajuste Idioma establecido en Inglés.

Por último, en la clase *Ajustes* hay adjuntas dos clases independientes más: *Personalización* y *EditarAlbumFavorito*.

Como se ha explicado anteriormente, a cada botón correspondiente se le asocia la acción de la llamada a cada una de ellas.

A continuación se explica el desarrollo de la que se encarga de la edición de un álbum personalizado para su uso en el aprendizaje de pictos.

4.5.3. Álbum de favoritos:

Este apartado pretende facilitar la selección de aquellos pictogramas que el adulto considere importantes y guardarlos en una álbum especial.

Es aquí, donde se cambia el aspecto de la aplicación Álbum en su pantalla inicial. La que contaba con tres botones: *Photos, Pizarra y Pictos*, como se ha comentado anteriormente varía, la *Pizarra* se ha unificado a *Photos*, y por su parte, se ha agregado otra visualización más: *Álbum Favorito.*

Figura 80. Selección final de los álbumes en la aplicación Álbum.

Desde ahí se visualizarán los pictos configurados y desde ésta nueva clase *Editar Álbum Favorito* es desde donde se editan.

El esquema inicial es sencillo conteniendo tres visores diferenciados.

Figura 81. Visualización de la composición de la vista Editar Álbum Favorito.

En la parte izquierda se observa un campo destinado a la selección de la categoría a personalizar.

Arriba tienen lugar todos los pictos existentes en la base de datos y debajo aquellos que se hayan seleccionado como favoritos dentro de la misma categoría.

En ambos visores de pictogramas existen flechas de navegación que permiten recorrer con facilidad todas las imágenes que estén en cada una de las partes.

Figura 82. Detalle de la parte de miniaturas de la Entidad Original.

En la parte central de la pantalla aparecen tres botones interactivos e intuitivos, que ayudan a definir qué imágenes se desea que se encuentren en el álbum personalizado y cuales no.

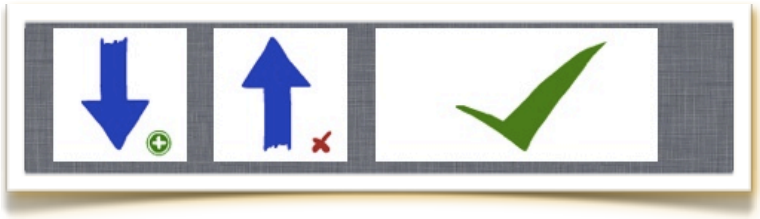

Figura 83. Botones interactivos para editar cada Categoría de Pictogramas.

El botón del *tic* simboliza la validación de los cambios y el guardado de la nueva información.

Siempre que se hable de guardar hay que tener en cuenta que el almacenamiento se da sobre una base de datos.

En este caso se decide modificar la ya existente de pictogramas para poder identificar qué elementos quedan seleccionados como favoritos, y cuales no.

En primer lugar entonces se añade una columna extra a la base de datos original, en la cual establecer unos o ceros, dependiendo de si el pictograma esté elegido como preferido o no.

ZNOMBRE	ZNOMBRE_MAY	ZNOMBRE_ING	ZNOMBRE_MAY_ING	ZCONT...	ZPK	ZNUEVO	ZACTIVO	ZALBUM_PERSONALIZADO
abuelos	ABUELOS	grandparents	GRANDPARENTS	5	1	0	1	0
adulto	ADULTO	adult	ADULT	2	2	0	1	0
amigo	AMIGO	friend	FRIEND	0	3	0	1	0
anciano	ANCIANO	elderly	ELDERLY	0	4	0	1	0
animale...	ANIMALES AC...	aquatic anima...	AQUATIC ANIMALS	0	5	0	1	0
animale...	ANIMALES DE ...	farm animals	FARM ANIMALS	0	6	0	1	0
animale...	ANIMALES SA...	wild animals	WILD ANIMALS	0	7	0	1	0
araña	ARANÑA	spider	SPIDER	0	8	0	1	0

Figura 84. Añadido de la columna especial para el Álbum Favorito.

A continuación se procede a la implementación del progreso de selección y almacenamiento de pictogramas.

MANUAL DE CONFIGURACIÓN

Hay numerosos botones interactivos en esta aplicación, por ello cada uno tiene funciones diferentes.

Aún así, las acciones que se llevan a cabo se realizan sobre tres variables principales en forma de *arrays* bien definidas:

- ***selecc_originales***: almacena los nombres de los pictogramas que estén seleccionados en la entidad completa.
- ***Selecc_favoritos***: almacena los nombres de los pictogramas que estén seleccionados en la entidad personalizada.
- ***Datos_favoritos***: contiene los nombres de los pictogramas que están actualmente en la entidad personalizada.

Mediante estas tres variables se llevan a cabo todas las tareas de intercambio de información en la aplicación de tal forma que la escritura final en la base de datos se dé únicamente mediante el botón de validación.

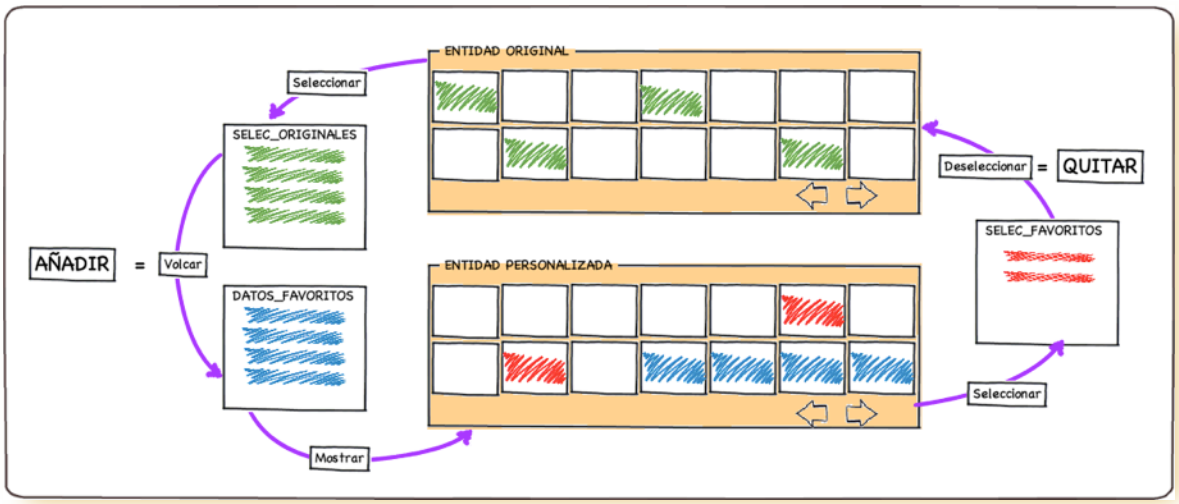

Figura 85. Diagrama de funcionamiento de la Edición del Álbum Favorito

A continuación, se citan cada una de las funciones asociadas a cada botón.

ON Entidad pulsado

El trabajo de los botones de entidad es sacar por pantalla los pictogramas que estén contenidos en esa categoría.

Arriba los que conforman la base de datos original y abajo los que el usuario ha definido en su álbum de favoritos.

Cada uno está asociado con ésta función y envía información por medio de su *tag*.

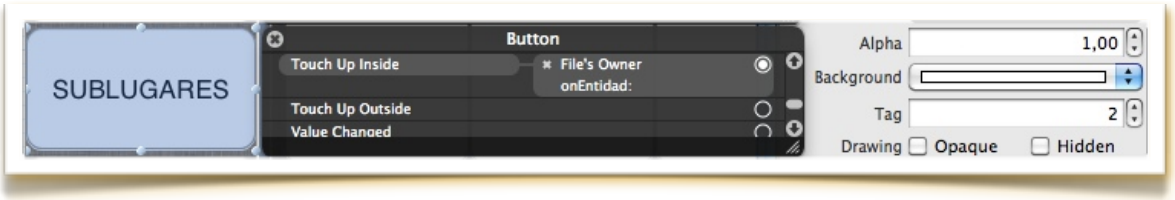

Figura 86. Detalle de la asociación de cada botón de Entidad a su acción y a su tag.

El código implementado es prácticamente igual al del álbum de fotos. Cuando se clicka en la categoría se sacan por pantalla las imágenes encontradas con esas condiciones en la base de datos.

En el caso de la entidad personalizada, se buscan unos (1) en la base de datos dentro de la columna anteriormente creada.

```
entityDesc_per=[NSEntityDescription entityForName:@"Personas" inManagedObjectContext:context_per];
request_favorito=[[NSFetchRequest alloc]init];
[request_favorito setEntity:entityDesc_per];
pred_favorito = [NSPredicate predicateWithFormat:@"ANY album_personalizado like '1'"];
[request_favorito setPredicate:pred_favorito];
objects_favoritos =[context_per executeFetchRequest:request_favorito error:&error_per];
```

Si no existe todavía ninguna personalización para la categoría se lanza una alerta dando el aviso:

Figura 87. Ejemplo de la alerta lanzada cuando no se ha personalizado la Entidad seleccionada.

Si por el contrario el usuario ya había configurado una entidad, los objetos cogidos son volcados directamente en una de las variables principales *datos_favoritos* y se procede a la representación de cada una de las imágenes en el visor inferior.

```
-(void)pasoPictosfavoritos
{
    objects_favoritos=nil;

    for (int i=0; i<[objectsCogidos_favoritos count]; i++)
    {
        [datos_favoritos addObject:[[objectsCogidos_favoritos objectAtIndex:i] valueForKey:@"nombre"]];
    }

    [self reloadfavoritoPics];
}
```

Al pulsar en cualquier otra entidad se borran automáticamente los datos contenidos en las variables principales.

ON Seleccionar Picto Original/Favorito

Todas las miniaturas están asociadas a esta acción. El envío de información se realiza mediante el *tag* correspondiente explicado en apartados anteriores de esta memoria.

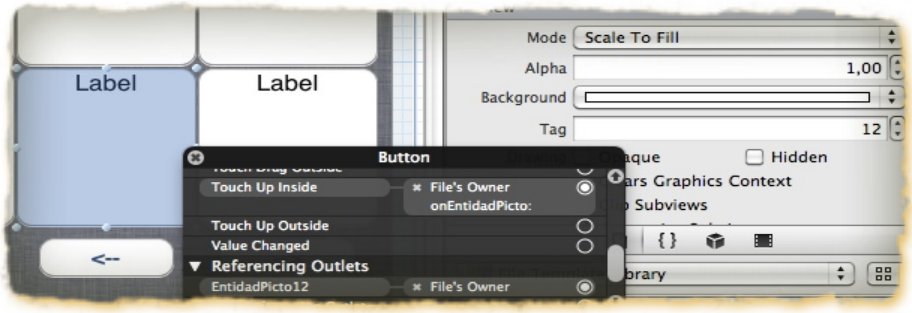

Figura 88. Asignación de cada miniatura a su acción común y su nº de identificación.

Las funciones de selección de los pictogramas tanto originales como favoritos para añadir o quitar imágenes posteriormente siguen el mismo patrón según el diagrama inicial esquematizado.

Lo que se hace principalmente es incluir el nombre del picto seleccionado dentro de la variable *selecc_originales* ó *selecc_favoritos* en el caso de que no esté.

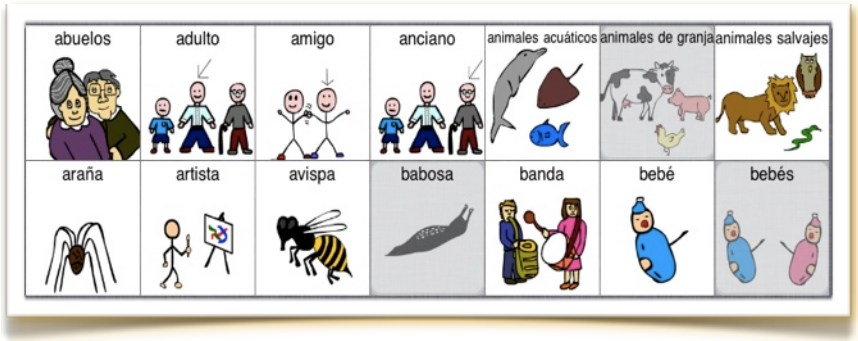

Figura 89. Ejemplo de selección de la Entidad Original.

Figura 90. Detalle de la consola donde se ve el contenido de la variable selec_originales.

Si ya existe dentro del *array* se elimina directamente.

```
-(IBAction)onEntidadPicto:(id)sender
{
    NSInteger ButtonTagInteger= [sender tag];
    NSLog(@"Imagen seleccionada = %i",ButtonTagInteger);

    // quitar los pictos que ya estén.
    [selecc_originales removeObjectIdenticalTo:[[objectsCogidos_entidad objectAtIndex:(indice+ButtonTagInteger)]
        valueForKey:@"nombre"]];

    if ([[Entidad_table objectAtIndex:ButtonTagInteger] alpha]==1) //picto no está seleccionado
    {
        NSString *nombre=[[objectsCogidos_entidad objectAtIndex:(indice+ButtonTagInteger)] valueForKey:@"nombre"]
            ;
        [selecc_originales addObject:nombre];
        [[Entidad_table objectAtIndex:ButtonTagInteger] setAlpha:0.5];
    }
    else //picto ya está seleccionado
    {
        [[Entidad_table objectAtIndex:ButtonTagInteger] setAlpha:1];
    }
}
```

A continuación se modifica la apariencia del botón para que quede pulsado o deseleccionado mediante su propiedad *alpha*.

Navegar entre pictos

El funcionamiento de esta parte es el mismo que el de la navegación dentro del álbum de fotos convencional.

La carga de imágenes con el paso de página es la misma, con una excepción: se comprueba si cada pictograma se encuentra dentro de las variables *selección_datos* para mantener la visualización de los botones que han sido pulsados anteriormente.

```
NSString *texto=[[objectsCogidos_entidad objectAtIndex:(i+indice)] valueForKey:@"nombre"];

[[Entidad_table objectAtIndex:i] setHidden:NO]; //Hacer visibles todos los botones que tengan imagen.
[[Entidad_table objectAtIndex:i]setAlpha:1];    //Poner la opacidad completa a todos

if ([selecc_originales containsObject:texto])
{
    [[Entidad_table objectAtIndex:i]setAlpha:0.5];
}
//Poner la opacidad a la mitad de aquellos botones que ya estén registrados en la selección original.
```

De esta forma, si en una página se han seleccionado varias imágenes se pasa a la siguiente y se pulsan otras tantas… A la vuelta atrás permanecen activos aquellos pictogramas que se habían seleccionado. Y en la página siguiente más de lo mismo.

Figura 91. Proceso de selección en varias páginas del Álbum Favorito.

Añadir Pictogramas

El botón que le corresponde pretende ser intuitivo con la punta de la flecha, al dar a entender que los pictos seleccionados de la parte de arriba pasarán al visor de abajo cuando se pulse.

Figura 92. Asociación del botón de agregar con su acción correspondiente.

Se comprueba si los pictogramas alojados en *selecc_originales* se encuentran ya en la variable *datos_favoritos*. Si es así no se hace nada.

```
-(IBAction)pulsarButAgregar:(id)sender
{
    for ( int i=0; i<[selecc_originales count]; i++)
    {
        if ([datos_favoritos containsObject:[selecc_originales objectAtIndex:i]])
        {

        }
        else
        {
            [datos_favoritos addObject:[selecc_originales objectAtIndex:i]];
            NSLog(@"datos_favoritos %@", datos_favoritos);
        }
    }
    [selecc_favoritos removeAllObjects];
    [self reloadfavoritoPics];

}
```

Si por el contrario no están, se vuelcan los nombres de la primera a la segunda variable y a continuación se actualiza su representación.

Quitar Pictogramas

Al igual que en el método anterior, este botón también indica *la subida* de los pictogramas seleccionados de nuevo a su lugar original, quitándolos de la categoría personalizada.

Figura 93. Asociación del botón quitar con su acción correspondiente.

Su funcionamiento es simple, va eliminando de la variable *datos_favoritos* aquella información que aparezca en *selecc_favoritos*.

```
-(IBAction)pulsarButQuitar:(id)sender
{
    for ( int i=0; i<[selecc_favoritos count]; i++)
    {
        [datos_favoritos removeObjectIdenticalTo:[selecc_favoritos objectAtIndex:i]];
    }
    [selecc_favoritos removeAllObjects];
    [self reloadfavoritoPics];
}
```

Validar y escribir en la BBDD.

Por último y tras todo este intercambio de información es necesario volcar los datos definitivos en la base de datos. Para ello se debe pulsar el botón *validar*.

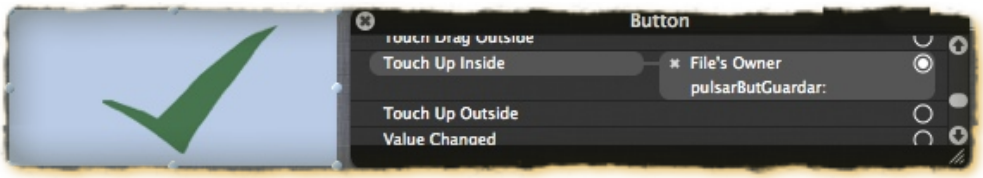

Figura 94. Asociación del botón validar con su acción correspondiente.

Este método recorre todos y cada uno de los objetos que existen en la base de datos de la entidad con la que se está trabajando.

```
for ( int i=0; i<[todos_objects count]; i++)
{
    if ([datos_favoritos containsObject:[[todos_objects objectAtIndex:i] valueForKey:@"nombre"]])
    {
        NSManagedObject *picto_favorito=[todos_objects objectAtIndex:i];
        [picto_favorito setValue:[NSNumber numberWithInt: 1] forKey:@"album_personalizado"];
    }
    else
    {
        NSManagedObject *picto_favorito=[todos_objects objectAtIndex:i];
        [picto_favorito setValue:[NSNumber numberWithInt: 0] forKey:@"album_personalizado"];
        NSLog(@"datos_favoritos %@", datos_favoritos);
    }
}
```

Por cada uno de ellos comprueba si dicha imagen se encuentra dentro de la variable *datos_favoritos* y si existe marca un uno (1) en la columna de *album_personalizado*, por el contrario, si no hay coincidencias, escribe un cero (0) en el mismo lugar.

Se observa que es en este apartado donde se realiza exclusivamente la escritura en la base de datos. Es una cuestión importante a la hora de organizar bien la programación ya que de no ser así, el código estaría llamando continuamente a la base de datos y a parte de ser muy engorroso podría consumir gran cantidad de recursos e impedir que la aplicación funcionara con fluidez.

4.6. DISEÑO GRÁFICO

La última parte de la que se encarga este proyecto es la realización de la línea editorial de toda la aplicación.

4.6.1. Objetivo:

El proyecto común está destinado a las personas con discapacidad en la comunicación, generalmente niños.

El aspecto necesita ser simple y sencillo, puesto que los usuarios que habitualmente lo usen pueden tener problemas de visión, o bien tendencia a despistarse con los detalles.

Por otra parte debe ser de uso intuitivo, de tal manera que sea fácil de utilizar y no haya dudas de su funcionamiento a pesar de que se pretenda hacer un pequeño manual de usuario. Por ello se establecen áreas de navegación y usabilidad.

De igual modo el diseño ha de ser atractivo y contar con recursos que llamen la atención del usuario para que éste lo utilice de buen gusto.

Por último, la intención es que el aspecto sea familiar y de alguna forma recuerde al método utilizado hasta ahora de cartulinas y velcros. Se trata de traspasar la frontera de lo digital, para hacer algo más natural.

En cualquier caso el objetivo es que el diseño sea algo unificado en todas las pantallas y siga un hilo común en torno a unas líneas y colores actuales.

4.6.2. Pictogramas:

Aunque no formen parte específica del entorno gráfico del programa son el pilar fundamental del mismo y es importante recalcar el trabajo que hay detrás de ellos.

Como las personas habituadas a este sistema saben, los pictogramas son unos pequeños dibujos dispuestos en cartulinas que representan lugares, acciones, personas o cosas, y que se la persona discapacitada distribuye de tal forma que su disposición ordenada forme frases y le sirva para comunicarse.

Pues bien, los dibujos que representan estos elementos, los que forman el lenguaje de comunicación no forman un sistema en concreto, es decir, cada cual utiliza las imágenes que más le convengan o se crea las suyas propias.

Aunque en general existen dos estándares que son algo utilizados por los educadores y las familias: ARASAAC y BOARDMAKER.

Son de procedencia española y americana respectivamente.

El primero está muy poco extendido en los centros especializados y a pesar de que tiene una gran biblioteca de imágenes, ninguna contiene la palabra asociada dentro de ella, y habría que tratar cada pictograma uno por uno para obtener lo deseado en este proyecto.

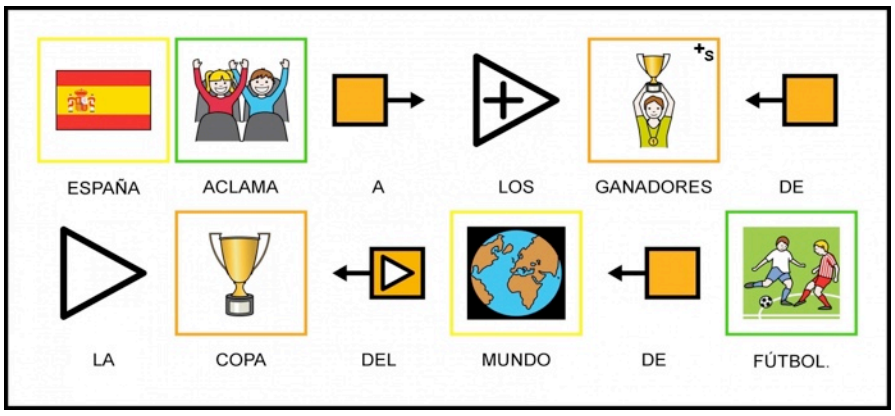

Figura 95. Ejemplo de uso de los pictogramas de ARASAAC.

El segundo de ellos es el más extendido mundialmente. Es comercializado por la empresa Mayer Johnson y contiene los nombres de cada pictograma dentro de cada imagen, además de contar con una gran librería de gráficos.

Figura 96. Ejemplo de uso de los pictogramas de BOARDMAKER.

El gran problema está cuando se piensa en distribuir este proyecto para que pueda ser utilizado por gente discapacitada. El coste de la adquisición de una de las librerías de Board Maker supondría un gran desembolso de dinero.

Así que la decisión fue tomada y se estableció que los tres proyectandos realizarían la labor de dibujar uno por uno los pictogramas más usados, completando una librería de 1500 imágenes.

Se llegó a la conclusión de que la forma más fácil y rápida de realizarlos era mediante el programa ***Adobe Flash CS3/CS4/CS5***.

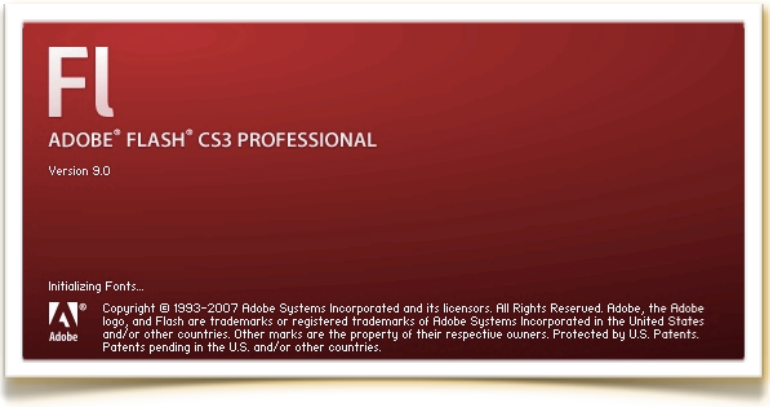

Figura 97. Logotipo del programa Adobe Flash CS3 Professional

Con este programa se dibujan de forma vectorial todos los pictogramas.

Su herramienta pincel permite realizar trazos suavizados sin necesidad de ser un experto dibujante ni de tener buen pulso con el ratón del ordenador.

El procedimiento es sencillo. Se establecen las dimensiones del lienzo en 250x250 pixeles.

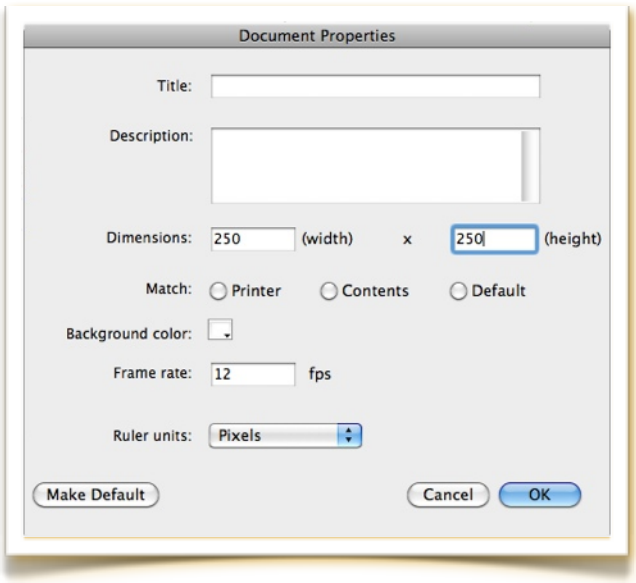

Figura 98. Ajuste de las dimensiones del lienzo.

A continuación se dibuja la imagen que se desee siempre dejando un espacio libre en la parte de arriba de unos 50 pixeles. Es el espacio que posteriormente ocupa el texto de cada picto con el idioma correspondiente y su formato en mayúsculas o minúsculas.

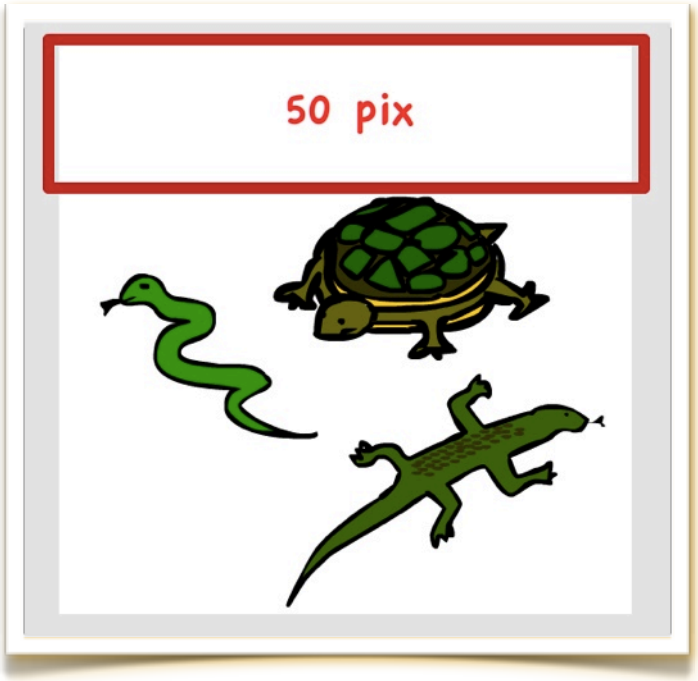

Figura 99. Distribución del espacio de los pictogramas.

La razón de éstas dimensiones en cada pictograma es porque en un principio la Base de Datos iba a gestionar cada imagen, y contando que iba a haber unas 1500 imágenes, el flujo de datos iba a ser enorme y por lo tanto la velocidad de procesado lenta.

Después se vio que lo mejor era almacenar en la Base de Datos nada más que el nombre de cada picto y entonces representar por pantalla el archivo con ese nombre. De esta forma la gestión fue mucho más fluida y no había problemas de procesado en la memoria de la aplicación.

Ésta solución fue mucho más tardía, y ya se habían dibujado los 1500 pictogramas, por lo que la exportación de los mismos a un tamaño mayor se postpone a un trabajo posterior.

Como consecuencia directa de esto, se observa que en *Visor de Fotos* de la aplicación *Álbum* las imágenes en la galería de pictogramas se ven pixeladas, pues no están creadas en el tamaño adecuado para su representación ampliada.

Continuando con el diseño, cada dibujo se realiza en una capa y un fotograma distinto, de tal manera que luego es mucho más fácil exportar la cadena de imágenes al mismo tiempo.

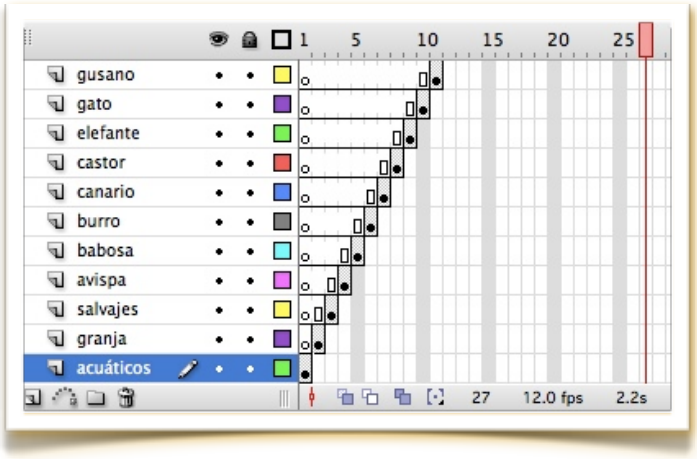

Figura 100. Distribución de las capas por cada pictograma.

Los pictos se guardan en *.png*, el formato perfecto para comprimir dibujos. Al no tener grandes detalles ni cambios suavizados de color, el contenedor no tiene pérdidas y comprime mucho mejor.

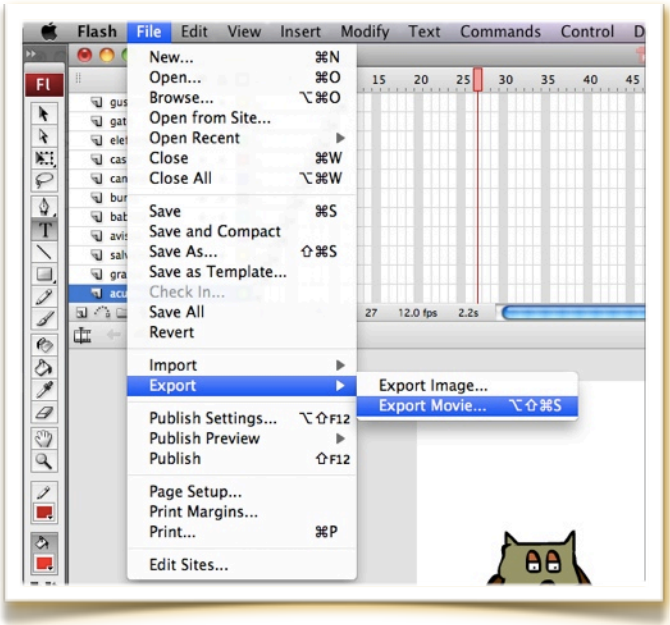

Figura 100. Exportación de cada elemento en forma de película.

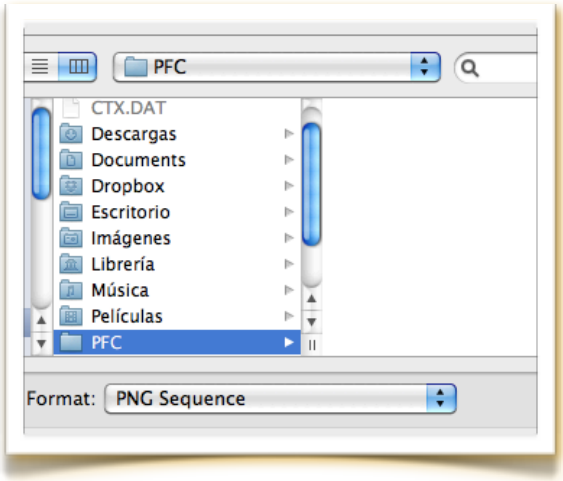

Figura 101. Las imágenes se guardan en formato de secuencia .png.

Se exporta la película completa como si fuera una secuencia de imágenes, y cada una de ellas se nombra con un nombre común.

Esto resulta más sencillo de cambiar uno a uno que exportar cada picto individualmente.

El proceso es completo y el trabajo se distribuye entre los tres estudiantes de tal forma que en un mes está terminado.

4.6.3. Botones:

El programa utilizado para el diseño de los botones es Adobe Photoshop CS4.

Figura 102. logotipo del programa Adobe Photoshop CS4 Extended.

Éste permite editar mediante capas, transparencias, pinturas, filtros y múltiples formas más imágenes en mapa de bits (rasterizadas).

Todos los botones integrados en el proyecto poseen el mismo patrón de diseño, pero hay que diferenciar dos tipos: *elementos de navegación* y *elementos de aplicación*.

ELEMENTOS DE NAVEGACIÓN

Se refieren a los botones que sirven para el desplazamiento del usuario entre las diferentes pantallas.

Su fondo es un cartón. Los símbolos que contienen son iconos conocidos y totalmente comprensibles. De esta forma se evita en todo momento el uso del texto como elemento de navegación.

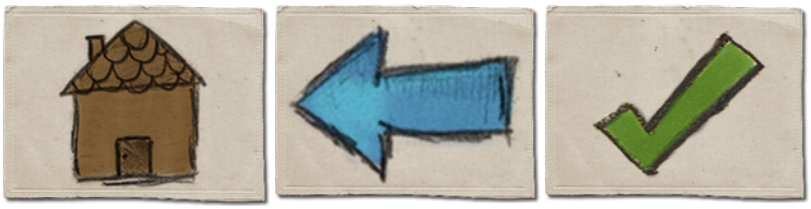

Figura 103. Muestra de los botones de navegación.

Son la clave de la interacción con la aplicación. Por ello deben estar distribuidos de forma ordenada en la pantalla y sobretodo constante para facilitar el proceso de aprendizaje y su usabilidad.

A lo largo de todo el programa los botones de navegación se establecen en la zona inferior de cada pantalla. Observando los referidos a *volver hacia atrás* en la parte izquierda, y los que interactúan con el proceso en ejecución en la parte central derecha.

Figura 104. Detalle de la disposición de los botones de navegación en la parte inferior de la pantalla.

Por otra parte, se observan los botones de aplicación.

ELEMENTOS DE APLICACIÓN

Igualmente están diseñados con Adobe Photoshop CS4.

Conforman el resto de botones y dan acceso a las diferentes mini-aplicaciones del programa, así como a determinadas funciones o eventos dentro de cada una de ellas.

También se disponen en un fondo de cartón, pero esta vez los dibujos son más elaborados y llamativos.

Así por ejemplo, en la pantalla principal se colocan los referidos a las mini-aplicaciones que contiene el proyecto.

Figura 105. Muestra de los botones de aplicación.

Su distribución no se establece mediante un patrón determinado, ya que puede estar dispersa por la pantalla, pero sí se realiza de forma ordenada e intuitiva.

4.6.4. Fondos:

Se plantean unos fondos que lleven un hilo común y que integren todas las mini-aplicaciones en una misma.

Se establece un papel *kraft verjurado* que cambiado de tonalidad diferencie cada apartado del proyecto.

Éste papel en sí mismo es uniforme, con una superficie ligeramente estriada que asemeja al cartón.

Figura 106. Detalle del papel kraft verjurado, fondo elegido.

Su escasez de detalle evita que el usuario pueda despistarse y a su vez la textura resulta familiar para su uso.

Cada mini-aplicación cuenta con un color diferente, el mismo dentro de sus pantallas internas, de tal forma que el niño puede identificar cada tarea con un color.

Por otra parte, se han combinado diferentes elementos a la hora de enmarcar objetos. Se han utilizado distintas texturas de papel, y sobre todo se ha hecho uso de las sombras para dar la sensación de tridimensionalidad y naturalidad buscadas.

Figura 107. Muestra de los fondos de la aplicación.

En cualquier caso, en todas las pantallas se han contrastado colores y elementos para mejorar las características intuitivas de la interfaz.

Capitulo 5: Conclusiones

Las conclusiones obtenidas tras la realización del proyecto han sido muchas y muy variadas.

Por ello se cree necesario diferencias dos tipos diferentes: las relacionadas con las conclusiones profesionales del trabajo, y las generales que abarcan más los objetivos a los que va destinado la realización del mismo.

5.1. PROFESIONALES

En primer lugar hay que destacar que todo diseño e implementación ha sido desarrollado pensando en el futuro de la aplicación. Todo planteamiento ha sido llevado a cabo con el objetivo de que si en un tiempo posterior se deseara ampliar y mejorar, todo fuera mucho más fácil.

Por otra parte mencionar que la decisión inicial de estudiar a fondo el lenguaje de programación ha sido considerada correcta ya que ha permitido trabajar con una gran fluidez dentro de los conocimientos adquiridos y sobretodo ha otorgado la capacidad de resolver problemas de un modo más sencillo y rápido.

Se ha visto que el mejor modo de solucionar problemas es muchas veces alejándose del mismo y dando la oportunidad de tener otro punto de vista, no obcecándose y permitiéndose a uno mismo pensar con claridad.

Como conclusión también el hecho de estar satisfecho con el trabajo realizado. En esta parte del proyecto se han construido mini-aplicaciones completas, desde el inicio hasta el final, con toda su estructura y funciones… de las cuales se comenzó con total desconocimiento.

Se ha comprendido un lenguaje de programación nuevo, lo que favorece la visión de implementación de cualquier ingeniero, y mejora sus cualidades y capacidades como tal.

También se ha aprendido una lección fundamental. El trabajo en equipo siempre es enriquecedor y los debates siempre son necesarios para poder seguir adelante y mejorar los resultados.

Como en todo proyecto, se han asimilado errores de proceso y ejecución. Fundamentos que se tendrán obviamente en cuenta para la próxima vez.

5.2. GENERALES

En cuanto a las conclusiones derivadas del trato directo con el entorno de trabajo, se pueden destacar una gran variedad.

Se ha observado un gran interés por parte de las personas implicadas dentro del mundo de la discapacidad por el funcionamiento de la aplicación y el lugar donde poder encontrarla.

En cuanto se presentó el proyecto en la primera charla se obtuvo un gran apoyo por parte de familias y educadores que querían preguntar, probar, dar nuevas ideas y sobretodo saber cuándo y donde se podía conseguir.

Es por esto por lo que la gran conclusión que se hace es que la evolución de las tecnologías a nivel mundial no está equiparado, ya que miles y miles de personas todavía no tienen acceso a mejoras.

Mejoras que en sí mismas son posibles porque actualmente se cuenta con las herramientas necesarias, las capacidades adquiridas para utilizarlas, y el personal adecuado para saber qué obtener.

Lo que está claro es que hay un hueco de mercado enorme, donde no hay de momento ningún interés en desarrollar, posiblemente porque el ser humano todavía no se ha dado cuenta que la mejor forma de avanzar y enriquecerse de conocimiento es compartiéndolos con las personas que abarquen otras disciplinas y desconozcan las propias.

Ésta razón es claramente otra conclusión más. El ritmo de trabajo obtenido en este proyecto ha sido gracias a la colaboración dentro de un equipo multidisciplinar. Las partes involucradas en el mundo de la educación orientado a la discapacidad, junto con las experiencias familiares y sabiduría técnica han sido la combinación perfecta para dar con el resultado obtenido.

Todas las personas que se han visto dentro del proyecto han aprendido del resto, y es una gran conclusión de la que todo el mundo debiera aprender.

Capitulo 6: Líneas futuras

Está claro que como proyecto, la presente aplicación se ha desarrollado al máximo posible y con ella cubrir todas las demandas del sector de la discapacidad comunicativa.

Aún así queda todavía mucho trabajo por hacer y mejorar.

6.1. ÁLBUM – COMUNICADOR

Del mismo modo que para aprender un idioma se utilizan primero las palabras más básicas y representativas y se ejercita con ellas, sería interesante que el álbum favorito estuviera disponible para su uso dentro del comunicador.

Actualmente el álbum favorito está destinado a ser una herramienta para el aprendizaje. El monitor que esté con el niño, el papá o la mamá, debe utilizarlo para enseñarle qué imágenes se asocian a cada elemento, verbo o persona. Pero esto se realizaría en un paso previo a la formación de frases.

La intención es que esta herramienta esté disponible también para la comunicación. Es decir, que el educador o familiar pudiera elegir los pictogramas asociados a la formación de frases, minimizar la cantidad inicial de imágenes y poder seleccionar aquellas que más le interesen para incluirlas en el lenguaje de comunicación.

De este modo, el aprendizaje sería paulatino y mucho más adecuado para el desarrollo comunicativo del usuario, al ir ejercitando sus conocimientos del lenguaje con la propia actividad.

Es por estas razones por las que se prevé su próximo desarrollo.

6.2. AGENDA – DIARIO

Por otra parte y como ya se ha visto en este documento anteriormente, el tema de la agenda y el diario se ha relegado a un proyecto posterior por salirse de los límites del presente trabajo.

Esta aplicación sigue siendo de las más importantes dentro del proyecto, ya que en tema de organización de tareas no hay nada desarrollado.

El concepto de tiempo es un problema para estas personas discapacitadas y a veces resulta muy difícil trabajar con ellas. Sobre todo en los momentos clave que son el inicio y el fin de una actividad. Por ello se necesita un apoyo visual que ayude a entender el suceso de acontecimientos y estimule la conciencia de rutinas o eventos.

Así mismo, en los centros especializados se utiliza un sistema diario para informar a los padres de las actividades realizadas durante el día, y del estado de sus hijos en general.

Con esta aplicación se pretende que sea el propio usuario quien interprete lo que pasa en cada momento y que sea capaz de registrarlo para posteriormente mostrarlo en casa o a sus amigos.

6.3. OTRAS

Por último y no por ello menos importantes, quedan por realizar otras muchas mejoras generales dentro de la propia aplicación.

Por ejemplo, la idea del caracol con su movimiento y su cuenta atrás sería muy interesante ampliarla a todas las pantallas del comunicador, si la tarea que se está realizando en ese momento es, por ejemplo, la formación de frases.

De este modo el usuario sabría en todo momento cuánto tiempo le queda para finalizar la actividad, esté haciendo lo que esté haciendo, siempre con el uso del dispositivo iPad.

También se pretende extender este proyecto a otros idiomas, ampliar el público de acceso y ayudar de este modo a muchas más personas que lo puedan estar necesitando fuera de España. Aún así, actualmente la aplicación cuenta con el idioma Inglés integrado en todas sus funciones.

Se recuerda que al disponer de una base de datos ya completa y gestionada, el añadir más campos destinados a más idiomas no supondría un gran esfuerzo.

Por otra parte, al igual que el resto de tecnologías, las de desarrollo de software están en continua evolución. Actualmente se encuentran en el mercado varias plataformas a las que se les da soporte y para las cuales hay mil y una aplicaciones desarrolladas.

Pues bien, también está la intención de expandir el presente software a otros sistemas tecnológicos, como puede ser el más común y utilizado Android.

Por último, cabe destacar que la intención de este proyecto es llegar a las más personas posibles y con ello facilitar y mejorar la vida de gente discapacitada y los que le rodean. Intentar hacer de este mundo algo más justo y tratar de que todos tengan las mismas posibilidades.

Es por esta razón por la que se pretende que el trabajo desarrollado no se quede sólo aquí, sino propagarlo.

En estos meses miembros del equipo de trabajo ya han comenzado a elaborar un plan de viabilidad que conciencie sobre la posibilidad de construir una empresa.

Capitulo 7: Presupuesto

Descripción detallada:

- **Ordenador MacBook Pro**, 15 Pulgadas, Aluminio **1.429,00 €**

  Configuración:
  Mando Apple Remote
  Accessory kit
  Teclado retroiluminado (Español) & Guía del usuario (En Español)
  SuperDrive 8x (DVD±R DL/DVD±RW/CD-RW)
  4GB 1066MHz DDR3 SDRAM - 2x2GB
  Core 2 Duo de Intel a 2,4 GHz
  250GB Serial ATA @ 7200

- **Impuesto por copyright***
  MacBook Pro, 15-inch, 2.4GHz, 2GB, 250GB **3,40 €**

- **MAC OS X 10.6.3** SNOW LEOPARD RETAIL-SPA **39,32 €**

- **Licencia Apple Developer** 99$ (Tasa=0,691) **68,37 €**

- **iPad** con Wi-Fi, 16GB, negro **479,00 €**

- **Salario Ingeniero Técnico de Telecomunicación**:
  1.450 € x 10 pagas = **14.500 €**

Total: 16.519,09 € (IVA incl)

Bibliografía.

Programación:

- http://www.elguruprogramador.com
- http://developer.apple.com
- http://www.desarrolloweb.com
- http://www.lenguajes-de-programacion.com
- www.video2brain.com
- http://cocoawithlove.com/
- https://github.com/klazuka/Kal
- http://stackoverflow.com/
- http://www.cocoalab.com/?q=BecomeAnXcoder-Español
- http://www.iphonesdkarticles.com/
- http://www.cristalab.com/tutoriales/fundamentos-de-programacion-para-iphone-c260l/
- http://www.cristalab.com/tutoriales/tutorial-de-programacion-para-iphone-c259l/
- http://www.video2brain.com/es/products-140.htm
- http://www.icodeblog.com/2010/07/08/asset-libraries-and-blocks-in-ios-4/
- http://blog.webscale.co.in/?p=228
- http://www.trekmedics.org/projects/ecal-iphone-app-tutorial/
- http://www.iphoneapplications.us/iphone-3gs-application-games-development-services.html

- http://www.markj.net/iphone-memory-debug-nszombie/

- http://idevzilla.com/2010/09/16/uiscrollview-a-really-simple-tutorial/

- http://kwigbo.tumblr.com/post/758575763/uiscrollview-image-gallery-tutorial

- http://forums.macrumors.com/

- http://ved-dimensions.blogspot.com/2009/04/iphone-development-creating-native_09.html

- http://code.google.com/p/iphonecal/

- http://mobileorchard.com/code-sharing-via-static-libraries-and-cross-project-references/

- http://appfinder.lisisoft.com/

- http://mobileorchard.com/31-iphone-applications-with-source-code/

- http://www.cocoadev.com/index.pl?NSUInteger

- http://www.edumobile.org/iphone/iphone-programming-tutorials/scrollview-example-in-iphone/

- http://cocoadevcentral.com/d/learn_objectivec/
-

Antecedentes:

- http://www.handholdadaptive.com

- http://www.catedu.es/arasaac/

- http://www.elblogdenits.com.ar/blog/2009/04/03/boardmaker-pictogramas-para-comunicacion-en-ninos-autistas/

Instituciones:

- http://www.anfasnavarra.org

- http://comunidad.cajanavarra.es/isterria-centro-educacion-para-ninos-con-discapacidad/

Anexos.

1. INTEGRACIÓN DE *KAL*

NOTA: Todos los archivos correspondientes a esta implementación *open – source* se encuentran alojados en la página web: https://github.com/klazuka/Kal

README.markdown

Kal - a calendar component for the iPhone

This project aims to provide an open-source implementation of the month view in Apple's mobile calendar app (MobileCal). When the user taps a day on the calendar, any associated data for that day will be displayed in a table view directly below the calendar. As a client of the Kal component, you have 2 responsibilities:

1. Tell Kal which days need to be marked with a dot because they have associated data.
2. Provide UITableViewCells which display the details (if any) for the currently selected day.

In order to use Kal in your application, you will need to provide an implementation of the KalDataSource protocol to satisfy these responsibilities. Please see KalDataSource.h and the included demo app for more details.

Release Notes

July 9, 2010

This is the iOS 4.0 / iPhone4 release. New features include:

1) A refactored project file. Kal is now built as a static library in a separate Xcode project. Regardless of whether you are a new or existing user of Kal, please read the section entitled "Integrating Kal into Your Project" below.

2) The project now specifies iOS 4.0 as the Base SDK. So if you want to upgrade to this release of Kal, you must upgrade your SDK.

3) Added hi-res graphics for Retina Display support. Extra special thanks to Paul Calnan for sending me the hi-res graphics.

4) Added a new example app, "NativeCal," which demonstrates how to integrate Kal with the EventKit framework that Apple made available in iOS 4.

NOTE I'm not crazy about the KalDataSource asynchronous/synchronous API. I will probably be changing it in the future and updating the example apps to use GCD and blocks.

March 11, 2010

A lot of people have emailed me asking for support for selecting and displaying an arbitrary date on the calendar. So today I pushed some commits that make this easy to do. You can specify which date should be initially selected and shown when the calendar is first created by using -[KalViewController initWithSelectedDate:]. If you would like to programmatically switch the calendar to display the month for an arbitrary date and select that date, use -[KalViewController showAndSelectDate:].

January 1, 2010

I have made significant changes to the KalDataSource API so that the client can respond to the data request asynchronously. The Kal demo app, "Holidays," now includes 2 example datasources:

1. HolidayJSONDataSource - retrieves data asynchronously from http://keith.lazuka.org/holidays.json
2. HolidaySqliteDataSource - queries an Sqlite database inside the application bundle and responds synchronously (because the query is fast enough that it doesn't affect UI responsiveness too badly).

December 19, 2009

Initial public release on GitHub.

Example Usage

Note: All of the following example code assumes that it is being called from within another UIViewController which is in a UINavigationController hierarchy.

How to display a very basic calendar (without any events):

```
KalViewController *calendar = [[[KalViewController alloc] init] autorelease];
[self.navigationController pushViewController:calendar animated:YES];
```

In most cases you will have some custom data that you want to attach to the dates on the calendar. The first thing you must do is provide an implementation of the KalDataSource protocol. Then all you need to do to display your annotated calendar is instantiate the KalViewController and tell it to use your KalDataSource implementation (in this case, "MyKalDataSource"):

```
id<KalDataSource> source = [[MyKalDataSource alloc] init];
KalViewController *calendar = [[[KalViewController alloc] initWithDataSource:source] autorelease];
[self.navigationController pushViewController:calendar animated:YES];
```

NOTE: KalViewController does not retain its datasource. You probably will want to store a reference to the dataSource in an instance variable so that you can release it after the calendar has been destroyed.

Integrating Kal into Your Project

Kal is compiled as a static library, and the recommended way to add it to your project is to use Xcode's "dependent project" facilities by following these step-by-step instructions:

1. Clone the Kal git repository: git clone git://github.com/klazuka/Kal.git. Make sure you store the repository in a permanent place because Xcode will need to reference the files every time you compile your project.
2. Locate the "Kal.xcodeproj" file under "Kal/src/". Drag Kal.xcodeproj and drop it onto the root of your Xcode project's "Groups and Files" sidebar. A dialog will appear -- make sure "Copy items" is unchecked and "Reference Type" is "Relative to Project" before clicking "Add".
3. Now you need to link the Kal static library to your project. Select the Kal.xcodeproj file that you just added to the sidebar. Under the "Details" table, you will see libKal.a. Check the checkbox on the far right for this file. This will tell Xcode to link against Kal when building your app.
4. Now you need to add Kal as a dependency of your project so that Xcode will compile it whenever you compile your project. Expand the "Targets" section of the sidebar and double-click your application's target. Under the "General" tab you will see a "Direct Dependencies" section. Click the "+" button, select "Kal" and click "Add Target".
5. Now you need to add the bundle of image resources internally used by Kal's UI. Locate "Kal.bundle" under "Kal/src" and drag and drop it into your project. A dialog will appear -- make sure "Create Folder References" is selected, "Copy items" is unchecked, and "Reference Type" is "Relative to Project" before clicking "Add".
6. Finally, we need to tell your project where to find the Kal headers. Open your "Project Settings" and go to the "Build" tab. Look for "Header Search Paths" and double-click it. Add the relative path from your project's directory to the "Kal/src" directory.
7. While you are in Project Settings, go to "Other Linker Flags" under the "Linker" section, and add "-all_load" to the list of flags.

8. You're ready to go. Just #import "Kal.h" anywhere you want to use KalViewController in your project.

Additional Notes

The Xcode project includes two demo apps:

1) "Holiday" demonstrates how to use Kal to display several 2009 and 2010 world holidays using both JSON and Sqlite datasources.

2) "NativeCal" demonstrates how to use Kal with the EventKit framework.

Kal is fully localized. The month name and days of the week will automatically use the appropriate language and style for the iPhone's current regional settings.

2. CÓDIGO FUENTE DE LA APLICACIÓN:

AJUSTES:

Ajustes(.h y .m):

```
//
//  Ajustes.h
//  COMUNICADOR
//
//  Created by Elisabeth Esandia Melero on 27/06/11.
//  Copyright 2011 __MyCompanyName__. All rights reserved.
//

#import <UIKit/UIKit.h>

# define KEY_TIPO_LETRA @"tipo_letra"
# define KEY_RESP_NEGATIVA @"resp_negativa"
# define KEY_TIEMPO_PULSACION @"tiempo_pulsacion"

# define ficheroPersistente @"ajustes.plist"

@class COMUNICADORViewController;
@class EditarAlbumFavorito;
@class Personalizacion;


@interface Ajustes : UIViewController {

    NSString *rutaFichero;
    NSMutableArray *ajustes;

    COMUNICADORViewController *controladorViewController;
    EditarAlbumFavorito * EditarAlbumFavoritoVista;
    Personalizacion * controladorPersonalizacion;

    int selectedUnit_resp_negativa;
        IBOutlet UISegmentedControl *segmentedControl_resp_negativa;

    int selectedUnit3;
        UISegmentedControl *segmentedControl_tiempo_pulsacion;

    int selectedUnit_minusc_mayusc;
        UISegmentedControl *segmentedControl_minusc_mayusc;

    int selectedUnit_cast_ing;
        UISegmentedControl *segmentedControl_cast_ing;

    IBOutlet UILabel * label_resp_negativa;
    IBOutlet UILabel * label_tiempo_pulsacion;
    IBOutlet UILabel * label_tamano_letra;
    IBOutlet UILabel * label_idioma;
    IBOutlet UILabel * label_pictos_personalizar;
    IBOutlet UILabel * label_album_favorito_personalizar;

    IBOutlet UIButton * but_pictos_personalizar;
    IBOutlet UIButton * but_album_favorito_personalizar;
    IBOutlet UIButton * but_guardar_ajustes;
    IBOutlet UIButton * but_inicio;


    IBOutlet id BOTON_AJUSTES;

    int resp_negativa;
```

```
    int tiempo_pulsacion;
    int tipo_letra;

    NSString * resp_negativa_string;
    NSString * tiempo_pulsacion_string;
    NSString * tipo_letra_string;


}


@property (nonatomic,retain) NSString *rutaFichero;
@property (nonatomic,retain) NSMutableArray *ajustes;

@property (nonatomic, retain) COMUNICADORViewController *controladorViewController;
@property (nonatomic, retain) EditarAlbumFavorito *EditarAlbumFavoritoVista;
@property (nonatomic, retain) Personalizacion *controladorPersonalizacion;
@property (nonatomic, retain) IBOutlet UISegmentedControl *segmentedControl_resp_negativa;
@property (nonatomic, retain) IBOutlet UISegmentedControl *segmentedControl_tiempo_pulsacion;
@property (nonatomic, retain) IBOutlet UISegmentedControl *segmentedControl_minusc_mayusc;
@property (nonatomic, retain) IBOutlet UISegmentedControl *segmentedControl_cast_ing;

@property (nonatomic, retain) IBOutlet UILabel * label_resp_negativa;
@property (nonatomic, retain) IBOutlet UILabel * label_tiempo_pulsacion;
@property (nonatomic, retain) IBOutlet UILabel * label_tamano_letra;
@property (nonatomic, retain) IBOutlet UILabel * label_idioma;
@property (nonatomic, retain) IBOutlet UILabel * label_pictos_personalizar;
@property (nonatomic, retain) IBOutlet UILabel * label_album_favorito_personalizar;

@property (nonatomic, retain)IBOutlet UIButton * but_pictos_personalizar;
@property (nonatomic, retain)IBOutlet UIButton * but_album_favorito_personalizar;
@property (nonatomic, retain)IBOutlet UIButton * but_guardar_ajustes;
@property (nonatomic, retain)IBOutlet UIButton * but_inicio;

@property (nonatomic) int resp_negativa;
@property (nonatomic) int tiempo_pulsacion;
@property (nonatomic) int tipo_letra;

@property (nonatomic,retain) NSString *resp_negativa_string;
@property (nonatomic,retain) NSString *tiempo_pulsacion_string;
@property (nonatomic,retain) NSString *tipo_letra_string;

- (IBAction)segmentedControlValueChangedRespNegativa;
- (void)cambioRespuestaNegativa;

- (IBAction)segmentedControlValueChanged3;
- (void)cambioTiempoPulsacion;

- (IBAction)segmentedControlValueChanged4;
- (void)cambioIdioma;

- (IBAction)segmentedControlValueChanged6;
- (void)cambioMayusculasMinusculas;

- (IBAction)ajustes_personalizar_pictos;
- (IBAction)ajustes_personalizar_album_favorito;

- (NSString *) rutaFichero;
- (void) mostrarAjustes;
- (void) guardarAjustes;

- (int)CargarRespuestaNegativa;

//- (IBAction) PULSAR_BOTON_OPCIONES: (id) sender;
- (IBAction) PULSAR_BOTON_SALIR_AJUSTES: (id) sender;
- (IBAction) PULSAR_BOTON_GUARDAR_AJUSTES: (id)sender;

@end
```

```
//
//  Ajustes.m
//  COMUNICADOR
//
//  Created by Elisabeth Esandia Melero on 27/06/11.
//  Copyright 2011 __MyCompanyName__. All rights reserved.
//

#import "Ajustes.h"
#import "COMUNICADORViewController.h"
#import "EditarAlbumFavorito.h"
#import "Personalizacion.h"

@implementation Ajustes

@synthesize rutaFichero;
@synthesize ajustes;

@synthesize controladorViewController;
@synthesize EditarAlbumFavoritoVista;
@synthesize controladorPersonalizacion;

@synthesize segmentedControl_resp_negativa;
@synthesize segmentedControl_tiempo_pulsacion;
@synthesize segmentedControl_minusc_mayusc;
@synthesize segmentedControl_cast_ing;

@synthesize label_resp_negativa;
@synthesize label_tiempo_pulsacion;
@synthesize label_tamano_letra;
@synthesize label_idioma;
@synthesize label_pictos_personalizar;
@synthesize label_album_favorito_personalizar;

@synthesize but_pictos_personalizar;
@synthesize but_album_favorito_personalizar;
@synthesize but_guardar_ajustes;
@synthesize but_inicio;

@synthesize resp_negativa;
@synthesize tiempo_pulsacion;
@synthesize tipo_letra;

@synthesize resp_negativa_string;
@synthesize tiempo_pulsacion_string;
@synthesize tipo_letra_string;



- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{
    self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];
    if (self) {
        // Custom initialization
    }
    return self;
}

- (void)dealloc
{
    [super dealloc];
}

- (void)didReceiveMemoryWarning
{
```

```
    // Releases the view if it doesn't have a superview.
    [super didReceiveMemoryWarning];

    // Release any cached data, images, etc that aren't in use.
}

#pragma mark - View lifecycle

- (void)viewDidLoad
{
    NSString *fichero= [self rutaFichero];
    if ([[NSFileManager defaultManager] fileExistsAtPath:fichero])
    {
        NSArray *listado =[[NSArray alloc] initWithContentsOfFile:fichero];
        ajustes=[[NSMutableArray alloc]initWithArray:listado];
        [self mostrarAjustes];
    }
    else
    {
        ajustes=[[NSMutableArray alloc]init];

        resp_negativa=1;
        tiempo_pulsacion=0.5;
        tipo_letra=20;

        [self guardarAjustes];
    }

    [super viewDidLoad];
    // Do any additional setup after loading the view from its nib.
}

- (int)CargarRespuestaNegativa
{

    return resp_negativa;
}

- (void)viewDidUnload
{
    [super viewDidUnload];
    // Release any retained subviews of the main view.
    // e.g. self.myOutlet = nil;
}

- (BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)interfaceOrientation
{
    // Return YES for supported orientations
        return YES;
}

- (void)viewDidAppear:(BOOL)animated {
        [super viewDidAppear:animated];
    selectedUnit_resp_negativa=segmentedControl_resp_negativa.selectedSegmentIndex;
    [self cambioRespuestaNegativa];
        selectedUnit3 = segmentedControl_tiempo_pulsacion.selectedSegmentIndex;
        [self cambioTiempoPulsacion];
    selectedUnit_cast_ing = segmentedControl_cast_ing.selectedSegmentIndex;
        [self cambioIdioma];
    selectedUnit_minusc_mayusc = segmentedControl_minusc_mayusc.selectedSegmentIndex;
        [self cambioMayusculasMinusculas];


}

- (IBAction)segmentedControlValueChangedRespNegativa
{
    selectedUnit_resp_negativa = segmentedControl_resp_negativa.selectedSegmentIndex;
```

```
    [self cambioRespuestaNegativa];
    [self.view setNeedsDisplay];
}

- (IBAction)segmentedControlValueChanged3 {
        selectedUnit3 = segmentedControl_tiempo_pulsacion.selectedSegmentIndex;
        [self cambioTiempoPulsacion];
        [self.view setNeedsDisplay];
}

- (IBAction)segmentedControlValueChanged4 {
        selectedUnit_cast_ing = segmentedControl_cast_ing.selectedSegmentIndex;
        [self cambioIdioma];
        [self.view setNeedsDisplay];
}

- (IBAction)segmentedControlValueChanged6 {
        selectedUnit_minusc_mayusc = segmentedControl_minusc_mayusc.selectedSegmentIndex;
        [self cambioMayusculasMinusculas];
        [self.view setNeedsDisplay];
}

- (void)cambioRespuestaNegativa
{
    if (selectedUnit_resp_negativa == 0) {
                resp_negativa=0;
            [   self.view setNeedsDisplay];
        }
        else if (selectedUnit_resp_negativa == 1) {
                resp_negativa=1;
            [   self.view setNeedsDisplay];
        }
}

- (void)cambioTiempoPulsacion {
        if (selectedUnit3 == 0) {
                tiempo_pulsacion=0.5;
            [   self.view setNeedsDisplay];
        }
        else if (selectedUnit3 == 1) {
                tiempo_pulsacion=1;
            [   self.view setNeedsDisplay];
        }
        else if (selectedUnit3 == 2) {
                tiempo_pulsacion=2;
            [   self.view setNeedsDisplay];
        }
        else if (selectedUnit3 == 3) {
                tiempo_pulsacion=3;
            [   self.view setNeedsDisplay];
        }
        else if (selectedUnit3 == 4) {
                tiempo_pulsacion=4;
            [   self.view setNeedsDisplay];
        }

    //[seleccionAlbumAjustes setAjustes:tiempo_pulsacion];

}

- (void)cambioIdioma
{
        if (selectedUnit_cast_ing == 0 && selectedUnit_minusc_mayusc==0)
    {
                tipo_letra=20;
            [   self.view setNeedsDisplay];
        }
        else if (selectedUnit_cast_ing == 0 && selectedUnit_minusc_mayusc==1)
```

```
    {
                        tipo_letra=30;
                [       self.view setNeedsDisplay];
        }
    else if (selectedUnit_cast_ing == 1 && selectedUnit_minusc_mayusc==0)
    {
                        tipo_letra=50;
                [       self.view setNeedsDisplay];
        }
    else if (selectedUnit_cast_ing == 1 && selectedUnit_minusc_mayusc==1)
    {
                        tipo_letra=60;
                [       self.view setNeedsDisplay];
        }
}

- (void)cambioMayusculasMinusculas
{
        if (selectedUnit_cast_ing == 0 && selectedUnit_minusc_mayusc==0)
    {
                        tipo_letra=20;
                [       self.view setNeedsDisplay];
        }
        else if (selectedUnit_cast_ing == 0 && selectedUnit_minusc_mayusc==1)
    {
                        tipo_letra=30;
                [       self.view setNeedsDisplay];
        }
    else if (selectedUnit_cast_ing == 1 && selectedUnit_minusc_mayusc==0)
    {
                        tipo_letra=50;
                [       self.view setNeedsDisplay];
        }
    else if (selectedUnit_cast_ing == 1 && selectedUnit_minusc_mayusc==1)
    {
                        tipo_letra=60;
                [       self.view setNeedsDisplay];
        }
}
/*
 -(IBAction) PULSAR_BOTON_OPCIONES: (id) sender
 {
 //NSUserDefaults *defaults = [NSUserDefaults standardUserDefaults];
 //resp_negativa = [defaults boolForKey:@"Respuesta negativa"];

 if (resp_negativa == NO)
 {
 [BOTON_AJUSTES setTitle:@"ON" forState:0];
 resp_negativa = YES;

 }
 else
 {
 [BOTON_AJUSTES setTitle:@"OFF" forState:0];
 resp_negativa = NO;
 }
 }
 */
- (IBAction)ajustes_personalizar_pictos
{
    [controladorPersonalizacion viewDidLoad];
    controladorPersonalizacion=[[Personalizacion alloc] initWithNibName:@"Personalizacion" bundle:[NSBundle
mainBundle]];

    [self.view addSubview:controladorPersonalizacion.view];
}

- (NSString *)rutaFichero
```

```
{

    NSArray *rutas = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES);
    NSLog(@"Directorios: %@", rutas);

    NSString *directorio = [rutas objectAtIndex:0];

    return [directorio stringByAppendingPathComponent:ficheroPersistente];
}

- (void) mostrarAjustes
{
    NSDictionary *item =[ajustes objectAtIndex:0];

    //se convierten los valores a números
    resp_negativa= [[item objectForKey:KEY_RESP_NEGATIVA] intValue];
    tiempo_pulsacion = [[item objectForKey:KEY_TIEMPO_PULSACION] intValue];
    tipo_letra = [[item objectForKey:KEY_TIPO_LETRA] intValue];



    int idioma;
    int tamano_letra;

    if (tipo_letra<40)
    {
        idioma=0;
        label_resp_negativa.text=@"RESPUESTA NEGATIVA";
        label_tiempo_pulsacion.text=@"TIEMPO DE PULSADO (segundos)";
        label_tamano_letra.text=@"TIPO DE LETRA";
        label_idioma.text=@"IDIOMA";
        label_pictos_personalizar.text=@"INTRODUCE TUS PICTOGRAMAS O MODIFICA LOS EXISTENTES";
        label_album_favorito_personalizar.text=@"PERSONALIZA TU ÁLBUM DE FAVORITOS";

        [segmentedControl_minusc_mayusc setTitle:@"minúsculas" forSegmentAtIndex:0];
        [segmentedControl_minusc_mayusc setTitle:@"MAYÚSCULAS" forSegmentAtIndex:1];
        [segmentedControl_cast_ing setTitle:@"Castellano" forSegmentAtIndex:0];
        [segmentedControl_cast_ing setTitle:@"Inglés" forSegmentAtIndex:1];

        [but_pictos_personalizar setTitle:@"Personalizar pictos" forState:normal];
        [but_album_favorito_personalizar setTitle:@"Personalizar álbum" forState:normal];
        [but_guardar_ajustes setTitle:@"GUARDAR" forState:normal];
        [but_inicio setTitle:@"INICIO" forState:normal];


    }
    else
    {
        idioma=1;
        label_resp_negativa.text=@"SET ACTIVE 'SAY NO'";
        label_tiempo_pulsacion.text=@"TOUCHING TIME (seconds)";
        label_tamano_letra.text=@"FONT TYPE";
        label_idioma.text=@"LANGUAGE";
        label_pictos_personalizar.text=@"ADD OR EDIT YOUR OWN PICTOS";
        label_album_favorito_personalizar.text=@"CHOOSE YOUR FAVOURITE ALBUM PICTURES";

        [segmentedControl_minusc_mayusc setTitle:@"small letter" forSegmentAtIndex:0];
        [segmentedControl_minusc_mayusc setTitle:@"CAPITAL LETTER" forSegmentAtIndex:1];
        [segmentedControl_cast_ing setTitle:@"Spanish" forSegmentAtIndex:0];
        [segmentedControl_cast_ing setTitle:@"English" forSegmentAtIndex:1];

        [but_pictos_personalizar setTitle:@"Pictos personalize" forState:normal];
        [but_album_favorito_personalizar setTitle:@"Album personalize" forState:normal];
        [but_guardar_ajustes setTitle:@"SAVE" forState:normal];
        [but_inicio setTitle:@"HOME" forState:normal];
    }
    if (tipo_letra==20 || tipo_letra==50)
    {
```

```
        tamano_letra=0;
    }
    else
    {
        tamano_letra=1;
    }

    NSLog(@"negativa %i, pulsacion %i, idioma %i, tamanoletra %i", resp_negativa, tiempo_pulsacion, idioma,
tamano_letra);

    //se colocan los segmentos seleccionados correspondientes.
    [segmentedControl_resp_negativa setSelectedSegmentIndex:resp_negativa];
        [segmentedControl_tiempo_pulsacion setSelectedSegmentIndex:tiempo_pulsacion];
    [segmentedControl_cast_ing setSelectedSegmentIndex:idioma];
    [segmentedControl_minusc_mayusc setSelectedSegmentIndex:tamano_letra];
}

- (void) guardarAjustes;
{
    if (selectedUnit_cast_ing == 0 && selectedUnit_minusc_mayusc==0)
    {
                    tipo_letra=20;
                [   self.view setNeedsDisplay];
        }
        else if (selectedUnit_cast_ing == 0 && selectedUnit_minusc_mayusc==1)
    {
                    tipo_letra=30;
                [   self.view setNeedsDisplay];
        }
    else if (selectedUnit_cast_ing == 1 && selectedUnit_minusc_mayusc==0)
    {
                    tipo_letra=50;
                [   self.view setNeedsDisplay];
        }
    else if (selectedUnit_cast_ing == 1 && selectedUnit_minusc_mayusc==1)
    {
                    tipo_letra=60;
                [   self.view setNeedsDisplay];
    }

    resp_negativa_string=[NSString stringWithFormat:@"%d",resp_negativa];
    tiempo_pulsacion_string=[NSString stringWithFormat:@"%d",tiempo_pulsacion];
    tipo_letra_string=[NSString stringWithFormat:@"%d",tipo_letra];

    NSLog(@"negativa %@, tipo letra %i",resp_negativa_string, tipo_letra);

    NSDictionary *item =[NSDictionary dictionaryWithObjectsAndKeys:tipo_letra_string, KEY_TIPO_LETRA,
tiempo_pulsacion_string, KEY_TIEMPO_PULSACION, resp_negativa_string, KEY_RESP_NEGATIVA, nil];
    [self.ajustes removeAllObjects];
    [self.ajustes addObject:item];

    [ajustes writeToFile:[self rutaFichero] atomically:YES];

    [self mostrarAjustes];



}

- (IBAction)ajustes_personalizar_album_favorito
{
    EditarAlbumFavoritoVista=[[EditarAlbumFavorito alloc] initWithNibName:@"EditarAlbumFavorito" bundle:[NSBundle
mainBundle]];

    [self.view addSubview:EditarAlbumFavoritoVista.view];
}
```

```
-(IBAction) PULSAR_BOTON_SALIR_AJUSTES: (id) sender
{
    [self guardarAjustes];
        [self.view removeFromSuperview];
}

- (IBAction) PULSAR_BOTON_GUARDAR_AJUSTES: (id)sender
{
   [self guardarAjustes];
   if (tipo_letra<40) // IDIOMA CASTELLANO
   {

      UIAlertView *alertView = [[UIAlertView alloc] initWithTitle:@"ENHORABUENA" message:@"Sus ajustes han sido
guardados correctamente" delegate:self cancelButtonTitle:nil otherButtonTitles:@"Ok", nil];
      [alertView show];
      [alertView release];


   }
   else //IDIOMA INGLÉS
   {
      UIAlertView *alertView = [[UIAlertView alloc] initWithTitle:@"CONGRATULATIONS" message:@"Your changes have
been saved correctly" delegate:self cancelButtonTitle:nil otherButtonTitles:@"Ok", nil];
      [alertView show];
      [alertView release];


   }

}


@end
```

EditarAlbumFavorito (.h y .m):

```
//
//  EditarAlbumFavorito.h
//  COMUNICADOR
//
//  Created by Elisabeth Esandia Melero on 07/06/11.
//  Copyright 2011 __MyCompanyName__. All rights reserved.
//

#import "COMUNICADORAppDelegate.h"
#import <UIKit/UIKit.h>

@class Ajustes;

@interface EditarAlbumFavorito : UIViewController
{
   ////////////////////////////
   //      AJUSTES          //
   Ajustes * ControladorAjustes;

   int ajuste_resp_negativa;
   int ajuste_tiempo_pulsacion;
   int ajuste_tipo_letra;

   //En Idioma irá el string correspondiente a la columna de la base de datos que quiero consultar
   NSString *Idioma;
   ////////////////////////////

   NSArray *objects_entidad;              // Objetos cogidos(entidad completa) de la llamada a la BD.
   NSMutableArray *objectsCogidos_entidad; // Objetos almacenados de "objects_entidad" para trabajar con ellos en la
recarga.
```

```
    NSArray *objects_favoritos;              // Objetos cogidos (favoritos) de la llamada a la BD.
    NSMutableArray *objectsCogidos_favoritos; // Objetos almacenados de "objects_favoritos" para trabajar con ellos en la
recarga.

    NSMutableArray *Entidad_table;          // Tabla de miniaturas de Entidad.
    NSMutableArray *Favorito_table;   // Tabla de miniaturas de Favorito.
    NSMutableArray *EntidadLabel_table;
    NSMutableArray *FavoritoLabel_table;
    int EntidadActual;
    int num_pag_view;                    // Índice de la página que estamos visualizando de objetos de entidad.
    int indice;                        // Índice del objeto que estamos viendo de la entidad.
    int num_pag_view_per;                 // Índice de la página que estamos visualizando de objetos de entidad personalizada.
    int indice_per;                    // Índice del objeto que estamos viendo de la entidad personalizada.

    NSMutableArray *selecc_originales;      // Array de elementos que se van seleccionando de cada entidad.
    NSMutableArray *selecc_favoritos;  // Array de elementos que se van seleccionando en la entidad personalizada.
    NSMutableArray *datos_favoritos;   // Array de elementos que aparecerán en cada entidad personalizada.

    //NSInteger *ButtonTag;                 // Valor del tag del botón miniatura que se pulsa.

    ////////////////////  LLAMADA A LA BD

    COMUNICADORAppDelegate *appDelegate;
    NSManagedObjectContext *context;
    NSEntityDescription *entityDesc;
    NSFetchRequest *request;
    NSError *error;

    COMUNICADORAppDelegate *appDelegate_per;
    NSManagedObjectContext *context_per;
    NSEntityDescription *entityDesc_per;
    NSFetchRequest *request_favorito;
    NSPredicate *pred_favorito;
    NSError *error_per;

    UIAlertView *alertaSimple;            // Alerta simple por si no hay objetos.
    UIAlertView *alertaSimplePersonalizada;
    UIAlertView *alertaSimpleGuardar;

    ////////////////////   BOTONES EDICIÓN

    IBOutlet UIButton *butAgregar;
    IBOutlet UIButton *butQuitar;
    IBOutlet UIButton *butGuardar;

    ////////////////////   BOTONES MINIATURAS

        IBOutlet UIButton * EntidadPicto0;
        IBOutlet UIButton * EntidadPicto1;
        IBOutlet UIButton * EntidadPicto2;
        IBOutlet UIButton * EntidadPicto3;
        IBOutlet UIButton * EntidadPicto4;
        IBOutlet UIButton * EntidadPicto5;
        IBOutlet UIButton * EntidadPicto6;
        IBOutlet UIButton * EntidadPicto7;
        IBOutlet UIButton * EntidadPicto8;
        IBOutlet UIButton * EntidadPicto9;
        IBOutlet UIButton * EntidadPicto10;
        IBOutlet UIButton * EntidadPicto11;
        IBOutlet UIButton * EntidadPicto12;
        IBOutlet UIButton * EntidadPicto13;

    IBOutlet UIButton * FavoritoPicto1;
        IBOutlet UIButton * FavoritoPicto2;
        IBOutlet UIButton * FavoritoPicto3;
        IBOutlet UIButton * FavoritoPicto4;
        IBOutlet UIButton * FavoritoPicto5;
        IBOutlet UIButton * FavoritoPicto6;
```

```
        IBOutlet UIButton * FavoritoPicto7;
        IBOutlet UIButton * FavoritoPicto8;
        IBOutlet UIButton * FavoritoPicto9;
        IBOutlet UIButton * FavoritoPicto10;
        IBOutlet UIButton * FavoritoPicto11;
        IBOutlet UIButton * FavoritoPicto12;
        IBOutlet UIButton * FavoritoPicto13;
        IBOutlet UIButton * FavoritoPicto14;

    IBOutlet UILabel * Label1;
        IBOutlet UILabel * Label2;
        IBOutlet UILabel * Label3;
        IBOutlet UILabel * Label4;
        IBOutlet UILabel * Label5;
        IBOutlet UILabel * Label6;
        IBOutlet UILabel * Label7;
        IBOutlet UILabel * Label8;
        IBOutlet UILabel * Label9;
        IBOutlet UILabel * Label10;
        IBOutlet UILabel * Label11;
        IBOutlet UILabel * Label12;
        IBOutlet UILabel * Label13;
        IBOutlet UILabel * Label14;

        IBOutlet UILabel * Label15;
    IBOutlet UILabel * Label16;
        IBOutlet UILabel * Label17;
        IBOutlet UILabel * Label18;
        IBOutlet UILabel * Label19;
        IBOutlet UILabel * Label20;
        IBOutlet UILabel * Label21;
        IBOutlet UILabel * Label22;
        IBOutlet UILabel * Label23;
        IBOutlet UILabel * Label24;
        IBOutlet UILabel * Label25;
        IBOutlet UILabel * Label26;
        IBOutlet UILabel * Label27;
        IBOutlet UILabel * Label28;


    ////////////////////   BOTONES NAVEGACIÓN

    IBOutlet UILabel *label_lugar;
    IBOutlet UILabel *label_sublugar;
    IBOutlet UILabel *label_personas;
    IBOutlet UILabel *label_verbos;
    IBOutlet UILabel *label_complementos;

    IBOutlet UIButton * BotonEntidadSiguiente;
    IBOutlet UIButton * BotonEntidadAnterior;

    IBOutlet UIButton * BotonFavoritoSiguiente;
    IBOutlet UIButton * BotonFavoritoAnterior;

    IBOutlet UIButton *butVolver;


    IBOutlet UILabel *EntidadTitle;     // Label controlador del título.
    IBOutlet UILabel *EntidadFavoritoTitle;

    BOOL onFavorito;
}

@property (nonatomic, retain) Ajustes *ControladorAjustes;

@property(nonatomic,retain)IBOutlet UILabel *label_lugar;
@property(nonatomic,retain)IBOutlet UILabel *label_sublugar;
```

```
@property(nonatomic,retain)IBOutlet UILabel *label_personas;
@property(nonatomic,retain)IBOutlet UILabel *label_verbos;
@property(nonatomic,retain)IBOutlet UILabel *label_complementos;

@property(nonatomic,retain)IBOutlet UIButton *butAgregar;
@property(nonatomic,retain)IBOutlet UIButton *butQuitar;
@property(nonatomic,retain)IBOutlet UIButton *butGuardar;

@property(nonatomic,retain)IBOutlet UIButton *EntidadPicto0;
@property(nonatomic,retain)IBOutlet UIButton *EntidadPicto1;
@property(nonatomic,retain)IBOutlet UIButton *EntidadPicto2;
@property(nonatomic,retain)IBOutlet UIButton *EntidadPicto3;
@property(nonatomic,retain)IBOutlet UIButton *EntidadPicto4;
@property(nonatomic,retain)IBOutlet UIButton *EntidadPicto5;
@property(nonatomic,retain)IBOutlet UIButton *EntidadPicto6;
@property(nonatomic,retain)IBOutlet UIButton *EntidadPicto7;
@property(nonatomic,retain)IBOutlet UIButton *EntidadPicto8;
@property(nonatomic,retain)IBOutlet UIButton *EntidadPicto9;
@property(nonatomic,retain)IBOutlet UIButton *EntidadPicto10;
@property(nonatomic,retain)IBOutlet UIButton *EntidadPicto11;
@property(nonatomic,retain)IBOutlet UIButton *EntidadPicto12;
@property(nonatomic,retain)IBOutlet UIButton *EntidadPicto13;

@property(nonatomic,retain)IBOutlet UIButton *FavoritoPicto1;
@property(nonatomic,retain)IBOutlet UIButton *FavoritoPicto2;
@property(nonatomic,retain)IBOutlet UIButton *FavoritoPicto3;
@property(nonatomic,retain)IBOutlet UIButton *FavoritoPicto4;
@property(nonatomic,retain)IBOutlet UIButton *FavoritoPicto5;
@property(nonatomic,retain)IBOutlet UIButton *FavoritoPicto6;
@property(nonatomic,retain)IBOutlet UIButton *FavoritoPicto7;
@property(nonatomic,retain)IBOutlet UIButton *FavoritoPicto8;
@property(nonatomic,retain)IBOutlet UIButton *FavoritoPicto9;
@property(nonatomic,retain)IBOutlet UIButton *FavoritoPicto10;
@property(nonatomic,retain)IBOutlet UIButton *FavoritoPicto11;
@property(nonatomic,retain)IBOutlet UIButton *FavoritoPicto12;
@property(nonatomic,retain)IBOutlet UIButton *FavoritoPicto13;
@property(nonatomic,retain)IBOutlet UIButton *FavoritoPicto14;

@property(nonatomic,retain)IBOutlet UILabel * Label1;
@property(nonatomic,retain)IBOutlet UILabel * Label2;
@property(nonatomic,retain)IBOutlet UILabel * Label3;
@property(nonatomic,retain)IBOutlet UILabel * Label4;
@property(nonatomic,retain)IBOutlet UILabel * Label5;
@property(nonatomic,retain)IBOutlet UILabel * Label6;
@property(nonatomic,retain)IBOutlet UILabel * Label7;
@property(nonatomic,retain)IBOutlet UILabel * Label8;
@property(nonatomic,retain)IBOutlet UILabel * Label9;
@property(nonatomic,retain)IBOutlet UILabel * Label10;
@property(nonatomic,retain)IBOutlet UILabel * Label11;
@property(nonatomic,retain)IBOutlet UILabel * Label12;
@property(nonatomic,retain)IBOutlet UILabel * Label13;
@property(nonatomic,retain)IBOutlet UILabel * Label14;

@property(nonatomic,retain)IBOutlet UILabel * Label15;
@property(nonatomic,retain)IBOutlet UILabel * Label16;
@property(nonatomic,retain)IBOutlet UILabel * Label17;
@property(nonatomic,retain)IBOutlet UILabel * Label18;
@property(nonatomic,retain)IBOutlet UILabel * Label19;
@property(nonatomic,retain)IBOutlet UILabel * Label20;
@property(nonatomic,retain)IBOutlet UILabel * Label21;
@property(nonatomic,retain)IBOutlet UILabel * Label22;
@property(nonatomic,retain)IBOutlet UILabel * Label23;
@property(nonatomic,retain)IBOutlet UILabel * Label24;
@property(nonatomic,retain)IBOutlet UILabel * Label25;
@property(nonatomic,retain)IBOutlet UILabel * Label26;
@property(nonatomic,retain)IBOutlet UILabel * Label27;
@property(nonatomic,retain)IBOutlet UILabel * Label28;
```

```
@property(nonatomic,retain)IBOutlet UILabel *EntidadTitle;
@property(nonatomic,retain)IBOutlet UILabel *EntidadFavoritoTitle;

@property(nonatomic,retain)IBOutlet UIButton *BotonEntidadSiguiente;
@property(nonatomic,retain)IBOutlet UIButton *BotonEntidadAnterior;

@property(nonatomic,retain)IBOutlet UIButton *BotonFavoritoSiguiente;
@property(nonatomic,retain)IBOutlet UIButton *BotonFavoritoAnterior;

@property(nonatomic,retain)IBOutlet UIButton *butVolver;


-(void) Ajustar;

-(void)reloadEntidadPics;
-(void)pasoPictosfavoritos;
-(void)reloadfavoritoPics;

-(IBAction)volverEntidadPictos:(id)sender;      // Volver a la pantalla anterior.

-(IBAction)onEntidad:(id)sender;              // Pulsación de la elección de entidad.
-(IBAction)onEntidadPicto:(id)sender;          // Pulsación del picto de la entidad seleccionada.
-(IBAction)onFavoritoPicto:(id)sender;    // Pulsación del picto configurado en la entidad "Favorito"

////////////////////////// ACCIONES DE NAVEGACIÓN

-(IBAction)onEntidadSiguiente:(id)sender;
-(IBAction)onEntidadAnterior:(id)sender;

-(IBAction)onFavoritoSiguiente:(id)sender;
-(IBAction)onFavoritoAnterior:(id)sender;

////////////////////////// ACCIONES DE EDICIÓN

-(IBAction)pulsarButAgregar:(id)sender;
-(IBAction)pulsarButQuitar:(id)sender;
-(IBAction)pulsarButGuardar:(id)sender;


@end



//
//  EditarAlbumFavorito.m
//  COMUNICADOR
//
//  Created by Elisabeth Esandia Melero on 07/06/11.
//  Copyright 2011 __MyCompanyName__. All rights reserved.
//

#import "EditarAlbumFavorito.h"
#import "Ajustes.h"

@implementation EditarAlbumFavorito

@synthesize ControladorAjustes;

@synthesize label_lugar;
@synthesize label_sublugar;
@synthesize label_personas;
@synthesize label_verbos;
@synthesize label_complementos;

@synthesize butAgregar;
@synthesize butQuitar;
@synthesize butGuardar;
```

```
@synthesize EntidadPicto0;
@synthesize EntidadPicto1;
@synthesize EntidadPicto2;
@synthesize EntidadPicto3;
@synthesize EntidadPicto4;
@synthesize EntidadPicto5;
@synthesize EntidadPicto6;
@synthesize EntidadPicto7;
@synthesize EntidadPicto8;
@synthesize EntidadPicto9;
@synthesize EntidadPicto10;
@synthesize EntidadPicto11;
@synthesize EntidadPicto12;
@synthesize EntidadPicto13;

@synthesize FavoritoPicto1;
@synthesize FavoritoPicto2;
@synthesize FavoritoPicto3;
@synthesize FavoritoPicto4;
@synthesize FavoritoPicto5;
@synthesize FavoritoPicto6;
@synthesize FavoritoPicto7;
@synthesize FavoritoPicto8;
@synthesize FavoritoPicto9;
@synthesize FavoritoPicto10;
@synthesize FavoritoPicto11;
@synthesize FavoritoPicto12;
@synthesize FavoritoPicto13;
@synthesize FavoritoPicto14;

@synthesize Label1;
@synthesize Label2;
@synthesize Label3;
@synthesize Label4;
@synthesize Label5;
@synthesize Label6;
@synthesize Label7;
@synthesize Label8;
@synthesize Label9;
@synthesize Label10;
@synthesize Label11;
@synthesize Label12;
@synthesize Label13;
@synthesize Label14;

@synthesize Label15;
@synthesize Label16;
@synthesize Label17;
@synthesize Label18;
@synthesize Label19;
@synthesize Label20;
@synthesize Label21;
@synthesize Label22;
@synthesize Label23;
@synthesize Label24;
@synthesize Label25;
@synthesize Label26;
@synthesize Label27;
@synthesize Label28;

@synthesize EntidadTitle;
@synthesize EntidadFavoritoTitle;

@synthesize BotonEntidadSiguiente;
@synthesize BotonEntidadAnterior;

@synthesize BotonFavoritoSiguiente;
@synthesize BotonFavoritoAnterior;
```

```
@synthesize butVolver;

- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{
    self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];
    if (self) {
        // Custom initialization
    }
    return self;
}

- (void)dealloc
{
    [objects_entidad release];
    [objectsCogidos_entidad release];
    //[request release];

    [Entidad_table release];
    [Favorito_table release];
    [EntidadLabel_table release];
    [FavoritoLabel_table release];

    [BotonEntidadAnterior release];
    [BotonFavoritoAnterior release];
    [BotonEntidadSiguiente release];
    [BotonFavoritoSiguiente release];

    //[VisorFotosVista release];
    [super dealloc];
}

- (void)didReceiveMemoryWarning
{
    // Releases the view if it doesn't have a superview.
    [super didReceiveMemoryWarning];

    // Release any cached data, images, etc that aren't in use.
}
- (void)viewDidLoad
{
    ControladorAjustes =[[Ajustes alloc]init];

    [self Ajustar];

    if (ajuste_tipo_letra<40) // IDIOMA CASTELLANO
    {
        [butVolver setTitle:@"VOLVER" forState:normal];
        label_lugar.text=@"LUGARES";
        label_sublugar.text=@"SUBLUGARES";
        label_personas.text=@"PERSONAS";
        label_verbos.text=@"VERBOS";
        label_complementos.text=@"COMPLEMENTOS";

        EntidadTitle.text=@"ENTIDAD COMPLETA";
        EntidadFavoritoTitle.text=@"ENTIDAD PERSONALIZADA";

        alertaSimplePersonalizada = [UIAlertView alloc];
        [alertaSimplePersonalizada initWithTitle:@"ALERTA"
                                         message:@"No hay pictos personalizados de esta entidad"
                                        delegate:self
                               cancelButtonTitle:@"Aceptar"
                               otherButtonTitles: nil];

        alertaSimple = [UIAlertView alloc];
        [alertaSimple initWithTitle:@"ALERTA"
                            message:@"No hay pictos asociados"
                           delegate:self
```

```
                cancelButtonTitle:@"Aceptar"
                otherButtonTitles: nil];

        alertaSimpleGuardar = [UIAlertView alloc];
        [alertaSimpleGuardar initWithTitle:@"ENTIDAD PERSONALIZADA GUARDADA"
                    message:@"Sus datos han sido guardados correctamente"
                   delegate:self
                cancelButtonTitle:@"Aceptar"
                otherButtonTitles: nil];
    }
    else //IDIOMA INGLÉS
    {
        [butVolver setTitle:@"BACK" forState:normal];
        label_lugar.text=@"PLACES";
        label_sublugar.text=@"SUB-PLACES";
        label_personas.text=@"PERSONS";
        label_verbos.text=@"VERBS";
        label_complementos.text=@"COMPLEMENTS";

        EntidadTitle.text=@"WHOLE ENTITY";
        EntidadFavoritoTitle.text=@"PERSONALIZED ENTITY";

        alertaSimplePersonalizada = [UIAlertView alloc];
        [alertaSimplePersonalizada initWithTitle:@"ALERT"
                                 message:@"There are not personalized pictos for this entity"
                                delegate:self
                       cancelButtonTitle:@"Ok"
                       otherButtonTitles: nil];

        alertaSimple = [UIAlertView alloc];
        [alertaSimple initWithTitle:@"ALERT"
                    message:@"There are not personalized pictos for this entity"
                   delegate:self
                cancelButtonTitle:@"Ok"
                otherButtonTitles: nil];

        alertaSimpleGuardar = [UIAlertView alloc];
        [alertaSimpleGuardar initWithTitle:@"EPERSONALIZED ENTITY SAVED"
                           message:@"Your data has been saved correctly"
                          delegate:self
                    cancelButtonTitle:@"Ok"
                    otherButtonTitles: nil];
    }

    objects_entidad = [[NSArray alloc] init]; // Inicializamos el array que hará la llamada de los objetos a la BD.
    objectsCogidos_entidad =[[NSMutableArray alloc]init];   // Inicializamos el array que almacenará los "objects_entidad"
para trabajar con ellos.

    objects_favoritos =[[NSArray alloc]init];
    objectsCogidos_favoritos=[[NSMutableArray alloc]init];

    // Inicializamos el array que leerá los elementos que se van seleccionando en cada página de cada entidad.
    selecc_originales =[[NSMutableArray alloc]init];
    // Inicializamos el array que leerá los elementos que se van seleccionando en cada página de cada entidad
personalizada.
    selecc_favoritos =[[NSMutableArray alloc] init];

    // Inicializamos el array que dispondrá de los elementos de cada entidad personalizada.
    datos_favoritos =[[NSMutableArray alloc]init];

    appDelegate = [[UIApplication sharedApplication] delegate]; // Parte de la BD
    context = [appDelegate managedObjectContext];   // Parte de la BD

    appDelegate_per = [[UIApplication sharedApplication] delegate]; // Parte de la BD
    context_per = [appDelegate_per managedObjectContext]; // Parte de la BD

    // Inicializamos la tabla de botones de entidad.
```

```
    Entidad_table = [[NSMutableArray alloc]  initWithObjects:EntidadPicto0, EntidadPicto1, EntidadPicto2, EntidadPicto3, EntidadPicto4, EntidadPicto5, EntidadPicto6, EntidadPicto7, EntidadPicto8, EntidadPicto9, EntidadPicto10, EntidadPicto11, EntidadPicto12, EntidadPicto13, nil];

    Favorito_table =[[NSMutableArray alloc] initWithObjects:FavoritoPicto1, FavoritoPicto2, FavoritoPicto3, FavoritoPicto4, FavoritoPicto5, FavoritoPicto6, FavoritoPicto7, FavoritoPicto8, FavoritoPicto9, FavoritoPicto10, FavoritoPicto11, FavoritoPicto12, FavoritoPicto13, FavoritoPicto14, nil];

    EntidadLabel_table =[[NSMutableArray alloc] initWithObjects:Label1, Label2, Label3, Label4, Label5,Label6, Label7, Label8, Label9, Label10, Label11, Label12, Label13, Label14, nil];

    FavoritoLabel_table =[[NSMutableArray alloc] initWithObjects:Label15, Label16, Label17, Label18, Label19, Label20, Label21, Label22, Label23, Label24, Label25, Label26, Label27, Label28, nil];


    for (int i=0; i<[Entidad_table count]; i++)
    {
        [[Entidad_table objectAtIndex:i] setHidden:YES];
    }
    for (int i=0; i<[Favorito_table count]; i++)
    {
        [[Favorito_table objectAtIndex:i] setHidden:YES];
    }

    for (int i=0; i<[EntidadLabel_table count]; i++)
    {
        [[EntidadLabel_table objectAtIndex:i] setText:@""];
    }

    for (int i=0; i<[FavoritoLabel_table count]; i++)
    {
        [[FavoritoLabel_table objectAtIndex:i] setText:@""];
    }

    num_pag_view = 0; // Establecemos el valor de la página inicial.
    num_pag_view_per =0;


    // Ocultamos e inhabilitamos los botones de navegación.
    [BotonEntidadSiguiente setEnabled:NO];
    [BotonEntidadAnterior setEnabled:NO];
    [BotonEntidadSiguiente setHidden:YES];
    [BotonEntidadAnterior setHidden:YES];

    ////////////////////
    // IF --> Si no hay fotos cargadas en Favorito--> evalúa
    ////////////////////
    [BotonFavoritoSiguiente setEnabled:NO];
    [BotonFavoritoAnterior setEnabled:NO];
    [BotonFavoritoSiguiente setHidden:YES];
    [BotonFavoritoAnterior setHidden:YES];
    ////////////////////


    //VisorFotosVista=[[VisorFotos alloc] initWithNibName:@"VisorFotos" bundle:[NSBundle mainBundle]]; //Eli

    [super viewDidLoad];
}

- (void)Ajustar
{
    [ControladorAjustes viewDidLoad];
    ajuste_resp_negativa=ControladorAjustes.resp_negativa;
    ajuste_tiempo_pulsacion=ControladorAjustes.tiempo_pulsacion;
    ajuste_tipo_letra=ControladorAjustes.tipo_letra;

    if (ajuste_tipo_letra==20){
        Idioma=@"nombre";
```

```
    }
    else if (ajuste_tipo_letra==30){
        Idioma=@"nombre_may";
    }
    else if (ajuste_tipo_letra==50){
        Idioma=@"nombre_ing";
    }
    else if (ajuste_tipo_letra==60){
        Idioma=@"nombre_may_ing";
    }

    NSLog(@"respuesta negativa %i, tiempo pulsacion %i, tipo letra
%i",ajuste_resp_negativa,ajuste_tiempo_pulsacion,ajuste_tipo_letra);
}

-(IBAction)onEntidad:(id)sender
{
    [selecc_originales removeAllObjects];
    [selecc_favoritos removeAllObjects];
    [datos_favoritos removeAllObjects];

    for (int i=0; i<[Favorito_table count]; i++)
    {
        [[Favorito_table objectAtIndex:i] setHidden:YES];
    }
    for (int i=0; i<[FavoritoLabel_table count]; i++)
    {
        [[FavoritoLabel_table objectAtIndex:i] setText:@""];
    }

    NSLog(@"Ahora hay %i objetos en datos favoritos", [datos_favoritos count]);

    [BotonEntidadAnterior setHidden:YES];
    [BotonEntidadAnterior setEnabled:NO];
    num_pag_view=0;
    NSInteger ButtonTagEntidad= [sender tag];

    dispatch_async(dispatch_get_main_queue(), ^
                   {
                       NSAutoreleasePool *pool = [[NSAutoreleasePool alloc] init];

                       NSLog(@"Estoy cargando Datos");

                       switch (ButtonTagEntidad)
                       {
                           case 1:
                           {
                               NSLog(@"Se está cargando el álbum LUGARES");
                               if (ajuste_tipo_letra<40) // IDIOMA CASTELLANO
                               {
                                   [EntidadTitle setText:@"LUGARES"];
                                   [EntidadFavoritoTitle setText:@"LUGARES PERSONALIZADOS"];
                               }
                               else // IDIOMA INGLÉS
                               {
                                   [EntidadTitle setText:@"PLACES"];
                                   [EntidadFavoritoTitle setText:@"PERSONALIZED PLACES"];
                               }
                               EntidadActual=1;

                               ////////////////////////////////////////////////////////
                               // LLAMADA A LA BD ENTIDAD ORIGINAL
                               ////////////////////////////////////////////////////////

                               entityDesc = [NSEntityDescription entityForName:@"Lugar" inManagedObjectContext:context];
                               request = [[NSFetchRequest alloc] init];
                               [request setEntity:entityDesc];
                               objects_entidad = [context executeFetchRequest:request error:&error];
```

```
    if ([objects_entidad count] == 0)
    {
        [alertaSimple show];
    }
    else
    {
        [objectsCogidos_entidad setArray:objects_entidad];
        [self reloadEntidadPics];
        [request release];
    }

    ////////////////////////////////////////////////////////
    // LLAMADA A LA BD ENTIDAD PERSONALIZADA
    ////////////////////////////////////////////////////////

    entityDesc_per=[NSEntityDescription entityForName:@"Lugar" inManagedObjectContext:context_per];
    request_favorito=[[NSFetchRequest alloc]init];
    [request_favorito setEntity:entityDesc_per];
    pred_favorito = [NSPredicate predicateWithFormat:@"ANY album_personalizado like '1'"];
    [request_favorito setPredicate:pred_favorito];
    objects_favoritos =[context_per executeFetchRequest:request_favorito error:&error_per];

    if ([objects_favoritos count] == 0)
    {
        [alertaSimplePersonalizada show];
    }
    else
    {
        [objectsCogidos_favoritos setArray:objects_favoritos];
        [self pasoPictosfavoritos];
        [request_favorito release];
    }
}
    break;
case 2:
{
    NSLog(@"Se está cargando el álbum SUBLUGARES");
    if (ajuste_tipo_letra<40) // IDIOMA CASTELLANO
    {
        [EntidadTitle setText:@"SUBLUGARES"];
        [EntidadFavoritoTitle setText:@"SUBLUGARES PERSONALIZADOS"];
    }
    else // IDIOMA INGLÉS
    {
        [EntidadTitle setText:@"SUB-PLACES"];
        [EntidadFavoritoTitle setText:@"PERSONALIZED SUB-PLACES"];
    }
    EntidadActual=2;

    ////////////////////////////////////////////////////////
    // LLAMADA A LA BD ENTIDAD ORIGINAL
    ////////////////////////////////////////////////////////

    entityDesc = [NSEntityDescription entityForName:@"Sublugar" inManagedObjectContext:context];
    request = [[NSFetchRequest alloc] init];
    [request setEntity:entityDesc];
    objects_entidad = [context executeFetchRequest:request error:&error];
    if ([objects_entidad count] == 0)
    {
        [alertaSimple show];
    }
    else
    {
        [objectsCogidos_entidad setArray:objects_entidad];
        [self reloadEntidadPics];
        [request release];
    }
```

```
                    ////////////////////////////////////////////////////////
                    // LLAMADA A LA BD ENTIDAD PERSONALIZADA
                    ////////////////////////////////////////////////////////

                    entityDesc_per=[NSEntityDescription entityForName:@"Sublugar"
inManagedObjectContext:context_per];
                    request_favorito=[[NSFetchRequest alloc]init];
                    [request_favorito setEntity:entityDesc_per];
                    pred_favorito = [NSPredicate predicateWithFormat:@"ANY album_personalizado like '1'"];
                    [request_favorito setPredicate:pred_favorito];
                    objects_favoritos =[context_per executeFetchRequest:request_favorito error:&error_per];

                    if ([objects_favoritos count] == 0)
                    {
                        [alertaSimplePersonalizada show];
                    }
                    else
                    {
                        [objectsCogidos_favoritos setArray:objects_favoritos];
                        [self pasoPictosfavoritos];
                        [request_favorito release];
                    }
                }
                    break;

                case 3:
                {
                    NSLog(@"Se está cargando el álbum PERSONAS");
                    if (ajuste_tipo_letra<40) // IDIOMA CASTELLANO
                    {
                        [EntidadTitle setText:@"PERSONAS"];
                        [EntidadFavoritoTitle setText:@"PERSONAS PERSONALIZADAS"];
                    }
                    else // IDIOMA INGLÉS
                    {
                        [EntidadTitle setText:@"PERSONS"];
                        [EntidadFavoritoTitle setText:@"PERSONALIZED PERSONS"];
                    }
                    EntidadActual=3;

                    ////////////////////////////////////////////////////////
                    // LLAMADA A LA BD ENTIDAD ORIGINAL
                    ////////////////////////////////////////////////////////

                    entityDesc = [NSEntityDescription entityForName:@"Personas" inManagedObjectContext:context];
                    request = [[NSFetchRequest alloc] init];
                    [request setEntity:entityDesc];
                    objects_entidad = [context executeFetchRequest:request error:&error];
                    if ([objects_entidad count] == 0)
                    {
                        [alertaSimple show];
                    }
                    else
                    {
                        [objectsCogidos_entidad setArray:objects_entidad];
                        [self reloadEntidadPics];
                        [request release];
                    }
                    ////////////////////////////////////////////////////////
                    // LLAMADA A LA BD ENTIDAD PERSONALIZADA
                    ////////////////////////////////////////////////////////

                    entityDesc_per=[NSEntityDescription entityForName:@"Personas"
inManagedObjectContext:context_per];
                    request_favorito=[[NSFetchRequest alloc]init];
                    [request_favorito setEntity:entityDesc_per];
                    pred_favorito = [NSPredicate predicateWithFormat:@"ANY album_personalizado like '1'"];
                    [request_favorito setPredicate:pred_favorito];
```

```
                objects_favoritos =[context_per executeFetchRequest:request_favorito error:&error_per];

                if ([objects_favoritos count] == 0)
                {
                    [alertaSimplePersonalizada show];
                }
                else
                {
                    [objectsCogidos_favoritos setArray:objects_favoritos];
                    [self pasoPictosfavoritos];
                    [request_favorito release];
                }
            }
                break;

            case 4:
            {
                NSLog(@"Se está cargando el álbum VERBOS");
                if (ajuste_tipo_letra<40) // IDIOMA CASTELLANO
                {
                    [EntidadTitle setText:@"VERBOS"];
                    [EntidadFavoritoTitle setText:@"VERBOS PERSONALIZADOS"];
                }
                else // IDIOMA INGLÉS
                {
                    [EntidadTitle setText:@"VERBS"];
                    [EntidadFavoritoTitle setText:@"PERSONALIZED VERBS"];
                }
                EntidadActual=4;

                ////////////////////////////////////////////////////////
                // LLAMADA A LA BD ENTIDAD ORIGINAL
                ////////////////////////////////////////////////////////

                entityDesc = [NSEntityDescription entityForName:@"Verbos" inManagedObjectContext:context];
                request = [[NSFetchRequest alloc] init];
                [request setEntity:entityDesc];
                objects_entidad = [context executeFetchRequest:request error:&error];
                if ([objects_entidad count] == 0)
                {
                    [alertaSimple show];
                }
                else
                {
                    [objectsCogidos_entidad setArray:objects_entidad];
                    [self reloadEntidadPics];
                    [request release];
                }
                ////////////////////////////////////////////////////////
                // LLAMADA A LA BD ENTIDAD PERSONALIZADA
                ////////////////////////////////////////////////////////

                entityDesc_per=[NSEntityDescription entityForName:@"Verbos"
inManagedObjectContext:context_per];
                request_favorito=[[NSFetchRequest alloc]init];
                [request_favorito setEntity:entityDesc_per];
                pred_favorito = [NSPredicate predicateWithFormat:@"ANY album_personalizado like '1'"];
                [request_favorito setPredicate:pred_favorito];
                objects_favoritos =[context_per executeFetchRequest:request_favorito error:&error_per];

                if ([objects_favoritos count] == 0)
                {
                    [alertaSimplePersonalizada show];
                }
                else
                {
                    [objectsCogidos_favoritos setArray:objects_favoritos];
                    [self pasoPictosfavoritos];
```

```
                    [request_favorito release];
                }
            }
                break;

            case 5:
            {
                NSLog(@"Se está cargando el álbum COMPLEMENTOS");
                if (ajuste_tipo_letra<40) // IDIOMA CASTELLANO
                {
                    [EntidadTitle setText:@"COMPLEMENTOS"];
                    [EntidadFavoritoTitle setText:@"COMPLEMENTOS PERSONALIZADOS"];
                }
                else // IDIOMA INGLÉS
                {
                    [EntidadTitle setText:@"COMPLEMENTS"];
                    [EntidadFavoritoTitle setText:@"PERSONALIZED COMPLEMENTS"];
                }
                EntidadActual=5;

                ////////////////////////////////////////////////////////
                // LLAMADA A LA BD ENTIDAD ORIGINAL
                ////////////////////////////////////////////////////////

                entityDesc = [NSEntityDescription entityForName:@"Complementos"
inManagedObjectContext:context];
                request = [[NSFetchRequest alloc] init];
                [request setEntity:entityDesc];
                objects_entidad = [context executeFetchRequest:request error:&error];
                if ([objects_entidad count] == 0)
                {
                    [alertaSimple show];
                }
                else
                {
                    [objectsCogidos_entidad setArray:objects_entidad];
                    [self reloadEntidadPics];
                    [request release];
                }
                ////////////////////////////////////////////////////////
                // LLAMADA A LA BD ENTIDAD PERSONALIZADA
                ////////////////////////////////////////////////////////

                entityDesc_per=[NSEntityDescription entityForName:@"Complementos"
inManagedObjectContext:context_per];
                request_favorito=[[NSFetchRequest alloc]init];
                [request_favorito setEntity:entityDesc_per];
                pred_favorito = [NSPredicate predicateWithFormat:@"ANY album_personalizado like '1'"];
                [request_favorito setPredicate:pred_favorito];
                objects_favoritos =[context_per executeFetchRequest:request_favorito error:&error_per];

                if ([objects_favoritos count] == 0)
                {
                    [alertaSimplePersonalizada show];
                }
                else
                {
                    [objectsCogidos_favoritos setArray:objects_favoritos];
                    [self pasoPictosfavoritos];
                    [request_favorito release];
                }
            }
                break;

            default:
                break;
        }
```

```
                [pool release];
            });
}

-(void)reloadEntidadPics
{
    indice=num_pag_view*[Entidad_table count];

    NSLog(@"indice= %i;num_pag_view= %i; thumbnail_table count= %i", indice, num_pag_view, [Entidad_table count]);
    NSLog(@"objetos cogidos %@", objectsCogidos_entidad);

    objects_entidad=nil;

    int contar2=[objectsCogidos_entidad count];
    NSLog(@"imagenes que tenemos %i ",contar2);

    for (int i=0; i<[Entidad_table count]; i++)
    {
        NSLog(@"i= %i; indice= %i",i,indice);
        if ((i+indice)<[objectsCogidos_entidad count])
        {
            [[Entidad_table objectAtIndex:i] setImage:[UIImage imageNamed:[[objectsCogidos_entidad objectAtIndex:(i+indice)]
valueForKey:@"nombre"]]forState:UIControlStateNormal]; //Colocar la imagen en cada botón

            [[EntidadLabel_table objectAtIndex:i] setText:[[objectsCogidos_entidad objectAtIndex:i+indice]
valueForKey:Idioma]];

            NSString *texto=[[objectsCogidos_entidad objectAtIndex:(i+indice)] valueForKey:@"nombre"];

            [[Entidad_table objectAtIndex:i] setHidden:NO]; //Hacer visibles todos los botones que tengan imagen.
            [[Entidad_table objectAtIndex:i]setAlpha:1];    //Poner la opacidad completa a todos

            if ([selecc_originales containsObject:texto]||[datos_favoritos containsObject:texto])
            {
                [[Entidad_table objectAtIndex:i]setAlpha:0.5];
            }
            //Poner la opacidad a la mitad de aquellos botones que ya estén registrados en la selección original.


            [BotonEntidadSiguiente setEnabled:YES];
            [BotonEntidadSiguiente setHidden:NO];
        }
        else
        {
            //Hide the button
            [[Entidad_table objectAtIndex:i] setHidden:YES];
            [[EntidadLabel_table objectAtIndex:i]setText:@""];
            [BotonEntidadSiguiente setEnabled:NO];
            [BotonEntidadSiguiente setHidden:YES];

            if (indice!=0)
            {
                [BotonEntidadAnterior setEnabled:YES];
                [BotonEntidadAnterior setHidden:NO];
            }
            else
            {
                NSLog(@"Estoy en la primera página!");
            }


        }
    }
}

-(void)pasoPictosfavoritos
{
    objects_favoritos=nil;
```

```
    for (int i=0; i<[objectsCogidos_favoritos count]; i++)
    {
        [datos_favoritos addObject:[[objectsCogidos_favoritos objectAtIndex:i] valueForKey:@"nombre"]];
    }

    [self reloadfavoritoPics];
}

-(void)reloadfavoritoPics
{
    indice_per=num_pag_view_per*[Favorito_table count];

    NSLog(@"indice_per= %i;num_pag_view_per= %i; Favorito_table count= %i", indice_per, num_pag_view_per,
[Favorito_table count]);

    for (int i=0; i<[Favorito_table count]; i++)
    {
        NSLog(@"i= %i; indice= %i",i,indice_per);

        ////////////////////////////////////////////////////
        // MOSTRAMOS LAS IMÁGENES DE LA LLAMADA A LA BD
        ////////////////////////////////////////////////////

        if ((i+indice_per)<[datos_favoritos count])
        {
            [[Favorito_table objectAtIndex:i] setImage:[UIImage imageNamed:[datos_favoritos objectAtIndex:(i+indice_per)]]
forState:UIControlStateNormal]; //Colocar la imagen en cada botón

            int indiceBDIdioma=[[objectsCogidos_entidad valueForKey:@"nombre"] indexOfObjectIdenticalTo:[datos_favoritos
objectAtIndex:i+indice_per]];
            [[FavoritoLabel_table objectAtIndex:i] setText:[[objectsCogidos_entidad objectAtIndex:indiceBDIdioma]
valueForKey:Idioma]];

            NSString *texto=[datos_favoritos objectAtIndex:(i+indice_per)];

            [[Favorito_table objectAtIndex:i] setHidden:NO]; //Hacer visibles todos los botones que tengan imagen.
            [[Favorito_table objectAtIndex:i]setAlpha:1];    //Poner la opacidad completa a todos

            if ([selecc_favoritos containsObject:texto]) //Poner la opacidad a la mitad de aquellos botones que ya estén
registrados en la selección original.
            {
                [[Favorito_table objectAtIndex:i]setAlpha:0.5];
            }

            [BotonFavoritoSiguiente setEnabled:YES];
            [BotonFavoritoSiguiente setHidden:NO];
        }

        ////////////////////////////////////////////////////
        // OCULTAMOS LOS BOTONES QUE SOBRAN
        ////////////////////////////////////////////////////

        else
        {
            //Hide the button
            [[Favorito_table objectAtIndex:i] setHidden:YES];
            [[FavoritoLabel_table objectAtIndex:i]setText:@""];
            [BotonFavoritoSiguiente setEnabled:NO];
            [BotonFavoritoSiguiente setHidden:YES];

            if (indice_per!=0)
            {
                [BotonFavoritoAnterior setEnabled:YES];
                [BotonFavoritoAnterior setHidden:NO];
            }
            else
            {
```

```
            NSLog(@"Estoy en la primera página de favorito!");
        }


    }
  }
}


-(IBAction)volverEntidadPictos:(id)sender
{
   for (int i=0; i<[Favorito_table count]; i++)
   {
      [[Favorito_table objectAtIndex:i] setHidden:YES];
      [[FavoritoLabel_table objectAtIndex:i]setText:@""];
   }

   for (int i=0; i<[Entidad_table count]; i++)
   {
      [[Entidad_table objectAtIndex:i] setHidden:YES];
      [[EntidadLabel_table objectAtIndex:i]setText:@""];
   }

   [BotonEntidadAnterior setHidden:YES];
   [BotonEntidadAnterior setEnabled:NO];
   [BotonEntidadSiguiente setHidden:YES];
   [BotonEntidadSiguiente setEnabled:NO];

   [EntidadTitle setText:@"ENTIDAD"];
   [self.view removeFromSuperview];

}

-(IBAction)onEntidadPicto:(id)sender
{
   NSInteger ButtonTagInteger= [sender tag];
   NSLog(@"Imagen seleccionada = %i",ButtonTagInteger);

   // quitar los pictos que ya estén.
   [selecc_originales removeObjectIdenticalTo:[[objectsCogidos_entidad objectAtIndex:(indice+ButtonTagInteger)]
valueForKey:@"nombre"]];

   if ([[Entidad_table objectAtIndex:ButtonTagInteger] alpha]==1) //picto no está seleccionado
   {
      NSString *nombre=[[objectsCogidos_entidad objectAtIndex:(indice+ButtonTagInteger)] valueForKey:@"nombre"];
      [selecc_originales addObject:nombre];
      [[Entidad_table objectAtIndex:ButtonTagInteger] setAlpha:0.5];
   }
   else //picto ya está seleccionado
   {
      [[Entidad_table objectAtIndex:ButtonTagInteger] setAlpha:1];
   }


   NSLog(@"seleccion originales = %@",selecc_originales);

}

-(IBAction)onFavoritoPicto:(id)sender
{
   NSInteger ButtonTagInteger= [sender tag];
   NSLog(@"Imagen seleccionada = %i",ButtonTagInteger);

   [selecc_favoritos removeObjectIdenticalTo:[datos_favoritos objectAtIndex:(indice_per+ButtonTagInteger)]];

   if ([[Favorito_table objectAtIndex:ButtonTagInteger] alpha]==1)
   {
      NSString *nombre=[datos_favoritos objectAtIndex:(indice_per+ButtonTagInteger)];
```

```
        [selecc_favoritos addObject:nombre];
        [[Favorito_table objectAtIndex:ButtonTagInteger] setAlpha:0.5];
    }
    else
    {
        [[Favorito_table objectAtIndex:ButtonTagInteger] setAlpha:1];
    }


    NSLog(@"seleccion selecc_favoritos = %@",selecc_favoritos);
}

-(IBAction)onEntidadSiguiente:(id)sender
{
    num_pag_view +=1;

    NSLog(@"A LA SIGUIENTE ENTIDAD!!--> %i",num_pag_view);

    [self reloadEntidadPics];

    [BotonEntidadAnterior setEnabled:YES];
    [BotonEntidadAnterior setHidden:NO];
}
-(IBAction)onEntidadAnterior:(id)sender
{
    NSLog(@"A LA ANTERIOR ENTIDAD!!");

    [BotonEntidadSiguiente setEnabled:YES];
    [BotonEntidadSiguiente setHidden:NO];

    if (num_pag_view>0) {
        num_pag_view -=1;
    }
    if(num_pag_view==0)
    {
        [BotonEntidadAnterior setEnabled:NO];
        [BotonEntidadAnterior setHidden:YES];
    }
    [self reloadEntidadPics];
}

-(IBAction)onFavoritoSiguiente:(id)sender
{
    num_pag_view_per +=1;

    NSLog(@"A LA SIGUIENTE ENTIDAD PERSONALIZADA!!--> %i",num_pag_view_per);

    [self reloadfavoritoPics];

    [BotonFavoritoAnterior setEnabled:YES];
    [BotonFavoritoAnterior setHidden:NO];
}
-(IBAction)onFavoritoAnterior:(id)sender
{
    NSLog(@"A LA ANTERIOR ENTIDAD PERSONALIZADA!!");

    [BotonFavoritoSiguiente setEnabled:YES];
    [BotonFavoritoSiguiente setHidden:NO];

    if (num_pag_view_per>0) {
        num_pag_view_per -=1;
    }
    if(num_pag_view_per==0)
    {
        [BotonFavoritoAnterior setEnabled:NO];
        [BotonFavoritoAnterior setHidden:YES];
    }
    [self reloadfavoritoPics];
```

```
}

-(IBAction)pulsarButAgregar:(id)sender
{
    for ( int i=0; i<[selecc_originales count]; i++)
    {
        if ([datos_favoritos containsObject:[selecc_originales objectAtIndex:i]])
        {

        }
        else
        {
            [datos_favoritos addObject:[selecc_originales objectAtIndex:i]];
            NSLog(@"datos_favoritos %@", datos_favoritos);
        }
    }
    [selecc_favoritos removeAllObjects];
    [self reloadfavoritoPics];

}
-(IBAction)pulsarButQuitar:(id)sender
{
    for ( int i=0; i<[selecc_favoritos count]; i++)
    {
        [datos_favoritos removeObjectIdenticalTo:[selecc_favoritos objectAtIndex:i]];
        //[datos_favoritos addObject:[selecc_originales objectAtIndex:i]];
        NSLog(@"datos_favoritos %@", datos_favoritos);

        [selecc_originales removeObjectIdenticalTo:[selecc_favoritos objectAtIndex:i]];
    }
    [selecc_favoritos removeAllObjects];
    [self reloadEntidadPics];
    [self reloadfavoritoPics];
}
-(IBAction)pulsarButGuardar:(id)sender
{
    NSString *textoEntidadActual=@"";

    switch (EntidadActual) {
        case 1:
            textoEntidadActual=@"Lugar";
            break;
        case 2:
            textoEntidadActual=@"Sublugar";
            break;
        case 3:
            textoEntidadActual=@"Personas";
            break;
        case 4:
            textoEntidadActual=@"Verbos";
            break;
        case 5:
            textoEntidadActual=@"Complementos";
            break;
        default:
            break;
    }

    COMUNICADORAppDelegate *appDelegate_guardar = [[UIApplication sharedApplication] delegate];
    NSManagedObjectContext *context_guardar = [appDelegate_guardar managedObjectContext];
    NSEntityDescription *entityDesc_guardar = [NSEntityDescription entityForName:[NSString
stringWithString:textoEntidadActual] inManagedObjectContext:context_guardar];
    NSFetchRequest *request_guardar = [[NSFetchRequest alloc] init];
    [request_guardar setEntity:entityDesc_guardar];
    NSError *error_guardar;
    NSArray *todos_objects=[context_guardar executeFetchRequest:request_guardar error:&error_guardar];
```

```
    // NSArray *todos_objects = [context executeFetchRequest:request error:&error];

    for ( int i=0; i<[todos_objects count]; i++)
    {
        if ([datos_favoritos containsObject:[[todos_objects objectAtIndex:i] valueForKey:@"nombre"]])
        {
            NSManagedObject *picto_favorito=[todos_objects objectAtIndex:i];
            [picto_favorito setValue:[NSNumber numberWithInt: 1] forKey:@"album_personalizado"];
        }
        else
        {
            NSManagedObject *picto_favorito=[todos_objects objectAtIndex:i];
            [picto_favorito setValue:[NSNumber numberWithInt: 0] forKey:@"album_personalizado"];
            NSLog(@"datos_favoritos %@", datos_favoritos);
        }
    }

    [request_guardar release];

    if (![context save:&error])
    {
        NSLog(@"error de coredata %@, %@", error, [error userInfo]);
        exit(-1);
    }

    [alertaSimpleGuardar show];
}

#pragma mark - View lifecycle

- (void)viewDidUnload
{
    self.EntidadPicto0=nil;
    self.EntidadPicto1=nil;
    self.EntidadPicto2=nil;
    self.EntidadPicto3=nil;
    self.EntidadPicto4=nil;
    self.EntidadPicto5=nil;
    self.EntidadPicto6=nil;
    self.EntidadPicto7=nil;
    self.EntidadPicto8=nil;
    self.EntidadPicto9=nil;
    self.EntidadPicto10=nil;
    self.EntidadPicto11=nil;
    self.EntidadPicto12=nil;
    self.EntidadPicto13=nil;

    self.FavoritoPicto1=nil;
    self.FavoritoPicto2=nil;
    self.FavoritoPicto3=nil;
    self.FavoritoPicto4=nil;
    self.FavoritoPicto5=nil;
    self.FavoritoPicto6=nil;
    self.FavoritoPicto7=nil;
    self.FavoritoPicto8=nil;
    self.FavoritoPicto9=nil;
    self.FavoritoPicto10=nil;
    self.FavoritoPicto11=nil;
    self.FavoritoPicto12=nil;
    self.FavoritoPicto13=nil;
    self.FavoritoPicto14=nil;

    self.Label1=nil;
    self.Label2=nil;
    self.Label3=nil;
    self.Label4=nil;
    self.Label5=nil;
    self.Label6=nil;
```

```
    self.Label7=nil;
    self.Label8=nil;
    self.Label9=nil;
    self.Label10=nil;
    self.Label11=nil;
    self.Label12=nil;
    self.Label13=nil;
    self.Label14=nil;
    self.Label15=nil;
    self.Label16=nil;
    self.Label17=nil;
    self.Label18=nil;
    self.Label19=nil;
    self.Label20=nil;
    self.Label21=nil;
    self.Label22=nil;
    self.Label23=nil;
    self.Label24=nil;
    self.Label25=nil;
    self.Label26=nil;
    self.Label27=nil;
    self.Label28=nil;

    self.BotonEntidadAnterior=nil;
    self.BotonEntidadSiguiente=nil;
    self.BotonFavoritoAnterior=nil;
    self.BotonFavoritoSiguiente=nil;

    [super viewDidUnload];
    // Release any retained subviews of the main view.
    // e.g. self.myOutlet = nil;
}

- (BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)interfaceOrientation
{
    // Return YES for supported orientations
        return YES;
}

@end
```

RELOJ (.h y .m):

```
//
//  Reloj.h
//  COMUNICADOR
//
//  Created by JOSE CRUZ PEREZ PI on 27/05/11.
//  Copyright 2011 UNIVERSIDAD PUBLICA DE NAVARRA. All rights reserved.
//

#import <UIKit/UIKit.h>
#import <QuartzCore/QuartzCore.h>
#import "COMUNICADORAppDelegate.h"


@class Ajustes;

@interface Reloj : UIViewController <UIPickerViewDelegate, UIPickerViewDataSource>{

    //Elementos usados para efectuar ajustes:
    Ajustes * ControladorAjustes;

    int ajuste_resp_negativa;
    int ajuste_tiempo_pulsacion;
    int ajuste_tipo_letra;
```

```
    //Array del picker con las actividades y etiquetas de los pickers.

        NSArray *arrayDatos;

        //IBOutlet UITextField *txt;
        IBOutlet UILabel *label;
        IBOutlet UILabel *label_crono;

    IBOutlet UIPickerView *picker; //picker actividades
        IBOutlet UIDatePicker *picker_crono;
        IBOutlet NSTimer *timer;
        int mainInt;
        int cuenta_atras;

        IBOutlet NSString *prueba;

    BOOL pulsacion_reloj;

    // ELEMENTOS PRIMERA PANTALLA
    IBOutlet UIView *VISTA_PRINCIPAL_RELOJ;
        IBOutlet UIImageView *VISTA_IMAGEN_RELOJ;
    IBOutlet UILabel *texto_imagen_primera;

    IBOutlet UIButton * but_reloj_ok;
    IBOutlet UIButton * but_reloj_inicio_primero;

    // ELEMENTOS SEGUNDA PANTALLA
    IBOutlet UIView *VISTA_SECUNDARIA_RELOJ;
        IBOutlet UIImageView *VISTA_SECUNDARIA_IMAGEN;
    IBOutlet UILabel *texto_imagen_segunda;

    IBOutlet UIButton * but_reloj_inicio_segundo;
    IBOutlet UIButton * but_reloj_volver;
    IBOutlet UIButton * but_reloj_empezar;
    IBOutlet UIButton * but_reloj_parar;
    IBOutlet UIButton * but_reloj_parar_oculto;

      //Caracol//

    IBOutlet UIImageView *reloj_caracol;
        IBOutlet UIView *mascara;
        IBOutlet UIView *animalillo;

      //Creo las capas del caracol

        CALayer *hoja;
        CALayer *babamask;
        CALayer *hierba;
        CALayer *caracol;
        CALayer *cabeza;
        CALayer *baba;

      //animaciones del caracol
        CABasicAnimation *cabezamove;
        CABasicAnimation *caracolmove;
        CABasicAnimation *maskmove;

}

@property (nonatomic, retain)IBOutlet UIButton * but_reloj_inicio_segundo;
@property (nonatomic, retain)IBOutlet UIButton * but_reloj_volver;
@property (nonatomic, retain)IBOutlet UIButton * but_reloj_empezar;
@property (nonatomic, retain)IBOutlet UIButton * but_reloj_parar;
@property (nonatomic, retain)IBOutlet UIButton * but_reloj_inicio_primero;
@property (nonatomic, retain)IBOutlet UIButton * but_reloj_ok;
@property (nonatomic, retain)IBOutlet UIButton * but_reloj_parar_oculto;
```

```
@property(nonatomic,retain)Ajustes * ControladorAjustes;

@property (nonatomic,retain) IBOutlet UIPickerView *picker;
@property (nonatomic,retain) IBOutlet UILabel *label;
@property (nonatomic, retain) IBOutlet NSTimer *timer;

- (void)Ajustar;

- (IBAction) PULSAR_VOLVER_RELOJ: (id) sender;

- (IBAction) PULSAR_VOLVER_OK_RELOJ: (id) sender;
- (IBAction) PULSAR_OK_RELOJ: (id) sender;
- (IBAction) PULSAR_VOLVER_OK_A_HOME: (id) sender;
- (IBAction) PULSAR_START: (id) sender;
- (IBAction) PULSAR_STOP: (id) sender;
- (IBAction) PULSAR_BOTON_OCULTO_PARAR: (id) sender;

@end


//
//  Reloj.m
//  COMUNICADOR
//
//  Created by JOSE CRUZ PEREZ PI on 27/05/11.
//  Copyright 2011 UNIVERSIDAD PUBLICA DE NAVARRA. All rights reserved.
//

#import "Reloj.h"
#import <AudioToolbox/AudioServices.h>
#import "COMUNICADORAppDelegate.h"

#import "Ajustes.h"


@implementation Reloj

@synthesize ControladorAjustes;

@synthesize picker;
@synthesize label;
@synthesize timer;

@synthesize but_reloj_inicio_segundo;
@synthesize but_reloj_volver;
@synthesize but_reloj_empezar;
@synthesize but_reloj_parar;
@synthesize but_reloj_inicio_primero;
@synthesize but_reloj_ok;
@synthesize but_reloj_parar_oculto;

- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{
    self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];
    if (self) {
        // Custom initialization
    }
    return self;
}

- (void)dealloc
{
    [super dealloc];
}

- (void)didReceiveMemoryWarning
{
    // Releases the view if it doesn't have a superview.
```

```
    [super didReceiveMemoryWarning];

    // Release any cached data, images, etc that aren't in use.
}

#pragma mark - View lifecycle

- (void)viewDidLoad
{
    ControladorAjustes =[[Ajustes alloc]init];
    [self Ajustar];

    if (ajuste_tipo_letra<40)
    {
        [but_reloj_inicio_primero setTitle:@"INICIO" forState:normal];
        [but_reloj_volver setTitle:@"VOLVER" forState:normal];
        [but_reloj_empezar setTitle:@"EMPEZAR" forState:normal];
        [but_reloj_parar setTitle:@"PARAR" forState:normal];
        [but_reloj_inicio_segundo setTitle:@"INICIO" forState:normal];

        arrayDatos  = [[NSArray alloc] initWithObjects:@"SELECCIONA ACTIVIDAD:",@"TRABAJAR", @"JUGAR",
@"COMER", @"BAILAR",  nil];
        label.text=@"SELECCIONA ACTIVIDAD:";
    }
    else
    {
        [but_reloj_inicio_primero setTitle:@"HOME" forState:normal];
        [but_reloj_volver setTitle:@"BACK" forState:normal];
        [but_reloj_empezar setTitle:@"START" forState:normal];
        [but_reloj_parar setTitle:@"STOP" forState:normal];
        [but_reloj_inicio_segundo setTitle:@"HOME" forState:normal];

        arrayDatos  = [[NSArray alloc] initWithObjects:@"CHOOSE ACTIVITY:",@"WORK", @"PLAY", @"EAT", @"DANCE",
nil];
        label.text=@"CHOOSE ACTIVITY:";
    }

    [picker reloadAllComponents];

          [picker_crono addTarget:self
                 action:@selector(changeDateInLabel:)
       forControlEvents:UIControlEventValueChanged];
        pulsacion_reloj=NO;

    [VISTA_IMAGEN_RELOJ setHidden:YES];
    [texto_imagen_primera setHidden:YES];

    [but_reloj_ok setHidden:YES];
    [but_reloj_ok setEnabled:NO];

    [super viewDidLoad];
    // Do any additional setup after loading the view from its nib.
}

- (void)Ajustar
{
    [ControladorAjustes viewDidLoad];
    ajuste_resp_negativa=ControladorAjustes.resp_negativa;
    ajuste_tiempo_pulsacion=ControladorAjustes.tiempo_pulsacion;
    ajuste_tipo_letra=ControladorAjustes.tipo_letra;

    NSLog(@"respuesta negativa %i, tiempo pulsacion %i, tipo letra
%i",ajuste_resp_negativa,ajuste_tiempo_pulsacion,ajuste_tipo_letra);
}


- (void)viewDidUnload
{
```

```
    [super viewDidUnload];
    // Release any retained subviews of the main view.
    // e.g. self.myOutlet = nil;
}

- (BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)interfaceOrientation
{
    // Return YES for supported orientations
        return YES;
}

#pragma mark - PickerView Delegate

- (NSInteger)pickerView:(UIPickerView *)pickerView numberOfRowsInComponent:(NSInteger)component
{
        return [arrayDatos count];
}

- (NSInteger)numberOfComponentsInPickerView:(UIPickerView *)pickerView
{
        return 1;
}

- (NSString *)pickerView:(UIPickerView *)pickerView titleForRow:(NSInteger)row forComponent:(NSInteger)component
{
        NSString *returnStr;
        if (component == 0 )
        {
                returnStr = [arrayDatos objectAtIndex:row];
    }

        return returnStr;

}


//picker de actividades.

- (void)pickerView:(UIPickerView *)pickerView didSelectRow:(NSInteger)row inComponent:(NSInteger)component{
        //[[[UIApplication sharedApplication] delegate] cargarVista:row];
        label.text = [arrayDatos objectAtIndex:row];
        prueba=@"00:00:00";

    /*
 if ((label.text==@"OTRO")||(label.text==@"OTHER"))
    {
         txt.hidden=NO;
                [VISTA_IMAGEN_RELOJ addSubview:VISTA_ACCION_OTRO];
         [txt becomeFirstResponder];
        }
 else
    {
         txt.hidden=YES;
                [txt resignFirstResponder];
        }
    */
        if (((label.text==@"SELECCIONA ACTIVIDAD:")||(label.text==@"CHOOSE ACTIVITY:")))
    {
       [VISTA_IMAGEN_RELOJ setHidden:YES];
       [texto_imagen_primera setHidden:YES];

                [      but_reloj_ok setHidden:YES];
       [but_reloj_ok setEnabled:NO];
        }
        else
    {
       [VISTA_IMAGEN_RELOJ setHidden:NO];
       [texto_imagen_primera setHidden:NO];
```

```
    if ([label_crono.text isEqualToString:prueba])
    {
        [but_reloj_ok setHidden:YES];
        [but_reloj_ok setEnabled:NO];
    }
    else
    {
        [but_reloj_ok setHidden:NO];
        [but_reloj_ok setEnabled:YES];
    }
        }

        //En función de lo seleccionado en el picker, saco una imagen u otra
        if ((label.text==@"TRABAJAR")||(label.text==@"WORK"))
{
[VISTA_IMAGEN_RELOJ setImage:[UIImage imageNamed:@"trabajar"]];
    if (ajuste_tipo_letra<40)
    {
        [texto_imagen_primera setText:@"trabajar"];
    }
    else
    {
        [texto_imagen_primera setText:@"work"];
    }
        }
        else if ((label.text==@"JUGAR")||(label.text==@"PLAY"))
{
    [VISTA_IMAGEN_RELOJ setImage:[UIImage imageNamed:@"jugar"]];
    if (ajuste_tipo_letra<40)
    {
        [texto_imagen_primera setText:@"jugar"];
    }
    else
    {
        [texto_imagen_primera setText:@"play"];
    }
        }
        else if ((label.text==@"COMER")||(label.text==@"EAT"))
{
    [VISTA_IMAGEN_RELOJ setImage:[UIImage imageNamed:@"comer"]];
    if (ajuste_tipo_letra<40)
    {
        [texto_imagen_primera setText:@"comer"];
    }
    else
    {
        [texto_imagen_primera setText:@"eat"];
    }
}
        else if ((label.text==@"BAILAR")||(label.text==@"DANCE"))
{
    [VISTA_IMAGEN_RELOJ setImage:[UIImage imageNamed:@"bailar"]];
    if (ajuste_tipo_letra<40)
    {
        [texto_imagen_primera setText:@"bailar"];
    }
    else
    {
        [texto_imagen_primera setText:@"dance"];
    }
}
 /*
else if ((label.text==@"OTRO")||(label.text==@"OTRO"))
 {
                    [VISTA_IMAGEN_RELOJ addSubview:VISTA_ACCION_OTRO];
        }
 */
```

```
}

//Actualizar etiqueta label_crono en funcion del picker_crono

- (void)changeDateInLabel:(id)sender{
        //Use NSDateFormatter to write out the date in a friendly format
        NSDateFormatter *df = [[NSDateFormatter alloc] init];
 df.dateStyle =          NSDateFormatterMediumStyle;
 [df          setDateFormat:@"HH:mm:ss"];
        label_crono.text = [NSString stringWithFormat:@"%@",
                [df stringFromDate:picker_crono.date]];
 [df          release];

   if (((label.text==@"SELECCIONA ACTIVIDAD:")||(label.text==@"CHOOSE ACTIVITY:")))
   {
      [VISTA_IMAGEN_RELOJ setHidden:YES];
      [texto_imagen_primera setHidden:YES];

                  [     but_reloj_ok setHidden:YES];
      [but_reloj_ok setEnabled:NO];
         }
         else
   {
      [VISTA_IMAGEN_RELOJ setHidden:NO];
      [texto_imagen_primera setHidden:NO];

      if ([label_crono.text isEqualToString:prueba])
      {
         [but_reloj_ok setHidden:YES];
         [but_reloj_ok setEnabled:NO];
      }
      else
      {
         [but_reloj_ok setHidden:NO];
         [but_reloj_ok setEnabled:YES];
      }
         }
}

-(IBAction) PULSAR_OK_RELOJ: (id) sender
{
        [VISTA_PRINCIPAL_RELOJ addSubview:VISTA_SECUNDARIA_RELOJ];

   [but_reloj_volver setHidden:NO];
   [but_reloj_volver setEnabled:YES];

   [but_reloj_empezar setHidden:NO];
   [but_reloj_empezar setEnabled:YES];

   [but_reloj_inicio_segundo setHidden:NO];
   [but_reloj_inicio_segundo setEnabled:YES];

        [VISTA_SECUNDARIA_RELOJ addSubview:reloj_caracol];
        [VISTA_SECUNDARIA_RELOJ addSubview:mascara];
        [VISTA_SECUNDARIA_RELOJ addSubview:animalillo];

   if ((label.text==@"TRABAJAR")||(label.text==@"WORK"))
   {
      [VISTA_SECUNDARIA_IMAGEN setImage:[UIImage imageNamed:@"trabajar"]];
      if (ajuste_tipo_letra<40)
      {
         [texto_imagen_segunda setText:@"TRABAJAR"];
      }
      else
      {
         [texto_imagen_segunda setText:@"WORK"];
      }
         }
```

```
        else if ((label.text==@"JUGAR")||(label.text==@"PLAY"))
  {
     [VISTA_SECUNDARIA_IMAGEN setImage:[UIImage imageNamed:@"jugar"]];
     if (ajuste_tipo_letra<40)
     {
        [texto_imagen_segunda setText:@"JUGAR"];
     }
     else
     {
        [texto_imagen_segunda setText:@"PLAY"];
     }
        }
        else if ((label.text==@"COMER")||(label.text==@"EAT"))
  {
     [VISTA_SECUNDARIA_IMAGEN setImage:[UIImage imageNamed:@"comer"]];
     if (ajuste_tipo_letra<40)
     {
        [texto_imagen_segunda setText:@"COMER"];
     }
     else
     {
        [texto_imagen_segunda setText:@"EAT"];
     }
        }
        else if ((label.text==@"BAILAR")||(label.text==@"DANCE"))
  {
     [VISTA_SECUNDARIA_IMAGEN setImage:[UIImage imageNamed:@"bailar"]];
     if (ajuste_tipo_letra<40)
     {
        [texto_imagen_segunda setText:@"BAILAR"];
     }
     else
     {
        [texto_imagen_segunda setText:@"DANCE"];
     }
        }

  /*
 if (txt.hidden == NO)
  {
         label.text=txt.text;
                  [VISTA_SECUNDARIA_IMAGEN addSubview:VISTA_ACCION_OTRO];
        }
*/

        //Coloco las capas del caracol
        hierba=[CALayer layer];
        hierba=reloj_caracol.layer;  // "hierba" es una CALayer que va insertada dentro del UIView "reloj".
        hierba.bounds=CGRectMake(0.0f, 0.0f, 900.0f, 95.0f); // los límites de la imagen que contenga, en el origen 0,0
con respecto al total de pantalla.
        hierba.anchorPoint=CGPointMake(0.5f, 0.0f); // el punto de referencia de la capa se sitúa en el 0,5; 0,0 de la capa
en la que está contenida.
        hierba.contents= (id)[[UIImage imageNamed:@"hierba.png"] CGImage]; // Esta CALayer contiene la imagen
"hierba.png"

        hoja=[CALayer layer];
        hoja.bounds=CGRectMake(0.0f, 0.0f, 134.0f, 146.0f);
        hoja.position=CGPointMake(820.0f, 30.0f); // Se señala la posición con respecto a la capa que va a ocupar.
(sublayer de "hierba")
        hoja.contents= (id)[[UIImage imageNamed:@"hoja_reloj.png"] CGImage];

        [hierba addSublayer:hoja];

        baba=[CALayer layer]; // "baba" es una CALayer que va insertada dentro del UIView "mascara",
        baba=mascara.layer;   // puesto que va a tener un efecto así, y no queremos que el efecto se aplique a otros
UIViews.(como "hierba")
        baba.bounds=CGRectMake(0.0f, 0.0f, 900.0f, 95.0f);
        baba.anchorPoint=CGPointMake(0.5f, 0.4f);
```

```
        baba.contents= (id)[[UIImage imageNamed:@"baba.png"] CGImage]; // Contiene la imagen propia de la baba.

        babamask=[CALayer layer]; // "babamask" es una CALayer que va a hacer el efecto de máscara sobre la CALayer
que la coloquemos.
        // La imagen que contenga, será la forma del hueco que dejará ver.
        babamask.bounds=CGRectMake(0.0f, 0.0f, 150.0f, 95.0f);
        babamask.anchorPoint=CGPointMake(0.0f, 0.0f);
        babamask.contents=(id)[[UIImage imageNamed:@"mascara.png"] CGImage]; // contiente la imagen "mascara.png"
consistente en un rectángulo.

        [baba setMask:babamask]; // Se aplica la máscara sólamente a la capa deseada.

        caracol=[CALayer layer];
        caracol=animalillo.layer;
        caracol.bounds=CGRectMake(0.0f, 0.0f, 200.0f, 174.0f);
        caracol.anchorPoint=CGPointMake(2.3f, 0.25f);
        caracol.contents= (id)[[UIImage imageNamed:@"caracolillo.png"] CGImage];

        cabeza=[CALayer layer];
        cabeza.bounds=CGRectMake(0.0f, 0.0f, 200.0f, 174.0f);
        cabeza.position=CGPointMake(0.0f, 174.0f);
        cabeza.anchorPoint=CGPointMake(0.0f, 1.0f);
        cabeza.contents= (id)[[UIImage imageNamed:@"cabeza_reloj.png"] CGImage];

        [caracol addSublayer:cabeza];
}

- (IBAction) PULSAR_START: (id) sender;
{
        mainInt=[picker_crono countDownDuration];
        cuenta_atras=[picker_crono countDownDuration];
        timer=[NSTimer scheduledTimerWithTimeInterval:(1.0/1.0) target:self selector:@selector(updateLabel) userInfo:nil
repeats:YES];

    [self Ajustar];

    if (ajuste_tipo_letra<40)
    {
        [but_reloj_empezar setTitle:@"EMPEZAR" forState:normal];
    }
    else
    {
        [but_reloj_empezar setTitle:@"START" forState:normal];
    }

    [but_reloj_empezar setHidden:YES];
    [but_reloj_empezar setEnabled:NO];

    [but_reloj_volver setHidden:YES];
    [but_reloj_volver setEnabled:NO];

    [but_reloj_inicio_segundo setHidden:YES];
    [but_reloj_inicio_segundo setEnabled:NO];

    [but_reloj_parar_oculto setHidden:NO];
    [but_reloj_parar_oculto setEnabled:YES];

    [but_reloj_parar setHidden:YES];
    [but_reloj_parar setEnabled:NO];

        //Creamos la animacion del caracol
 caracolmove=[CABasicAnimation animationWithKeyPath:        @"anchorPoint"];
        caracolmove.fromValue=[NSValue valueWithCGPoint: CGPointMake(2.3, 0.25)];
        caracolmove.toValue=[NSValue valueWithCGPoint: CGPointMake(-0.8, 0.25)];
 caracolmove.duration=cuenta_atras;
        caracolmove.fillMode = kCAFillModeForwards; //estas dos líneas hacen que la animación se mantenga en su
posición final.
        caracolmove.removedOnCompletion = NO;
```

```
 maskmove=[CABasicAnimation animationWithKeyPath:          @"bounds"];
          maskmove.fromValue=[NSNumber valueWithCGRect: CGRectMake(0.0f, 0.0f, 150.0f, 95.0f)];
          maskmove.toValue=[NSNumber valueWithCGRect: CGRectMake(0.0f, 0.0f, 730.0f, 95.0f)];
 maskmove.duration=cuenta_atras;
          maskmove.fillMode = kCAFillModeForwards;
          maskmove.removedOnCompletion=NO;


          // Añadimos la animación a la capa correspondiente y así la inicializamos:

 babamask.speed=          1.0; //Activamos la velocidad en caso de que haya sido desactivada.
 caracol.speed=          1.0;

 [caracol addAnimation:caracolmove forKey:          @"caracolmove"];
 [babamask addAnimation:maskmove forKey:          @"maskmove"];
}

- (IBAction) PULSAR_STOP: (id) sender;
{
 [timer invalidate];

          [animalillo removeFromSuperview];
          [mascara removeFromSuperview];

   [self Ajustar];

   if (ajuste_tipo_letra<40)
   {
      [but_reloj_parar setTitle:@"PARAR" forState:normal];
   }
   else
   {
      [but_reloj_parar setTitle:@"STOP" forState:normal];
   }

   [but_reloj_parar setHidden:YES];
   [but_reloj_parar setEnabled:NO];

   [but_reloj_parar_oculto setHidden:YES];
   [but_reloj_parar_oculto setEnabled:NO];

   [but_reloj_inicio_segundo setHidden:NO];
   [but_reloj_inicio_segundo setEnabled:YES];

}

//Actualizar label_crono y funcionamiento de reloj.

-(void) updateLabel
{

   // CUANDO EL LABEL LLEGA A CERO...

 mainInt = mainInt -          1;
          if (mainInt == 0.0) {

           [timer invalidate];

       [but_reloj_parar_oculto setHidden:YES];
       [but_reloj_parar_oculto setEnabled:NO];

       [but_reloj_parar setHidden:YES];
       [but_reloj_parar setEnabled:NO];

       [but_reloj_inicio_segundo setHidden:NO];
       [but_reloj_inicio_segundo setEnabled:YES];
```

```
            cabezamove=[CABasicAnimation animationWithKeyPath:              @"transform.rotation.z"];
                        cabezamove.fromValue = [NSNumber numberWithFloat:0];
                        cabezamove.toValue = [NSNumber numberWithFloat:0.05];
            cabezamove.duration=            1;
                        cabezamove.autoreverses=YES;
                        cabezamove.repeatCount=2;

            [cabeza addAnimation:cabezamove forKey:             @"cabezamove"];
            hoja.contents= (            id)[[UIImage imageNamed:@"hoja_mordida.png"] CGImage];

                        //Hacemos que suene el mordisco
                        SystemSoundID mordisco;
                        AudioServicesCreateSystemSoundID((CFURLRef)[NSURL fileURLWithPath:[[NSBundle mainBundle]
pathForResource:@"mordisco" ofType:@"wav"]], &mordisco);
                        AudioServicesPlaySystemSound (mordisco);
            }


            NSDate *nuevaFecha = [NSDate dateWithTimeIntervalSinceReferenceDate:mainInt-3600];
 NSDateFormatter *df = [[NSDateFormatter alloc] init];
 [df setDateFormat:          @"HH:mm:ss"];
            label_crono.text =  [df stringFromDate:nuevaFecha];
            [df release];
}

-(IBAction) PULSAR_VOLVER_RELOJ: (id) sender
{
    [self Ajustar];

    if (ajuste_tipo_letra<40)
    {
        [but_reloj_volver setTitle:@"VOLVER" forState:normal];
        [but_reloj_inicio_primero setTitle:@"INICIO" forState:normal];

        label.text=@"SELECCIONA ACTIVIDAD:";
    }
    else
    {
        [but_reloj_volver setTitle:@"BACK" forState:normal];
        [but_reloj_inicio_primero setTitle:@"HOME" forState:normal];

        label.text=@"SELECT ONE ACTIVITY:";
    }

            [self.view removeFromSuperview];

    //txt.hidden=YES;
    //txt.text=@"";
 [picker selectRow:          0 inComponent:0 animated:NO];
            picker_crono.date=[NSDate distantPast];

 NSDateFormatter *format = [[NSDateFormatter alloc] init];
 [format setDateFormat:          @"hh:mm:ss"];
            NSString *prod=@"00:00:00";
            NSDate *myDate = [format dateFromString: prod];
            picker_crono.date=myDate;

 NSDateFormatter *df = [[NSDateFormatter alloc] init];
 df.dateStyle = NSDateFormatterMediumStyle;
 [df setDateFormat:          @"HH:mm:ss"];
            label_crono.text = [NSString stringWithFormat:@"%@",
                                                                [df stringFromDate:picker_crono.date]];
            [df release];

}

-(IBAction) PULSAR_VOLVER_OK_RELOJ: (id) sender
{
```

```
    [but_reloj_parar setHidden:YES];
    [but_reloj_parar setEnabled:NO];

            [VISTA_SECUNDARIA_RELOJ removeFromSuperview];

    /*
  else if ((label.text==@"OTRO")||(label.text==@"OTRO"))
    {
                        [VISTA_IMAGEN_RELOJ addSubview:VISTA_ACCION_OTRO];
            }
     */

}

-(IBAction) PULSAR_VOLVER_OK_A_HOME: (id) sender
{
    [self Ajustar];

    if (ajuste_tipo_letra<40) // IDIOMA CASTELLANO
    {
        label.text=@"SELECCIONA ACTIVIDAD:";
        [but_reloj_inicio_segundo setTitle:@"INICIO" forState:normal];
    }
    else //IDIOMA INGLÉS
    {
        label.text=@"SELECT ONE ACTIVITY:";
        [but_reloj_inicio_segundo setTitle:@"HOME" forState:normal];
    }

    [VISTA_SECUNDARIA_RELOJ removeFromSuperview];
    [self.view removeFromSuperview];

  babamask.speed=          0.0; //Inhabilitamos la velocidad
  caracol.speed=          0.0;
            [cabeza removeFromSuperlayer];
            [hoja removeFromSuperlayer];

            //txt.text=@"";
            //txt.hidden=YES;

    [but_reloj_ok setHidden:YES];
    [but_reloj_ok setEnabled:NO];

  [picker selectRow:          0 inComponent:0 animated:NO];

  NSDateFormatter *format = [[NSDateFormatter alloc] init];
  [format setDateFormat:          @"hh:mm:ss"];
            NSString *prod=@"00:00:00";
            NSDate *myDate = [format dateFromString: prod];
            picker_crono.date=myDate;

  NSDateFormatter *df = [[NSDateFormatter alloc] init];
  df.dateStyle = NSDateFormatterMediumStyle;
  [df setDateFormat:          @"HH:mm:ss"];
            label_crono.text = [NSString stringWithFormat:@"%@",
                                                                [df stringFromDate:picker_crono.date]];
            [df release];
}

- (IBAction) PULSAR_BOTON_OCULTO_PARAR: (id) sender
{
            if (pulsacion_reloj == NO)
            {
                        pulsacion_reloj = YES;

        [but_reloj_parar setHidden:NO];
        [but_reloj_parar setEnabled:YES];
```

```
        [but_reloj_empezar setHidden:YES];
        [but_reloj_empezar setEnabled:NO];
            }
            else
            {
                        pulsacion_reloj = NO;

        [but_reloj_parar setHidden:YES];
        [but_reloj_parar setEnabled:NO];

        [but_reloj_empezar setHidden:NO];
        [but_reloj_empezar setEnabled:YES];

            }
}


@end
```

ÁLBUM:

SeleccionAlbum (.h y .m):

```
//
//  seleccionAlbum.h
//  COMUNICADOR
//
//  Created by Elisabeth Esandia Melero on 29/03/11.
//  Copyright 2011 __MyCompanyName__. All rights reserved.
//

#import <UIKit/UIKit.h>


@class cargarFotos;
@class seleccionEntidadPictos;
@class Ajustes;

@interface seleccionAlbum : UIViewController {

        cargarFotos * cargarFotosVista; //creamos una instancia de cargarFotos de la que utilizaremos su vista.
        seleccionEntidadPictos * EntidadPictosVista; //creamos una instancia de seleccionEntidadPictos de la que
utilizaremos su vista.

    ////////////////////////////
    //      AJUSTES           //
    Ajustes * ControladorAjustes;

    int ajuste_resp_negativa;
    int ajuste_tiempo_pulsacion;
    int ajuste_tipo_letra;

    NSTimer *pulsacionTimer;

    ////////////////////////////

    IBOutlet UIButton *butPredeterminado;
    IBOutlet UIButton *butVolver;

}

@property (nonatomic, retain) Ajustes *ControladorAjustes;
@property(nonatomic,retain)cargarFotos *cargarFotosVista;
@property(nonatomic,retain)seleccionEntidadPictos *EntidadPictosVista;
```

```
@property(nonatomic,retain)IBOutlet UIButton *butPredeterminado;
@property(nonatomic,retain)IBOutlet UIButton *butVolver;

@property (nonatomic, retain) NSTimer *pulsacionTimer;

-(void) Ajustar;

-(IBAction)touchDown:(id)sender;
-(IBAction)touchUp:(id)sender;
-(void)showVista;

-(IBAction)volverPpal:(id)sender;

@end


    //
//  seleccionAlbum.m
//  COMUNICADOR
//
//  Created by Elisabeth Esandia Melero on 29/03/11.
//  Copyright 2011 __MyCompanyName__. All rights reserved.
//

#import "seleccionAlbum.h"
#import "cargarFotos.h"
#import "seleccionEntidadPictos.h"
#import "Ajustes.h"

@implementation seleccionAlbum

@synthesize cargarFotosVista;
@synthesize EntidadPictosVista;
@synthesize ControladorAjustes;

@synthesize pulsacionTimer;

@synthesize butPredeterminado;
@synthesize butVolver;

/*
 // The designated initializer.  Override if you create the controller programmatically and want to perform customization that
is not appropriate for viewDidLoad.
- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil {
    if ((self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil])) {
        // Custom initialization
                }
    return self;
}
*/


// Implement viewDidLoad to do additional setup after loading the view, typically from a nib.
- (void)viewDidLoad {
    ControladorAjustes =[[Ajustes alloc]init];
    cargarFotosVista=[[cargarFotos alloc] initWithNibName:@"cargarFotos" bundle:Nil];
    EntidadPictosVista=[[seleccionEntidadPictos alloc] initWithNibName:@"seleccionEntidadPictos" bundle:Nil]; //Eli


    [self Ajustar];

    if (ajuste_tipo_letra<40) // IDIOMA CASTELLANO
    {
        [butVolver setTitle:@"INICIO" forState:normal];
    }
    else //IDIOMA INGLÉS
    {
```

```
        [butVolver setTitle:@"HOME" forState:normal];
    }

            //cargarFotosVista=[[cargarFotos alloc] initWithNibName:@"cargarFotos" bundle:[NSBundle mainBundle]]; //Eli
            //EntidadPictosVista=[[seleccionEntidadPictos alloc] initWithNibName:@"seleccionEntidadPictos" bundle:
[NSBundle mainBundle]]; //Eli
    //EditarAlbumFavoritoVista=[[EditarAlbumFavorito alloc] initWithNibName:@"EditarAlbumFavorito" bundle:[NSBundle
mainBundle]];

    [super viewDidLoad];
}

- (void)Ajustar
{
    [ControladorAjustes viewDidLoad];
    ajuste_resp_negativa=ControladorAjustes.resp_negativa;
    ajuste_tiempo_pulsacion=ControladorAjustes.tiempo_pulsacion;
    ajuste_tipo_letra=ControladorAjustes.tipo_letra;

    NSLog(@"respuesta negativa %i, tiempo pulsacion %i, tipo letra
%i",ajuste_resp_negativa,ajuste_tiempo_pulsacion,ajuste_tipo_letra);
}

-(IBAction)touchDown:(id)sender
{
            self.pulsacionTimer = [NSTimer scheduledTimerWithTimeInterval:ajuste_tiempo_pulsacion target:self
selector:@selector(showVista) userInfo:sender repeats:NO];
}

-(IBAction)touchUp:(id)sender
{
            [pulsacionTimer invalidate];
}

-(void)showVista
{
    switch([pulsacionTimer.userInfo  tag]){

            /*
             Caso 0: Fotos & pizarra
             Caso 1: Pictos
             Caso 2: Album Favorito
             */

                        case 0:
            [cargarFotosVista viewDidLoad];
            [cargarFotosVista cargarDatosEnArray:0];
            [self.view addSubview:cargarFotosVista.view];
            NSLog(@"Estoy cambiando la vista");
                                    break;

                        case 1:
            [EntidadPictosVista onVerPersonalizado:NO];
            [EntidadPictosVista viewDidLoad];
            [self.view addSubview:EntidadPictosVista.view];
            break;
        case 2:
            [EntidadPictosVista onVerPersonalizado:YES];
            [EntidadPictosVista viewDidLoad];
            [self.view addSubview:EntidadPictosVista.view];
            break;
        default:
                                    break;
            }
}

-(IBAction)volverPpal:(id)sender
{
```

```
        [self.view removeFromSuperview];
}

- (BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)interfaceOrientation {
    // Overriden to allow any orientation.
    return YES;
}

# pragma mark -
# pragma mark Memory Management

- (void)didReceiveMemoryWarning {
    // Releases the view if it doesn't have a superview.
    [super didReceiveMemoryWarning];

    // Release any cached data, images, etc that aren't in use.
}


- (void)viewDidUnload {
    self.cargarFotosVista=nil;
    [super viewDidUnload];
    // Release any retained subviews of the main view.
    // e.g. self.myOutlet = nil;
}


- (void)dealloc {
    [cargarFotosVista release];
    [EntidadPictosVista release];
    [super dealloc];
}


@end
```

SeleccionEntidadPictos (.h y .m):

```
//
//  seleccionEntidadPictos.h
//  COMUNICADOR
//
//  Created by Elisabeth Esandia Melero on 26/05/11.
//  Copyright 2011 __MyCompanyName__. All rights reserved.
//

#import <UIKit/UIKit.h>

@class cargarFotos;
@class Ajustes;

@interface seleccionEntidadPictos : UIViewController {

        cargarFotos * cargarFotosVista; //creamos una instancia de cargarFotos de la que utilizaremos su vista.

    ////////////////////////////
    //      AJUSTES           //
    Ajustes * ControladorAjustes;

    int ajuste_resp_negativa;
    int ajuste_tiempo_pulsacion;
    int ajuste_tipo_letra;

    NSTimer *pulsacionTimer;

    ////////////////////////////
```

```
    IBOutlet UILabel *label_lugar;
    IBOutlet UILabel *label_sublugar;
    IBOutlet UILabel *label_personas;
    IBOutlet UILabel *label_verbos;
    IBOutlet UILabel *label_complementos;

    IBOutlet UILabel *tipo_entidad;
    IBOutlet UIButton *butVolver;
    BOOL ONPersonalizado;
}

@property(nonatomic,retain)IBOutlet UILabel *tipo_entidad;
@property(nonatomic,retain)IBOutlet UIButton *butVolver;
@property(nonatomic,retain)cargarFotos *cargarFotosVista;
@property (nonatomic, retain) Ajustes *ControladorAjustes;
@property(nonatomic)BOOL VerPersonalizado;

@property(nonatomic,retain)IBOutlet UILabel *label_lugar;
@property(nonatomic,retain)IBOutlet UILabel *label_sublugar;
@property(nonatomic,retain)IBOutlet UILabel *label_personas;
@property(nonatomic,retain)IBOutlet UILabel *label_verbos;
@property(nonatomic,retain)IBOutlet UILabel *label_complementos;

@property (nonatomic, retain) NSTimer *pulsacionTimer;

-(void) Ajustar;

-(IBAction)touchDown:(id)sender;
-(IBAction)touchUp:(id)sender;
-(void)showVista;

-(void)onVerPersonalizado:(BOOL)verPersonalizado;
-(IBAction)volverAlbum:(id)sender;


@end



//
//  seleccionEntidadPictos.m
//  COMUNICADOR
//
//  Created by Elisabeth Esandia Melero on 26/05/11.
//  Copyright 2011 __MyCompanyName__. All rights reserved.
//

#import "seleccionEntidadPictos.h"
#import "cargarFotos.h"
#import "Ajustes.h"

@implementation seleccionEntidadPictos;

@synthesize label_lugar;
@synthesize label_sublugar;
@synthesize label_personas;
@synthesize label_verbos;
@synthesize label_complementos;

@synthesize tipo_entidad;
@synthesize cargarFotosVista;
@synthesize VerPersonalizado;
@synthesize ControladorAjustes;

@synthesize pulsacionTimer;
@synthesize butVolver;
```

```
/*
- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{
    self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];
    if (self) {
        // Custom initialization
    }
    return self;
}
 */

- (void)dealloc
{
    [cargarFotosVista release];
    [super dealloc];
}

- (void)didReceiveMemoryWarning
{
    // Releases the view if it doesn't have a superview.
    [super didReceiveMemoryWarning];

    // Release any cached data, images, etc that aren't in use.
}

#pragma mark - View lifecycle

- (void)viewDidLoad
{
    ControladorAjustes =[[Ajustes alloc]init];
    cargarFotosVista=[[cargarFotos alloc] initWithNibName:@"cargarFotos" bundle:[NSBundle mainBundle]]; //Eli

    if (ONPersonalizado)
    {
        [tipo_entidad setHidden:NO];
    }
    else
    {
        [tipo_entidad setHidden:YES];

    }

    [self Ajustar];

    if (ajuste_tipo_letra<40) // IDIOMA CASTELLANO
    {
        [butVolver setTitle:@"VOLVER" forState:normal];
        label_lugar.text=@"LUGARES";
        label_sublugar.text=@"SUBLUGARES";
        label_personas.text=@"PERSONAS";
        label_verbos.text=@"VERBOS";
        label_complementos.text=@"COMPLEMENTOS";
        tipo_entidad.text=@"ENTIDADES PERSONALIZADAS";
    }
    else //IDIOMA INGLÉS
    {
        [butVolver setTitle:@"BACK" forState:normal];
        label_lugar.text=@"PLACES";
        label_sublugar.text=@"SUB-PLACES";
        label_personas.text=@"PERSONS";
        label_verbos.text=@"VERBS";
        label_complementos.text=@"COMPLEMENTS";
        tipo_entidad.text=@"PERSONALIZED ENTITIES";

    }

    [super viewDidLoad];
```

```
    // Do any additional setup after loading the view from its nib.
}

-(void)onVerPersonalizado:(BOOL)verPersonalizado
{
    ONPersonalizado=verPersonalizado;
}

- (void)Ajustar
{
    [ControladorAjustes viewDidLoad];
    ajuste_resp_negativa=ControladorAjustes.resp_negativa;
    ajuste_tiempo_pulsacion=ControladorAjustes.tiempo_pulsacion;
    ajuste_tipo_letra=ControladorAjustes.tipo_letra;

    NSLog(@"respuesta negativa %i, tiempo pulsacion %i, tipo letra
%i",ajuste_resp_negativa,ajuste_tiempo_pulsacion,ajuste_tipo_letra);
}

-(IBAction)touchDown:(id)sender
{
        self.pulsacionTimer = [NSTimer scheduledTimerWithTimeInterval:ajuste_tiempo_pulsacion target:self
selector:@selector(showVista) userInfo:sender repeats:NO];
}

-(IBAction)touchUp:(id)sender
{
        [pulsacionTimer invalidate];
}

-(void)showVista
{
    [cargarFotosVista viewDidLoad];
    if (ONPersonalizado)
    {
        [cargarFotosVista onVerPersonalizado:YES];
    }
    else
    {
        [cargarFotosVista onVerPersonalizado:NO];

    }
    NSInteger ButtonTagInteger= [pulsacionTimer.userInfo tag];
    NSLog(@"Entidad seleccionada = %i onverpersonalizado %i",ButtonTagInteger, ONPersonalizado);


    [cargarFotosVista cargarDatosEnArray:ButtonTagInteger];
    [self.view addSubview:cargarFotosVista.view];
}

-(IBAction)volverAlbum:(id)sender
{
    [self.view removeFromSuperview];
}

- (void)viewDidUnload
{
    [super viewDidUnload];
    // Release any retained subviews of the main view.
    // e.g. self.myOutlet = nil;
}

- (BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)interfaceOrientation
{
    // Return YES for supported orientations
        return NO;
}
```

```
@end
```

CargarFotos (.h y.m):

```
//
//  cargarFotos.h
//  COMUNICADOR
//
//  Created by Elisabeth Esandia Melero on 30/03/11.
//  Copyright 2011 __MyCompanyName__. All rights reserved.
//
#import "COMUNICADORAppDelegate.h"
#import "cargarFotos.h"
#import "VisorFotos.h"
#import <UIKit/UIKit.h>
#import <AssetsLibrary/AssetsLibrary.h>

@class VisorFotos;
@class Ajustes;

@interface cargarFotos : UIViewController <UIAlertViewDelegate>{

    ////////////////////////////
    //      AJUSTES           //
    Ajustes * ControladorAjustes;

    int ajuste_resp_negativa;
    int ajuste_tiempo_pulsacion;
    int ajuste_tipo_letra;

    //En Idioma irá el string correspondiente a la columna de la base de datos que quiero consultar
    NSString *Idioma;

    NSTimer *pulsacionTimer;

    ////////////////////////////

        NSMutableArray *assets;
    ALAssetsLibrary *library;
    NSArray *objects;
    NSMutableArray *objectsCogidos;
    NSMutableArray *thumbnail_table;
    NSMutableArray *label_table;
    BOOL ONPersonalizado;
    int num_pag_view;
    int indice;

    COMUNICADORAppDelegate *appDelegate;
    NSManagedObjectContext *context;
    NSEntityDescription *entityDesc;
    NSFetchRequest *request;
    NSError *error;

    COMUNICADORAppDelegate *appDelegate_per;
    NSManagedObjectContext *context_per;
    NSEntityDescription *entityDesc_per;
    NSFetchRequest *request_personalizado;
    NSPredicate *pred_personalizado;
    NSError *error_per;


    BOOL metodoCarga;

    VisorFotos * VisorFotosVista;
```

```
    UIAlertView *alertaSimple;
    UIAlertView *alertaSimplePersonalizada;

    IBOutlet UIActivityIndicatorView *activity;

        IBOutlet UIButton * Thumbnail1;
        IBOutlet UIButton * Thumbnail2;
        IBOutlet UIButton * Thumbnail3;
        IBOutlet UIButton * Thumbnail4;
        IBOutlet UIButton * Thumbnail5;
        IBOutlet UIButton * Thumbnail6;
        IBOutlet UIButton * Thumbnail7;
        IBOutlet UIButton * Thumbnail8;
        IBOutlet UIButton * Thumbnail9;
        IBOutlet UIButton * Thumbnail10;
        IBOutlet UIButton * Thumbnail11;
        IBOutlet UIButton * Thumbnail12;
        IBOutlet UIButton * Thumbnail13;
        IBOutlet UIButton * Thumbnail14;
        IBOutlet UIButton * Thumbnail15;

    IBOutlet UILabel * Label1;
        IBOutlet UILabel * Label2;
        IBOutlet UILabel * Label3;
        IBOutlet UILabel * Label4;
        IBOutlet UILabel * Label5;
        IBOutlet UILabel * Label6;
        IBOutlet UILabel * Label7;
        IBOutlet UILabel * Label8;
        IBOutlet UILabel * Label9;
        IBOutlet UILabel * Label10;
        IBOutlet UILabel * Label11;
        IBOutlet UILabel * Label12;
        IBOutlet UILabel * Label13;
        IBOutlet UILabel * Label14;
        IBOutlet UILabel * Label15;

    IBOutlet UIButton * BotonSiguiente;
    IBOutlet UIButton * BotonAnterior;
    IBOutlet UIButton * BotonVolver;


}

@property(nonatomic,retain)IBOutlet UIActivityIndicatorView *activity;

@property(nonatomic,retain)IBOutlet UIButton *Thumbnail1;
@property(nonatomic,retain)IBOutlet UIButton *Thumbnail2;
@property(nonatomic,retain)IBOutlet UIButton *Thumbnail3;
@property(nonatomic,retain)IBOutlet UIButton *Thumbnail4;
@property(nonatomic,retain)IBOutlet UIButton *Thumbnail5;
@property(nonatomic,retain)IBOutlet UIButton *Thumbnail6;
@property(nonatomic,retain)IBOutlet UIButton *Thumbnail7;
@property(nonatomic,retain)IBOutlet UIButton *Thumbnail8;
@property(nonatomic,retain)IBOutlet UIButton *Thumbnail9;
@property(nonatomic,retain)IBOutlet UIButton *Thumbnail10;
@property(nonatomic,retain)IBOutlet UIButton *Thumbnail11;
@property(nonatomic,retain)IBOutlet UIButton *Thumbnail12;
@property(nonatomic,retain)IBOutlet UIButton *Thumbnail13;
@property(nonatomic,retain)IBOutlet UIButton *Thumbnail14;
@property(nonatomic,retain)IBOutlet UIButton *Thumbnail15;

@property(nonatomic,retain)IBOutlet UILabel * Label1;
@property(nonatomic,retain)IBOutlet UILabel * Label2;
@property(nonatomic,retain)IBOutlet UILabel * Label3;
@property(nonatomic,retain)IBOutlet UILabel * Label4;
@property(nonatomic,retain)IBOutlet UILabel * Label5;
@property(nonatomic,retain)IBOutlet UILabel * Label6;
```

```
@property(nonatomic,retain)IBOutlet UILabel * Label7;
@property(nonatomic,retain)IBOutlet UILabel * Label8;
@property(nonatomic,retain)IBOutlet UILabel * Label9;
@property(nonatomic,retain)IBOutlet UILabel * Label10;
@property(nonatomic,retain)IBOutlet UILabel * Label11;
@property(nonatomic,retain)IBOutlet UILabel * Label12;
@property(nonatomic,retain)IBOutlet UILabel * Label13;
@property(nonatomic,retain)IBOutlet UILabel * Label14;
@property(nonatomic,retain)IBOutlet UILabel * Label15;

@property(nonatomic,retain)IBOutlet UIButton *BotonSiguiente;
@property(nonatomic,retain)IBOutlet UIButton *BotonAnterior;
@property(nonatomic,retain)IBOutlet UIButton *BotonVolver;

@property(nonatomic,retain)VisorFotos * VisorFotosVista;
@property (nonatomic, retain) Ajustes *ControladorAjustes;

@property (nonatomic, retain) NSTimer *pulsacionTimer;

-(void) Ajustar;

-(IBAction)touchDown:(id)sender;
-(IBAction)touchUp:(id)sender;
-(void)showVista;

-(void)onVerPersonalizado:(BOOL)verPersonalizado;

-(void)reloadRenderPics;
-(void)cargarDatosEnArray:(int)tipo_album;
-(IBAction)volverAlbum:(id)sender;
-(IBAction)pulsarPagSiguiente:(id)sender;
-(IBAction)pulsarPagAnterior:(id)sender;


@end



   //
//  cargarFotos.m
//  COMUNICADOR
//
//  Created by Elisabeth Esandia Melero on 30/03/11.
//  Copyright 2011 __MyCompanyName__. All rights reserved.
//

#import "COMUNICADORAppDelegate.h"
#import "cargarFotos.h"
#import "VisorFotos.h"
#import <AssetsLibrary/AssetsLibrary.h>
#import "Ajustes.h"


@implementation cargarFotos

@synthesize activity;

@synthesize Thumbnail1;
@synthesize Thumbnail2;
@synthesize Thumbnail3;
@synthesize Thumbnail4;
@synthesize Thumbnail5;
@synthesize Thumbnail6;
@synthesize Thumbnail7;
@synthesize Thumbnail8;
@synthesize Thumbnail9;
@synthesize Thumbnail10;
@synthesize Thumbnail11;
```

```
@synthesize Thumbnail12;
@synthesize Thumbnail13;
@synthesize Thumbnail14;
@synthesize Thumbnail15;

@synthesize Label1;
@synthesize Label2;
@synthesize Label3;
@synthesize Label4;
@synthesize Label5;
@synthesize Label6;
@synthesize Label7;
@synthesize Label8;
@synthesize Label9;
@synthesize Label10;
@synthesize Label11;
@synthesize Label12;
@synthesize Label13;
@synthesize Label14;
@synthesize Label15;

@synthesize BotonSiguiente;
@synthesize BotonAnterior;
@synthesize BotonVolver;

@synthesize VisorFotosVista;
@synthesize ControladorAjustes;

@synthesize pulsacionTimer;



/*
 // The designated initializer.  Override if you create the controller programmatically and want to perform customization that is not appropriate for viewDidLoad.
- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil {
    if ((self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil])) {
        // Custom initialization
    }
    return self;
}
*/


// Implement viewDidLoad to do additional setup after loading the view, typically from a nib.
- (void)viewDidLoad {

    ControladorAjustes =[[Ajustes alloc]init];
    [self Ajustar];

    if (ajuste_tipo_letra<40) // IDIOMA CASTELLANO
    {
        [BotonVolver setTitle:@"VOLVER" forState:normal];
        alertaSimplePersonalizada = [UIAlertView alloc];
        [alertaSimplePersonalizada initWithTitle:@"ALERTA"
                                         message:@"No hay pictos personalizados de esta entidad"
                                        delegate:self
                               cancelButtonTitle:@"Aceptar"
                               otherButtonTitles: nil];

        alertaSimple = [UIAlertView alloc];
        [alertaSimple initWithTitle:@"ALERTA"
                                         message:@"No hay pictos asociados"
                                        delegate:self
                               cancelButtonTitle:@"Aceptar"
                               otherButtonTitles: nil];
    }
    else //IDIOMA INGLÉS
```

```
    {
        [BotonVolver setTitle:@"BACK" forState:normal];
        alertaSimplePersonalizada = [UIAlertView alloc];
        [alertaSimplePersonalizada initWithTitle:@"ALERT"
                                         message:@"There are not personalized pictos for this entity"
                                        delegate:self
                               cancelButtonTitle:@"Ok"
                               otherButtonTitles: nil];

        alertaSimple = [UIAlertView alloc];
        [alertaSimple initWithTitle:@"ALERT"
                                         message:@"There are not personalized pictos for this entity"
                                        delegate:self
                               cancelButtonTitle:@"Ok"
                               otherButtonTitles: nil];
    }

    [activity startAnimating];

    assets = [[NSMutableArray alloc] init];
    library  = [[ALAssetsLibrary alloc] init];

    objects = [[NSArray alloc] init];
    objectsCogidos =[[NSMutableArray alloc]init];

    appDelegate = [[UIApplication sharedApplication] delegate];
    context = [appDelegate managedObjectContext];

    appDelegate_per = [[UIApplication sharedApplication] delegate]; // Parte de la BD
    context_per = [appDelegate_per managedObjectContext]; // Parte de la BD

    thumbnail_table = [[NSMutableArray alloc]  initWithObjects:Thumbnail1, Thumbnail2, Thumbnail3, Thumbnail4,
Thumbnail5, Thumbnail6, Thumbnail7, Thumbnail8, Thumbnail9, Thumbnail10, Thumbnail11, Thumbnail12, Thumbnail13,
Thumbnail14, Thumbnail15, nil];

    label_table=[[NSMutableArray alloc] initWithObjects:Label1, Label2, Label3, Label4, Label5, Label6, Label7, Label8,
Label9, Label10, Label11, Label12, Label13, Label14, Label15, nil];

    for (int i=0; i<[thumbnail_table count]; i++)
    {
        [[thumbnail_table objectAtIndex:i] setHidden:YES];
    }

    for (int i=0; i<[label_table count]; i++)
    {
        [[label_table objectAtIndex:i] setText:@""];
    }

    num_pag_view = 0;
    [BotonAnterior setEnabled:NO];
    [BotonAnterior setHidden:YES];
    [BotonSiguiente setEnabled:NO];
    [BotonSiguiente setHidden:YES];

    VisorFotosVista=[[VisorFotos alloc] initWithNibName:@"VisorFotos" bundle:[NSBundle mainBundle]]; //Eli

    [super viewDidLoad];
}

- (void)alertView:(UIAlertView *)actionSheet clickedButtonAtIndex:(NSInteger)buttonIndex {
    // the user clicked one of the OK/Cancel buttons
    if (buttonIndex == 0)
    {
        [self.view removeFromSuperview];
    }
}

- (void)Ajustar
```

```
{
    [ControladorAjustes viewDidLoad];
    ajuste_resp_negativa=ControladorAjustes.resp_negativa;
    ajuste_tiempo_pulsacion=ControladorAjustes.tiempo_pulsacion;
    ajuste_tipo_letra=ControladorAjustes.tipo_letra;

    if (ajuste_tipo_letra==20)
    {
        Idioma=@"nombre";
    }
    else if (ajuste_tipo_letra==30)
    {
        Idioma=@"nombre_may";
    }
    else if (ajuste_tipo_letra==50)
    {
        Idioma=@"nombre_ing";
    }
    else if (ajuste_tipo_letra==60)
    {
        Idioma=@"nombre_may_ing";
    }

    NSLog(@"respuesta negativa %i, tiempo pulsacion %i, tipo letra
%i",ajuste_resp_negativa,ajuste_tiempo_pulsacion,ajuste_tipo_letra);
}

-(IBAction)touchDown:(id)sender
{
        self.pulsacionTimer = [NSTimer scheduledTimerWithTimeInterval:ajuste_tiempo_pulsacion target:self
selector:@selector(showVista) userInfo:sender repeats:NO];
}

-(IBAction)touchUp:(id)sender
{
        [pulsacionTimer invalidate];
}

-(void)showVista
{
    [self.view addSubview:VisorFotosVista.view];

    NSInteger ButtonTagInteger= [pulsacionTimer.userInfo tag];
    NSLog(@"Imagen seleccionada = %i",ButtonTagInteger);

    NSMutableArray*nuevasImagenes=[[NSMutableArray alloc]init];

    if (metodoCarga==0)
    {
        nuevasImagenes=assets;
    }
    else
    {
        nuevasImagenes=objectsCogidos;
    }

    //[VisorFotosVista viewDidLoad];
    [VisorFotosVista cargarImagen:((num_pag_view*[thumbnail_table count])+ButtonTagInteger)
cargarArray:nuevasImagenes metodo:metodoCarga];
}

- (void)onVerPersonalizado:(BOOL)verPersonalizado
{

    ONPersonalizado=verPersonalizado;
    NSLog(@"ESTOY VIENDO PERSONALIZADOS %i", ONPersonalizado);
}
```

```
- (void)cargarDatosEnArray:(int)tipo_album
{
    dispatch_async(dispatch_get_main_queue(), ^
                   {
                       NSAutoreleasePool *pool = [[NSAutoreleasePool alloc] init];

                NSLog(@"Estoy cargando Datos");
                     NSLog(@"ESTOY VIENDO PERSONALIZADOS %i", ONPersonalizado);

    switch (tipo_album)
    {
        case 0:
        {
        [library enumerateGroupsWithTypes:ALAssetsGroupAll
                                usingBlock:
                               // Group enumerator Block
                                ^(ALAssetsGroup *group, BOOL *stop)
                               {
                                   if(group != nil)
                                   {
                                       [group enumerateAssetsUsingBlock:

                                        // Asset enumerate Block
                                        ^(ALAsset *result, NSUInteger index, BOOL *stop)
                                       {
                                           if(result != NULL)
                                           {
                                               [assets addObject:result];
                                           };
                                       }
                                        ];
                                   };
                                   [self.activity stopAnimating];
                                   [self.activity setHidden:YES];
                                   metodoCarga=0;
                                   [self reloadRenderPics];
                               }
                                failureBlock:
                                ^(NSError *error) {
                                    NSLog(@"Failure");}
         ];
        [library release];
        }
            break;
        case 1:
        {
            [activity startAnimating];

            if (ONPersonalizado)  // LLAMAREMOS A LOS PERSONALIZADOS
            {
                ////////////////////////////////////////////////////////
                // LLAMADA A LA BD ENTIDAD PERSONALIZADA
                ////////////////////////////////////////////////////////
                NSLog(@"Se está cargando el álbum LUGARES PERSONALIZADOS");
                entityDesc_per=[NSEntityDescription entityForName:@"Lugar" inManagedObjectContext:context_per];
                request_personalizado=[[NSFetchRequest alloc]init];
                [request_personalizado setEntity:entityDesc_per];
                pred_personalizado = [NSPredicate predicateWithFormat:@"ANY album_personalizado like '1'"];
                [request_personalizado setPredicate:pred_personalizado];
                objects =[context_per executeFetchRequest:request_personalizado error:&error_per];

                if ([objects count] == 0)
                {
                    [alertaSimplePersonalizada show];
                }
                else
                {
                    [self.activity stopAnimating];
```

```
            [self.activity setHidden:YES];
            metodoCarga=1;
            [objectsCogidos setArray:objects];
            [self reloadRenderPics];
            [self.activity stopAnimating];
            [request_personalizado release];
        }
    }
    else
    {
        ////////////////////////////////////////////////////////
        // LLAMADA A LA BD ENTIDAD ORIGINAL
        ////////////////////////////////////////////////////////
        NSLog(@"Se está cargando el álbum LUGARES");
        entityDesc = [NSEntityDescription entityForName:@"Lugar" inManagedObjectContext:context];
        request = [[NSFetchRequest alloc] init];
        [request setEntity:entityDesc];
        objects = [context executeFetchRequest:request error:&error];

        if ([objects count] == 0)
        {
            [alertaSimple show];
        }
        else
        {
            [self.activity stopAnimating];
            [self.activity setHidden:YES];
            metodoCarga=1;
            [objectsCogidos setArray:objects];

            [self reloadRenderPics];
            [self.activity stopAnimating];
            [request release];
        }
    }
    break;

}
case 2:
{
    [activity startAnimating];
    if (ONPersonalizado)  // LLAMAREMOS A LOS PERSONALIZADOS
    {
        ////////////////////////////////////////////////////////
        // LLAMADA A LA BD ENTIDAD PERSONALIZADA
        ////////////////////////////////////////////////////////
        NSLog(@"Se está cargando el álbum SUBLUGARES PERSONALIZADOS");
        entityDesc_per=[NSEntityDescription entityForName:@"Sublugar" inManagedObjectContext:context_per];
        request_personalizado=[[NSFetchRequest alloc]init];
        [request_personalizado setEntity:entityDesc_per];
        pred_personalizado = [NSPredicate predicateWithFormat:@"ANY album_personalizado like '1'"];
        [request_personalizado setPredicate:pred_personalizado];
        objects =[context_per executeFetchRequest:request_personalizado error:&error_per];

        if ([objects count] == 0)
        {
            [alertaSimplePersonalizada show];
        }
        else
        {
            [self.activity stopAnimating];
            [self.activity setHidden:YES];
            metodoCarga=1;
            [objectsCogidos setArray:objects];
            [self reloadRenderPics];
            [self.activity stopAnimating];
            [request_personalizado release];
        }
```

```
    }
    else
    {
        ////////////////////////////////////////////////////////
        // LLAMADA A LA BD ENTIDAD ORIGINAL
        ////////////////////////////////////////////////////////
        NSLog(@"Se está cargando el álbum SUBLUGARES");
        entityDesc = [NSEntityDescription entityForName:@"Sublugar" inManagedObjectContext:context];
        request = [[NSFetchRequest alloc] init];
        [request setEntity:entityDesc];
        objects = [context executeFetchRequest:request error:&error];

        if ([objects count] == 0)
        {
            [alertaSimple show];
        }
        else
        {
            [self.activity stopAnimating];
            [self.activity setHidden:YES];
            metodoCarga=1;
            [objectsCogidos setArray:objects];

            [self reloadRenderPics];
            [self.activity stopAnimating];
            [request release];
        }
    }

}
    break;
case 3:
{
    [activity startAnimating];
    if (ONPersonalizado)  // LLAMAREMOS A LOS PERSONALIZADOS
    {
        ////////////////////////////////////////////////////////
        // LLAMADA A LA BD ENTIDAD PERSONALIZADA
        ////////////////////////////////////////////////////////
        NSLog(@"Se está cargando el álbum PERSONAS PERSONALIZADOS");
        entityDesc_per=[NSEntityDescription entityForName:@"Personas" inManagedObjectContext:context_per];
        request_personalizado=[[NSFetchRequest alloc]init];
        [request_personalizado setEntity:entityDesc_per];
        pred_personalizado = [NSPredicate predicateWithFormat:@"ANY album_personalizado like '1'"];
        [request_personalizado setPredicate:pred_personalizado];
        objects =[context_per executeFetchRequest:request_personalizado error:&error_per];

        if ([objects count] == 0)
        {
            [alertaSimplePersonalizada show];
        }
        else
        {
            [self.activity stopAnimating];
            [self.activity setHidden:YES];
            metodoCarga=1;
            [objectsCogidos setArray:objects];
            [self reloadRenderPics];
            [self.activity stopAnimating];
            [request_personalizado release];
        }
    }
    else
    {
        ////////////////////////////////////////////////////////
        // LLAMADA A LA BD ENTIDAD ORIGINAL
        ////////////////////////////////////////////////////////
        NSLog(@"Se está cargando el álbum PERSONAS");
```

```
        entityDesc = [NSEntityDescription entityForName:@"Personas" inManagedObjectContext:context];
        request = [[NSFetchRequest alloc] init];
        [request setEntity:entityDesc];
        objects = [context executeFetchRequest:request error:&error];

        if ([objects count] == 0)
        {
            [alertaSimple show];
        }
        else
        {
            [self.activity stopAnimating];
            [self.activity setHidden:YES];
            metodoCarga=1;
            [self.activity stopAnimating];
            [self.activity setHidden:YES];
            metodoCarga=1;
            [objectsCogidos setArray:objects];

            [self reloadRenderPics];
            [self.activity stopAnimating];
            [request release];

        }
    }

}
    break;
case 4:
{
    [activity startAnimating];
    if (ONPersonalizado)  // LLAMAREMOS A LOS PERSONALIZADOS
    {
        ////////////////////////////////////////////////////////////
        // LLAMADA A LA BD ENTIDAD PERSONALIZADA
        ////////////////////////////////////////////////////////////
        NSLog(@"Se está cargando el álbum VERBOS PERSONALIZADOS");
        entityDesc_per=[NSEntityDescription entityForName:@"Verbos" inManagedObjectContext:context_per];
        request_personalizado=[[NSFetchRequest alloc]init];
        [request_personalizado setEntity:entityDesc_per];
        pred_personalizado = [NSPredicate predicateWithFormat:@"ANY album_personalizado like '1'"];
        [request_personalizado setPredicate:pred_personalizado];
        objects =[context_per executeFetchRequest:request_personalizado error:&error_per];

        if ([objects count] == 0)
        {
            [alertaSimplePersonalizada show];
        }
        else
        {
            [self.activity stopAnimating];
            [self.activity setHidden:YES];
            metodoCarga=1;
            [objectsCogidos setArray:objects];
            [self reloadRenderPics];
            [self.activity stopAnimating];
            [request_personalizado release];
        }
    }
    else
    {
        ////////////////////////////////////////////////////////////
        // LLAMADA A LA BD ENTIDAD ORIGINAL
        ////////////////////////////////////////////////////////////
        NSLog(@"Se está cargando el álbum VERBOS");
        entityDesc = [NSEntityDescription entityForName:@"Verbos" inManagedObjectContext:context];
        request = [[NSFetchRequest alloc] init];
        [request setEntity:entityDesc];
```

```
            objects = [context executeFetchRequest:request error:&error];

            if ([objects count] == 0)
            {
                [alertaSimple show];
            }
            else
            {
                [self.activity stopAnimating];
                [self.activity setHidden:YES];
                metodoCarga=1;
                [objectsCogidos setArray:objects];

                [self reloadRenderPics];
                [self.activity stopAnimating];
                [request release];
            }
        }

    }
        break;
    case 5:
    {
        [activity startAnimating];
        if (ONPersonalizado)  // LLAMAREMOS A LOS PERSONALIZADOS
        {
            ////////////////////////////////////////////////////////
            // LLAMADA A LA BD ENTIDAD PERSONALIZADA
            ////////////////////////////////////////////////////////
            NSLog(@"Se está cargando el álbum COMPLEMENTOS PERSONALIZADOS");
            entityDesc_per=[NSEntityDescription entityForName:@"Complementos" inManagedObjectContext:context_per];
            request_personalizado=[[NSFetchRequest alloc]init];
            [request_personalizado setEntity:entityDesc_per];
            pred_personalizado = [NSPredicate predicateWithFormat:@"ANY album_personalizado like '1'"];
            [request_personalizado setPredicate:pred_personalizado];
            objects =[context_per executeFetchRequest:request_personalizado error:&error_per];

            if ([objects count] == 0)
            {
                [alertaSimplePersonalizada show];
            }
            else
            {
                [self.activity stopAnimating];
                [self.activity setHidden:YES];
                metodoCarga=1;
                [objectsCogidos setArray:objects];
                [self reloadRenderPics];
                [self.activity stopAnimating];
                [request_personalizado release];
            }
        }
        else
        {
            ////////////////////////////////////////////////////////
            // LLAMADA A LA BD ENTIDAD ORIGINAL
            ////////////////////////////////////////////////////////
            NSLog(@"Se está cargando el álbum COMPLEMENTOS");
            entityDesc = [NSEntityDescription entityForName:@"Complementos" inManagedObjectContext:context];
            request = [[NSFetchRequest alloc] init];
            [request setEntity:entityDesc];
            objects = [context executeFetchRequest:request error:&error];

            if ([objects count] == 0)
            {
                [alertaSimple show];
            }
            else
```

```
            {
                [self.activity stopAnimating];
                [self.activity setHidden:YES];
                metodoCarga=1;
                [objectsCogidos setArray:objects];

                [self reloadRenderPics];
                [self.activity stopAnimating];
                [request release];
            }
        }

    }
        break;
    default:
        break;
}

[pool release];
    });
}

-(void)reloadRenderPics
{
    indice=num_pag_view*[thumbnail_table count];

    NSLog(@"indice= %i;num_pag_view= %i; thumbnail_table count= %i; metodoCarga = %i", indice, num_pag_view,
[thumbnail_table count], metodoCarga);

    objects=nil;

    if (metodoCarga==0)
    {
        int contar=[assets count];
        NSLog(@"imagenes que tenemos %i ",contar);

        for (int i=0; i<[thumbnail_table count]; i++)
        {
            if ((i+indice)<[assets count])
            {
                [[thumbnail_table objectAtIndex:i] setImage:[UIImage imageWithCGImage:[[assets objectAtIndex:(i+indice)]
thumbnail]
                                                    scale:0.5 orientation:UIImageOrientationUp] forState:UIControlStateNormal];
                [[thumbnail_table objectAtIndex:i] setHidden:NO];
                [BotonSiguiente setEnabled:YES];
                [BotonSiguiente setHidden:NO];
            }
            else
            {
                //Hide the button
                [[thumbnail_table objectAtIndex:i] setHidden:YES];
                [BotonSiguiente setEnabled:NO];
                [BotonSiguiente setHidden:YES];
                if (indice!=0)
                {
                    [BotonAnterior setEnabled:YES];
                    [BotonAnterior setHidden:NO];
                }
                else
                {
                    NSLog(@"Estoy en la primera página!");
                }
            }
        }
    }
    else
    {
        int contar2=[objectsCogidos count];
```

```
    NSLog(@"imagenes que tenemos %i ",contar2);

    for (int i=0; i<[thumbnail_table count]; i++)
    {
        NSLog(@"i= %i; indice= %i",i,indice);
        if ((i+indice)<[objectsCogidos count])
        {
            [[thumbnail_table objectAtIndex:i] setImage:[UIImage imageNamed:[[objectsCogidos objectAtIndex:(i+indice)]
                                                        valueForKey:@"nombre"]]forState:UIControlStateNormal];
            [[label_table objectAtIndex:i] setText:[[objectsCogidos objectAtIndex:i+indice]valueForKey:Idioma]];
            [[thumbnail_table objectAtIndex:i] setHidden:NO];
            [BotonSiguiente setEnabled:YES];
            [BotonSiguiente setHidden:NO];
        }
        else
        {
            //Hide the button
            [[thumbnail_table objectAtIndex:i] setHidden:YES];
            [[label_table objectAtIndex:i]setText:@""];
            [BotonSiguiente setEnabled:NO];
            [BotonSiguiente setHidden:YES];
            if (indice!=0)
            {
                [BotonAnterior setEnabled:YES];
                [BotonAnterior setHidden:NO];
            }
            else
            {
                NSLog(@"Estoy en la primera página!");
            }
        }
    }
  }


}


-(IBAction)volverAlbum:(id)sender
{
    num_pag_view=0;

    [BotonAnterior setHidden:YES];
    [BotonAnterior setEnabled:NO];
    [BotonSiguiente setHidden:YES];
    [BotonSiguiente setEnabled:NO];
        [self.view removeFromSuperview];
}

-(IBAction)pulsarPagSiguiente:(id)sender
{
    num_pag_view +=1;

    [self reloadRenderPics];

    [BotonAnterior setEnabled:YES];
    [BotonAnterior setHidden:NO];

}

-(IBAction)pulsarPagAnterior:(id)sender
{
    [BotonSiguiente setEnabled:YES];
    [BotonSiguiente setHidden:NO];

    if (num_pag_view>0) {
        num_pag_view -=1;
    }
```

```
    if(num_pag_view==0)
    {
        [BotonAnterior setEnabled:NO];
        [BotonAnterior setHidden:YES];
    }
    [self reloadRenderPics];
}

- (BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)interfaceOrientation {
    // Overriden to allow any orientation.
    return NO;
}
# pragma mark -
# pragma mark Memory Management

- (void)didReceiveMemoryWarning {
    // Releases the view if it doesn't have a superview.
    [super didReceiveMemoryWarning];

    // Release any cached data, images, etc that aren't in use.
}


- (void)viewDidUnload {

    self.activity=nil;

    self.Thumbnail1=nil;
    self.Thumbnail2=nil;
    self.Thumbnail3=nil;
    self.Thumbnail4=nil;
    self.Thumbnail5=nil;
    self.Thumbnail6=nil;
    self.Thumbnail7=nil;
    self.Thumbnail8=nil;
    self.Thumbnail9=nil;
    self.Thumbnail10=nil;
    self.Thumbnail11=nil;
    self.Thumbnail12=nil;
    self.Thumbnail13=nil;
    self.Thumbnail14=nil;
    self.Thumbnail15=nil;

    self.Label1=nil;
    self.Label2=nil;
    self.Label3=nil;
    self.Label4=nil;
    self.Label5=nil;
    self.Label6=nil;
    self.Label7=nil;
    self.Label8=nil;
    self.Label9=nil;
    self.Label10=nil;
    self.Label11=nil;
    self.Label12=nil;
    self.Label13=nil;
    self.Label14=nil;
    self.Label15=nil;

    self.BotonAnterior=nil;
    self.BotonSiguiente=nil;
    self.VisorFotosVista=nil;

    [super viewDidUnload];
    // Release any retained subviews of the main view.
    // e.g. self.myOutlet = nil;
}
```

```
- (void)dealloc {

    [assets release];
    [library release];

    [objects release];
    [objectsCogidos release];
    //[request release];

    [activity release];

    [thumbnail_table release];
    [label_table release];

    [BotonSiguiente release];
    [BotonAnterior release];
    [VisorFotosVista release];

    [super dealloc];
}


@end
```

VisorFotos (.h y .m):

```
//
//  VisorFotos.h
//  COMUNICADOR
//
//  Created by Elisabeth Esandia Melero on 10/04/11.
//  Copyright 2011 __MyCompanyName__. All rights reserved.
//

#import <UIKit/UIKit.h>
#import <AssetsLibrary/AssetsLibrary.h>

@class MyScrollView;
@class Ajustes;

@interface VisorFotos : UIViewController <UIScrollViewDelegate> {

    ////////////////////////////
    //      AJUSTES           //
    Ajustes * ControladorAjustes;

    int ajuste_resp_negativa;
    int ajuste_tiempo_pulsacion;
    int ajuste_tipo_letra;

    NSTimer *pulsacionTimer;

    //En Idioma irá el string correspondiente a la columna de la base de datos que quiero consultar
    NSString *Idioma;

    ////////////////////////////

    IBOutlet UIScrollView *scroll;

    IBOutlet UIImageView *album_fondo;

    NSArray *imageSet;
    int imagen_inicial;
    int NumImages;
    BOOL metodoCarga;
```

```
    UIImageView *nextView;
    UIImageView *view1;
        UIImageView *view2;

        int view1Index;
        int view2Index;

    IBOutlet UIButton * BotonSiguiente;
    IBOutlet UIButton * BotonAnterior;
    IBOutlet UIButton * BotonVolver;

    IBOutlet UILabel * pictosText1;
    IBOutlet UILabel * pictosText2;
    IBOutlet UILabel * pictosTextNext;

}

@property (nonatomic, retain) Ajustes *ControladorAjustes;

@property(nonatomic,retain)IBOutlet UIImageView *album_fondo;


@property(nonatomic,retain)IBOutlet UIScrollView *scroll;
@property(nonatomic,retain)IBOutlet UIButton *BotonSiguiente;
@property(nonatomic,retain)IBOutlet UIButton *BotonAnterior;
@property(nonatomic,retain)IBOutlet UIButton *BotonVolver;
@property(nonatomic,retain)IBOutlet UILabel *pictosText1;
@property(nonatomic,retain)IBOutlet UILabel *pictosText2;
@property(nonatomic,retain)IBOutlet UILabel *pictosTextNext;

-(void)Ajustar;

-(void)cargarImagen:(int)imagen_inic cargarArray:(NSMutableArray*)nuevasImagenes metodo:(BOOL)metodoDeCarga;
-(IBAction)volverFotos:(id)sender;
-(IBAction)pulsarImagSiguiente:(id)sender;
-(IBAction)pulsarImagAnterior:(id)sender;


@end


//
//  VisorFotos.m
//  COMUNICADOR
//
//  Created by Elisabeth Esandia Melero on 10/04/11.
//  Copyright 2011 __MyCompanyName__. All rights reserved.
//

#import "VisorFotos.h"
#import <AssetsLibrary/AssetsLibrary.h>
#import "Ajustes.h"

#define ScrollX 70
#define ScrollY 37
#define ScrollWidth 900
#define ScrollHeight 600

#define ScrollXPic 200
#define ScrollWidthPic 600

@interface VisorFotos (Private)

- (void) update;

@end
```

```
@implementation VisorFotos

@synthesize album_fondo;

@synthesize ControladorAjustes;
@synthesize scroll;
@synthesize BotonSiguiente;
@synthesize BotonAnterior;
@synthesize BotonVolver;
@synthesize pictosText1;
@synthesize pictosText2;
@synthesize pictosTextNext;



#pragma mark - View lifecycle

- (void)viewDidLoad

{

    ControladorAjustes =[[Ajustes alloc]init];
    [self Ajustar];

    if (ajuste_tipo_letra<40) // IDIOMA CASTELLANO
    {
        [BotonVolver setTitle:@"VOLVER" forState:normal];
    }
    else //IDIOMA INGLÉS
    {
        [BotonVolver setTitle:@"BACK" forState:normal];
    }

    if((self = [super init]))
            {
                        scroll = [[UIScrollView alloc] init];
                        scroll.scrollEnabled = YES;
                        scroll.pagingEnabled = YES;
                        scroll.directionalLockEnabled = YES;
                        scroll.showsVerticalScrollIndicator = NO;
                        scroll.showsHorizontalScrollIndicator = YES;
                        scroll.delegate = self;
                        scroll.backgroundColor = [UIColor clearColor];
        scroll.autoresizesSubviews = YES;


        pictosText1 = [[UILabel alloc] initWithFrame:CGRectMake(0, 5, 600, 115)];
        [pictosText1 setBackgroundColor:[UIColor clearColor]];
        [pictosText1 setTextAlignment:UITextAlignmentCenter];
        pictosText1.font = [UIFont boldSystemFontOfSize: 64.0];

        pictosText2 = [[UILabel alloc] initWithFrame:CGRectMake(0, 5, 600, 115)];
        [pictosText2 setBackgroundColor:[UIColor clearColor]];
        [pictosText2 setTextAlignment:UITextAlignmentCenter];
        pictosText2.font = [UIFont boldSystemFontOfSize: 64.0];

        pictosTextNext = [[UILabel alloc] initWithFrame:CGRectMake(0, 5, 600, 115)];
        [pictosTextNext setBackgroundColor:[UIColor clearColor]];
        [pictosTextNext setTextAlignment:UITextAlignmentCenter];
        pictosTextNext.font = [UIFont boldSystemFontOfSize: 64.0];


        // Cambiamos el tamaño del visor dependiendo de si estamos mostrando pictos o no.
        if (metodoCarga==1)//mostramos pictos.
        {
            [album_fondo setImage:[UIImage imageNamed:[NSString stringWithFormat:@"album_fondo_pictos.png"]]];
            scroll.frame = CGRectMake(ScrollXPic, ScrollY, ScrollWidthPic, ScrollHeight);
```

```
        //scroll.contentSize = CGSizeMake(NumImages*ScrollWidthPic, ScrollHeight); //tamaño total del scroll
    }
    else if(metodoCarga==0) // Mostramos assets
    {
        [album_fondo setImage:[UIImage imageNamed:[NSString stringWithFormat:@"album_fondo_assets.png"]]];
        scroll.frame = CGRectMake(ScrollX, ScrollY, ScrollWidth, ScrollHeight);
        //scroll.contentSize = CGSizeMake(NumImages*ScrollWidth, ScrollHeight); //tamaño total del scroll
    }

                [    self.view addSubview:scroll];

    nextView = [[UIImageView alloc] init];

                    view1 = [[UIImageView alloc] init];
                [    scroll addSubview:view1];

                    view2 = [[UIImageView alloc] init];
                [    scroll addSubview:view2];

     //cargarFotosPagActual=[[cargarFotos alloc]initWithNibName:@"cargarFotos" bundle:[NSBundle mainBundle]];
        }

        //return self;

}

- (void)Ajustar
{
    [ControladorAjustes viewDidLoad];
    ajuste_resp_negativa=ControladorAjustes.resp_negativa;
    ajuste_tiempo_pulsacion=ControladorAjustes.tiempo_pulsacion;
    ajuste_tipo_letra=ControladorAjustes.tipo_letra;

    if (ajuste_tipo_letra==20){
        Idioma=@"nombre";
    }
    else if (ajuste_tipo_letra==30){
        Idioma=@"nombre_may";
    }
    else if (ajuste_tipo_letra==50){
        Idioma=@"nombre_ing";
    }
    else if (ajuste_tipo_letra==60){
        Idioma=@"nombre_may_ing";
    }

    NSLog(@"respuesta negativa %i, tiempo pulsacion %i, tipo letra
%i",ajuste_resp_negativa,ajuste_tiempo_pulsacion,ajuste_tipo_letra);
}

- (BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)interfaceOrientation
{
    // Return YES for supported orientations
        return NO;
}

/*
- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{
    self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];
    if (self) {
        // Custom initialization
    }
    return self;
}
*/
-(void)cargarImagen:(int)imagen_inic cargarArray:(NSMutableArray*)nuevasImagenes metodo:(BOOL)metodoDeCarga
{
```

```
    [imageSet autorelease];

        imageSet = [nuevasImagenes retain]; //imageSet:array de assets(imagenes).

    metodoCarga = metodoDeCarga;

    NumImages= [imageSet count]; // nº de imagenes dentro del array

    imagen_inicial = imagen_inic; //imagen que se ha clickado y hay que colocar la primera.

    [self viewDidLoad];

    if (imagen_inicial<NumImages-1) //Vemos si es la primera, la última o una imagen entre medio.
    {
        if (metodoDeCarga==1) // NSMutableArray = Assets
        {

            view1.frame = CGRectMake((ScrollWidthPic*imagen_inicial), 0, ScrollWidthPic, ScrollHeight); //Establecemos la
posicion correspondiente a la imagen elegida dentro del scroll
            view2.frame = CGRectMake((ScrollWidthPic*(imagen_inicial+1)), 0, ScrollWidthPic, ScrollHeight);

            view1.image = [UIImage imageNamed:[[imageSet objectAtIndex:(imagen_inicial)] valueForKey:@"nombre"]];
            view2.image = [UIImage imageNamed:[[imageSet objectAtIndex:(imagen_inicial+1)] valueForKey:@"nombre"]];

            [pictosText1 setText:[[imageSet objectAtIndex:(imagen_inicial)] valueForKey:Idioma]];
            [pictosText2 setText:[[imageSet objectAtIndex:(imagen_inicial+1)] valueForKey:Idioma]];

            [view1 addSubview:pictosText1];
            [view2 addSubview:pictosText2];

        }
        else //NSMutableArray = imagenes normales

        {
            view1.frame = CGRectMake((ScrollWidth*imagen_inicial), 0, ScrollWidth, ScrollHeight); //Establecemos la posicion
correspondiente a la imagen elegida dentro del scroll
            view2.frame = CGRectMake((ScrollWidth*(imagen_inicial+1)), 0, ScrollWidth, ScrollHeight);

            view1.image = [UIImage imageWithCGImage:[[[imageSet objectAtIndex:imagen_inicial]defaultRepresentation]
fullScreenImage]];//Imagen elegida y siguiente
            view2.image = [UIImage imageWithCGImage:[[[imageSet objectAtIndex:imagen_inicial+1]defaultRepresentation]
fullScreenImage]];

        }
    }
    else
    {
        if (metodoDeCarga==1)
        {
            view1.frame = CGRectMake((ScrollWidthPic*imagen_inicial), 0, ScrollWidthPic, ScrollHeight); //Establecemos la
posicion correspondiente a la imagen elegida dentro del scroll
            view2.frame = CGRectMake((ScrollWidthPic*(imagen_inicial-1)), 0, ScrollWidthPic, ScrollHeight);

            view1.image = [UIImage imageNamed:[[imageSet objectAtIndex:(imagen_inicial)] valueForKey:@"nombre"]];
            view2.image = [UIImage imageNamed:[[imageSet objectAtIndex:(imagen_inicial-1)] valueForKey:@"nombre"]];

            [pictosText1 setText:[[imageSet objectAtIndex:(imagen_inicial)] valueForKey:Idioma]];
            [pictosText2 setText:[[imageSet objectAtIndex:(imagen_inicial-1)] valueForKey:Idioma]];

            [view1 addSubview:pictosText1];
            [view2 addSubview:pictosText2];
        }
        else
        {
            view1.frame = CGRectMake((ScrollWidth*imagen_inicial), 0, ScrollWidth, ScrollHeight); //Establecemos la posicion
correspondiente a la imagen elegida dentro del scroll
            view2.frame = CGRectMake((ScrollWidth*(imagen_inicial-1)), 0, ScrollWidth, ScrollHeight);
```

```
            view1.image = [UIImage imageWithCGImage:[[[imageSet objectAtIndex:imagen_inicial]defaultRepresentation]
fullScreenImage]];//Imagen elegida y siguiente
            view2.image = [UIImage imageWithCGImage:[[[imageSet objectAtIndex:imagen_inicial-1]defaultRepresentation]
fullScreenImage]];
        }
    }

    if (metodoCarga==1)
    {
        [scroll setContentOffset:CGPointMake((ScrollWidthPic*imagen_inicial), ScrollY)]; //Cargar imagen inicial en el scroll
        scroll.contentSize = CGSizeMake([imageSet count]*ScrollWidthPic, ScrollHeight); //tamaño total del scroll
    }
    else
    {
        [scroll setContentOffset:CGPointMake((ScrollWidth*imagen_inicial), ScrollY)]; //Cargar imagen inicial en el scroll
        scroll.contentSize = CGSizeMake([imageSet count]*ScrollWidth, ScrollHeight); //tamaño total del scroll
    }

    NSLog(@"Imagen pasada = %i",imagen_inicial);

}


-(IBAction)volverFotos:(id)sender
{
    [self.view removeFromSuperview];
}


-(IBAction)pulsarImagSiguiente:(id)sender
{
            float currPos = scroll.contentOffset.x;

    if (metodoCarga==1)
    {
        int selectedPage = roundf(currPos / ScrollWidthPic);

        [scroll scrollRectToVisible:CGRectMake(ScrollWidthPic*(selectedPage+1), ScrollY, ScrollWidthPic, ScrollHeight)
animated:YES];
    }
    else
    {
        int selectedPage = roundf(currPos / ScrollWidth);

        [scroll scrollRectToVisible:CGRectMake(ScrollWidth*(selectedPage+1), ScrollY, ScrollWidth, ScrollHeight)
animated:YES];
    }
}

-(IBAction)pulsarImagAnterior:(id)sender
{
            float currPos = scroll.contentOffset.x;

    if (metodoCarga==1)
    {
        int selectedPage = roundf(currPos / ScrollWidthPic);

        [scroll scrollRectToVisible:CGRectMake(ScrollWidthPic*(selectedPage-1), ScrollY, ScrollWidthPic, ScrollHeight)
animated:YES];
    }
    else
    {
        int selectedPage = roundf(currPos / ScrollWidth);

        [scroll scrollRectToVisible:CGRectMake(ScrollWidth*(selectedPage-1), ScrollY, ScrollWidth, ScrollHeight)
animated:YES];
    }
}
```

```
# pragma mark -
# pragma mark UIScrollView Delegate

- (void) scrollViewDidScroll:(UIScrollView *) scrollView
{
        [self update];
}


- (void) update
{
   CGFloat pageWidth=0;

   if (metodoCarga==1)
   {
      pageWidth = ScrollWidthPic;
   }
   else
   {
      pageWidth = ScrollWidth;
   }

   float currPos = scroll.contentOffset.x; //Posición del scroll en cada momento

        int selectedPage = roundf(currPos / pageWidth); //Página que se ve en cada momento (empieza en la cero=0)

        float truePosition = selectedPage*pageWidth; //coordenadas de la página actual dentro del scroll

        int zone = selectedPage % 2; //Comprobamos el resto de la pág/2. ZONE=0 --> pág.Par - ZONE=1 --> pág.Impar

        BOOL view1Active = zone == 0; //view1Active=1 si ZONA PAR

   ////////////////////////////////////////
   // OCULTAR / MOSTRAR BOTONES NAVEGACION //
   ////////////////////////////////////////

   if (0<selectedPage)
   {
      [BotonSiguiente setHidden:NO];
      [BotonSiguiente setEnabled:YES];
      [BotonAnterior setHidden:NO];
      [BotonAnterior setEnabled:YES];
      if ((selectedPage==[imageSet count]-1))
      {
         [BotonSiguiente setHidden:YES];
         [BotonSiguiente setEnabled:NO];
      }
   }
   else
   {
      [BotonAnterior setHidden:YES];
      [BotonAnterior setEnabled:NO];

      if ((selectedPage==[imageSet count]-1))
      {
         [BotonSiguiente setHidden:YES];
         [BotonSiguiente setEnabled:NO];
      }
   }

   ////////////////////////////////////////
   // SACAR LA IMAGEN CORRESPONDIENTE //
   ////////////////////////////////////////

   if(imagen_inicial%2==0)
   {
```

```
        NSLog(@"la imagen seleccionada es par");
        if (view1Active)// if ZONA==PAR --> nextView=view2; else -->nextView=view1
        {
            nextView=view2;
            pictosTextNext=pictosText2;
        }
        else
        {
            nextView=view1;
            pictosTextNext=pictosText1;
        }
    }
    else
    {
        NSLog(@"la imagen seleccionada es impar");

        if (view1Active)// if ZONA==IMPAR --> nextView=view1; else -->nextView=view2
        {
            nextView=view1;
            pictosTextNext=pictosText1;
        }
        else
        {
            nextView=view2;
            pictosTextNext=pictosText2;
        }
    }

            int nextpage = truePosition > currPos ? selectedPage-1 : selectedPage+1; //Nos movemos hacia la izq. o la dcha.

            if(nextpage >= 0 && nextpage < [imageSet count])
            {
                        if((view1Active && nextpage == view1Index) || (!view1Active && nextpage == view2Index)) return;

                        NSLog(@"Load next image!");

        if (metodoCarga==1)
        {
            nextView.frame = CGRectMake(nextpage*ScrollWidthPic, 0, ScrollWidthPic, ScrollHeight);

            nextView.image = [UIImage imageNamed:[[imageSet objectAtIndex:(nextpage)] valueForKey:@"nombre"]];
            [pictosTextNext setText:[[imageSet objectAtIndex:(nextpage)] valueForKey:Idioma]];
        }
        else
        {
            nextView.frame = CGRectMake(nextpage*ScrollWidth, 0, ScrollWidth, ScrollHeight);

            nextView.image = [UIImage imageWithCGImage:[[[imageSet objectAtIndex:nextpage]defaultRepresentation]
fullScreenImage]];
        }

                        if(view1Active)
            view1Index = nextpage;
                        else
            view2Index = nextpage;
            }


}

# pragma mark -
# pragma mark Memory Management

- (void)viewDidUnload
{
    [super viewDidUnload];
    self.BotonSiguiente=nil;
    self.BotonAnterior=nil;
```

```
    // Release any retained subviews of the main view.
    // e.g. self.myOutlet = nil;
}

- (void)dealloc
{
    //[cargarFotosPagActual release];
    [scroll release];
    [nextView release];
    [pictosText1 release];
    [pictosText2 release];
    [pictosTextNext release];


    [view1 release];
    [view2 release];
    [imageSet release];
    [BotonAnterior release];
    [BotonSiguiente release];
    [super dealloc];
}

- (void)didReceiveMemoryWarning
{
    // Releases the view if it doesn't have a superview.
    [super didReceiveMemoryWarning];

    // Release any cached data, images, etc that aren't in use.
}



@end
```

AGENDA:

ControlAgenda (.h y .m):

```
//
//  controlAgenda.h
//  COMUNICADOR
//
//  Created by Elisabeth Esandia Melero on 03/05/11.
//  Copyright 2011 __MyCompanyName__. All rights reserved.
//

#import <UIKit/UIKit.h>


@interface controlAgenda : UIViewController <UITabBarDelegate>{

    IBOutlet UITabBar *controlVistasAgenda;
    UIViewController *vistaMesController;
    UIViewController *vistaSemanaController;
    UIViewController *vistaDiaController;
    UIViewController *vistaActualController;

    IBOutlet UIButton * BotonAgregarEvento;
    IBOutlet UIButton * BotonEditarEvento;

}

@property(nonatomic,retain)IBOutlet UITabBar *controlVistasAgenda; // <-- connect to the tab bar in IB
@property (nonatomic, retain) UIViewController *vistaMesController;
```

```
@property (nonatomic, retain) UIViewController *vistaSemanaController;
@property (nonatomic, retain) UIViewController *vistaDiaController;
@property (nonatomic, assign) UIViewController *vistaActualController;


@property(nonatomic,retain)IBOutlet UIButton *BotonAgregarEvento;
@property(nonatomic,retain)IBOutlet UIButton *BotonEditarEvento;


- (void)activateTab:(int)index;
-(IBAction)volverMenu:(id)sender;
-(IBAction)onAgregarEvento:(id)sender;
-(IBAction)onEditarEvento:(id)sender;


@end



//
//  controlAgenda.m
//  COMUNICADOR
//
//  Created by Elisabeth Esandia Melero on 03/05/11.
//  Copyright 2011 __MyCompanyName__. All rights reserved.
//

#import "controlAgenda.h"
#import "vistaMes.h"
#import "vistaSemana.h"
#import "vistaDia.h"


@implementation controlAgenda

@synthesize controlVistasAgenda;
@synthesize vistaMesController;
@synthesize vistaSemanaController;
@synthesize vistaDiaController;
@synthesize vistaActualController;

@synthesize BotonAgregarEvento;
@synthesize BotonEditarEvento;

- (void)tabBar:(UITabBar *)tabBar didSelectItem:(UITabBarItem *)item {
          NSLog(@"didSelectItem: %i", item.tag);
    [self activateTab:item.tag];
}

- (void)activateTab:(int)index {
    switch (index) {
        case 1:
            if (vistaActualController != nil)
                [vistaActualController.view removeFromSuperview];
            vistaActualController = vistaMesController;
            if (vistaMesController == nil) {
                self.vistaMesController =
                [[vistaMes alloc] initWithNibName:@"vistaMes" bundle:nil];
            }
            [self.view insertSubview:vistaMesController.view belowSubview:controlVistasAgenda];
            break;
        case 2:
            if (vistaActualController != nil)
                [vistaActualController.view removeFromSuperview];
            vistaActualController = vistaSemanaController;
            if (vistaSemanaController == nil) {
                self.vistaSemanaController =
                [[vistaSemana alloc] initWithNibName:@"vistaSemana" bundle:nil];
```

```
            }
            [self.view insertSubview:vistaSemanaController.view belowSubview:controlVistasAgenda];

            break;
        case 3:
            if (vistaActualController != nil)
                [vistaActualController.view removeFromSuperview];
            vistaActualController = vistaDiaController;
            if (vistaDiaController == nil) {
                self.vistaDiaController =
                [[vistaDia alloc] initWithNibName:@"vistaDia" bundle:nil];
            }
            [self.view insertSubview:vistaDiaController.view belowSubview:controlVistasAgenda];
            break;
        default:
            break;
    }
}

-(IBAction)volverMenu:(id)sender
{
   [        self.view removeFromSuperview];
}

-(IBAction)onAgregarEvento:(id)sender
{

}

-(IBAction)onEditarEvento:(id)sender
{

}


- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{
    self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];
    if (self) {
        // Custom initialization
    }
    return self;
}

- (void)dealloc
{
    [super dealloc];
}

- (void)didReceiveMemoryWarning
{
    // Releases the view if it doesn't have a superview.
    [super didReceiveMemoryWarning];

    // Release any cached data, images, etc that aren't in use.
}

#pragma mark - View lifecycle

- (void)viewDidLoad
{
    [super viewDidLoad];
    controlVistasAgenda.delegate=self;
    [controlVistasAgenda setSelectedItem:[controlVistasAgenda.items objectAtIndex:0]];
    [self activateTab:1];
    // Do any additional setup after loading the view from its nib.
}
```

```
- (void)viewDidUnload
{
    [super viewDidUnload];
    // Release any retained subviews of the main view.
    // e.g. self.myOutlet = nil;
}

- (BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)interfaceOrientation
{
    // Return YES for supported orientations
        return YES;
}

@end
```