Universidad Zaragoza

Proyecto Fin de Carrera
Ingeniería Informática
Junio 2012

Creación de herramientas software de apoyo a la comunicación alternativa y aumentativa

Marta García Azpiroz

Directores:

Dra. Sandra Baldassarri
Dr. Javier Marco Rubio

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

Departamento de Informática e Ingeniería de Sistemas Universidad Zaragoza

Datos Técnicos

Título:	Creación de herramientas software de apoyo a la comunicación alternativa y aumentativa.
Autor:	Marta García Azpiroz
DNI:	72992309-S
Titulación:	Ingeniería Informática
Directora:	Dra. Sandra Baldassarri
Codirector:	Dr. Javier Marco Rubio
Departamento:	Informática e Ingeniería de Sistemas
Centro:	Escuela de Ingeniería y Arquitectura
Universidad:	Universidad de Zaragoza
Fecha:	Junio 2012

Creación de herramientas software de apoyo a la comunicación alternativa y aumentativa

RESUMEN

En el Colegio Público de Educación Especial Alborada se escolarizan alumnos que padecen graves trastornos en el habla, lo que les limita a la hora de comunicarse con el entorno que les rodea. En la gran mayoría de los alumnos se encuentran afectadas de forma importante las funciones de comunicación, así como los procesos de relación con el entorno físico y social, los de adquisición de habilidades de autonomía y, en general, los procesos básicos de aprendizaje.

Para facilitarles la expresión sin utilizar la palabra hablada surgen los llamados Sistemas Aumentativos y Alternativos de Comunicación que complementan o sustituyen el lenguaje oral. Muchos de ellos recurren al uso de imágenes y símbolos pictográficos que representen, de forma clara, las palabras y conceptos más habituales en la comunicación cotidiana.

Este Proyecto Fin de Carrera comprende el desarrollo de dos sistemas de apoyo a la Comunicación Alternativa y Aumentativa.

La primera herramienta desarrollada se denomina *AraBoard*, y consiste en un software para la creación y reproducción de tableros de comunicación que permiten trasladar el concepto tradicional de tablero a diversas plataformas informáticas y sistemas operativos como dispositivos móviles y tablets, con sistema operativo Android; y pizarras digitales y equipos informáticos convencionales, con sistema operativo Windows. *AraBoard* se compone de dos aplicaciones diferenciadas pero complementarias entre sí:

- *AraBoard Constructor*: Esta herramienta se utiliza para la creación y edición de los tableros de comunicación, mediante la colección de pictogramas ARASAAC y cualquier otra imagen y audio almacenados en el dispositivo del usuario.
- *AraBoard Player:* Esta herramienta se utiliza para ejecutar los tableros de comunicación previamente creados con *AraBoard Constructor.*

La segunda herramienta desarrollada se denomina *Acoti*, y permite la generación de actividades pedagógicas interactivas destinadas a funcionar el *tabletop* NIKVision, prototipo de superficie horizontal interactiva desarrollado por el Affective Lab y el cual se encuentra disponible en el colegio Alborada. Esta herramienta facilita la labor de los educadores del centro permitiéndoles, sin necesidad de tener conocimientos de programación, configurar las actividades de apoyo al aprendizaje utilizando sus propios recursos de imágenes o sonidos y ejecutarlos sobre el *tabletop* NIKVision.

Ambas aplicaciones han sido probadas por los usuarios finales, tanto profesores como alumnos del CPEE Alborada a lo largo de todo el proceso de desarrollo, lo que ha permitido distribuir una versión robusta de las aplicaciones.

Agradecimientos

Quiero dar las gracias a mis directores, Sandra Baldassarri y Javier Marco, por su implicación y por toda la ayuda que me han prestado a lo largo de este proyecto.

Gracias a José Manuel Marco, profesor del CPEE Alborada, por su ayuda incondicional y por haber apoyado mi trabajo con tanto entusiasmo.

Gracias a mi familia, en especial, a mis padres Jesús y Rosa, y a mi hermano Guillermo, quienes me han apoyado siempre y sin los que no habría llegado hasta aquí.

Gracias a Miguel por estar ahí siempre y enseñarme el valor del esfuerzo.

Por último, gracias a mi amigo Ismael, por su ayuda a lo largo de toda la carrera, y en general, gracias a todos los amigos que siempre me han apoyado y con los que he pasado tan buenos momentos.

Índice general

Capítulo 1. Introducción

En este primer capítulo se ofrece una visión general del proyecto. La sección 1.1 describe el contexto de desarrollo del proyecto. La sección 1.2 describe cuáles son los objetivos que han motivado la realización del mismo. Por último, en la sección 1.3 se presenta una breve descripción de la estructura que sigue la memoria y los anexos, para facilitar la lectura de la misma.

1.1 Contexto de desarrollo

Este Proyecto Fin de Carrera surge de la colaboración entre el Colegio Público de Educación Especial Alborada y el Affective Lab, del Grupo de Informática Gráfica Avanzada (GIGA) de la Universidad de Zaragoza, y tiene como principal objetivo desarrollar nuevas tecnologías que den respuestas adaptadas a las limitaciones que presentan los alumnos del colegio Alborada en los ámbitos de la comunicación, el aprendizaje y el desarrollo social.

En el Colegio Público de Educación Especial (CPEE) Alborada [1] se escolarizan alumnos/as que manifiestan necesidades, capacidades e intereses muy diferentes, si bien todos ellos presentan necesidades específicas de apoyo educativo significativas, como consecuencia de padecer de un grado de discapacidad muy severo. El desarrollo personal de cada alumno en los ámbitos cognitivo, social o comunicativo condiciona sus capacidades de relación con sus iguales, con los adultos y con el medio que le rodea. En la gran mayoría de los alumnos se encuentran afectadas de forma importante las funciones de comunicación, así como los procesos de relación con el entorno físico y social, los de adquisición de habilidades de autonomía y, en general, los procesos básicos de aprendizaje.

Es frecuente que la discapacidad que afecta a los alumnos tenga consecuencias importantes en su capacidad de comunicación. En estos casos, la comunicación debe apoyarse en el uso de recursos de distinta complejidad tecnológica como por ejemplo, tableros de comunicación, comunicadores, pulsadores, etc. En general los llamados Sistemas de Comunicación Aumentativos y Alternativos (SAAC) que permiten al alumno superar esas dificultades y transmitir, aunque sea de manera muy elemental, la esencia del mensaje, así como tomar conciencia de la posibilidad de influir en el entorno mediante intervenciones de naturaleza comunicativa.

En el CPEE Alborada es muy habitual el uso de este tipo de sistemas. Sin embargo, existen ciertas limitaciones en su adaptabilidad debidas, en su mayor parte, a que la capacidad para usar estos programas varía de un usuario a otro. Dentro de este contexto de desarrollo, se vio la necesidad de desarrollar nuevas tecnologías que diesen respuestas adaptadas a las diversas limitaciones presentes en los alumnos del colegio.

Este Proyecto Fin de Carrera (en adelante PFC) presenta un trabajo fuertemente vinculado al área de la Educación Especial y de las Tecnologías de la Información y la Comunicación (TIC), y se propone el desarrollo de herramientas educativas de apoyo a la Comunicación Aumentativa y Alternativa.

1.2 Objetivos del proyecto

El objetivo principal de este PFC consiste en desarrollar un conjunto de herramientas software de apoyo a la Comunicación Alternativa y Aumentativa (CAA) para mejorar la comunicación de los alumnos con su entorno y en sus relaciones sociales. Para lograr este objetivo, se llevaran a cabo reuniones con el personal docente del colegio con el objeto de analizar las necesidades de los alumnos y establecer las pautas y los requerimientos de las herramientas a desarrollar. En concreto, se van a desarrollar las siguientes herramientas:

- Software para tableros de comunicación que permitirá trasladar el concepto tradicional de tablero a diversas plataformas informáticas como: dispositivos móviles, tablets, pizarras digitales y equipos informáticos convencionales. Concretamente, la aplicación deberá permitir:

  - La creación de Tableros de Comunicación. De este modo, tanto padres como docentes dispondrán de una herramienta de autor con la cual podrán generar, personalizar y guardar tableros de comunicación adaptados a las necesidades y características individuales de cada niño.

  - La reproducción y visualización de Tableros de Comunicación de forma que el niño pueda interactuar con los tableros previamente generados con la funcionalidad de creación.

  Los recursos para ambas funcionalidades se obtendrán mediante accesos a bases de datos de pictogramas en internet, y permitiendo a los usuarios emplear sus propios recursos: imágenes, gráficos, audios, etc.

- Aplicación para la generación de juegos pedagógicos interactivos destinados a funcionar sobre el *tabletop* NIKVision, prototipo de superficie horizontal interactiva desarrollado por el Affective Lab y el cual se encuentra disponible en el colegio Alborada. El objetivo de esta aplicación consiste en facilitar la labor de los educadores del centro permitiéndoles, sin necesidad de tener conocimientos de programación, crear nuevos juegos de apoyo al aprendizaje utilizando sus propios recursos de imágenes, vídeos o sonidos.

Estas herramientas se diseñarán e implementarán atendiendo a los requerimientos y las directrices proporcionadas por los expertos del colegio, tanto en lo referente a objetivos y funcionalidades de las mismas, como en los métodos didácticos y dispositivos empleados con los alumnos y en sus limitaciones físicas y/o cognitivas. Además, las herramientas a desarrollar en este PFC deben atender a los principios del diseño universal, buscándose la usabilidad, la accesibilidad y la adaptabilidad a distintos niveles de discapacidad comunicativa.

1.3 Estructura de la memoria

Este documento se estructura en dos partes: una memoria que resume el trabajo realizado, y un conjunto de anexos con documentación técnica más detallada.

La memoria está dividida en los siguientes capítulos principales:

- **Capítulo 1 – *Introducción*:** Este capítulo presenta brevemente el contexto y los objetivos generales del proyecto a desarrollar y se explica la estructura de la memoria.

- **Capítulo 2 – *Conceptos previos*:** Este capítulo aborda los conceptos del ámbito de la educación especial necesarios para comprender el resto de la memoria.

- **Capítulo 3 – *Análisis del problema***: Este capítulo describe el análisis del problema a resolver. El capítulo comienza realizando un estudio de las herramientas de apoyo a la CAA disponibles actualmente y se plantean una serie de necesidades no cubiertas que se materializarán en la descripción de los escenarios a desarrollar. A continuación se describe el modelo de proceso seguido a lo largo del desarrollo del proyecto. Y finalmente, el capítulo concluye con la definición de los requisitos funcionales y no funcionales de las aplicaciones a desarrollar.

- **Capítulo 4 – *Documentación del desarrollo software de AraBoard*:** En este capítulo se presenta la herramienta software *AraBoard* que permite la construcción y reproducción de tableros de comunicación. A lo largo del capítulo se analizan en detalle las partes en que se divide el sistema, junto con la explicación de su desarrollo.

- **Capítulo 5 – *Documentación del desarrollo software de Acoti*:** En este capítulo se analizan en detalle los módulos de que consta la herramienta *Acoti*, que permite la generación de actividades pedagógicas. A lo largo del capítulo se analizan en detalle los distintos módulos del sistema, junto con la explicación de su desarrollo.

- **Capítulo 6 – *Resultados obtenidos:*** Esta sección muestra brevemente cuáles han sido los resultados logrados una vez finalizado el desarrollo completo de las aplicaciones.

- **Capítulo 7 – *Conclusiones y trabajo futuro*:** En esta sección se analiza el cumplimiento de los objetivos, se describen las posibles líneas de trabajo futuro de ambas aplicaciones, y se muestra la opinión personal de la autora sobre todo el trabajo realizado a lo largo del PFC.

La memoria principal viene acompañada de los siguientes anexos que amplían la información señalada a largo de la misma:

- **Anexo A – *Estudio de las herramientas de apoyo a la CAA***: Se explican de forma detallada todas aquellas aplicaciones de apoyo a la Comunicación Alternativa y Aumentativa que fueron estudiadas.

- **Anexo B – *Estudio de las tecnologías de desarrollo de AraBoard*:** Se describen las tecnologías que fueron estudiadas previamente al desarrollo de ambas aplicaciones.

- **Anexo C – *Gestión del proyecto*:** Describe el modelo de proceso seguido para la realización del proyecto, junto con sus hitos y fases más importantes. También se explican los aspectos más destacados seguidos durante las fases de análisis, diseño e implementación del proceso de desarrollo.

- **Anexo D – *Proceso de desarrollo del software AraBoard*:** Se muestra toda la información referente al proceso de ingeniería del software detallando las fases de análisis, diseño, implementación y pruebas de la aplicación *AraBoard*.

- **Anexo E – *Proceso de desarrollo del software Acoti*:** Se muestra toda la información referente al proceso de ingeniería del software detallando las fases de análisis, diseño, implementación y pruebas de la aplicación *Acoti*.

- **Anexo F – *Manual de usuario de AraBoard*:** Se dan las pautas a seguir para la instalación de la aplicación *AraBoard* y su correcta utilización.

Por último se incluyen los índices de figuras y tablas, y la bibliografía consultada a lo largo del desarrollo del proyecto.

Capítulo 2. Conceptos previos

En este capítulo se presentan los conceptos y la terminología necesaria para la correcta compresión de esta memoria. Estos conceptos son la base para comprender la terminología y el alcance de este proyecto. En primer lugar se describen los conceptos de Comunicación Alternativa y Aumentativa y de Sistema de Comunicación Aumentativo y Alternativo. En segundo lugar se presentan los principales Sistemas de Comunicación Aumentativos y Alternativos asistidos que se emplean en las aulas de Educación Especial.

2.1 La Comunicación Alternativa y Aumentativa

La comunicación constituye una de las formas en que las personas interactúan entre sí y es la base del aprendizaje y la adquisición de conocimientos. Existen muchas formas de comunicación: gestual, a través de signos, mediante imágenes, verbal, escrita, etc. A veces, en determinadas personas, pueden aparecer problemas en la comunicación debidos, entre otras causas, a trastornos auditivos, retrasos en su desarrollo o lesiones cerebrales, como puede ser la parálisis cerebral infantil (PCI) en el caso de los niños. Esto puede provocar graves dificultades en el habla, lo que les impide expresar sus necesidades, deseos o pensamientos de una forma adecuada. En este contexto surge el término de Comunicación Alternativa y Aumentativa.

La **Comunicación Alternativa y Aumentativa** (CAA) [2] [3] es el término general que abarca los métodos de comunicación utilizados para complementar o sustituir el habla o la escritura en personas que presentan discapacidad en la producción o la comprensión del lenguaje hablado o escrito.

Como apoyo para facilitar la comunicación, surgen los **Sistemas de Comunicación Aumentativa y Alternativa** (SAAC) [4], que son formas de lenguaje diferentes del habla, que se utilizan cuando ésta se encuentra seriamente afectada. Los sistemas aumentativos complementan el lenguaje oral cuando, por sí sólo, no es suficiente para entablar una comunicación efectiva con el entorno. Los alternativos, lo sustituyen cuando éste no es comprensible o está ausente. Son usuarios de SAAC un amplio espectro de personas con Necesidades Complejas de Comunicación (NCC). Para ellos, la CAA puede ser una herramienta de apoyo al lenguaje expresivo y/o comprensivo, un medio de expresión, o un medio de expresión y comprensión.

Los SAAC se clasifican en [5]:

- "**No asistidos**" o "**sin ayuda**": se basan en el cuerpo del usuario para transmitir mensajes y no requieren del uso de un instrumento exterior. El lenguaje de señas o deletreo es un ejemplo claro de SAAC no asistido.

- "**Asistidos**" o "**con ayuda**": precisan de un dispositivo externo que actúe como soporte del sistema. Entre los asistidos, se puede hablar de SAAC de baja, media y alta tecnología, según el soporte empleado.

A continuación se explican en detalle los SAAC asistidos que son los que se abordan en este trabajo.

2.2 Los SAAC asistidos

Existen diferentes tipos de SAAC asistidos, cuya finalidad es facilitar la comunicación considerando los distintos grados de afectación y la individualidad de cada persona. Según los materiales y el soporte que se utilicen, los métodos de comunicación pueden variar desde dispositivos que producen una salida de voz y/o producción escrita en el caso de los comunicadores, a papel y lápiz para los libros o tableros de comunicación en soporte papel [6].

Los SAAC con ayuda, o asistidos, se clasifican acorde a la tecnología implicada en el dispositivo. De esta manera, se categorizan en:

- BT: de baja tecnología, o "tablero o cuaderno de comunicación".
- MT: de media tecnología, o "comunicador portátil".
- AT: de alta tecnología, o "programa informático de comunicación".

Entre los SAAC con ayuda destacan por su popularidad los comunicadores (MT-AT) y los tableros de comunicación (BT, si son en soporte papel; AT si son reproducidos en un dispositivo informático).

Los **comunicadores** (Figura 1) son dispositivos electrónicos creados específicamente para la comunicación. Su contenido es fijo y viene predefinido. Con ellos, el usuario puede producir mensajes de forma relativamente rápida y sencilla presionando las teclas del dispositivo, y éste se encarga de transformar el mensaje asociado a cada tecla en voz.

Los **tableros de comunicación** (Figura 2) son uno de los SAAC más utilizados hoy en día. Un tablero de comunicación es una superficie donde se sitúan los elementos necesarios para la comunicación, como pueden ser pictogramas, letras, sílabas, etc. Estos elementos representan acciones, sentimientos, objetos y personas que puedan tener cierta relación con el usuario y su entorno. El usuario señala las imágenes para poder realizar demandas, comunicarse y compartir sus ideas y pensamientos.

Figura 1. Diferentes comunicadores electrónicos

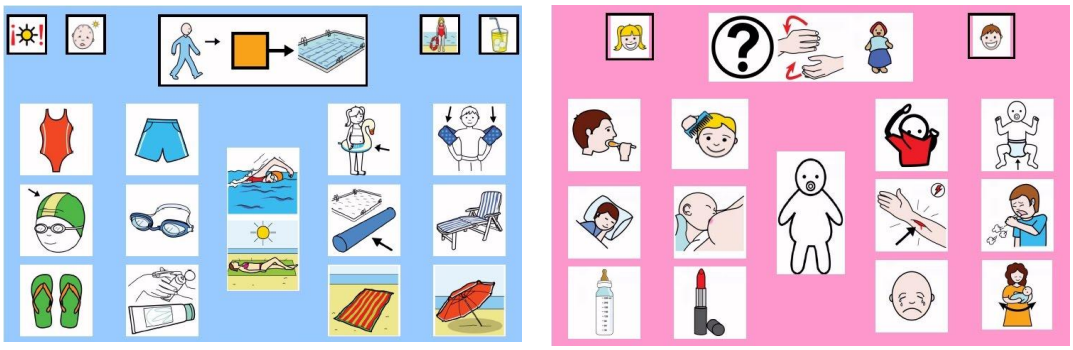


Figura 2. Tableros de comunicación impresos en papel

Los tableros de comunicación proporcionan un lenguaje visual a personas que no pueden hablar, leer, ni escribir, o tienen dificultades para hacerlo. Están constituidos por símbolos pictográficos, dibujos sencillos e iconográficos, que permiten comunicarse de una forma fácil, ya que representan las palabras y conceptos más característicos y habituales que el usuario utiliza.

Los elementos pictográficos que componen los tableros de comunicación reciben el nombre de pictogramas.

Un **pictograma**, o también denominado gráfica de imágenes o pictografía, es un diagrama que utiliza imágenes o símbolos para mostrar datos para una rápida comprensión. En un pictograma, se utiliza una imagen o un símbolo para representar una cantidad específica. Los pictogramas se clasifican por categorías, empleando un código de colores: los verbos en color verde, los sustantivos en color naranja, etc.

Los sistemas pictográficos de ARASAAC[1], SPC[2] y Bliss[3] son los más utilizados actualmente para la composición de los tableros de comunicación.

---

[1] Los pictogramas del ARASAAC fueron creados en España por el Centro Aragonés de Tecnologías para la Educación (CATEDU), profesionales del Colegio Público de Educación Especial Alborada y el diseñador gráfico Sergio Palao, en año 2008 bajo la licencia Creative Commons (BY-NC-SA).
Sus imágenes pueden descargarse del Portal Aragonés de la Comunicación Aumentativa y Alternativa, sitio: http://www.catedu.es/arasaac/index.php

[2] El Sistema Pictográfico de Comunicación (SPC) fue creado por Roxana Mayer Johnson, en 1981 en Minessotta (EEUU). Hizo dos ampliaciones en 1985, y en 1992, donde añadió nuevos símbolos. Fue desarrollado por la empresa Meyer-Jonson bajo licencia de tipo comercial. Sitio http://www.mayer-johnson.com/. Hasta la aparición del ARASAAC, éste fue el lenguaje de CAA más utilizado a nivel mundial en las instituciones educativo-terapéuticas.

Dentro de los SAAC de alta tecnología, cabe destacar un SAAC asistido, de naturaleza muy diferente a los comunicadores y los tableros de comunicación. Este sistema, que ha sido desarrollado por el Affective Lab en colaboración con el CPEE Alborada, consiste en una aplicación educativa basada en la superficie de interacción tangible (*tabletop*) denominada NIKVision (Figura 3) [7]. Los niños interactúan con la aplicación a través de la manipulación de objetos convencionales (normalmente juguetes) sobre su superficie. Estos objetos tienen asociada una representación virtual (por ejemplo una imagen del juguete) que se proyecta sobre la superficie NIKVision. De esta manera las imágenes de los objetos sirven de guía a los usuarios para realizar manipulaciones de los objetos sobre la mesa.

La aplicación está orientada a alumnos con Necesidades Complejas de Comunicación (NCC) en el marco de la Educación Especial, asistiéndoles en el proceso de adquisición de competencias comunicacionales, y en particular, a su preparación para poder llegar a utilizar otro tipo de SAAC, como por ejemplo un comunicador o un tablero de comunicación sobre soporte electrónico.

Figura 3. Alumno del CPEE Alborada interactuando sobre la superficie NIKVision

[3] El sistema Bliss fue creado por Karl Blitz (quien en Inglaterra se cambia el nombre por Charles Bliss), entre 1942 y 1965. En 1974 McNaughton del Instituto de niños discapacitados de Notario, se pone en contacto con Bliss y obtiene el Copy Right del sistema, quedando Bliss como consultor. Se crea a partir de ahí el BCIL Instituto para la comunicación Bliss simbólica y se habla de Blissimbolismo.

Capítulo 3. Análisis del problema

En este capítulo se presenta el análisis del problema a resolver. Este problema reside principalmente en la existencia de ciertas limitaciones en las aplicaciones del mercado para dar respuestas adaptadas a un grupo de usuarios tan heterogéneo como es el alumnado del colegio Alborada. Para ello, en la sección 3.1, se realiza un estudio de las herramientas de apoyo a la CAA empleadas en las aulas y en el entorno de los alumnos del CPEE Alborada. A partir de las conclusiones obtenidas en el estudio, en la sección 3.2 se presentan los escenarios que se van a desarrollar. A continuación, en la sección 3.3 se describe el modelo de proceso que se seguirá para el desarrollo del proyecto. Por último, en la sección 3.4 se enumeran los requisitos generales del proyecto derivados del estudio previo y de las reuniones con los profesores del colegio.

3.1 Estudio de aplicaciones existentes

Hoy en día, la informática y las nuevas tecnologías juegan un papel decisivo en los SAAC asistidos, concretamente, en los tableros de comunicación y en los comunicadores. Los nuevos dispositivos informáticos ofrecen un amplio abanico de posibilidades comunicativas no sólo mediante recursos gráficos, sino también mediante la inclusión de contenidos multimedia que pueden ser fácilmente personalizados.

Por otro lado, el elevado coste de los comunicadores electrónicos está haciendo emerger el desarrollo de nuevas herramientas software de autor que permiten construir y configurar comunicadores, que pueden ser ejecutados en los distintos dispositivos electrónicos actuales. Con estas herramientas software, los ordenadores convencionales, los portátiles y en particular los dispositivos móviles y tablets, se convierten en sistemas alternativos a los comunicadores electrónicos con muchas más prestaciones y a un coste mucho más económico. De igual forma, el traslado de la definición tradicional de tablero de comunicación en modo impreso a un entorno informático pone la tecnología al servicio de las personas discapacitadas, ampliando así sus posibilidades de comunicación, aprendizaje y entretenimiento a través de un acceso simplificado a los diversos dispositivos informáticos.

Este apartado presenta de manera resumida el estudio de las herramientas software de apoyo a la CAA empleadas en las aulas y en el entorno de los alumnos del CPEE Alborada. En concreto se estudian aquellas aplicaciones que permiten informatizar el concepto de comunicador y de tablero de comunicación adaptándolo a las últimas tecnologías.

Para la realización del estudio se tomó la decisión de estudiar las aplicaciones que emplean como recursos pictogramas de la colección de ARASAAC [8], ya que:

1. El alumnado del CPEE Alborada está familiarizado a trabajar con la colección de pictogramas de ARASAAC.
2. La colección de pictogramas es gratuita y tiene una gran difusión a nivel nacional e internacional.
3. Parte del personal docente del colegio Alborada son los creadores de ARASAAC.

Por otro lado, se consultó con profesionales del CPEE Alborada acerca de las aplicaciones software más populares en el entorno de los alumnos de educación especial. En base a sus opiniones y comentarios acerca de cada una de ellas, se valoraron los aspectos que debían ser mejorados en lo referente a la adaptación de las aplicaciones a los distintos perfiles de discapacidad comunicativa presentes en el alumnado del colegio.

En la Tabla 1 se presenta un resumen de las aplicaciones de tableros de comunicación más relevantes. Esta tabla incluye en primer lugar una breve descripción de la aplicación. En segundo lugar, una valoración de la sencillez de su uso, seguido del grado de personalización que permite. Y por último, las plataformas soportadas por la aplicación. El estudio detallado de las aplicaciones incluidas en la Tabla 1 se presenta con más detalle en el Anexo A de esta memoria.

El sistema de valoración que ha sido empleado para medir el **grado de sencillez de la interfaz** es el siguiente: **Difícil**, cuando resulta complicado descubrir cómo acceder a todas las funcionalidades de la aplicación, bien sea en la interfaz de autor o en la de usuario. **Medio**, cuando el uso de la aplicación es sencillo, pero la abundancia de funcionalidades complica la interfaz para un usuario con conocimientos medios-bajos de informática. **Fácil**, cuando todas las opciones de la aplicación se encuentran rápidamente y su ejecución resulta sencilla.

El sistema empleado para valorar el **grado de personalización** es: **Alto**, cuando el usuario puede incluir sus propios recursos en la aplicación, además de los recursos predefinidos, disponiéndolos de la manera que prefiera. **Medio**, cuando es posible usar una gran cantidad de pictogramas distintos y la inclusión de recursos propios no es tan sencilla. **Bajo**, cuando la disposición de los elementos viene preestablecida, o el conjunto de pictogramas empleados es limitado y no permite la incorporación de recursos propios.

Aplicación	Descripción	Sencillez de la interfaz	Grado de personalización	Plataformas			
				Windows	Mac	Android	iOS
TICO (Tableros Interactivos de Comunicación)	Consta de dos aplicaciones diferenciadas: Editor e Intérprete. Permite generar y utilizar tableros de comunicación de forma interactiva y generar frases mediante la acumulación de pictogramas.	Media	Alto	Windows XP, Vista, 7	--	--	--
In-TIC	Permite adaptar y simplificar las principales funcionalidades del dispositivo móvil o tablet a las características o necesidades de cada persona. Además, para las personas con dificultades de comunicación, también se puede configurar el dispositivo como un comunicador dinámico.	Media	Alto	Windows XP, Vista, 7	--	Sí	--
e-Mintza	Es un sistema de comunicación mediante pictogramas, distribuidos en diferentes categorías, en el que éstos se van acumulando para ser leídos como una frase con estructura normalizada.	Difícil	Alto	Windows XP, Vista, 7	Sí	Sí	--
Picto Droid Lite	El sistema de comunicación es muy sencillo y directo ya que, al pulsar sobre uno los pictogramas, la voz sintetizada del dispositivo lee el nombre del archivo de imagen introducido en cada categoría.	Fácil	Medio	--	--	Sí	--
Baluh	Proporciona una solución de comunicación para las personas que tienen dificultad para hablar mediante la creación de libros interactivos de comunicación aumentativa y alternativa.	Media	Medio - Alto	--	--	--	Iphone Ipod/ Ipad
CPA (Comunicador Personal Adaptable)	Sistema de comunicación basado en la utilización de pictogramas e imágenes con sonidos asociados. Permite editar y personalizar tus tableros de comunicación con tus propios recursos.	Fácil	Alto	Windows XP, Vista, 7	--	--	Iphone Ipod/ Ipad

Tabla 1. Resumen de las aplicaciones de tableros de comunicación estudiadas

A partir de las directrices de los profesionales del CPEE Alborada, y tomando como base las aplicaciones estudiadas y las conclusiones obtenidas (Anexo A), se deduce que este tipo de aplicaciones tienen que cumplir tres características principales:

- Adaptabilidad al mayor número de plataformas informáticas posibles, principalmente a aquellas que sean más accesibles tanto económicamente como funcionalmente.
- La construcción de los tableros debe ser rápida y sencilla, y debe permitir la incorporación de recursos con los que los usuarios están más familiarizados (audios/imágenes de su vida cotidiana).
- Adaptabilidad y accesibilidad del mayor número de usuarios posibles, y diseñar las aplicaciones con el fin de abarcar a cuantos más colectivos sea posible.

De este estudio y de las carencias que presentan algunas de las aplicaciones estudiadas a la hora de adaptar una misma herramienta a un grupo tan heterogéneo de usuarios, surgen los dos escenarios que se presentan a continuación.

3.2 Definición de escenarios

Una vez establecidas las líneas que guían este proyecto en el marco de la CAA, y en las reuniones establecidos con el cliente (profesorado del CPEE Alborada), se explican los escenarios que se van a desarrollar.

El trabajo realizado se centra en el desarrollo de dos escenarios diferentes: en primer lugar, una herramienta software que permite construir y reproducir tableros de comunicación; y en segundo lugar, una aplicación para la generación de juegos pedagógicos interactivos destinados a funcionar sobre superficies interactivas. A continuación se expone la dinámica de ambas herramientas como escenarios donde implantar las conclusiones obtenidas con el cliente.

3.2.1 AraBoard: Construcción y reproducción de tableros de comunicación

Esta primera herramienta consiste en un software para crear y visualizar Tableros de Comunicación que permiten trasladar el concepto tradicional de tablero a las diversas plataformas informáticas actuales: dispositivos móviles, tablets, pizarras digitales y equipos informáticos convencionales.

La aplicación se divide en dos interfaces diferenciadas: por un lado, la interfaz de **usuario** final, correspondiente a la reproducción y visualización de tableros de comunicación, y por otro lado, la interfaz del **tutor**, donde los tutores del usuario (familiares, profesores, terapeutas o cualquier otro personal de apoyo y, en algunos casos particulares, el propio usuario final) pueden crear y editar tableros de comunicación, y adaptarlos a las necesidades específicas de cada persona.

Para facilitar la construcción y personalización de los tableros, la aplicación de construcción/edición permite incluir pictogramas de la colección de de ARASAAC mediante un buscador de pictogramas. Este factor la diferencia resto de aplicaciones estudiadas, ya que no es necesario almacenar la base de datos de pictogramas en la máquina del usuario, puesto que está en la web. La principal ventaja de este factor es que la base de datos siempre va a estar actualizada. En el mismo momento en que un pictograma sea modificado desde el portal web de ARASAAC, este pictograma estará disponible desde nuestra aplicación. La desventaja que introduce es la necesidad de trabajar en red para buscar los recursos. Sin embargo, la ausencia de conexión a Internet, no impide que los usuarios hagan uso de la aplicación, ya que la aplicación también permite que los usuarios incorporen sus propias imágenes, gráficos o ficheros de sonido que consideren apropiados, permitiendo de esta manera un gran número de posibilidades a la hora de personalizar y adaptar los tableros de comunicación y no depender únicamente del acceso a la colección de ARASAAC.

3.2.2 Acoti: Generación y ejecución de actividades pedagógicas

La segunda herramienta desarrollada permite la creación y ejecución de actividades pedagógicas interactivas destinadas a funcionar el *tabletop* NIKVision, y el cual se encuentra disponible en el laboratorio Affective Lab y en el CPEE Alborada.

Esta herramienta facilita la labor de los educadores del centro permitiéndoles, sin necesidad de tener conocimientos de programación, configurar las actividades de apoyo al aprendizaje utilizando sus propios recursos de imágenes o sonidos y ejecutarlos sobre el *tabletop* NIKVision.

Las actividades que pueden ser configuradas son actividades de aprendizaje de abstracción y clasificación de objetos. El tutor puede crear cualquier tipo de actividad que consista en definir una o varias áreas en una determinada zona de la superficie tangible, y asignar a cada área un conjunto de juguetes correctos. De esta manera, el profesorado puede elaborar actividades ajustadas a los intereses del alumno o grupo de alumnos que vayan a participar en la actividad, y guardarlas en ficheros independientes para usarlas posteriormente. La aplicación busca la participación activa del alumno, proporcionándole respuestas inmediatas a sus acciones, reforzando el valor del éxito y del fracaso frente al objetivo a completar.

Ambos escenarios se explicarán de manera más detallada a lo largo de los siguientes capítulos y anexos de la memoria.

3.3 Modelo de proceso de desarrollo

El primer paso para el desarrollo de los sistemas fue la elección del modelo de proceso software a seguir, seleccionando el *modelo incremental*, explicado con más detalle en el Anexo C.

El **modelo incremental**, descrito en [9], consiste en ir realizando secuencialmente sucesivas modificaciones o extensiones a la aplicación hasta obtener el sistema final. La elección de este modelo en contraposición con otros modelos analizados como el de cascada

o el evolutivo se realizó porque combina las ventajas de éstos y es el que más se adapta al desarrollo de la aplicación.

La principal característica de este modelo es que permite crear cada vez versiones más completas del software, para ello se construyen versiones sucesivas de la aplicación. Se crea una primera versión que utiliza el usuario (los profesores del colegio), y éste provee retroalimentación al desarrollador, y según los requerimientos especificados de este usuario se crea una segunda versión. El desarrollo del sistema deberá ir pasando por diversas fases que podrán ir desarrollándose en paralelo y modificándose a gusto de los profesores del colegio e incluso añadir nuevas funcionalidades en cada versión.

El siguiente paso consistió en llevar a cabo el análisis del sistema, explicado detalladamente en los Anexos D.1 y E.1. Como metodología a aplicar se eligió OMT (Object-Modeling Technique) [10]. Se ha creído oportuno añadir la fase de análisis de requisitos, propia de la metodología UML, ya que en nuestro caso permite estudiar más a fondo las necesidades de la aplicación. Por último, para realizar el diseño del sistema se siguió también la metodología OMT. El proceso en detalle se presenta en los Anexos D.2 y E.2.

3.4 Análisis de Requisitos

La primera etapa del desarrollo software consistió en establecer los requisitos que debían cumplir las aplicaciones para así conseguir una mayor comprensión del problema a resolver. Para ello, se realizaron reuniones con los directores del proyecto y con los profesores y especialistas en CAA del CPEE Alborada, en las se establecieron los objetivos y funcionalidades que debía cubrir cada una de las aplicaciones por separado, en función de las características de los usuarios potenciales de CAA y de la metodología de enseñanza y aprendizaje desarrollada en las aulas.

Los requisitos recogidos para cada herramienta sufrieron varias modificaciones a lo largo del desarrollo del software correspondiente. A continuación se listan de forma resumida los requisitos tal y como quedaron en su versión final, agrupados en funcionales y no funcionales. El listado detallado de los requisitos funcionales y no funcionales de cada aplicación se encuentra en los Anexos D (sección D.1.1) y E (sección E.1.1).

<u>RF de la herramienta de construcción y reproducción de Tableros de comunicación:</u>

1. Permitir la creación, edición, carga y almacenamiento de tableros de comunicación.
2. Permitir la visualización y reproducción de los tableros de comunicación creados.
3. Facilitar la personalización de los tableros mediante la modificación de sus dimensiones, títulos, fondo, y recursos que contiene.
4. Posibilitar la incorporación de los pictogramas de la colección de ARASAAC como recursos del tablero.
5. Flexibilidad para integrar dichos pictogramas en la aplicación, así como posibilitar su edición y eliminación.

6. Flexibilidad para crear nuevos pictogramas e insertar en ellos recursos propios, tanto gráficos como auditivos, según las necesidades del usuario.
7. Garantizar la ejecución de la aplicación sobre las siguientes plataformas: PCs convencionales, pizarras digitales, dispositivos móviles y tablets.
8. Garantizar que los tableros creados con las distintas distribuciones de la aplicación para cada sistema operativo/plataforma sean compatibles entre sí.

RF de la herramienta de creación y ejecución de actividades pedagógicas:

1. Permitir a los profesores la configuración de actividades de clasificación y abstracción de cualidades de objetos, destinadas a funcionar sobre el *tabletop* NIKVision.
2. El sistema permitirá configurar las actividades sin necesidad de programación, mediante la edición de ficheros XML de configuración.
3. El sistema permitirá definir cualquier tipo de actividad que consista en definir una o varias áreas sobre la mesa en una determinada posición, y asociar un conjunto de objetos correctos a cada una de ellas.
4. Flexibilidad en la elaboración de las actividades, puesto que en el colegio existen distintos perfiles de alumnos discapacitados, para los que tendrá que ser adaptada.
5. Flexibilidad para incluir recursos propios: imágenes, pictogramas y sonidos.
6. El sistema permitirá identificar las acciones de poner/modificar/añadir un objeto sobre *tabletop*.
7. Las actividades deben reforzar el valor del éxito y del fracaso frente al objetivo a completar. Para ello debe proporcionar *feedback* visual y auditivo asociado a la acción que el alumno haya realizado el alumno sobre la mesa.

RNF comunes a ambas herramientas:

1. Las aplicaciones serán implementada en el lenguaje en el *ActionScript* 3.0.
2. Para la gestión y almacenamiento de las actividades se empleará la tecnología XML. El fichero XML referenciará a los ficheros de audio y de imagen necesarios para el tablero de comunicación o actividad.
3. Junto al fichero XML se guardarán las carpetas que almacenarán los ficheros de imágenes y los ficheros de audio.
4. Los formatos de imagen que serán soportados por la aplicación son PNG y JPEG.
5. Los formatos de audio que serán soportados por la aplicación es MP3.
6. La aplicación de construcción/reproducción de tableros de comunicación necesita conexión a Internet para tener acceso a la colección de pictogramas de ARASAAC.

7. El sistema de construcción/reproducción de tableros de comunicación podrá ser ejecutado en distintas plataformas y dispositivos. Y la interfaz del sistema estará adaptada a las características del dispositivo.
8. La interfaz del sistema de configuración de actividades didácticas estará adaptada a las características de la superficie interactiva o *tabletop* NIKVision.

Capítulo 4. Desarrollo de la herramienta AraBoard

En este capítulo se describe el proceso de desarrollo de la herramienta de creación y edición de tableros de comunicación, denominada *AraBoard*. En la sección 4.1 se presenta la visión general del sistema. En la sección 4.2 se explica la visión general del sistema y las tecnologías empleadas para desarrollarlo. En las sucesivas secciones se explica y justifica el papel de cada una de las aplicaciones que la conforman: la aplicación de construcción y edición de tableros, y la aplicación de reproducción y visualización. Para cada una se explican sus aspectos de diseño de alto nivel. En el Anexo D se ofrece información detallada de las distintas fases (análisis, diseño, implementación, pruebas, etc.) del proceso de desarrollo del sistema.

4.1 Visión general del sistema

El entorno de la aplicación *AraBoard* se puede observar en la Figura 4. En ella se observa como los usuarios pueden interaccionar con *AraBoard* a través de diversos dispositivos como PC, móvil o tablet.

En el centro de la figura se observa la aplicación *AraBoard*, que reside dentro del dispositivo del usuario. En ella se distinguen dos partes diferenciadas: la aplicación de construcción y personalización de tableros de comunicación, *AraBoard Constructor*, donde los tutores del usuario (familiares, profesores, terapeutas o cualquier otro personal de apoyo y, en algunos casos particulares, el propio usuario final) podrán crear y configurar tableros de comunicación, y adaptarlos a las necesidades específicas de cada persona. Y por otro lado, la aplicación de visualización de tableros de comunicación, *AraBoard Player,* que será utilizada por usuarios (generalmente niños) que presenten algún tipo de discapacidad comunicativa. Como se puede apreciar en la esquina inferior izquierda de la figura, *AraBoard* se comunica directamente a través de Internet con el servidor del portal web de ARASAAC [8]. Esta comunicación permite la búsqueda de cualquier pictograma contenido en la base de datos y su descarga instantánea junto con la locución correspondiente.

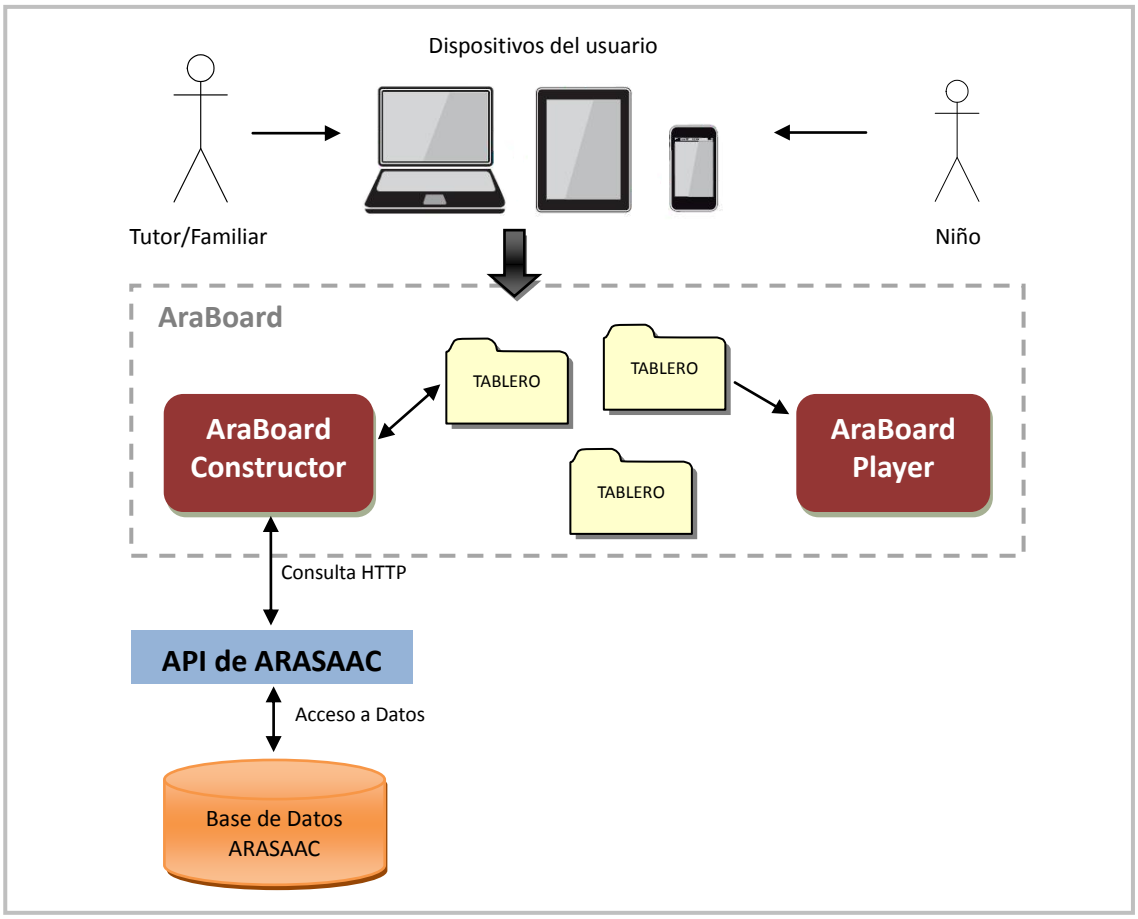


Figura 4. Entorno de aplicación del sistema AraBoard

La arquitectura de las aplicaciones *AraBoard Constructor* y *AraBoard Player* se detallará a lo largo de las siguientes secciones.

4.2 Arquitectura del sistema

El funcionamiento global del sistema durante proceso de construcción y visualización de los tableros de comunicación se puede ver en la Figura 5, en la que se observan los distintos módulos que la componen y su relación.

El sistema *AraBoard* consta de dos aplicaciones independientes y diferenciadas, pero complementarias entre sí: *AraBoard Constructor* y *AraBoard Player*.

La aplicación ***AraBoard Constructor*** posibilita la creación y edición de tableros de comunicación adaptados a las necesidades particulares de cada usuario.

Desde esta aplicación es posible definir y configurar todos los elementos -visuales, auditivos, textuales, de apariencia, etc.- que componen el tablero de comunicación, así como modificar sus dimensiones (número de filas y columnas y disposición de celdas).

Para la personalización de los elementos que componen el tablero, la aplicación permite que el usuario incorpore:

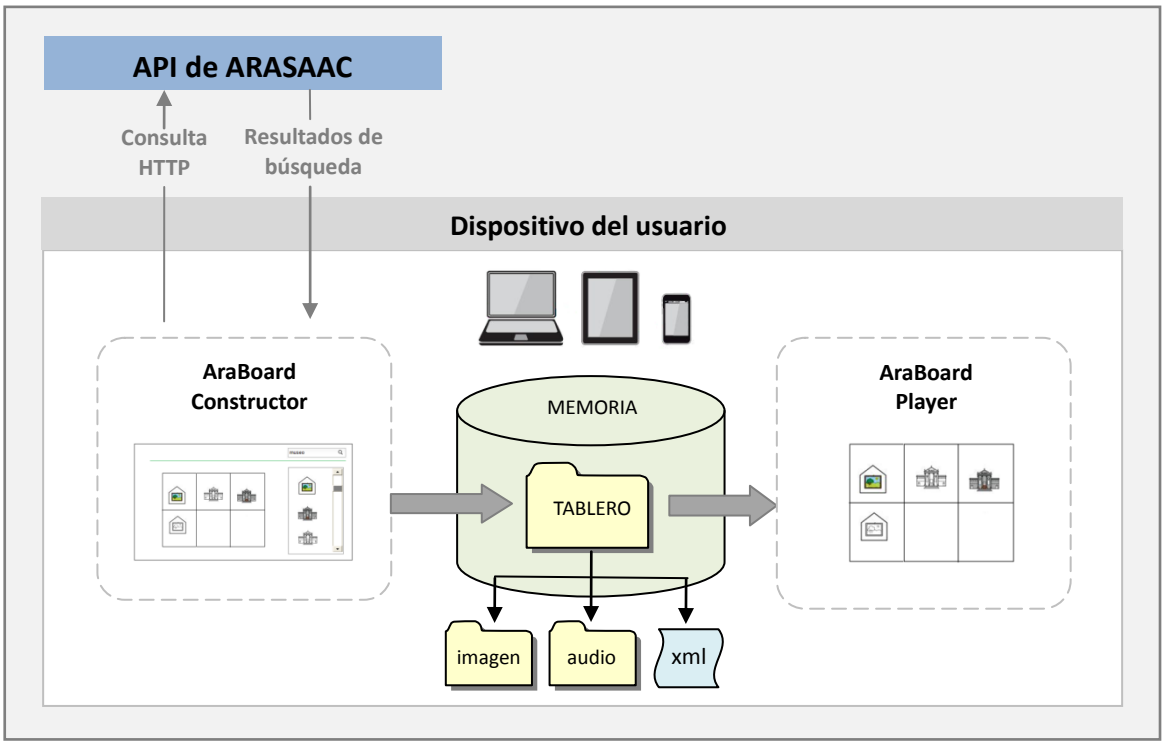


Figura 5.Esquema del proceso de construcción y visualización de un tablero de comunicación

- Recursos procedentes de la colección de pictogramas de ARASAAC: Para la obtención de estos recursos, la aplicación necesita comunicarse con la API de ARASAAC para que le transfiera los resultados de las búsquedas coincidentes con la palabra y el idioma que el usuario haya introducido a través del buscador de la interfaz. Para establecer esta comunicación, la aplicación envía una consulta http a la API, especificando el idioma y la palabra a buscar, y ésta le proporciona todos los datos coincidentes, que son tratados por la aplicación y mostrados al usuario.

- Recursos propios: Para la incorporación de recursos propios, la aplicación facilita la carga de imágenes y de audio almacenados en la máquina del usuario.

Una vez se ha construido el tablero, la aplicación permite guardar el proyecto del tablero de comunicación, creando un directorio en la máquina del usuario donde se almacena un fichero XML con la estructura del tablero, junto con todos los elementos que lo componen (audios e imágenes). Los tableros de comunicación construidos serán utilizados por la aplicación *AraBoard Player*, que permitirá al usuario interactuar con ellos.

Por otro lado, ***AraBoard Player*** es una aplicación que permite al usuario final utilizar los tableros de comunicación generados previamente por la aplicación *AraBoard Constructor*. El usuario interactúa con el tablero señalando las distintas celdas que lo componen, y una vez se ha señalado una celda, se ejecuta el audio asociado a ella. El mecanismo de reproducción de audio está diseñado de manera que aunque no se señale con exactitud la celda deseada, siempre se ejecutará el audio asociado a la celda más cercana a la zona en la que se ha pulsado.

Otro aspecto importante del sistema es su adaptación al mayor número de plataformas posible, tanto en lo referente a apariencia gráfica como en lo referente a funcionalidades dependientes del sistema operativo. En este PFC se han desarrollado dos versiones de la aplicación:

- Versión para dispositivos móviles y tablet con sistema operativo Android.
- Versión para pizarras digitales y ordenadores convencionales con sistema operativo Windows.

La elección del sistema operativo Android para llevar a cabo la aplicación sobre plataformas móviles y tablet, fue una decisión tomada conjuntamente con el cliente, en la que se valoró principalmente el coste económico de los dispositivos físicos (Smartphone y tablet, frente a Iphone e Ipad).

Para poder llevar a cabo el sistema con soporte multiplataforma se realizó un estudio previo de las tecnologías de desarrollo (que se explica detalladamente en el Anexo B), que permitiesen por un lado, la fácil adaptación de las aplicaciones de escritorio a aplicaciones móviles; y por otro lado, el acceso al sistema de archivos y al almacenamiento local sin restringir dónde y cómo se almacenan y se accede a los datos guardados en la máquina del usuario.

4.3 Implementación de AraBoard

Para llevar a cabo la implementación de *AraBoard* se dividieron en tareas las funcionalidades que había que implementar para cumplir los requisitos fruto de las reuniones y visitas al CPEE Alborada. En las reuniones con el cliente se decidió separar la herramienta *AraBoard* en dos aplicaciones diferenciadas en función del usuario final.

4.3.1 Implementación de AraBoard Constructor

Las principales tareas realizadas durante la implementación de la aplicación *AraBoard Constructor* se detallan a continuación siguiendo el orden de su desarrollo. Sin embargo, muchas de estas tareas se han solapado en el tiempo y han requerido un perfeccionamiento conforme se iban probando a lo largo de las iteraciones del modelo de proceso incremental que se ha seguido para el desarrollo de este sistema.

Tarea 1. Conexión con la API de ARASAAC y gestión de los datos.

Para facilitar al usuario realizar búsquedas en la colección de pictogramas de ARASAAC, la aplicación incorpora un buscador de pictogramas en varios idiomas. El usuario introduce una palabra o frase de búsqueda, y ésta se envía dentro de una consulta http a la API de ARASAAC. Esta consulta permite fijar varios criterios de búsqueda como:

- Idioma de los pictogramas
- Formato en color o en blanco y negro
- Si el pictograma contiene la palabra clave de búsqueda, si coincide exactamente, etc.

- Número máximo de resultados
- …

Una vez se ha construido y enviado la consulta, la API de ARASAAC devuelve la información correspondiente a todos los pictogramas coincidentes con la palabra buscada: URLs de los archivos de audio e imagen, así como un conjunto de metadatos asociados (categoría, texto asociado, etc.). Los datos devueltos por el servicio de la API son parseados (ya que su formato es cadena de caracteres), y con ellos se compone una lista de pictogramas resultados de búsqueda, que el usuario podrá visualizar e incorporar al tablero.

El principal beneficio de este mecanismo es que no es necesario tener la base de datos descargada en el dispositivo del usuario, sino que está disponible a través de la web, y por tanto, en el mismo instante que un nuevo pictograma se añade el portal web de ARASAAC, es accesible desde *AraBoard*. La desventaja de este mecanismo reside es la necesidad de conexión a Internet, lo cual tampoco resulta un impedimento para el usuario puesto que también puede incorporar recursos desde su dispositivo.

Tarea 2: Implementación de la interfaz gráfica.

Para realizar la interfaz gráfica se decidió emplear la librería de componentes gráficos *MinimalComps* [11], ya que sus componentes son compatibles con aplicaciones de escritorio y con aplicaciones móviles. Tras realizar el estudio de la librería, algunos componentes tuvieron que ser modificados debido a que, o bien su apariencia, o bien su comportamiento, no se ajustaba a las necesidades de las aplicaciones. Por otra parte, fue necesario crear nuevos componentes, cuando los existentes no se ajustaban a los requerimientos gráficos de la interfaz. Las modificaciones realizadas sobre la librería de componentes gráficos se detallan en el Anexo D, sección D.2.2.

Debido a que la aplicación está disponible para versiones con SO Windows y Android, la interfaz gráfica fue adaptada a ambas plataformas. La principal diferencia de la interfaz gráfica para Android y para Windows es la barra de opciones del menú. En el primer caso, el menú aparece visible en la pantalla cuando se presiona la tecla menú del dispositivo; mientras que en el caso de aplicaciones de escritorio, el menú se encuentra situado en parte superior de la aplicación.

Por otro lado, surgieron problemas la versión Android debido a la interacción táctil de los usuarios con la aplicación. Para solventaron se estudiaron otros componentes gráficos adaptados a dispositivos con pantallas táctiles, concretamente la librería *Thankmister* [12].

El diseño de ventanas de ambas aplicaciones se muestra en el Anexo D, sección D.2.3.

Tarea 3. Programación del comportamiento del tablero y sus elementos.

Los elementos principales de un tablero de comunicación son los pictogramas. Los pictogramas pueden formar parte de los resultados de búsqueda, o bien estar posicionados en las celdas del tablero. El usuario puede realizar las siguientes acciones con ellos:

- Situarlos en una celda vacía de tablero
- Moverlos de una celda a otra del tablero
- Reproducir el audio asociado a ellos
- Crear un nuevo pictograma a partir de una celda vacía

- Editar su contenido mediante el arrastre de los pictogramas sobe el icono de edición
- Eliminarlos mediante su arrastrarte sobre el icono de la papelera

El comportamiento del tablero es dinámico, de manera que sus propiedades -número de filas, de columnas, color de fondo, color de letra, contenido- se pueden modificar en cualquier momento de forma rápida. Un tablero se puede construir desde cero o a partir de un tablero existe, por lo que se distinguen las siguientes acciones asociadas a los tableros: Crear un nuevo tablero, editar un tablero existente y guardar un tablero. La manera en la que se muestran estas opciones al usuario varía de la versión Android a la versión PC debido a las restricciones de la organización del sistema de directorios del SO.

Tarea 4: Acceso al sistema de ficheros del sistema operativo: Almacenamiento y Carga.

El sistema operativo impone restricciones a la hora de implementar la funcionalidad de carga y almacenamiento en ambas versiones.

Así como en la versión PC el usuario puede seleccionar dónde se guardará y de dónde se cargará su tablero, esto no es posible en la versión Android, ya que la navegación por directorios no está soportada. Para solventar este problema, se realizó un estudio de la estructura de los ficheros y directorios en Android [13] y se optó por desarrollar dos enfoques diferentes debido a esta incompatibilidad. En la versión Android se optó por la creación de un directorio fijo en la carpeta de datos de usuario, y ahí se almacenarán/cargarán todos los tableros de comunicación. Para la versión para PC, se muestra una ventana de diálogo en la que el usuario puede seleccionar la ubicación de su tablero.

Tarea 5. Almacenamiento de los tableros de comunicación en ficheros XML.

Para el almacenamiento de los tableros se ha utilizado la tecnología XML [14], ya que permite un acceso rápido, sencillo y estructurado a los datos de cada tablero.

Cada tablero de comunicación se almacena en un fichero XML de nombre "*tablero_comunicación.xml*". Desde el fichero XML se referencia a cada uno de los recursos gráficos y los audios que componen cada una de las celdas del tablero. Estos recursos se almacenan en dos directorios separados: los ficheros de imagen en el directorio "*\imágenes"*; y los ficheros de en el directorio "*\audios"*. La descripción detallada de la estructura de un fichero XML de actividad se muestra la sección D.3.2.

Una vez fijada la tecnología a emplear y la estructura de los ficheros, se implementaron las funcionalidades de carga y almacenamiento de tableros. Ambas funcionalidades son dependientes del sistema de ficheros del SO, y por tanto varían de la versión PC (en la que se puede elegir la ubicación de almacenamiento) a la versión Android (en la que el almacenamiento se realiza en una carpeta predeterminada por la aplicación).

4.3.2 Implementación de AraBoard Player

La aplicación *AraBoard Player* es complementaria a la aplicación *AraBoard Constructor* y permite que el usuario pueda visualizar los tableros creados con la herramienta de construcción. Las distintas tareas realizadas durante la implementación de esta aplicación se detallan a continuación siguiendo el orden de su desarrollo.

Tarea 1. Extracción de los tableros a partir del fichero XML.

Esta tarea consiste en desarrollar un módulo que permite leer y tratar el contenido definido en el fichero XML del tablero de comunicación. Este módulo se encarga de recorrer el fichero y de extraer sus propiedades, las de sus celdas y acceder a los directorios donde se almacenan los recursos de cada celda. A partir de la información extraída de cada elemento, se programaron las funciones correspondientes para crear cada tipo de componente y fijar su comportamiento en la aplicación.

Tarea 2. Implementación de las celdas del tablero.

Esta tarea tuvo por objetivo programar la visualización y el comportamiento de las celdas cuando son pulsadas por el usuario.

El comportamiento de las celdas se ha programado según las especificaciones proporcionadas por los profesores del CPEE Alborada, de forma que los tableros de comunicación también puedan ser usados por personas que presenten trastornos en la motricidad. De esta manera, aunque el usuario no señale con exactitud la celda deseada, siempre se ejecutará el audio asociado a la celda más cercana a la zona en la que se ha pulsado. La celda más cercana se determina mediante el cálculo de la *distancia Euclídea*[4] entre las coordenadas donde se ha pulsado, y los centros de todas las celdas del tablero, y seleccionando la menos de todas ellas.

Tarea 3. Implementación y adaptación de la interfaz gráfica.

De la misma manera que en la aplicación *AraBoard Constructor*, la interfaz gráfica se debe adaptar al dispositivo en el que se va a ejecutar.

En el caso de la aplicación para Windows, el menú de opciones se encuentra disponible en la parte superior de la aplicación. En el caso de la aplicación Android, es accesible por medio del botón menú del dispositivo. Además, la versión para Android es capaz de detectar la rotación automática del dispositivo, y redimensiona el tablero y todos sus componentes de acuerdo al movimiento realizado por el usuario.

Por último, es importante señalar que los tableros de comunicación son independientes de la plataforma sobre la que se han creado, es decir, un tablero creado con la aplicación *AraBoard Constructor* para Windows puede ser reproducido por la aplicación *AraBoard Player* para Android, y viceversa.

[4] En un espacio bidimensional, la distancia euclidiana entre dos puntos P_1 y P_2, de coordenadas (x_1, y_1) y (x_2, y_2) respectivamente, es: $De\ (P1, P2) = \sqrt{(x2 - x1)2\ + (y2 - y1)2}$

Capítulo 5. Desarrollo de la herramienta Acoti

En este capítulo se describe el proceso de desarrollo de la herramienta software de generación de actividades pedagógicas, denominada *Acoti*. En la sección 5.1 se presenta el entorno de aplicación del sistema. En la sección 5.2 se explica la visión general del sistema y las tecnologías empleadas para desarrollarlo. En las sucesivas secciones se explica y justifica el papel de las tareas que la componen. Para cada una se explican sus aspectos de diseño de alto nivel. En el Anexo E se ofrece información detallada de las distintas partes del proceso de desarrollo del sistema.

5.1 Entorno de aplicación del sistema

El sistema *Acoti* permite la creación y ejecución de actividades pedagógicas interactivas destinadas a funcionar sobre el *tabletop* NIKVision [7], superficie horizontal interactiva desarrollada por el Affective Lab del grupo GIGA de la Universidad de Zaragoza. La superficie NIKVision está diseñada principalmente para que niños pequeños puedan aprender y jugar utilizando objetos y juguetes con los que están acostumbrados a interactuar.

En la figura 6 se pueden observar los elementos de la arquitectura hardware de NIKVision. Para el reconocimiento y seguimiento de los objetos dispuestos en la mesa (Figura 6 - 1), se ha adoptado el *framework* ReacTIVision [15] [16], el cual analiza la imagen capturada por una cámara de vídeo colocada en el interior de la mesa (Figura 6 - 2) con el fin de detectar visualmente los objetos colocados en la superficie. La ventaja de este sistema es que no requiere añadir electrónica especial en los objetos a ser utilizados en las aplicaciones, ya que basta con pegar un marcador impreso, o *fiducial* (Figura 7), en la base del objeto seleccionado para la interacción de manera que pueda ser identificado por ReacTIVision.

Tanto el software de reconocimiento visual, como el software de las aplicaciones informáticas, se ejecutan en un ordenador convencional convencional (Figura 6 - 3), que gestiona las salidas de la imagen digital de NIKVision: una en la superficie de la mesa

donde tiene lugar la manipulación de los objetos (Figura 6 - 4 y 5) y la otra a través de un monitor colocado en la mesa frente a los niños (Figura 6 - 6).

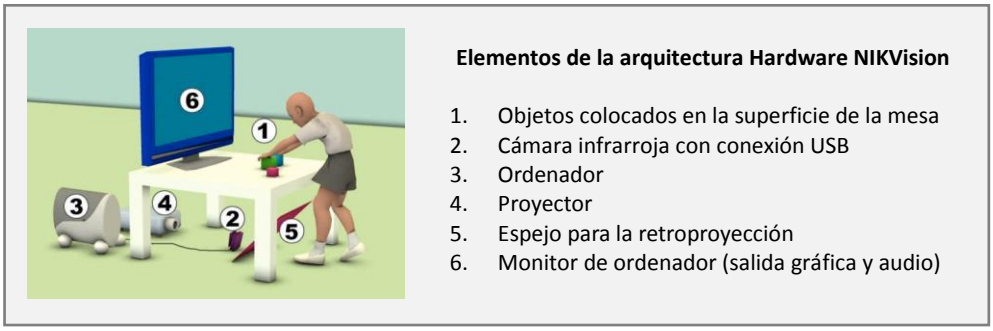


Figura 6.Arquitectura hardware de NIKVision

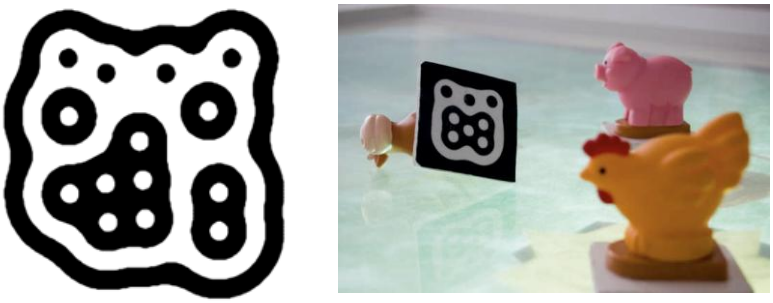


Figura 7. Izquierda: Fiducial de ReacTIVision. Derecha: Fiducial pegado a la base de un juguete.

La figura 8 muestra los componentes de la arquitectura software de NIKVision. En ella se distinguen tres partes diferentes:

- ***Tabletop* NIKVision**: Superficie horizontal interactiva sobre la que los usuarios interactúan colocando objetos. Sobre dicha superficie se proyecta la aplicación que sirve de guía para las manipulaciones de los objetos sobre la mesa. Los objetos colocados sobre su superficie son captados por una cámara web, que se encarga de enviar las imágenes a un software de reconocimiento visual.

- **ReacTIVision**: Permite abstraernos del acceso al hardware y de los algoritmos de detección visual de objetos e interacciones del usuario sobre la mesa. Esto se consigue mediante la comunicación con el entorno de desarrollo Adobe Flash, a través del envío de mensajes mediante el protocolo de comunicación TUIO [17] [18], basado en el envío de paquetes UDP.

- **Aplicación en el entorno Adobe Flash (*Acoti*):** La aplicación *Acoti* es la encargada de cargar y gestionar todos los elementos que componen una actividad o juego, así como de gestionar el transcurso de la misma en función de las acciones realizadas por el usuario. Para detectar y programar el comportamiento asociado a dichas acciones, *Acoti* establece comunicación con las interfaces del protocolo TUIO, que le permitirán identificar si se ha añadido, eliminado, o modificado un juguete de la superficie del *tabletop*, y cuáles son sus características actuales (*fiducial*, posición en pantalla,…).

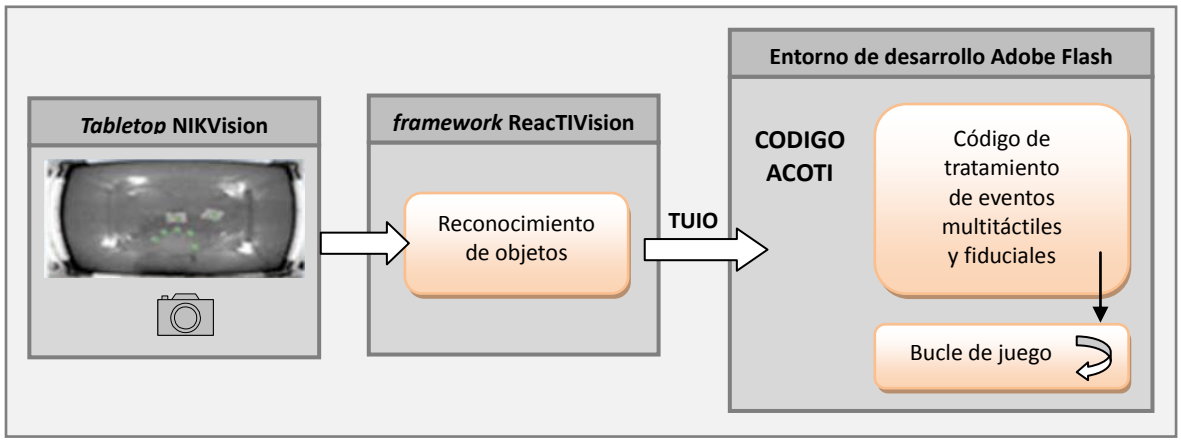


Figura 8.Arquitectura software de NIKVision

Este PFC se centra en la última etapa de la arquitectura software de NIKVision mediante la implementación de la herramienta software *Acoti*.

5.2 Visión general del sistema

La aplicación *Acoti* es una herramienta que permite configurar juegos y actividades pedagógicas interactivas destinadas a funcionar sobre el *tabletop* NIKVision. Las actividades que se pueden diseñar con esta aplicación son juegos didácticos de abstracción y clasificación de objetos. De manera general, se puede crear cualquier actividad que consista en definir una o varias aéreas en una determinada posición del *tabletop*, y asociar a cada una de ellas un conjunto de juguetes correctos en dicha área. Cada uno de los juguetes lleva asociado en su base un identificador o *fiducial*, que será identificado por el software de ReacTIVision, y accesible desde la aplicación *Acoti* a través de las funciones de las interfaces TUIO.

En la dinámica de interacción de esta actividad, el alumno aborda la tarea de completar las áreas visibles en la representación gráfica del escenario proyectado sobre la superficie del *tabletop*, ubicando los objetos en las zonas adecuadas. Para indicar a los alumnos si están jugando de forma correcta o incorrecta se les proporciona *feedback* gráfico y auditivo, que les sirve de guía para completar la actividad, y refuerza el concepto de éxito o los guía para un nuevo intento.

El software desarrollado durante este proyecto permite la configuración de actividades para el *tabletop* sin necesidad de programación, mediante la edición ficheros XML de configuración de la actividad. De esta manera se permite la variación del nivel de dificultad de las actividades según el alumno.

Para el almacenamiento de las actividades se ha optado por utilizar la tecnología XML [14], ya que permite un acceso rápido, sencillo y estructurado a los datos de cada actividad. Cada uno de estos ficheros XML almacena una secuencia de tareas que el niño debe completar para superar la actividad. Dentro de cada tarea puede haber una o más áreas de juego definidas, que contienen una lista de *fiduciales* correctos e incorrectos. La descripción de la estructura de un fichero XML de actividad se muestra en el Anexo E (sección E.3.2). Cuando el niño manipula un objeto sobre la mesa, la aplicación comprueba

si el *fiducial* del objeto manipulado está colocado dentro del área a la que pertenece, y en función del resultado de esta comprobación actualiza el estado del juego.

5.3 Implementación de Acoti

Para llevar a cabo la implementación de *Acoti* se estableció una metodología que consistió en dividir en tareas las funcionalidades que había que implementar para cumplir los requisitos fruto de las reuniones y visitas al CPEE Alborada. Las principales tareas realizadas durante la implementación de la aplicación *Acoti* se detallan a continuación.

Tarea 1. Extracción de los elementos del fichero XML.

Esta tarea consiste en desarrollar un módulo que permite leer y tratar el contenido definido en el fichero XML. Este módulo se encarga de recorrer el fichero y de extraer los elementos que lo componen: tareas, fondos, áreas, *feedback* y *fiduciales*. A partir de la información extraída de cada uno, se programaron las funciones correspondientes para crear cada tipo de componente y fijar su comportamiento en la actividad.

Tarea 2. Programación del comportamiento de los elementos definidos en el XML.

Los elementos que componen una actividad pueden tener distintos comportamientos asociados en función de las acciones realizadas por el usuario al manipular los objetos sobre la superficie del *tabletop*. Estos comportamientos están relacionados generalmente con la consecución de tareas de la actividad, y con el tipo de *feedback* proporcionado a cada tipo de acción (*feedback* positivo si la acción realizada es correcta, negativo cuando la acción es incorrecta, y neutro en los demás casos). Esta tarea consiste en contemplar todos los estados por los que puede pasar una actividad en función de las acciones del usuario y proporcionar las respuestas adecuadas en cada caso.

Tarea 3. Programación del comportamiento de las funciones de la interfaz TUIO

Las funciones de las interfaces de TUIO (concretamente la interfaz *TuioListener*) actúan de nexo de unión entre ReacTIVision y el entorno de desarrollo Flash Profesional (tal y como se muestra en la Figura 8) y proporcionan una infraestructura que nos permite programar el comportamiento asociado a los objetos colocados sobre la mesa y a su manipulación, abstrayéndonos del acceso al hardware y de los algoritmos de detección visual de objetos del *tabletop* NIKVision.

Siempre que se produce un evento de añadir/modificar/quitar un objeto sobre el *tabletop*, es necesario actualizar el estado de la actividad ya que el estado (correcto/neutro/error) de una o más áreas ha podido cambiar. Por ejemplo, si se ha quitado un objeto de un área que era correcta ésta pasa a incorrecta, o si se ha añadido el objeto con el *fiducial* que faltaba para completarla pasa a correcta, etc.

Tarea 4. Elaboración de actividades y pruebas realizadas.

Con el fin de probar el correcto funcionamiento de la aplicación, se crea un juego sencillo de clasificación de juguetes por colores. Este juego consiste en colocar juguetes de diferentes colores sobre las zonas de la pantalla del color correspondiente. La implementación y uso de este juego permitió depurar errores (sobre todo en lo referente al tratamiento del *feedback* proporcionado al alumno) y obtener como resultado una versión más robusta de la aplicación.

La realización de las pruebas fue de vital importancia ya que permitieron garantizar que la aplicación funcionaba correctamente sobre el *tabletop* NIKVision. En primer lugar, se realizaron pruebas sobre el simulador de TUIO (*tuioSimulator*), y una vez se aseguró el funcionamiento básico del software, se realizaron pruebas sobre los *tabletops* disponibles en el laboratorio Affective Lab y en el del CPEE Alborada.

Capítulo 6. Resultados obtenidos

Este capítulo presenta los resultados obtenidos como consecuencia del desarrollo completo de las herramientas de apoyo a la CAA implementadas: *AraBoard* y *Acoti*. En la sección 6.1 se muestra el aspecto final las aplicaciones *AraBoard Constructor* y *AraBoard Player*, en sus versiones para Windows y Android. En la sección 6.2 se presenta el aspecto final de la aplicación *Acoti* cuando es ejecutada sobre la superficie NIKVision.

6.1 Aplicación AraBoard

Tal y como se ha ido comentando a lo largo de la memoria, la herramienta *AraBoard* permite construir y reproducir tableros de comunicación de forma personalizada y adaptada al usuario, y se ha implementado para los sistemas operativos Windows y Android. A continuación se presentan el aspecto de las aplicaciones que la forman:

1. ***Araboard Constructor:*** se utiliza para la creación y edición de los tableros de comunicación mediante la colección de pictogramas ARASAAC y cualquier otro recurso (audio o imagen) almacenado en el dispositivo del usuario.

   En las Figuras 9 y 10 se observa la interfaz de construcción y personalización del tablero para las versiones Android y Windows respectivamente.

Figura 9. AraBoard Constructor para Android: Construcción de un tablero

Figura 10. AraBoard Constructor para PC: Construcción de un tablero

En la Figura 11 se muestra la ventana de edición de pictogramas que aparece cuando se arrastra el pictograma de la celda (1,1) sobre el icono de edición.

Figura 11. AraBoard Constructor para PC: Edición de un pictograma

En la figura 12 se muestra la interfaz de almacenamiento en la que el usuario puede guardar los tableros creados, en la ubicación que desee, para su posterior uso a través de la aplicación *AraBoard Player*.

Figura 12. AraBoard Constructor para PC: Almacenamiento de un tablero

2. ***AraBoard Player:*** se utiliza para ejecutar los tableros de comunicación previamente creados en *AraBoard Constructor*.

   En la Figura 13 se muestra un tablero ejecutado sobre un dispositivo tablet Android.

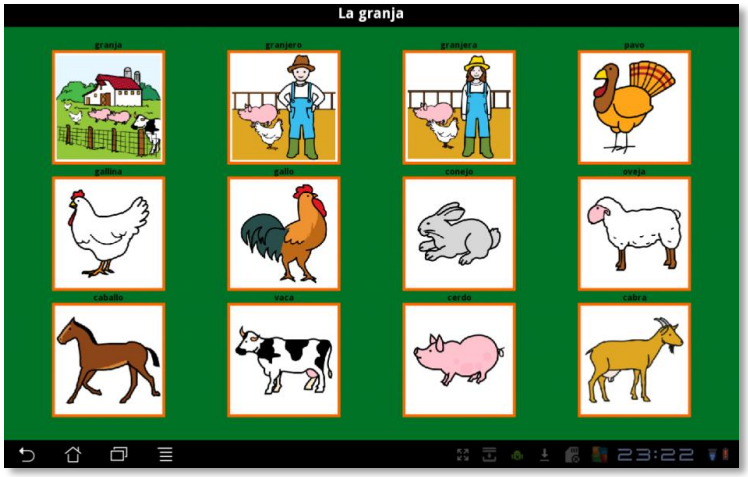


Figura 13. AraBoard Player para Android

En las figuras 14 y 15 se muestra la el proceso de carga del tablero creado con la aplicación de construcción en la Figura 10.

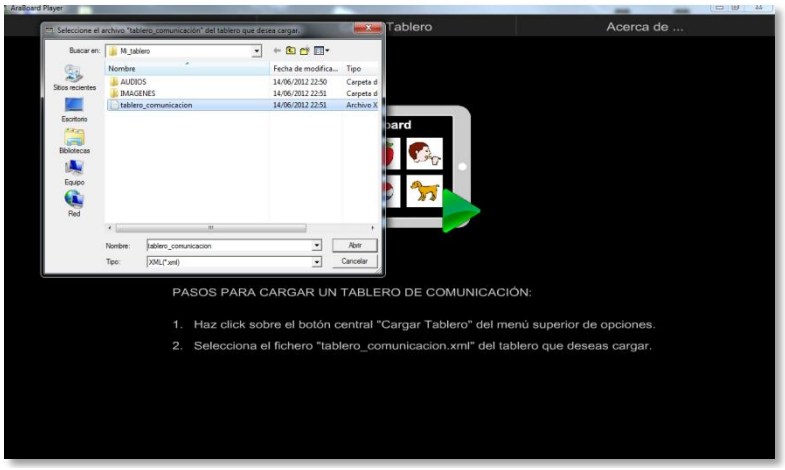


Figura 14. AraBoard Player para PC – Carga de un tablero

Figura 15. AraBoard Player para PC

6.2 Aplicación Acoti

La herramienta *Acoti* permite la generación de gran variedad de aplicaciones pedagógicas adaptadas al nivel cognitivo del alumno, destinadas a funcionar sobre el *tabletop* NIKVision.

En la Figura 16 se muestran imágenes de actividades creadas con *Acoti* y del uso de las mismas. Como se puede comprobar, tanto las actividades como los recursos empleados son muy variados. En las Figuras 16.a y 16.b, se observa la ejecución de un juego de secuencias en el que los alumnos deben completar secuencias de colores con los juguetes del color correspondiente. En las figuras 16.c y 16.d se contempla el típico juego de vestidos, en el que los alumnos tienen que vestir a los personajes que aparecen en la pantalla según las estaciones que se muestre como fondo de la aplicación. Las figuras 16.e y 16.f muestran un juego en el que van apareciendo pictogramas de ARASAAC de alimentos, y los niños deben colocarlos dentro del carrito de la compra.

En cuanto a los recursos empleados, se observa en algunos de los juegos se usan juguetes propios, como es el caso de las figuras 16.a, 16.b y 16.f. En otras se emplean dibujos elaborados por los alumnos, Figura 16.c y 16.d. Y en la figura 16.e se recurre al uso de pictogramas de ARASAAC.

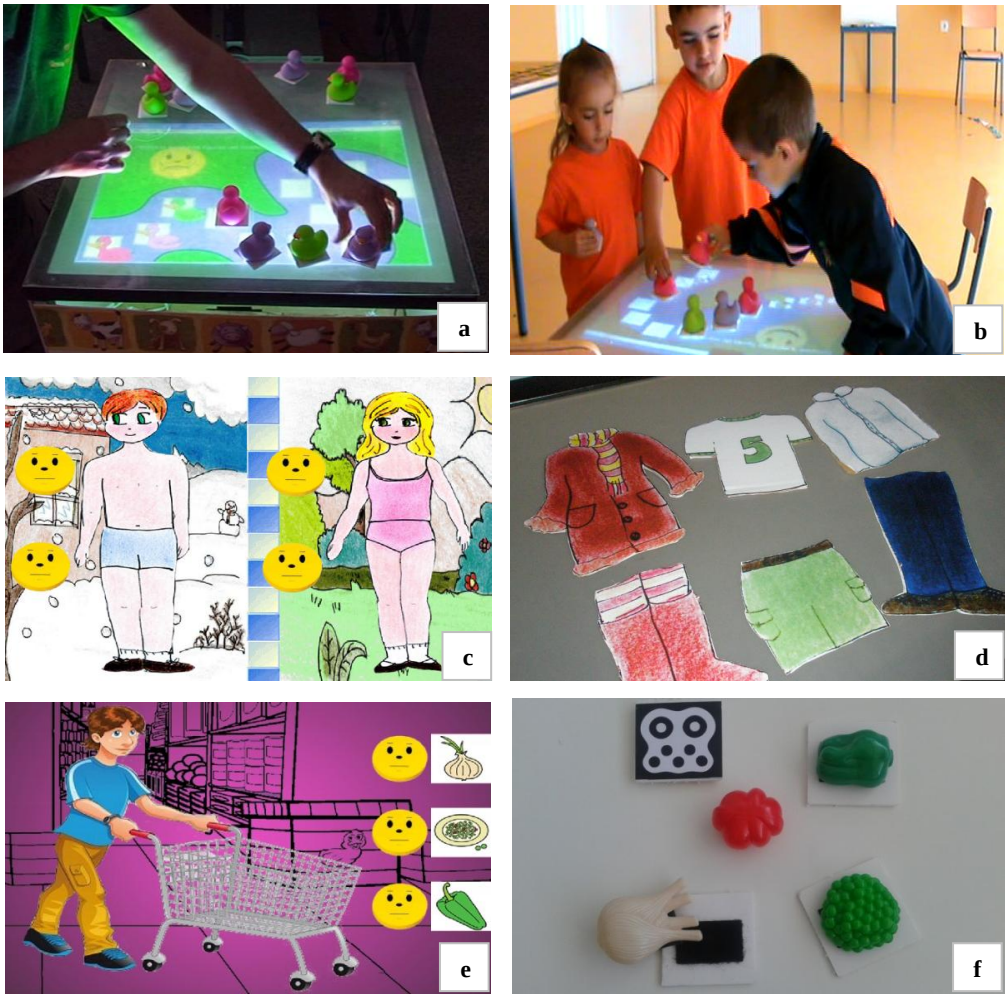


Figura 16. Juegos ejecutados por Acoti sobre la superficie NIKVision

Capítulo 7. Conclusiones y trabajo futuro

En este último capítulo se reflexiona y se presentan las conclusiones derivadas de la realización del proyecto. El capítulo se divide en tres bloques, la consecución de los objetivos, las posibles ideas para un trabajo futuro y la valoración y conclusiones del trabajo personal.

7.1 Consecución de objetivos

El propósito de este proyecto fin de carrera era diseñar e implementar un conjunto de herramientas de apoyo a la comunicación aumentativa y alternativa que fueran fácilmente configurables para adaptarlas a los distintos niveles de discapacidad comunicativa presentes en el alumnado del CPEE Alborada, y que fueran fáciles de usar por los profesores, tutores y familiares de los alumnos, así como por los propios alumnos. Concretamente, las herramientas propuestas eran:

- Herramienta para la construcción y reproducción de tableros de comunicación.
- Herramienta para la generación de actividades didácticas fácilmente.

En el caso de la primera herramienta, se han estudiado las aplicaciones existentes en el mercado, se han realizado reuniones con los profesores y especialistas del colegio y se ha definido, diseñado, e implementado una herramienta que permite construir, editar y reproducir tableros de comunicación personalizados con pictogramas de ARASAAC y con recursos propios del usuario. La herramienta se ha desarrollado para distintas plataformas y ofrece todas las funcionalidades acordadas con el personal docente del CPEE Alborada bajo una interfaz atractiva y sencilla de manejar.

Con la finalización de este PFC se ha obtenido una versión de la aplicación lo suficientemente estable para ser distribuida con garantías de tener un buen funcionamiento. La aplicación se distribuirá a través de Google Play en el caso de la versión para Android, y a través del portal web de ARASAAC para el caso de la versión para PC. Actualmente la aplicación está disponible para que los usuarios la descarguen a través del blog oficial de ARASAAC [19], el blog "*Informática para la educación especial*" [20], la web del GIGA Affective Lab [21], y SourceForge [22].

La aplicación, ha estado en permanente fase de pruebas y actualmente está siendo utilizada actualmente por los alumnos y profesores del CPEE Alborada. Por otro lado, los miembros del portal web ARASAAC presentaron una versión beta de esta aplicación en la "*1ª Jornada Por Dereito: Ferramentas para a INCLUSION*" [23], celebrada el pasado 12 de Mayo en la Isla de Arosa, Pontevedra, y tuvo una gran acogida entre los asistentes.

También se ha diseñado e implementado una herramienta que permite la generación y ejecución de actividades didácticas sobre la superficie interactiva NIKVision. La aplicación ha permitido generar diversas actividades definidas y adaptadas a los distintos niveles cognitivos de los alumnos. Esta herramienta está siendo utilizada actualmente por los profesores y alumnos del CPEE Alborada, y ha sido expuesta a alumnos de primaria (sin ningún tipo de discapacidad), participantes en la IV Semana de la Ingeniería y la Arquitectura [24], celebrada en la Universidad de Zaragoza el pasado mes de Noviembre de 2011.

En ambos casos, las dos aplicaciones han permitido cumplir con los objetivos planteados al inicio del proyecto. Se ha conseguido desarrollar toda la funcionalidad especificada en los requisitos funcionales y no funcionales del sistema, superando con éxito las expectativas del cliente.

7.2 Trabajo futuro

En esta sección se describen posibles trabajos futuros que podrían complementar o ampliar el proyecto realizado. A continuación se plantean las posibilidades de mejora para la aplicación *AraBoard*, y posteriormente para la aplicación *Acoti*.

- **Portabilidad a dispositivos IOs:** La adaptación de la aplicación a dispositivos IOs no resultaría especialmente complicada ya que el código y la estructura del programa serían muy similares, puesto que el código de la aplicación fue desarrollado en Adobe ActionScript 3.0 bajo el entorno de ejecución AIR, y este entorno también facilita la adaptación de aplicaciones de escritorio a dispositivos móviles como Android e IOs. Sin embargo, sería necesario estudiar el sistema de ficheros propio del sistema IOs y adaptar las funcionalidades de carga y almacenamiento de tableros de comunicación, ya que son dependientes del SO del dispositivo. Por otro lado, también sería necesario estudiar el acceso a los botones físicos del dispositivo ya que en IOs sólo existe un único botón que suple la funcionalidad de los 3 botones de Android (*Inicio*, *Back* y *Menu*), y por tanto la gestión de los menús sería distinta.

- **Mejora de la descarga de imágenes:** En función del estado de la conexión a Internet, la visualización de las imágenes de los pictogramas puede tardar en aparecer. Sería interesante poder mejorar la gestión de este recurso permitiendo almacenar todos los pictogramas que forman parte de la interfaz, como archivos temporales mientras se está construyendo un tablero. De esta forma, los archivos sólo tardarían en aparecer la primera vez que se descargasen.

- **Acceso a la cámara y al micrófono del dispositivo:** Incorporar el acceso a la cámara y al micrófono de los dispositivos Android en la interfaz de edición pictogramas facilitaría su construcción y edición ya que se podría realizar de manera más rápida y sencilla.

Posibilidades de mejora para la aplicación *Acoti*:

- **Interfaz gráfica de creación de actividades:** La elaboración de una interfaz que permitiera crear actividades de forma sencilla, y sin necesidad de editar ficheros XML, facilitaría la labor de los profesores que no tengan excesivos conocimientos de informática.

7.3 Valoración personal

Respecto al trabajo realizado, me siento muy satisfecha con los resultados que he obtenido. Todos los objetivos que se plantearon en los inicios del proyecto se han cubierto y actualmente las aplicaciones están siendo usadas por los alumnos y profesores del CPEE Alborada, que es la mejor garantía que se puede tener. La realización de este proyecto ha supuesto una experiencia muy enriquecedora desde varios puntos de vista:

Desde un punto de vista técnico, uno de los objetivos principales de este proyecto era el estudio y familiarización nuevos tipos de aplicaciones, tecnologías y herramientas usadas para desarrollar aplicaciones de escritorio que fueran fácilmente adaptables a otras plataformas. Además de haber conseguido adaptar una misma aplicación a dispositivos tan distintos del ordenador como son los Tablets y los Smartphones, he tenido la gran oportunidad de trabajar con la superficie interactiva NIKVision, y explotar una visión de interacción totalmente nueva para mí.

En cuanto a la aplicación *AraBoard,* decir que ha sido la primera aplicación Android que he desarrollado, ya que el resto de aplicaciones realizadas durante la carrera habían sido aplicaciones de escritorio. Para el diseño y la adaptación de la aplicación ha sido necesario estudiar la organización de los ficheros del sistema operativo así como los permisos de acceso. Este estudio ha incrementado mis conocimientos sobre este campo, y me ha permitido al adquirir experiencia de gran valor en el mercado laboral.

Desde el punto de vista de gestión de proyectos, se han aplicado los conocimientos teóricos y prácticos adquiridos a lo largo la carrera. He podido comprobar la importancia que tiene gestionar bien el tiempo y fijar unas metas realistas, sobre todo cuando los objetivos se van ampliando conforme se cubren etapas. La posibilidad de haber realizado un proyecto para un cliente real, y el hecho de haber trabajado con profesionales de un ámbito tan distinto al mío, ha sido una experiencia muy enriquecedora ya que siempre me han aportado un punto de vista diferente y me han mostrado una perspectiva más cercana al entorno real y a los niños para los que las aplicaciones están destinadas. Esto sin ninguna duda ha supuesto la mayor motivación para seguir adelante.

Finalmente quiero valorar lo mucho que he aprendido de la experiencia de mis directores, a no dar demasiadas vueltas a un mismo problema y aprender a simplificar. A enfrentarme a los problemas paso a paso y a fijarme metas realistas. A ellos debo agradecer el tiempo invertido en explicaciones e intercambios de puntos de vista, y todo el apoyo y la confianza que han depositado en mí a lo largo de estos meses.

Anexo A. Estudio de las aplicaciones de apoyo a la CAA

Este primer anexo complementa al capítulo 3 con información referente al estudio de diferentes aplicaciones software de apoyo a la Comunicación Alternativa y Aumentativa (CAA) disponibles actualmente. Concretamente se han estudiado las aplicaciones más difundidas en el entorno del colegio Alborada y del portal Aragonés de Comunicación Alternativa y Aumentativa (ARASAAC), y por consiguiente, aquellas que emplean la colección de pictogramas de ARASAAC. Para realizar el estudio detallado de cada una de las aplicaciones, primero se proporciona una breve descripción indicando en qué consiste cada una y a continuación se citan sus principales características: usuarios a los que va dirigida, características de las interfaces de usuario, plataformas soportadas, etc. Finalmente, una vez se descritas todas las aplicaciones, se proporciona una valoración general en la que se plantean los motivos por los cuales se ha desarrollado la herramienta de construcción y reproducción de tableros de comunicación.

A.1 Proyecto TICO

TICO (Tableros Interactivos de Comunicación) [25] (Figura A.1) es una aplicación informática para generar y utilizar tableros de comunicación de forma interactiva. El Proyecto TICO traslada la definición tradicional de tablero de comunicación a las nuevas posibilidades que ofrece su uso en un soporte informático, facilitando la interacción y modificación del entorno inmediato a personas con graves trastornos en la expresión oral. El programa se compone de dos aplicaciones independientes y diferenciadas pero complementarias entre sí:

- El **Editor** (Figura A.1, izquierda) posibilita la creación de los tableros de comunicación adaptados a las necesidades y características individuales de cada uno de los usuarios. Desde esta aplicación, se pueden definir y configurar todos los elementos -visuales, auditivos, de control de entorno, etc.- que componen el tablero y que serán utilizados tanto para la emisión de mensajes como para la realización de acciones referidas a control del entorno.

- El **Intérprete** (Figura A.1, derecha) permite al usuario final utilizar el tablero previamente diseñado superando así las limitaciones comunicativas. Esta es la aplicación dotada con la función de barrido para posibilitar el acceso, en todo momento, a aquellas personas que presentan trastornos graves en la motricidad.

El proyecto TICO nace como un comunicador, es decir, como distintos tableros combinados entre sí, que permiten al usuario comunicarse con las personas que tiene alrededor. Al estar dotada de la posibilidad de barrido, puede ser utilizada con otros dispositivos físicos distintos al ratón, permitiendo su accesibilidad a personas con dificultades motrices.

Al estar formado por dos aplicaciones diferenciadas -el Editor y el Intérprete-, el número de posibilidades de creación y personalización de tableros no distrae la atención del usuario ya que éste empleará únicamente la aplicación Intérprete para comunicarse.

Dada la versatilidad y sencillez del programa, TICO puede ser utilizado como elemento de acceso a distintas áreas del currículo, mediante el diseño de actividades orientadas a ello. Además, incorpora la posibilidad de introducir vídeos en las actividades.

El sistema de comunicación es muy sencillo y directo ya que, al pulsar sobre uno de los pictogramas, la voz sintetizada del dispositivo lee el nombre del archivo de imagen.

A.1.1 Características principales

- Los elementos del tablero se pueden agrupar para construir frases.
- Permite generar y articular frases con una estructura sintáctica similar a la utilizada en la comunicación oral espontánea.
- Utiliza pictogramas de ARASAAC.
- Permite inserción de sonido y de vídeo en los elementos del tablero (celdas).
- Permite diseñar varias actividades (también actividades curriculares)
- Incorpora barrido dirigido y barrido automático.
- Plataformas soportadas: Windows XP, Vista, Windows 7.

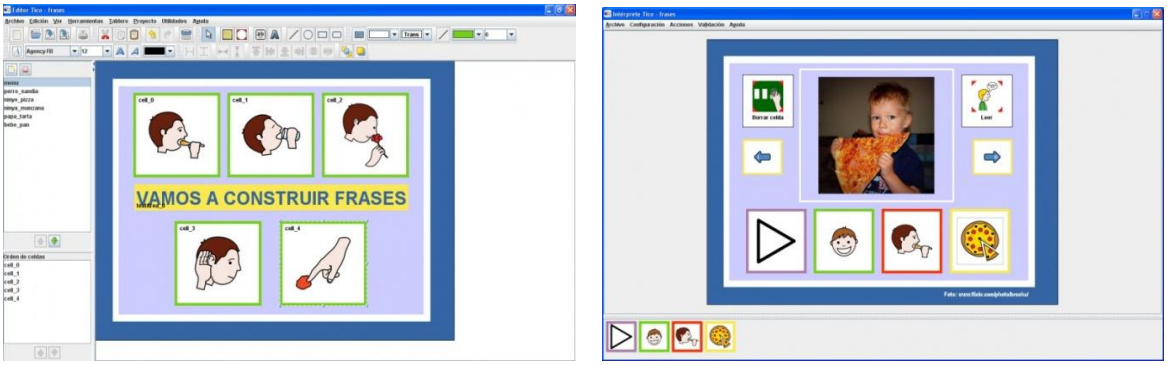


Figura A.1 Interfaz de las aplicaciones Editor(derecha) e Intérprete(izquierda) de TICO

A.2 Proyecto In-Tic

In-TIC [26] (Figura A.2 y A.3) posibilita que personas con diversidad funcional puedan mejorar su autonomía personal a través de dos vías:

- **Sistema de acceso simplificado a la tecnología**: Permite crear entornos personalizados para un acceso simplificado al ordenador y al móvil, configurando aquellas aplicaciones que el usuario desee y con el aspecto que prefiera. Mediante la creación y configuración de teclados virtuales personalizados para cada usuario, se posibilita que personas con dificultades a nivel físico, cognitivo o sensorial puedan utilizar el ordenador, acceder a Internet o a las funciones básicas del dispositivo móvil, ejecutar aplicaciones específicas, juegos, etc.

- **Comunicador dinámico:** Las personas que no hayan adquirido el lenguaje oral o tengan dificultades en el desarrollo de la habilidad lecto-escritora, pueden utilizar In-TIC como un comunicador, tanto en ordenadores convencionales como portátiles, *Tablet* PC y teléfonos móviles, creando plantillas con texto e imágenes que reproducen ficheros de sonido personalizados o que utilizan las capacidades de síntesis de voz del propio entorno de interacción.

In-Tic es un sistema de comunicación mediante pictogramas por pulsación directa, en el que, además, se pueden utilizar otras funcionalidades asociadas directamente a un pictograma como, por ejemplo, llamar por teléfono, abrir la galería de imágenes, escuchar música, …etc.

Las aplicaciones In- Tic están dirigidas a personas con un alto nivel cognitivo y que sean capaces de manejar todas las funcionalidades de un teléfono móvil. Ya que en ocasiones, el entorno de edición y creación de los tableros puede resultar complejo.

A.2.1 Características principales

- Permite facilitar las habilidades de comunicación a personas con dificultades en el habla, utilizando los recursos de voz sintetizada incorporados en los ordenadores.
- Permite acceder y utilizar las aplicaciones y opciones básicas del ordenador a través de la creación de un entorno virtual simplificado y personalizable.
- Permite adaptar y simplificar las principales funcionalidades del dispositivo Android (móvil o Tablet) a las características o necesidades individuales.
- Permite configurar el dispositivo Android como un comunicador dinámico (funcionalidades asociadas directamente a un pictograma.
- Integra una biblioteca multimedia con las colecciones de imágenes (indexadas y categorizadas) de ARASAAC y de Aumentativa.net.
- Incorpora una selección de sonidos descargados del portal del Ministerio de Educación del Gobierno de España.
- Plataformas: Android, Windows XP, Windows Vista y Windows 7 (actualmente se está trabajando en la migración a Linux).

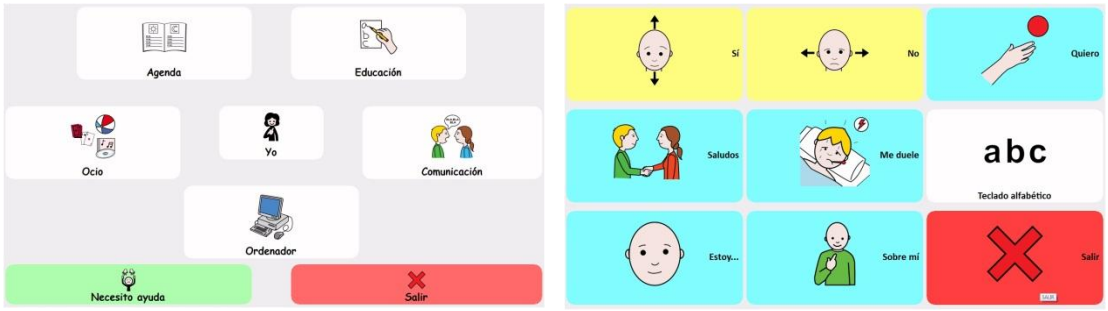


Figura A.2 Interfaz del Comunicador Dinámico In-TIC para PC

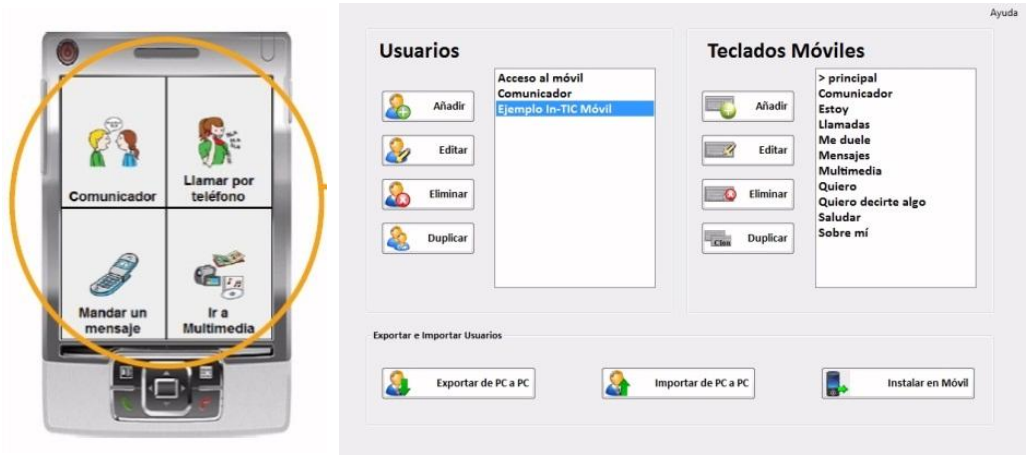


Figura A.3 Interfaz del Comunicador Dinámico In-TIC para Android

A.3 e-Mintza

La aplicación e-Mintza [27] (Figura A.4) es un sistema personalizable y dinámico de comunicación aumentativa y alternativa dirigido a personas con autismo o con barreras de comunicación oral o escrita. Presenta un tablero de comunicación con pictogramas o imágenes y sonidos asociados que permiten una comunicación directa y sencilla. El tablero es fácilmente personalizable en cuanto a la lengua utilizada, textos, imágenes, vídeos o sonidos, en función de las necesidades del usuario, quien podrá interactuar preferentemente a través de una pantalla táctil en un dispositivo tipo tablet, pero también a través del ratón en el caso de una pantalla de ordenador no táctil.

La aplicación está dirigida a personas con un buen nivel cognitivo, que sean capaces de estructurar frases sencillas para obtener el máximo rendimiento de la aplicación y que tengan cierta autonomía en el manejo del dispositivo.

El entorno de edición y creación de los tableros puede resultar complejo ya que la interfaz tiene demasiadas celdas de funciones que pueden distraer la atención del usuario, y ser confundidas con las celdas del tablero. Por otro lado, en la Figura A.4 se puede observar que la interfaz gráfica distorsiona la visión de los pictogramas.

A.3.1 Características principales

- Alta capacidad de personalización en cuanto a la lengua utilizada, los textos, imágenes, vídeos o sonidos, en función de las necesidades del usuario
- Posibilita la personalización del tablero de comunicación con fotos personales en vez de pictogramas o letras.
- Incorpora la funcionalidad de agenda, en la que el usuario puede ubicar el plan temporal secuenciado hasta seis actividades por día o espacio temporal.
- Permite la inserción de vídeo.
- Utiliza pictogramas de ARASAAC.

Plataformas: Windows XP, Windows Vista, Windows 7, Android

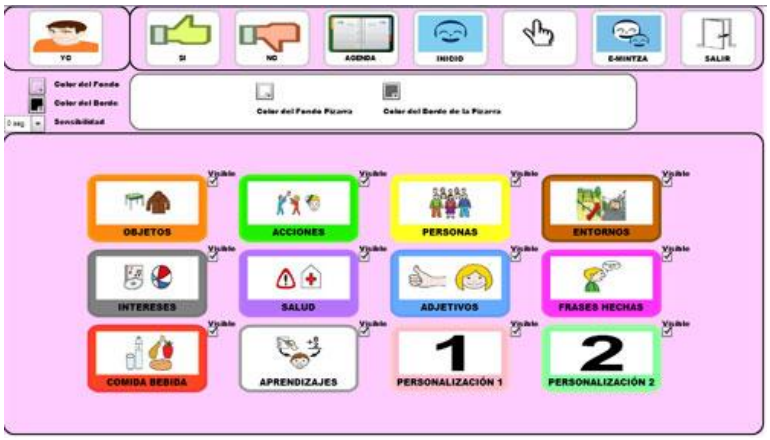


Figura A.4 Interfaz de autor de e-Mintza

A.4 Picto Droid Lite

PictoDroid Lite [28] [30] (Figura A.5) es una aplicación para dispositivos móviles y tablet con sistema operativo Android, que permite a los usuarios comunicarse a través del uso de pictogramas. Permite al usuario la creación de frases sencillas en modo acumulativo (*vamos a…, quiero jugar…, quiero ir al baño, quiero beber…, quiero comer…., estoy…)*, mediante la selección de sujeto, verbo, predicado, adverbios y adjetivos. Una vez que se completa la frase seleccionando los distintos pictogramas, el sistema lee la frase.

El sistema de comunicación es muy sencillo y directo ya que, al pulsar sobre uno de los pictogramas, la voz sintetizada del dispositivo lee el nombre del archivo de imagen introducido en cada categoría. Además se puede configurar para personas con bajo nivel cognitivo.

En cuanto a las opciones de personalización, el número mínimo de celdas para construir un tablero es cuatro, por lo que no se pueden crear tableros con una o dos celdas. Para construir e implementar los tableros de comunicación, es necesario navegar hasta el directorio raíz del dispositivo móvil y moverse entre las distintas carpetas del SO hasta encontrar la carpeta donde se almacenan los tableros de PictoDroid Lite. Por otro lado,

para crear la pantalla de inicio de la aplicación, hay que crear y modificar archivos en formato XML. Estas dos últimas opciones exigen conocimientos bastante avanzados de informática, por lo que es posible que no todos los usuarios de la aplicación sean capaces de llevarlas a cabo.

A.4.1 Características principales

- Permite elaborar de manera sencilla frases cortas.
- Lee la frase histórica acumulada, es decir, reproduce de forma continuada el audio de la secuencia de pictogramas que el usuario ha ido pulsando.
- Permite reproducir los clips de audio asociados a cada imagen.
- Utiliza pictogramas de ARASAAC.
- Muestra un conjunto de pictogramas predefinidos, clasificados por los colores de su categoría semántica.
- Plataformas: Android

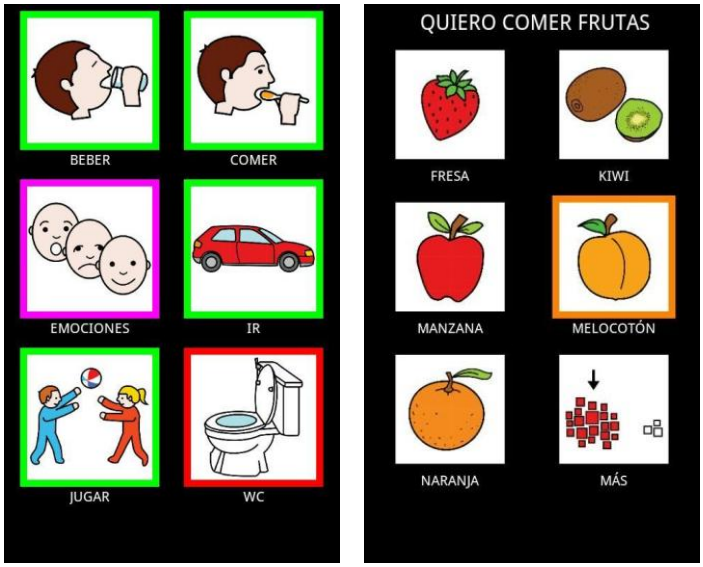


Figura A.5 Interfaz de PictoDroid Lite

A.5 Baluh

Baluh [29] [30] (Figura A.6) proporciona una solución de comunicación con todas las funciones aumentativas y alternativas para las personas que tienen dificultad para hablar. Es un sistema de comunicación mediante pictogramas (almacena una selección de 400 pictogramas), distribuidos en diferentes categorías, en el que éstos se van acumulando para ser leídos como una frase con estructura normalizada (Figura A.6 - Derecha). El funcionamiento habitual consiste en que el usuario toque el pictograma de "Mi libro" para acceder a su libro de imágenes, formado por categorías y sub-categorías (un total de 2 niveles) de pictogramas. Cada vez que se selecciona una imagen, se añade, más pequeña, en el espacio de reproducción.

Cuando la frase ya está construida, puede pulsarse el botón verde de "Play" para oír la concatenación de clips de audio asociados a cada imagen. En el espacio de reproducción caben siete imágenes, más que suficientes para construir frases a base de imágenes.

Por otro lado, Baluh permite acceder a un menú de herramientas de configuración de la aplicación. Algunas de las opciones disponibles en este menú son: Descarga de pictogramas de ARASAAC, crear un propio pictograma, navegar por las categorías, editar "Mi Libro". A pesar de que este menú hace de Baluh una herramienta muy completa y proporciona un alto grado de personalización, tiene la desventaja de que el entorno de edición y creación de los tableros es complejo, por lo que es posible que usuarios con conocimientos informáticos medios o bajos no puedan hacer uso de ellas.

La aplicación está especialmente diseñada para niños y adultos con TGD (Trastorno Generalizado del Desarrollo). Además puede ser de gran utilidad a usuarios con parálisis cerebral, Síndrome de Down, discapacidades del desarrollo, apraxia, accidente cerebro-vascular o una lesión cerebral traumática. Baluh puede ser una solución muy útil en hospitales y en servicios específicos de rehabilitación ya que permite al usuario establecer una comunicación con su entorno, de manera ágil y duradera.

A.5.1 Características principales

- Permite elaborar de manera sencilla frases cortas.
- Lee la frase histórica acumulada.
- Permite reproducir los clips de audio asociados a cada imagen.
- Utiliza pictogramas de ARASAAC.
- Muestra un conjunto de pictogramas clasificados por categorías.
- Plataformas: iPhone, iPod Touch, iPad

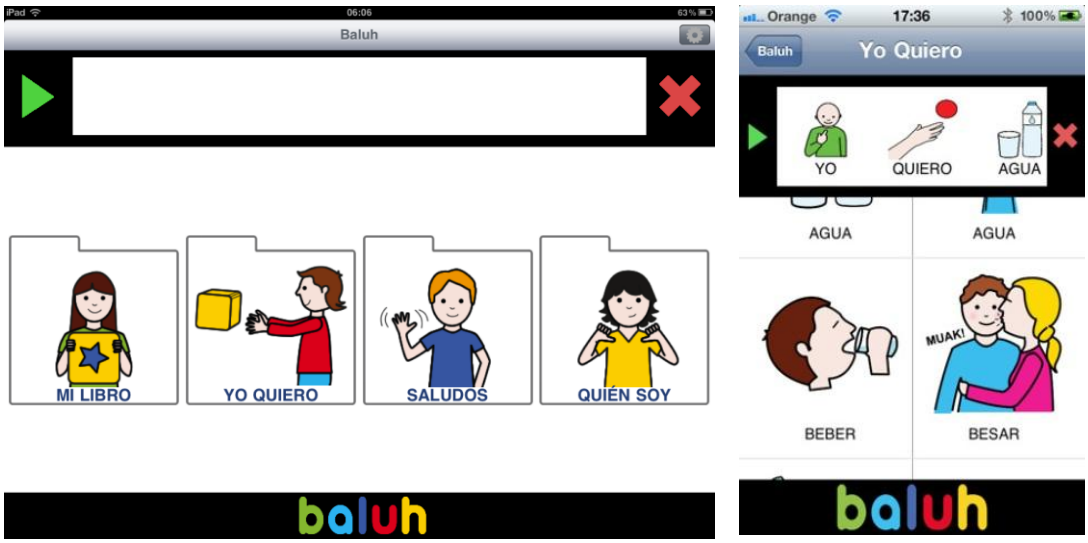


Figura A.6 Interfaz de Baluh en Ipad (izquierda) y en Iphone (derecha)

A.6 Comunicador Personal Adaptable

La aplicación Comunicador Personal Adaptable (CPA) [31] se trata de un sistema de comunicación para personas con problemas graves de comunicación (autismo, trastornos neurológicos, discapacidades motoras, afasia). CPA es un sistema de comunicación basado en la utilización de pictogramas e imágenes con sonidos asociados, a través de un software informático, y empleando como soporte PCs, Pocket PC y dispositivos iOS.

La aplicación el CPA para iPhone prescinde de las aplicaciones auxiliares presentes en la versión para Pocket PC (organizador del día, gestor de rutinas y organizador de frases) y se centra en la que la más funcional de ellas: el comunicador por palabras agrupadas por categorías. Como novedad importante con respecto a otras aplicaciones, se puede seleccionar cuantas imágenes (ya sean categorías o palabras contenidas en ellas) aparecerán por página: 1, 2, 3 ó 4 pudiendo adaptarlo de este modo mejor a las necesidades individuales de la persona que vaya a utilizarlo.

La interfaz es muy sencilla y consigue evitar distracciones por parte del usuario final. La navegación también resulta intuitiva: un toque en la categoría para entrar, y otro en la flecha "Categorías" que se encuentra en la barra superior para volver. En esta barra también aparece el nombre de la categoría en la que nos situemos, facilitando de este modo la ubicación. Para reproducir el sonido de las palabras sólo es necesario tocar cada una de las imágenes elegidas.

El CPA para iPhone viene con un vocabulario de base bastante amplio (600 palabras), empleando pictogramas de ARASAAC, pero, al igual que su antecesor, nos permite incorporar más vocabulario y así personalizar el programa.

En cuanto a la personalización de la aplicación, el Comunicador Personal Adaptable facilita insertar imágenes desde la galería de iPhone, y grabar sonidos que lo acompañen.

A.6.1 Características principales

- Plataformas: iPhone, iPod Touch, iPad, PC con sistema operativo Windows XP y Pocket PC con sistema widows mobile 2003 o superior.

Versión Ipod/Iphone (Figura A.7, arriba):

- Sencillo entorno de edición.
- Posibilidad de añadir nuevas palabras con imágenes de tu dispositivo o desde la cámara.
- Permite grabar directamente los sonidos o incorporarlos desde la biblioteca iTunes.

Versión Ipad (Figura A.7, abajo):

- Posibilidad de barra de escritura.
- Muestra categorías y vocabulario en una misma pantalla.

- Gestión de usuarios.
- Permite el acceso directamente a todo el vocabulario de ARASAAC.
- Permite añadir nuevas palabras con imágenes de tu dispositivo o desde la cámara.
- Disponible en varios idiomas.

Versión PC/ Pocket PC:

- Funcionalidad de comunicador simple (un solo pictograma) y ampliado (varios pictogramas).
- Funcionalidad de tablero de comunicación, con la posibilidad de construir frases de 5 elementos.
- Posibilidad de configuración del comunicador de forma que el usuario puede componer frases simples formadas por sujeto, verbo y complemento.
- Gestión de usuarios.
- Gestión de vocabulario, que permite seleccionar el vocabulario concreto para cada usuario.
- Creación y gestión de rutinas.
- Organizador diario.
- Opciones de configuración.

Figura A.7 Interfaz de CPA
(Arriba: iPhone/IPod. Abajo: iPad)

A.7 Conclusiones finales

En este apartado se presentan las conclusiones obtenidas del estudio de las aplicaciones software presentadas en las secciones anteriores.

Las aplicaciones estudiadas, bien se traten de aplicaciones para ordenador o de aplicaciones para dispositivos móviles o tablets, tienen la Base de Datos (BBDD) de pictogramas de ARASAAC (o parte de ella) almacenada en la memoria del dispositivo. Esto supone una ventaja, ya que los accesos a los pictogramas de la BBDD se producen con rapidez, al estar almacenados en la máquina local del usuario. Por otro lado, supone una desventaja ya que la BBDD ocupa demasiado espacio para ser almacenada en dispositivos móviles o tablets (la colección completa consta de alrededor de 13500 pictogramas; los pictogramas en color ocupan 532 MB, y la colección de pictogramas en blanco y negro 447MB). Este conlleva que la mayor parte de las aplicaciones opten por no almacenar la BBDD completa, sino solamente una selección de pictogramas, limitando de esta manera el acceso a la totalidad de los recursos de la colección.

Por otra parte, es importante señalar que la colección de pictogramas de ARASAAC está en continua actualización, por lo que el usuario tendrá que descargar la BBDD actualizada si quiere tener acceso a los últimos pictogramas elaborados.

Para solucionar estas desventajas, se plantea la siguiente opción: aprovechar la conexión a Internet para que las aplicaciones puedan acceder directamente a los recursos almacenados en la BBDD de ARASAAC, y de esta manera evitar la descarga de la BBDD completa en la máquina del usuario. Para llevar a cabo las búsquedas de pictogramas, la aplicación incorporará un buscador que permita al usuario realizar búsquedas en la colección de ARASAAC.

Este factor marca la diferencia entre la aplicación a desarrollar y el resto de aplicaciones disponibles en el mercado, ya que no es necesario almacenar la BBDD de pictogramas en la máquina del usuario, puesto que está en la web. La principal ventaja de este factor es que la base de datos siempre va a estar actualizada. En el mismo momento en que un pictograma sea modificado desde el portal web de ARASAAC, este pictograma estará disponible desde nuestra aplicación. La desventaja que introduce es la necesidad de trabajar en red para buscar los recursos, puesto que nos obliga a disponer de conexión a Internet para su obtención. Sin embargo, la ausencia de conexión no impide que los usuarios hagan uso de la aplicación, puesto que también permite la incorporación de los recursos propios del usuario (imágenes, gráficos o ficheros de sonido), permitiendo de esta manera un gran número de posibilidades a la hora de personalizar y adaptar los tableros de comunicación y no depender únicamente del acceso a la colección de ARASAAC.

La personalización de los comunicadores y de los tableros de comunicación resulta de gran importancia, por lo que se descartarán todas aquellas aplicaciones que no permitan la inclusión de recursos propios con los que el usuario final está familiarizado.

Otro aspecto que se ha valorado es el soporte multiplataforma de las aplicaciones. Es importante que la aplicación a desarrollar esté disponible en versiones para ordenador y para dispositivos móviles o tablets. De esta manera la aplicación es accesible a un mayor número de usuarios, y no restringe su uso a tener o no tener un móvil o tablet. Por otro

lado, los tableros creados con la versión de escritorio y la versión móvil deben ser compatibles e independientes de la plataforma sobre la que se hayan creado.

Del estudio realizado y de las conclusiones obtenidas, se identifican dos características principales que debe cumplir la aplicación a desarrollar:

- La construcción de los tableros debe permitir la incorporación de recursos con los que los usuarios están más familiarizados (audios/imágenes de su vida cotidiana, y pictogramas de la colección de ARASAAC).
- Adaptabilidad al mayor número de plataformas informáticas posibles, principalmente a aquellas que sean más accesibles tanto económicamente como funcionalmente.

Anexo B. Estudio de las tecnologías de desarrollo de AraBoard

En este anexo se presenta un análisis completo de las principales tecnologías utilizadas para el desarrollo de la herramienta *AraBoard*. En la primera sección se describen los requisitos tecnológicos del sistema. En la segunda sección se describe el lenguaje de programación ActionScript 3.0 empleado en el desarrollo del sistema. En la tercera sección se describe el entorno de desarrollo Adobe Flash, y de reproducción Adobe Flash Player. En la cuarta sección se describe la tecnología finalmente empleada: Adobe AIR. Finalmente, en la quinta sección se justifican las tecnologías elegidas para satisfacer los requisitos tecnológicos del sistema.

B.1 Requisitos Tecnológicos

El primer requisito tecnológico es la necesidad de utilizar una plataforma contrastada y de desarrollo estable, que soporte el paradigma de programación orientado a objetos, predominante en la mayoría de aplicaciones actuales.

El segundo requisito tecnológico que surge a la vista del análisis del problema es la necesidad de cargar y almacenar información relativa a los recursos que componen un tablero de comunicación. Por lo tanto, se requiere que el sistema sea capaz de acceder y almacenar recursos, tanto si estos proceden del sistema de archivos local, como si deben ser descargados de Internet y almacenados en la máquina del usuario.

El tercer requisito tecnológico es la necesidad de utilizar una tecnología que permita generar aplicaciones multiplataforma móvil y también de escritorio. Es decir, los programas de escritorio se podrán utilizar en cualquier ordenador, independientemente del sistema operativo que tenga, y podrán ser adaptables a dispositivos móviles o tablets.

El cuarto requisito tecnológico surge de la necesidad de desarrollar interfaces con alto contenido en recursos gráficos y audio, por lo que la inclusión de contenido multimedia debe realizarse de manera fácil y rápida. De la misma manera, el desarrollo de

las interfaces de usuario así como sus componentes ha de ser sencillo y no debe variar de una plataforma a otra.

B.2 El lenguaje ActionScript 3.0

En los últimos años, el Affective Lab del grupo GIGA ha desarrollado varias aplicaciones interactivas en colaboración con el CPEE Alborada. Estas aplicaciones han sido creadas con el entorno de desarrollo Adobe Flash Profesional y ejecutadas en el reproductor o máquina virtual Adobe Flash Player. Todas ellas han sido programadas en el lenguaje ActionScript, principalmente por la facilidad que presenta para crear juegos y aplicaciones con un alto número de recursos gráficos: animaciones, audios, imágenes, etc. Partiendo de la predisposición a emplear este lenguaje, se estudiaron sus características y se valoró si se ajustaba a los requisitos de la aplicación. A continuación se describen las principales características de este lenguaje.

ActionScript [32] es el lenguaje de programación para los entornos de reproducción y ejecución Adobe Flash Player y Adobe AIR. Este lenguaje se ejecuta mediante la máquina virtual ActionScript (AVM), que forma parte de Flash Player y de AIR. El código de ActionScript se suele compilar en un formato de código de bytes mediante un compilador, como el incorporado en Adobe Flash CS5 Professional. El código de bytes está incorporado en los archivos SWF ejecutados por Flash Player y AIR.

Desde su origen ActionScript ha pasado de ser un lenguaje muy básico a un lenguaje avanzado con soporte de programación orientada a objetos, comparable en funciones y uso al lenguaje JavaScript.

ActionScript, en concreto su versión 3.0, ofrece un modelo de programación robusto que también resultará familiar lenguaje de programación orientado a objetos como Java o C++. Los objetos constituyen la base del lenguaje ActionScript 3.0. Son sus componentes esenciales. Cada variable que se declare, cada función que se escriba y cada instancia de clase que se cree es un objeto. Un programa ActionScript 3.0 se puede considerar como un grupo de objetos que realizan tareas, responden a eventos y se comunican entre sí.

A continuación se presentan los entornos de desarrollo y ejecución estudiados para realizar esta aplicación, concretamente los entornos Adobe Flash, Adobe Flash Player y Adobe AIR.

B.3 Descripción de Adobe Flash y Adobe Flash Player

Adobe Flash [33], se refieren tanto al programa de desarrollo de aplicaciones como al reproductor. Estrictamente hablando, Adobe Flash es el entorno de desarrollo y Adobe Flash Player el reproductor o máquina virtual. Sin embargo, en lenguaje coloquial, se usa el término Flash para referirse al entorno, al reproductor e, incluso, a los archivos generados. Adobe Flash tiene soporte para el lenguaje de programación ActionScript, explicado en la sección anterior.

Adobe Flash Player [34] es una aplicación en forma de reproductor que permite reproducir archivos SWF que pueden ser creados con la herramienta de autoría Adobe Flash, con Adobe Flex[5], Adobe AIR o con otras herramientas de Adobe y de terceros. Estos archivos se reproducen en un entorno determinado. En un sistema operativo tiene el formato de aplicación del sistema, mientras que si el entorno es un navegador, su formato es el de un Plug-in u objeto ActiveX. Originalmente, Adobe Flash Player fue creado para mostrar animaciones vectoriales en 2 dimensiones, pero actualmente es una de las crear aplicaciones Web que incluyen flujo de audio y vídeo e interactividad.

El principal problema con Flash Player reside en sus restricciones de seguridad a la hora de configurar la aplicación de construcción, ya que ésta necesita acceder a servicios externos de red (al solicitar y recibir recursos del Portal Web de ARASAAC), e implementa funciones de acceso a ficheros en modo local (al almacenar los ficheros XML y los de imagen y sonido).

El problema aparece por la manera en la que el entorno de desarrollo permite configurar la seguridad de la reproducción local del programa, dado que solamente proporciona las dos opciones siguientes:

- Seguridad de reproducción local permitiendo sólo acceder a archivos locales, y
- Seguridad de reproducción de red permitiendo sólo el acceso a la red.

Sin proporcionar una combinación de ambas que nos permita configurar la seguridad tanto en la reproducción local como en la red.

Partiendo de este problema surge la necesidad de estudiar un entorno de ejecución distinto que permita la ejecución de la aplicación en ambos modos de trabajo sin imponer restricciones de seguridad. De estos inconvenientes surge el estudio de Adobe AIR.

B.4 Descripción de Adobe AIR

Adobe AIR [35] es una tecnología que permite la creación de aplicaciones de escritorio (de propósito general) a partir de tecnologías de desarrollo de páginas web, como pueden ser HTML, Ajax o Flash. Adobe AIR fue creado como un entorno de ejecución versátil que permite usar código Flash, Actionscript, HTML o JavaScript para crear aplicaciones basadas en internet con muchas características de los programas tradicionales de escritorio. Adobe lo define como un entorno de ejecución que no necesita navegador, para poder portar RIAs (aplicaciones de internet enriquecidas) al escritorio.

Una de las ventajas de las aplicaciones generadas por el entorno de ejecución AIR es que son multiplataforma. Es decir, los programas de escritorio de Adobe AIR se podrán utilizar en cualquier ordenador, independientemente del sistema operativo que tenga. Asimismo, el entorno de ejecución AIR también permite el desarrollo de aplicaciones móviles para dispositivos IOs y Android. El flujo de trabajo para crear aplicaciones AIR para dispositivos móviles es, en general, muy similar al utilizado para la creación de

[5] Adobe Flex (hasta 2005 Macromedia Flex) es un término que agrupa una serie de tecnologías publicadas desde Marzo de 2004 por Macromedia para dar soporte al despliegue y desarrollo de Aplicaciones Enriquecidas de Internet, basadas en su plataforma propietaria Flash.

aplicaciones de escritorio. La principal diferencia radica cuando llega el momento de empaquetar, depurar e instalar la aplicación. Por ejemplo, las aplicaciones de AIR para Android utilizan el formato del paquete nativo APK (.apk) de Android en lugar del formato del paquete de AIR (.air) o ejecutable (.exe).

La multiplataforma se consigue gracias a que cuando se descarga una aplicación con Adobe Air, se comprueba si el entorno está instalado en nuestro dispositivo. Si no estaba instalado, se descarga el entorno de ejecución necesario para que las aplicaciones funcionen en nuestro sistema operativo concreto. Una vez se haya instalado el entorno en nuestro sistema, no se tiene que volver a descargar ni instalar de nuevo. Con ello, Adobe AIR consigue acercarse al paradigma, "programa una vez y ejecuta donde quieras", a la vez que se hace universal y al alcance de todos los usuarios y plataformas.

Es importante destacar que Adobe AIR es una herramienta de distribución gratuita tanto para los usuarios que deseen instalarla y usar programas creados con AIR, como para los desarrolladores que quieran crear aplicaciones de escritorio a partir de sus proyectos web.

B.5 Conclusiones finales

A partir del estudio de las tecnologías empleadas en el grupo GIGA - Affective Lab para el desarrollo de aplicaciones interactivas con un alto grado de recursos gráficos, se identificaron una serie de restricciones que imposibilitaban la realización de la aplicación de construcción *AraBoard* con el entorno de desarrollo y reproducción empleado hasta el momento: Adobe Flash Profesional y Adobe Flash Player. Como alternativa al problema planteado se estudió el entorno de ejecución Adobe AIR y sus características [36].

Adobe AIR es una tecnología muy novedosa y con grandes posibilidades en la comunidad de desarrolladores de aplicaciones de escritorio, de RIAs y de aplicaciones móviles. Del estudio realizado en secciones anteriores se deducen sus ventajas e inconvenientes:

Ventajas

1. Multiplataforma móvil y de escritorio.

2. Capacidad de leer y escribir en todo el sistema de archivos local

3. ActionScript 3.0 es un lenguaje muy potente que permite el uso de patrones y estructuras complejas en los desarrollos.

4. Adobe AIR se ejecuta sobre la misma tecnología que Flash. Del mismo modo que las aplicaciones creadas con Flash, las aplicaciones creadas con AIR se ejecutan en el reproductor Adobe Flash Player, por lo que la portabilidad de una a otra es prácticamente trivial. La diferencia primordial entre ambas reside en las librerías adicionales de Adobe AIR, las cuales permiten crear RIAs de forma más sencilla, en oposición a Flash, que está más dirigido al ámbito de la animación y la creación de juegos.

- El producto final de una aplicación AIR compilada es un archivo .air o un archivo nativamente empaquetado .exe.

- El archivo .air es simplemente un archivo zip que contiene un archivo .swf y directorios relevantes del proyecto (por ejemplo, los activos no-embedidos).

5. El archivo .exe es el mismo a excepción de que también contiene un código para descargar e instalar el entorno Adobe AIR en el caso en que el usuario no lo tenga instalado en su sistema.

6. El entorno Adobe AIR es lo que permite al sistema operativo interpretar los archivos .air e instalarlos y/o ejecutarlos en el sistema. Además, contiene aquellas bibliotecas más relevantes de AIR que no están incluidas en Flash.

Inconvenientes

1. Aunque el entorno de ejecución Adobe AIR es gratuito, el entorno de desarrollo Adobe Flash Profesional es de pago.

2. El entorno Adobe AIR no funciona en todos los dispositivos Android, solo en aquellos modelos que tengan arquitectura Arm7.

3. Rendimiento regular, renderización no muy suave en IOS. Las aplicaciones Air de escritorio consumen mucha CPU, sobre todo en Mac.

El principal motivo por el que se ha seleccionado Adobe AIR como entorno de ejecución es porque ofrece un contrato de seguridad distinto al de Adobe Flash, ya que Adobe AIR permite leer y escribir en todo el sistema de archivos local puesto que las aplicaciones de AIR se instalan de forma nativa (segundo requisito cumplido).

Por otro lado, cabe destacar que las aplicaciones AIR se ejecutan en varios sistemas operativos distintos sin suponer trabajo adicional para el desarrollador. El motor de ejecución asegura una presentación e interacciones constantes y predecibles en todos los sistemas operativos compatibles con AIR (tercer requisito cumplido).

En cuanto a la inclusión de contenido multimedia, Adobe AIR no introduce ningún problema ya que permite usar el lenguaje de programación ActionScript 3.0, con el que el grupo ha trabajado en los últimos años. Además, como se ha citado en la sección B.2, ActionScript 3.0 es un lenguaje orientado a objetos que permite mucha más eficiencia en las aplicaciones de la plataforma Flash, permitiendo construir animaciones de todo tipo, desde simples a complejas, ricas en datos y en interfaces interactivas (primer y último requisito cubiertos).

Anexo C. Gestión del proyecto

Este anexo describe el modelo de proceso utilizado en el desarrollo de las aplicaciones software de este PFC, y explica cuáles son las ventajas y los inconvenientes de la metodología empleada. A continuación se muestra la distribución temporal del proyecto, describiendo para ello las diferentes etapas con sus objetivos y resultados.

C.1 Modelo de proceso

El **modelo incremental** (Figura C.1), descrito en [9], combina las ventajas del modelo en cascada y del modelo evolutivo. En un proceso de desarrollo incremental los clientes identifican, a grandes rasgos, los servicios que proporcionará el sistema. En primer lugar, identifican qué servicios son más importantes y cuáles menos. Entonces, se definen varios incrementos en donde cada uno proporciona un subconjunto de la funcionalidad del sistema. La asignación de servicios a los incrementos depende de la prioridad del servicio, siendo con los servicios de prioridad más alta. Una vez que los incrementos del sistema se han identificado, los requerimientos que se van a entregar en el primer incremento se definen en detalle, y éste se desarrolla (Figura C.1), por lo que los clientes pueden ponerlo en servicio una vez se ha completado y entregado. Esto significa que tienen una entrega temprana de parte de la funcionalidad del sistema. Pueden experimentar con el sistema, lo cual les ayuda a clarificar sus requerimientos para los incrementos posteriores y para las últimas versiones del incremento actual. Tan pronto como se completan los nuevos incrementos, se integran en los existentes de tal forma que la funcionalidad del sistema mejora con cada incremento entregado.

La principal característica de este modelo es que permite crear cada vez versiones más completas del software, para ello se construyen versiones sucesivas de la aplicación. Se crea una primera versión que utiliza el usuario, y éste provee retroalimentación al desarrollador, y según los requerimientos especificados de este usuario se crea una segunda versión.

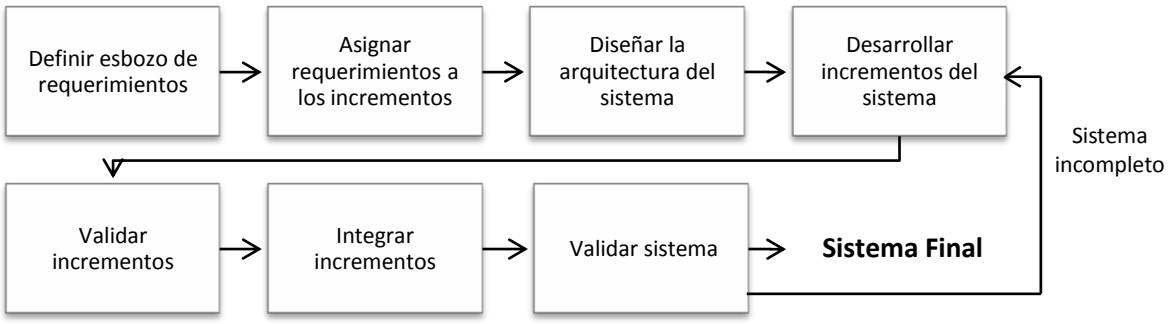


Figura C.1. Modelo incremental de proceso

El modelo proceso de desarrollo incremental tiene varias ventajas:

1. Los clientes no tienen que esperar hasta que el sistema completo se entregue para sacar provecho de él. El primer incremento satisface los requerimientos más críticos de tal forma que pueden utilizar el software inmediatamente.
2. Los clientes pueden utilizar los incrementos iniciales como prototipos y obtener experiencia sobre los requerimientos de los incrementos posteriores del sistema.
3. Existe un bajo riesgo de un fallo total del proyecto. Aunque se pueden encontrar problemas en algunos incrementos, lo normal es que el sistema se entregue de forma satisfactoria al cliente.
4. Puesto que los servicios de más alta prioridad se entregan primero, y los incrementos posteriores se integran en ellos, es inevitable que los servicios más importantes del sistema sean a los que se les hagan más pruebas. Esto significa que es menos probable que los clientes encuentren fallos de funcionamiento del software en las partes más importantes del sistema.

Sin embargo, existen algunos problemas en el desarrollo incremental. Los incrementos deben ser relativamente pequeños y cada uno debe entregar alguna funcionalidad del sistema. Puede ser difícil adaptar los requerimientos del cliente a incrementos de tamaño apropiado. Es más, muchos de los sistemas requieren un conjunto de recursos que se utilizan en diferentes partes del sistema. Puesto que los requerimientos no se definen en detalle hasta que un incremento se implementa, puede ser difícil identificar los recursos comunes que requieren todos los incrementos.

A pesar de las desventajas que pueda tener este modelo, el desarrollo de este proyecto no se ha visto afectado por ellas, y sus ventajas lo convierten en una buena opción de modelo de proceso software a seguir, ya que el sistema a desarrollar ha de pasar por diferentes versiones, y cada una de éstas por las pruebas del personal del colegio Alborada, para seguir definiendo el resto de partes y realizar las modificaciones pertinentes en cada fase.

C.2 Esquema temporal

La realización de este proyecto fin de carrera comenzó en Octubre de 2011 y termino a finales de Junio de 2012. Durante este tiempo el trabajo realizado se puede dividir en tres etapas temporales.

1. Estudio previo, tanto de las aplicaciones en las aulas, como de las tecnologías de desarrollo.

2. Desarrollo de la aplicación Acoti

3. Desarrollo de la aplicación AraBoard

La distribución temporal, y las diferentes fases en las que se dividió el proyecto, se muestran en el diagrama de Gantt de la figura C.3. A continuación, se describen brevemente los objetivos y resultados obtenidos en cada una de las fases del proyecto.

C.2.1 Fase de formación y análisis de las tecnologías

Esta fase comenzó a finales de Septiembre y terminó a principios de Noviembre, y consta dos objetivos principales:

El primero de ellos consiste en adquirir los conocimientos técnicos necesarios para desarrollar la aplicación *Acoti* sobre el *tabletop* NIKVision, así como el estudio del lenguaje de programación *ActionScript* 3.0.

El segundo objetivo consiste en estudiar, analizar y probar las tecnologías y herramientas que más se adaptaban al desarrollo la aplicación *AraBoard*, con el fin de elegir aquellas que más se adecuasen a los requisitos de la aplicación. El estudio previo de las tecnologías de desarrollo se encuentra detallado en el apéndice B.

Sin embargo, la formación a lo largo de todo el proyecto ha sido continua ya que conforme se planteaban nuevos problemas y necesidades (tanto en el análisis, como en el diseño y en la implementación) era necesario adquirir formación sobre las diferentes tecnologías y herramientas utilizadas.

C.2.2 Fase de análisis del problema

La fase de análisis tuvo lugar durante el mes de Noviembre. Los objetivos de esta fase fueron: analizar las necesidades de los usuarios del sistema y capturar los requisitos del mismo. Para ello se utilizaron varias técnicas: reuniones con el personal docente del colegio Alborada y con los directores del presente proyecto, estudio de las aplicaciones empleadas en las aulas y en el entorno de los alumnos del colegio, tablas de requisitos, casos de uso, etc. Esta fase se realizó de manera conjunta para ambas aplicaciones ya que el grupo de usuarios al que van dirigidas era el mismo.

Esta fase finalizó con el análisis de la herramienta de construcción y reproducción de tableros de comunicación, a principios de Diciembre, coincidiendo con una de las reuniones en el colegio Alborada con los directores del proyecto. Como consecuencia de esta reunión, y de las reuniones previas, se establecieron los requisitos de las herramientas a desarrollar.

C.2.3 Fase de diseño de la solución

Esta fase abarca dos periodos temporales separados, debidos principalmente al periodo de exámenes de Enero. El primer periodo se corresponde con el diseño de la herramienta *Acoti*, y tuvo lugar a finales de Noviembre y duró hasta principios de Diciembre. El segundo periodo comenzó a mediados de Enero hasta principios de Febrero, periodo en el que tuvo lugar la fase de diseño de la aplicación *AraBoard*.

El objetivo principal de esta fase consistió en definir el diseño general del sistema, en primer lugar para la aplicación *Acoti*, y más adelante, para la aplicación *AraBoard*. Una vez definida el sistema general, se diseñó, para cada aplicación, cada uno de los sus componentes y las relaciones establecidas entre ellos. Tras elaborar un primer diseño del sistema, se planificó la fase de implementación. Se priorizaron las funcionalidades del sistema y se estimó su duración en jornadas de trabajo. En el diagrama Gantt de la Figura C.3 se muestra la fase de implementación desglosada en sus iteraciones señalando la parte a la que afecta cada iteración.

C.2.4 Fase de implementación

El objetivo de esta fase era codificar los sistemas siguiendo el diseño elaborado en la fase anterior.

En primer lugar se presenta la fase de implementación de la aplicación *AraBoard*. Según la metodología seguida (modelo incremental), se divide la aplicación en 6 iteraciones, en donde cada una proporciona un subconjunto de la funcionalidad del sistema:

1. En la **primera iteración** se implementó el esqueleto general de la aplicación *AraBoard* para PC, con una interfaz básica que ofrecía una visión completa de las diferentes opciones disponibles (siguiendo la filosofía de hacer que el sistema completo funcionase antes de hacerlo atractivo a nivel de interfaz de usuario).

2. En la **segunda iteración** se implementaron las funcionalidades básicas de la aplicación: buscador de pictogramas, construcción, almacenamiento, ejecución y carga del tablero de comunicación, etc.

3. En la **tercera iteración** se adaptó la versión de PC implementada hasta el momento a una versión para Android. En concreto, esta iteración se centró en aspectos que afectaban al sistema completo: adaptación de la aplicación al sistema de ficheros del dispositivo, almacenamiento y gestión de tableros, operaciones de cargar un tablero existente o crear nuevo tablero.

4. En la **cuarta iteración**, se implementaron las funcionalidades secundarias de la aplicación para la versión Android: personalización del tablero y de los pictogramas que lo componen, búsquedas en varios lenguajes, etc.

5. En la **quinta iteración**, se implementaron las funcionalidades secundarias de la aplicación para la versión PC: personalización del tablero y de los pictogramas que lo componen, búsquedas en varios lenguajes, etc.

6. Finalmente, en la **sexta iteración**, se mejoraron todas las interfaces (en primer lugar la aplicación para Android, y en segundo lugar la aplicación para PC). El resultado de esta fase es el sistema funcionando en PCs convencionales con Windows XP, Vista y Windows 7; y dispositivos tablets y smartphones con sistema operativo Android.

Por otro lado, para la aplicación *Acoti* el número de iteraciones se dividió en 2.

1. En la **primera iteración** se implemento el esqueleto general de la aplicación siguiendo la primera versión de estructura de fichero XML, en la que las actividades constan de una única tarea.

2. En la **segunda iteración** se modificó la estructura del XML al incorporar nuevos tipos de *feedback* para el alumno (visible y auditivo) así como secuencias de tareas a realizar. La estructura del fichero XML se presenta en el Anexo E, sección E.3.2.

El resultado de estas dos iteraciones es el sistema funcionando sobre la superficie táctil NIKVision del laboratorio Affective Lab, y posteriormente del colegio Alborada.

C.2.5 Fase de pruebas

El objetivo de esta fase era detectar, y posteriormente corregir errores en el sistema. Las pruebas abarcaron toda la funcionalidad del sistema. Particularmente, las pruebas realizadas con la versión Android de *AraBoard* se realizaron sobre varios modelos de dispositivos, con el fin de verificar que la variedad de dispositivos, y que sus distintas características no afectaban al correcto funcionamiento de las funcionalidades implementadas. En el caso de la aplicación *Acoti*, se realizaron pruebas sobre el *tabletop* disponible en el laboratorio del GIGA Affective Lab, y sobre el *tabletop* disponible en el CPEE Alborada.

Como resultado de esta fase, ambas aplicaciones fueron probadas por los usuarios finales, tanto profesores como alumnos del CPEE Alborada, lo que ha permitido distribuir una versión robusta de las aplicaciones *AraBoard* y *Acoti*.

C.2.6 Fase de documentación

Durante esta fase se escribió esta memoria junto con el resumen del proyecto así como los anexos, utilizando como base toda la documentación que había sido generada durante las diferentes fases del proyecto.

C.3 Gestión del esfuerzo

En la figura C.2 se puede apreciar el tiempo dedicado a cada una de las fases del proyecto en comparación con el resto de fases del mismo.

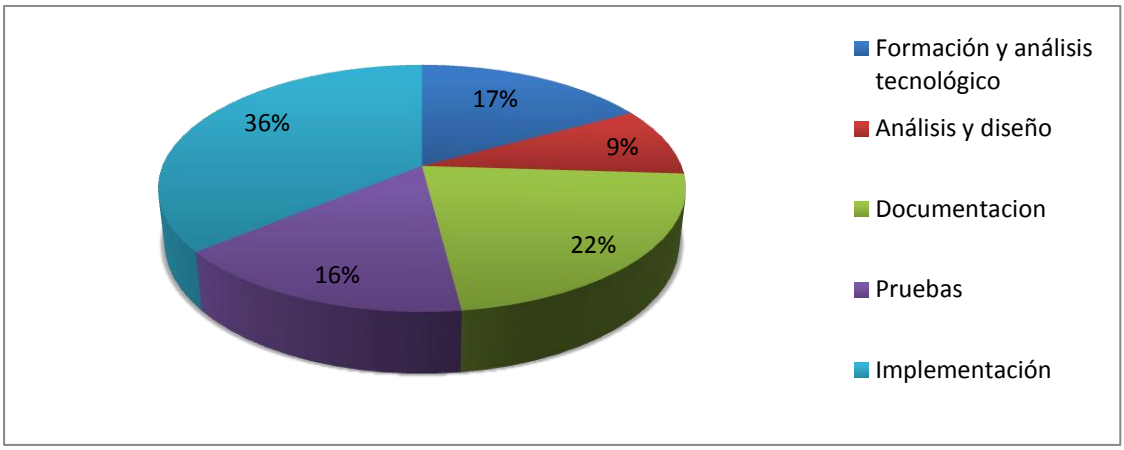
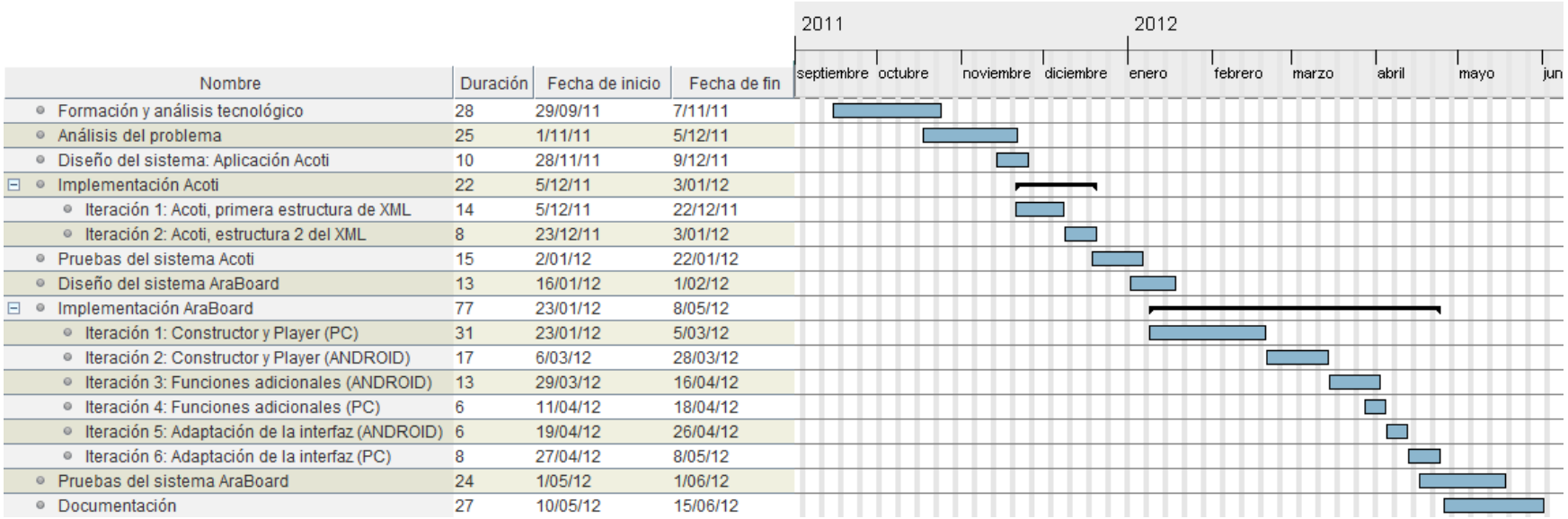


Figura C.2. Porcentaje dedicado a cada fase del proyecto

En total, la duración del proyecto asciende a 677 horas. El desglose de las mismas se muestra a continuación:

- Formación y análisis de tecnologías: 17%, 115h.
- Análisis y diseño: 9%, 61h.
- Implementación: 36 %, 243h.
- Pruebas: 16%, 108h.
- Documentación: 22%,145h.



Nombre	Duración	Fecha de inicio	Fecha de fin
Formación y análisis tecnológico	28	29/09/11	7/11/11
Análisis del problema	25	1/11/11	5/12/11
Diseño del sistema: Aplicación Acoti	10	28/11/11	9/12/11
Implementación Acoti	22	5/12/11	3/01/12
Iteración 1: Acoti, primera estructura de XML	14	5/12/11	22/12/11
Iteración 2: Acoti, estructura 2 del XML	8	23/12/11	3/01/12
Pruebas del sistema Acoti	15	2/01/12	22/01/12
Diseño del sistema AraBoard	13	16/01/12	1/02/12
Implementación AraBoard	77	23/01/12	8/05/12
Iteración 1: Constructor y Player (PC)	31	23/01/12	5/03/12
Iteración 2: Constructor y Player (ANDROID)	17	6/03/12	28/03/12
Iteración 3: Funciones adicionales (ANDROID)	13	29/03/12	16/04/12
Iteración 4: Funciones adicionales (PC)	6	11/04/12	18/04/12
Iteración 5: Adaptación de la interfaz (ANDROID)	6	19/04/12	26/04/12
Iteración 6: Adaptación de la interfaz (PC)	8	27/04/12	8/05/12
Pruebas del sistema AraBoard	24	1/05/12	1/06/12
Documentación	27	10/05/12	15/06/12

Figura C.3. Diagrama de Gantt con la duración del proyecto

Anexo D. Documentación del desarrollo software de AraBoard

Este anexo complementa al capítulo 4 con información referente a las distintas fases del desarrollo software de la herramienta *AraBoard* de creación y reproducción de tableros de comunicación. La realización del software de este proyecto se ha compuesto de dos tareas: por un lado, el desarrollo de la aplicación de construcción de tableros de comunicación, *AraBoard Constructor*; y por otro lado, la aplicación de visualización de tableros, *AraBoard Player*. Para el desarrollo de cada tarea se ha seguido la misma metodología de trabajo, que ha comenzado, para ambos casos, con una fase de análisis donde se ha definido el problema a resolver. Posteriormente se ha continuado con la fase del diseño detallado, e implementación. Por último, durante el desarrollo y tras su finalización, se han realizado distintos tipos de pruebas para garantizar un buen funcionamiento de la aplicación.

D.1 Metodología de análisis

En esta sección se explica la metodología de análisis, en la que se definen los requisitos y los diagramas utilizados, y los pasos seguidos para desarrollar cada modelo de descripción del sistema. La metodología de análisis seleccionada es OMT [10]. Según esta metodología la fase de análisis se realiza en tres pasos:

- **Modelo de objetos:** Describe la estructura estática de los objetos del sistema (relaciones, atributos y operaciones), el cual se representa mediante diagramas de objetos.

- **Modelo dinámico:** Describe los aspectos de un sistema estudiando la organización de estados y la secuencia de operaciones mediante diagramas de estado.

- **Modelo funcional:** Describe las transformaciones que pueden sufrir los datos dentro del sistema, representado gráficamente mediante diagramas de flujo de datos.

Además, se ha considerado oportuno añadir previamente a estos tres pasos la fase de análisis de requisitos, que es propio de la metodología UML, pero que en nuestro caso permite estudiar más a fondo las necesidades de la aplicación.

D.1.1 Análisis de requisitos

La primera etapa del desarrollo consiste en establecer los requisitos que debe cumplir la aplicación para así conseguir una mayor comprensión del problema a resolver. Para ello, antes de comenzar el desarrollo, se realizaron reuniones con los directores del proyecto y con los profesores del CPEE Alborada, en las se establecieron los objetivos y funcionalidades que debía cubrir la aplicación

Los requisitos recogidos sufrieron varias modificaciones a lo largo del desarrollo del software correspondiente. A continuación se listan de forma conjunta, tal y como quedaron en su versión final, agrupados en funcionales y no funcionales. Los requisitos funcionales definen el comportamiento del software, así como las características de sus funcionalidades. Los requisitos no funcionales describen las restricciones tanto software como hardware de la aplicación y las facilidades que debe proporcionar.

Requisitos funcionales: AraBoard Constructor

RF- 1: El sistema permite al usuario construir tableros de comunicación personalizados.

RF- 2: Los tableros de comunicación estarán formados por una o varias celdas que contendrán un pictograma y un título.

RF- 3: El sistema permite al usuario realizar búsquedas en distintos idiomas sobre la colección de pictogramas de ARASAAC.

RF- 4: El sistema permite al usuario seleccionar el idioma de las búsquedas de pictogramas.

RF- 5: El sistema permite al usuario incorporar al tablero un pictograma resultado de búsqueda.

RF- 6: El sistema permite al usuario editar cualquier pictograma que se encuentre sobre el tablero, tanto si éste es un pictograma resultado de búsqueda como si es un pictograma creado por el usuario. Las opciones de edición son:

- Editar el texto asociado a un pictograma.
- Editar la imagen asociada a un pictograma dando opción a cargarla desde el dispositivo.
- Editar el audio asociado a un pictograma dando opción a cargarla desde el dispositivo.
- Cambiar la categoría del pictograma.

RF- 7: El sistema permite al usuario crear un nuevo pictograma e incorporarlo al tablero.

RF- 8: El sistema permite al usuario ver la vista previa del pictograma que está creando y reproducir su audio asociado.

RF- 9: El sistema permite eliminar cualquier pictograma que se encuentre sobre el tablero.

RF- 10: El sistema permite mover y recolocar cualquier pictograma que se encuentre sobre el tablero.

RF- 11: El sistema permite reproducir el audio asociado a los pictogramas de las celdas que compone el tablero.

RF- 12: El sistema permite la construcción de un nuevo tablero.

RF- 13: El sistema permite la carga de un tablero existente.

RF- 14: El sistema permite la edición de un nuevo tablero y de un tablero existente.

RF- 15: El sistema permite editar un tablero mediante las siguientes opciones:

- Permite modificar el título del tablero.
- Permite modificar el número de columnas del tablero.
- Permite modificar el número de filas del tablero.
- Permite modificar el color de fondo del tablero.
- Permite modificar el color de fuente de los títulos de las celdas que lo componen.
- Permite recolocar las celdas que lo componen.

RF- 16: La dimensión máxima del tablero será de 4 filas x 8 columnas (32 celdas en total).

RF- 17: El sistema permite guardar el tablero creado.

RF- 18: El sistema guardará el tablero junto con todos los recursos gráficos y auditivos que lo componen. Los archivos de audio se guardarán en una carpeta y los de audio en otra.

Requisitos funcionales: AraBoard Player

RF- 1: El sistema permite al usuario visualizar y reproducir tableros de comunicación que hayan sido creados con *AraBoard Constructor*.

RF- 2: El sistema permite al usuario reproducir el audio asociado a cada celda del tablero cuando pulse sobre ella.

RF- 3: El sistema detectará las coordenadas de pantalla donde ha pulsado el usuario y ejecutará el audio asociado a la celda más cercana.

Requisitos no funcionales comunes a ambas aplicaciones

RNF- 1: El sistema podrá ser ejecutado en distintas plataformas y dispositivos.

RNF- 2: La interfaz del sistema estará adaptada a las características del dispositivo.

RNF- 3: La aplicación será ejecutada en un dispositivo con conexión a Internet.

RNF- 4: La aplicación será implementada en lenguaje ActionScript 3.0, bajo el entorno de ejecución Adobe Air.

RNF- 5: Los formatos de imagen que serán soportados por la aplicación son PNG y JPEG.

RNF- 6: El formato de audio que será soportados por la aplicación es MP3.

RNF- 7: Los tableros de comunicación deben ser independientes de la plataforma sobre la que se han creado.

RNF- 8: Para la gestión, almacenamiento y carga de los tableros de comunicación se empleará la tecnología XML. El fichero XML contendrá referencias a las rutas de los ficheros de audio y de imagen.

RNF- 9: Junto al fichero XML se guardarán dos carpetas, una de ellas tendrá el nombre *"/imágenes"* y almacenará los ficheros de imágenes; y la otra se llamará *"/audios"* y almacenará los ficheros de audio.

D.1.2 Modelo de objetos

Como se ha explicado anteriormente el modelo de objetos define la estructura estática de los objetos del sistema y proporciona el entorno en el que situar el modelo dinámico y funcional. En esta sección se muestra el diagrama de clases que componen la aplicación y el diccionario de datos. Como anotación a la Figura D.1, decir que para simplificar el diagramas de clases ambas aplicaciones se han representado conjuntamente en el mismo diagrama. Las clases *AraBoard Constructor y AraBoard Player,* serán dos aplicaciones separadas en distintos paquetes. Primero se hace un estudio de las principales clases mostrando las relaciones que existen entre todas ellas, y una vez construido el esquema de clases, se elabora el diccionario de datos.

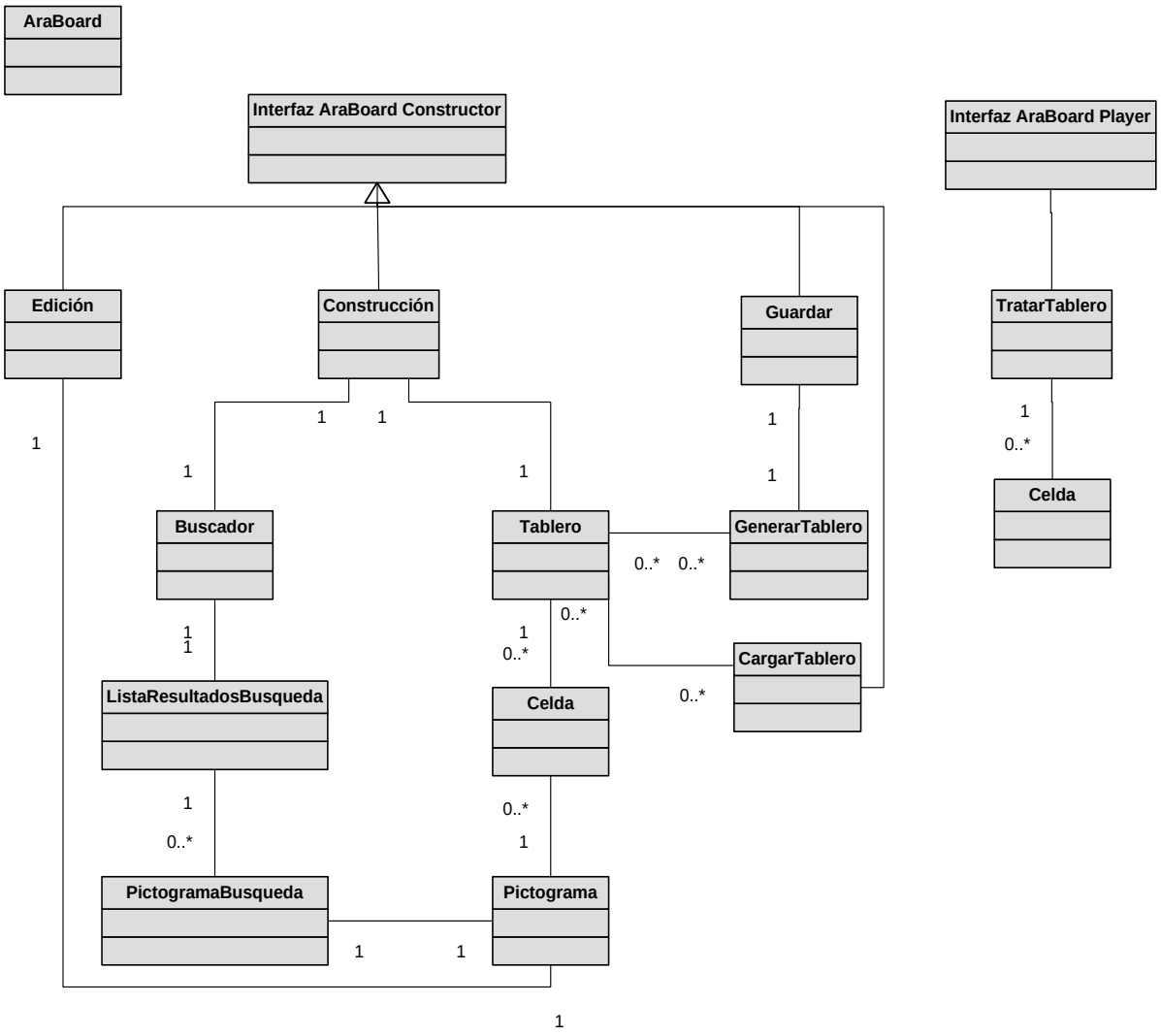


Figura D.1 Diagrama de clases del sistema AraBoard

D.1.3 Diccionario de Datos

El diccionario de datos contiene la descripción de las clases extraídas en el diagrama anterior, haciendo referencia al alcance de cada una de ellas dentro del problema y cualquier suposición o restricción relativa a su uso.

Aplicación *AraBoard*	Sistema que genera una interfaz interactiva.
Interfaz *AraBoard* Player	Diálogo para la comunicación persona/ordenador en el que se muestra información compuesta por componentes con los que el usuario puede interactuar.
Interfaz *AraBoard* Constructor	Diálogo para la comunicación persona/ordenador en el que se muestra información compuesta por componentes con los que el usuario puede interactuar.
Construcción	Diálogo para la comunicación persona/ordenador en el que se posibilita al usuario construir un tablero.
Edición	Diálogo para la comunicación persona/ordenador en el que se posibilita al usuario crear y editar un pictograma.
Guardar	Diálogo para la comunicación persona/ordenador en el que se posibilita al usuario guardar un tablero.
Tablero	Define un tablero de comunicación mediante el número de filas, columnas, fondo, título y listado de celdas que lo componen.
Celda	Define una celda del tablero mediante la fila y columna que ocupa en él, si la celda está ocupada define también un pictograma.
Buscador	Mediante una palabra de búsqueda permite buscar pictogramas en ARASAAC y generar una lista de resultados de búsqueda.
ListaResultados	Define una lista encargada de mostrar por pantalla los resultados de la búsqueda realizada por medio del buscador.
PictogramaBusqueda	Define cada uno de los resultados de búsqueda mediante un pictograma.
Pictograma	Define un pictograma mediante una url de una imagen, una url de un audio, un texto, una categoría y un color.
GenerarTablero	Sistema que tomando las características del tablero construido en pantalla, almacena el tablero de comunicación y todos sus elementos.
CargarTablero	Sistema encargado de mostrar por pantalla un tablero de comunicación previamente construido.

Tabla D.1. Diccionario de datos de AraBoard

D.1.4 Modelo dinámico

El modelo dinámico describe las operaciones y el orden en que se llevan a cabo. Para ello se cuenta con los diagramas de casos de uso.

Diagrama de casos de uso

Los diagramas de casos de uso se crean para representar las operaciones que realiza cada usuario dentro del sistema. Los posibles roles que pueden tener los usuarios en *AraBoard* son:

- **Profesor/Tutor/Familiar:** Es el usuario que, junto con el alumno, realiza actividades educativas o de comunicación. Se encarga de crear, cargar, modificar y guardar los tableros de comunicación.

- **Niño:** Es el usuario final de la aplicación que necesita el uso de los tableros de comunicación. A través del *AraBoard Player* puede utilizar tableros generados con la aplicación de construcción, *AraBoard Constructor*, que le permitan realizar actividades de comunicación.

La Figura D.2 muestra el diagrama de casos de uso que se ha modelado para la aplicación *AraBoard Player*. Y la Figura D.3 muestra el diagrama de casos de uso que se ha modelado para la aplicación *AraBoard Constructor*.

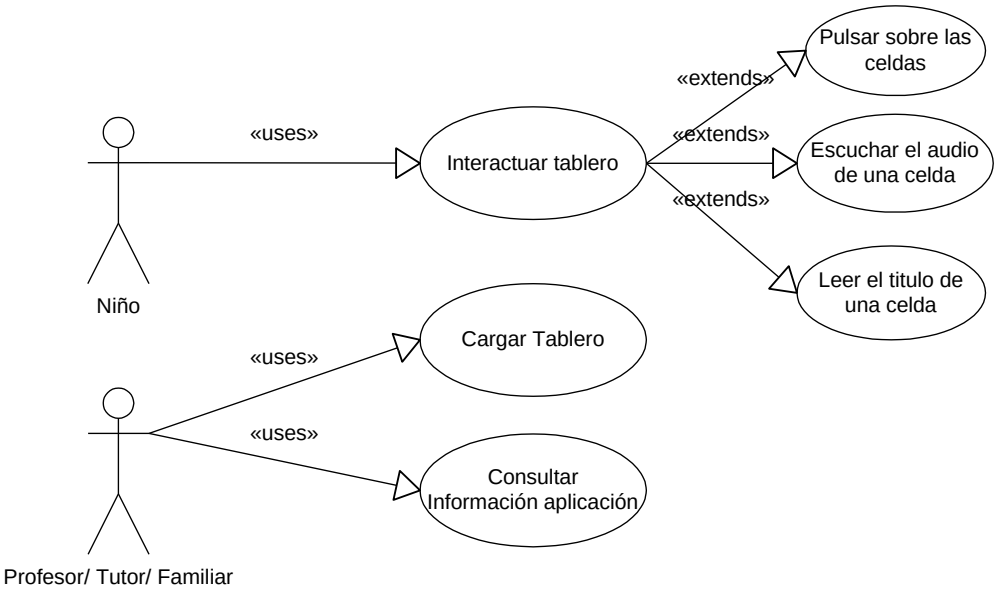


Figura D.2 Diagrama de casos de uso para AraBoard Player

Figura D.3 Diagrama de casos de uso para AraBoard Constructor

D.1.5 Modelo funcional

El modelo funcional se emplea para especificar el significado de las operaciones en el modelo de objetos y las acciones o actividades en el modelo dinámico. Para representar estas actividades se han utilizado los Diagramas de Flujos de Datos, en adelante DFD. Los DFD son grafos que están compuestos de arcos y nodos, donde los arcos pueden ser valores de entrada o salida; o flujos de datos (valores intermedios). Los nodos representan actores, que son los que producen o consumen datos; procesos, que son los que transforman los datos; o almacenes de datos, elementos pasivos que guardan datos.

En la Figura D.4 se muestra el funcionamiento global del sistema de búsqueda de un pictograma mediante el diagrama de flujo de datos de nivel 0. En este nivel, la aplicación interactúa con el usuario que está construyendo el tablero y con la base de datos de pictogramas de ARASAAC. El usuario introduce a través de la interfaz una palabra de búsqueda y el idioma en que desea buscar. *AraBoard* trabaja sobre la palabra proporcionada haciendo uso de un servicio web de la API de ARASAAC, y se encarga de elaborar una lista de pictogramas coincidentes con la palabra de búsqueda.

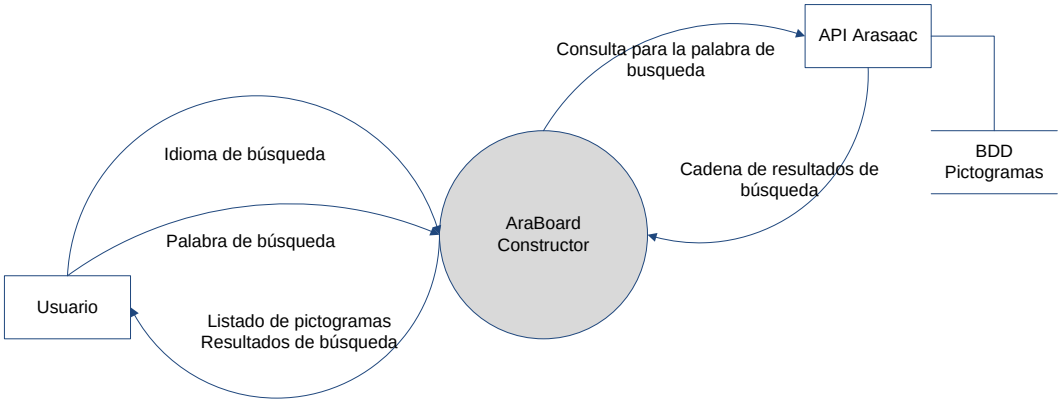


Figura D.4 DFD de nivel 0 para el proceso de búsqueda de pictogramas

En la figura D.5 se muestra el DFD de nivel 1 en el que se ve el funcionamiento del sistema de búsqueda en detalle.

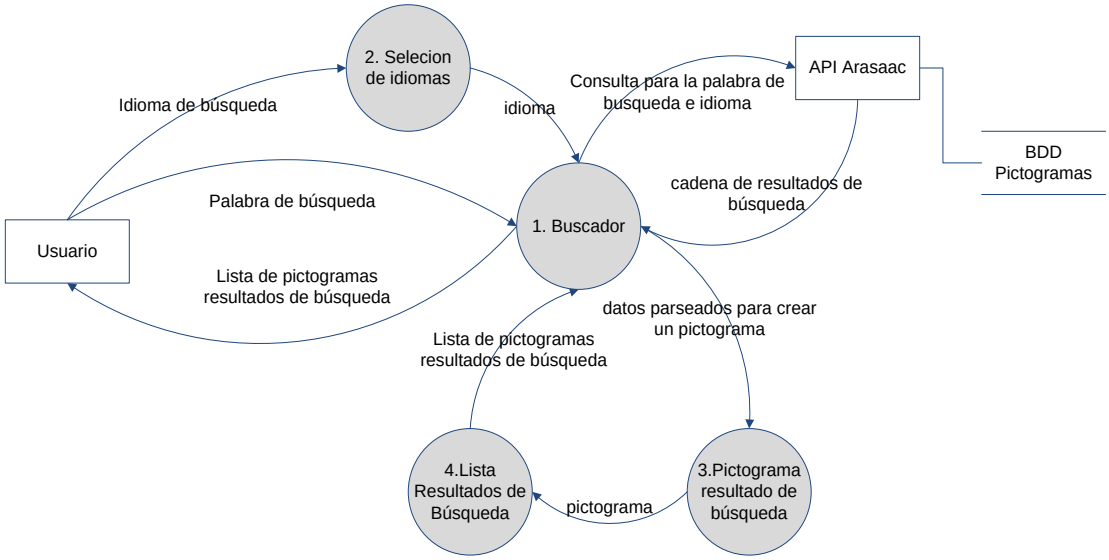


Figura D.5 DFD de nivel 1 para el proceso de búsqueda de pictogramas

En el DFD de nivel 1 se observa en primer lugar el proceso 1, "Buscador", este proceso se encarga de conectar, por medio de un servicio web, con la API de ARASAAC, la cual consultará en su base de datos, y devolverá los datos coincidentes. A partir de estos datos el buscador elabora una lista de pictogramas que mostrará al usuario a través de la interfaz del programa. En el DFD se distingue también el proceso 2, "Selección de idiomas", que permite al usuario seleccionar el idioma en que realizará la búsqueda. Dicha selección se envía al "Buscador" para que incluya el idioma en la consulta a la API. Por otro lado se distinguen los procesos 3, "Pictograma resultado de búsqueda", y 4, "Lista

resultados de búsqueda", que se encargarán de construir los resultados a partir de la información que les envía el "Buscador".

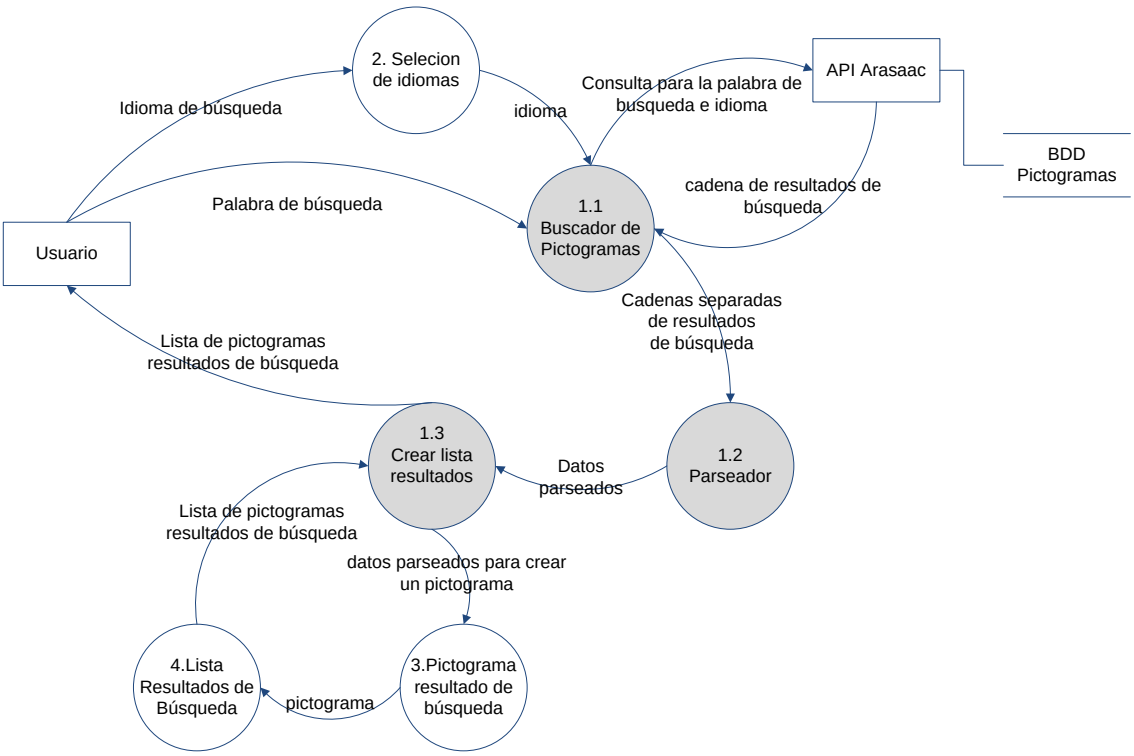


Figura D.6 DFD de nivel 2 para el proceso de búsqueda de pictogramas

La Figura D.6 muestra el Diagrama de Flujo de Datos de nivel 2. El proceso 1.1 "Buscador de pictogramas" se encarga de construir y enviar la consulta http a la API de ARASAAC a partir de la palabra de búsqueda y del idioma que el usuario ha seleccionado (Proceso 2 "Selección de Idioma"). Ésta le devuelve los resultados coincidentes con la palabra de búsqueda, y el buscador los envía al proceso "Parseador". El proceso 1.2 "Parseador" se encarga de tratar estos datos para obtener los datos necesarios para construir un pictograma (Imagen, audio, texto, categoría, color). Cada uno de los datos tratados se transmiten al proceso 1.3 "Crear lista resultados" el cual se encarga de crear los pictogramas resultados de búsqueda y añadirlos a una lista de resultados. Una vez construida esta lista, se muestran los datos al usuario a través de la interfaz de *AraBoard*.

D.2 Metodología de diseño e implementación

La etapa de diseño dentro de la metodología OMT [10]. Consta de dos fases principales.

- **La división en subsistemas,** para dividir la aplicación en componentes o módulos más pequeños**.**
- **El diseño de objetos,** donde se muestran en detalle las clases que componen cada subsistema.

A continuación, se detallan los distintos subsistemas, tanto internos como externos, que conforman la aplicación. En cuanto al diseño del *software* se explica con más detalle la parte referente a la aplicación de construcción aunque también se explican aspectos de la aplicación de reproducción.

D.2.1 Subsistemas de la aplicación AraBoard

El primer paso en el diseño consiste en dividir la aplicación a desarrollar en subsistemas agrupando aquellas clases o funciones con comportamiento similar.

En primer lugar, se describe el sistema de construcción de tableros. Dentro de este sistema se observan diferentes módulos, los cuales se muestran en la Figura D.7, para el caso de la aplicación para PC; y en la Figura D.8, para el caso de la aplicación para Android. Ambos sistemas presenta tres niveles diferentes:

- **Nivel inferior:** En este nivel se encuentran los tableros de comunicación definidos bajo la estructura del fichero XML que se detalla posteriormente en la sección D.3.2. Estos tableros, creados por la aplicación de construcción, podrán ser cargados (en la propia aplicación de construcción) para ser editados.
- **Nivel medio:** Este nivel gestiona todos los datos que forman un tablero de comunicación y los componentes de la interfaz gráfica de usuario. La "inteligencia" del sistema se encuentra en este nivel, puesto que en él se implementan todas las funcionalidades de la aplicación, desde el buscador de pictogramas hasta todas aquellas funciones que permiten construir, modificar, cargar y guardar un tablero de comunicación.
- **Nivel superior:** Está compuesto por los componentes de la interfaz gráfica de usuario. Cada uno de estos componentes controla la interacción del usuario y la presentación de los datos gestionados por el nivel medio. Este nivel varía ligeramente de la interfaz para PC a la interfaz Android, puesto que varios componentes se han adaptado a las características del dispositivo, y en algunos casos particulares ha sido necesario incluir compontes de librerías específicas.

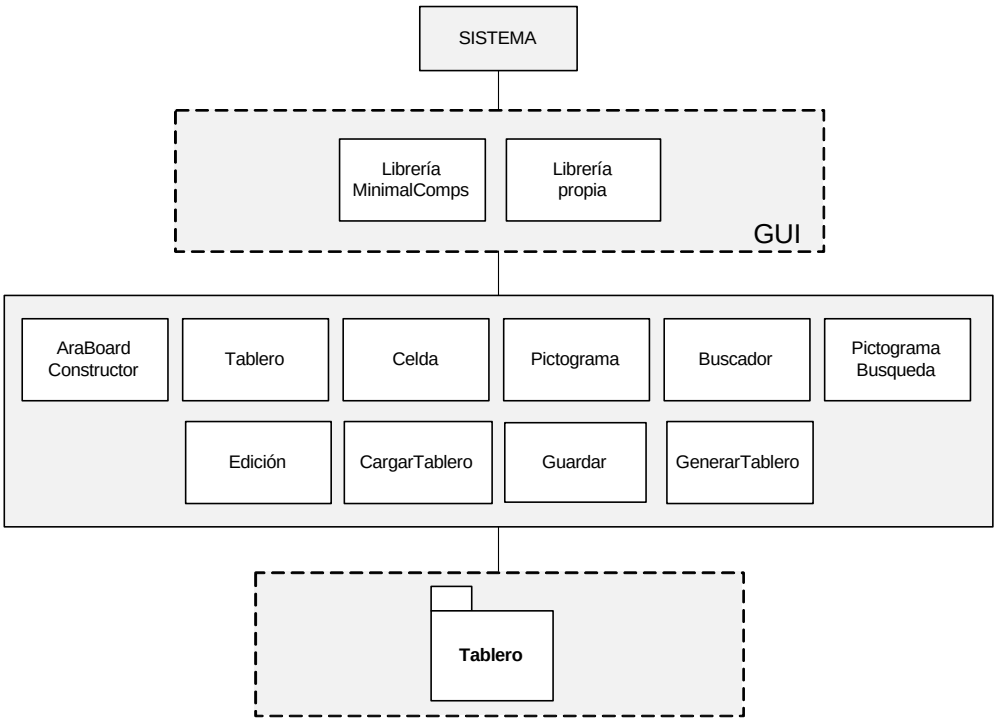


Figura D.7 Diagrama de subsistemas de AraBoard Constructor PC

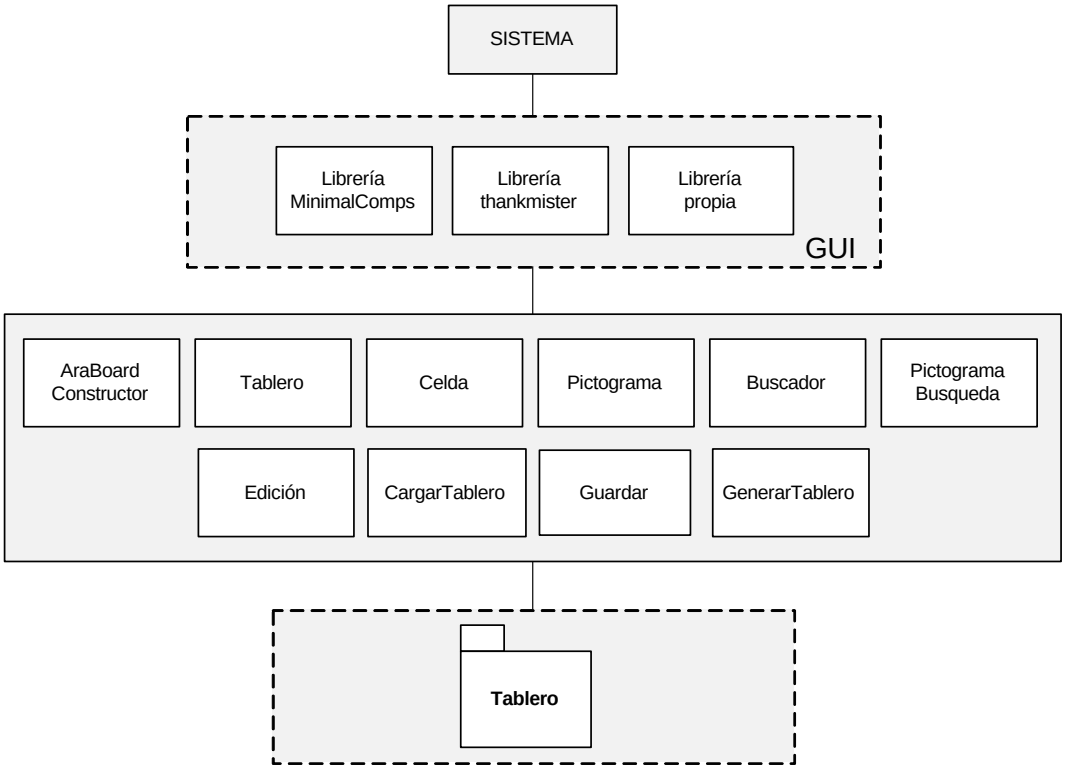


Figura D.8 Diagrama de subsistemas de AraBoard Constructor Android

A continuación se muestra sistema de visualización/reproducción de tableros. Dentro de este sistema se observan diferentes módulos, los cuales se muestran en la Figura D.9 - izquierda (para la versión de PC), y D.9 - derecha (para la versión de Android). La

estructura de estos sistemas es similar a la de la aplicación de construcción. En ellas también se distinguen tres niveles diferentes, cuya explicación es similar a la de la aplicación anterior, a excepción del nivel intermedio, cuyas funcionalidades se centran en la representación gráfica del tablero y en la reproducción del contenido de sus celdas.

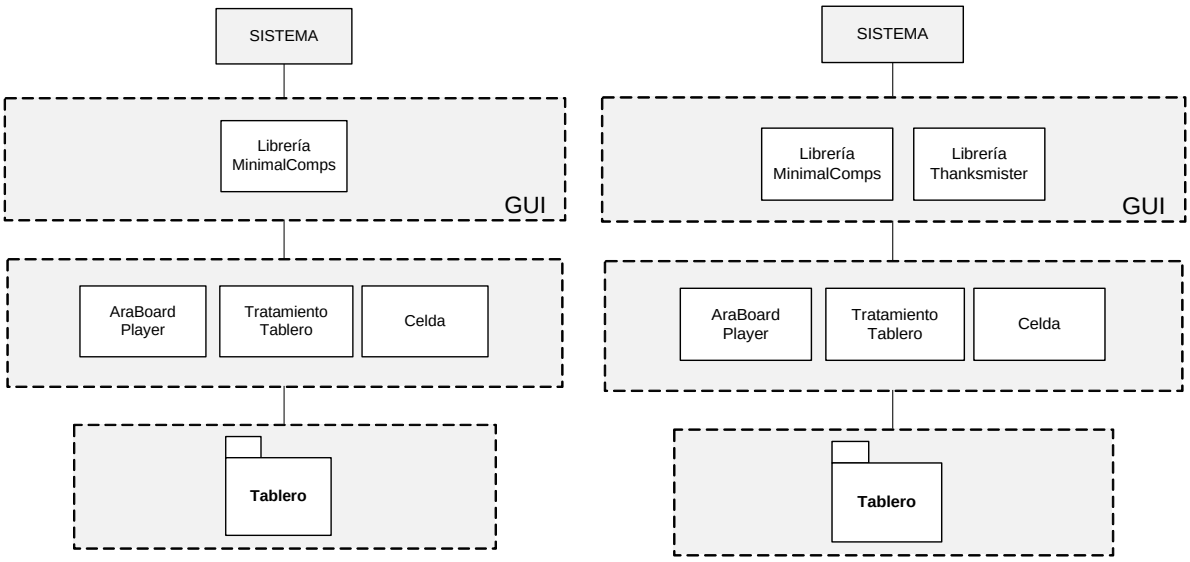


Figura D.9 Diagrama de subsistemas de AraBoard Player PC (izq) y Android (dcha)

A continuación se explica de forma breve cada uno de los subsistemas utilizados por las aplicaciones *AraBoard Constructor y Player*, en sus versiones Android y PC, para cumplir los objetivos fijados.

Subsistemas internos para AraBoard Constructor

Los subsistemas internos propios de la aplicación se dividen en los siguientes módulos:

- *AraBoardC*: Paquete principal de la aplicación de construcción que contiene todos los subsistemas.
- *AraBoardC.AraBoardConstructor*: Módulo principal del paquete *AraBoard*C. Se encarga de dibujar todos los componentes de la interfaz gráfica y de su gestión. Posibilita al usuario realizar operaciones sobre el tablero, sobre el buscador y sobre los pictogramas, tanto los resultados de búsqueda como los que componen las celdas del tablero.
- *AraBoardC.Tablero*: Módulo correspondiente al subsistema *Tablero*. Dibuja en la ventana principal de la aplicación el tablero en construcción y sus elementos principales. Se encarga de gestionar el tablero y la lista de celdas que lo constituyen. Además posibilita al usuario su edición a través de una serie de componentes gráficos.

- *AraBoardC.Celda*: Módulo correspondiente al subsistema *Celda*. Se encarga la inserción, recolocación, edición, y borrado de las celdas que componen un tablero.

- *AraBoardC.Pictograma*: Módulo correspondiente al subsistema *Pictograma*. Se encarga de gestionar la información de un pictograma, gestionar su almacenamiento y la descarga de los elementos que lo componen: fichero de imagen y de audio.

- *AraBoardC.Buscador*: Módulo correspondiente al subsistema *Buscador*. Posibilita al usuario la búsqueda de pictogramas en la colección disponible en el portal web de ARASAAC. Se encarga de conectar con la BBDD de dicho portal a través de la API para obtener una lista de pictogramas resultados de búsqueda.

- *AraBoardC.PictogramaBúsqueda*: Módulo correspondiente al subsistema *Pictograma Búsqueda*. Se encarga de gestionar la información de los resultados de búsqueda y de la incorporación de éstos al tablero en construcción.

- *AraBoardC.Edición*: Módulo correspondiente al subsistema *Edición*. Contiene la ventana principal del panel de edición de pictogramas y sus elementos principales. Posibilita la creación y edición de pictogramas.

- *AraBoardC.Guardar*: Módulo correspondiente al subsistema Almacenamiento. Contiene la ventana principal "*Guardar Tablero*" y sus elementos principales. Posibilita guardar un tablero en la memoria del dispositivo.

- *AraBoardC.CargarTablero*: Módulo correspondiente al subsistema *Cargar*. Posibilita cargar un tablero externo a la aplicación.

- *AraBoardC.GenerarTablero*: Módulo correspondiente al subsistema *GenerarTablero*. Posibilita escribir en el sistema el tablero en construcción de la aplicación.

- *AraBoardC.LibreríaPropia*: Módulo correspondiente a la "librería de componentes propios" de la interfaz GUI. En este módulo incluye algunos componentes gráficos que no se encontraban en las librerías gráficas externas y fueron implementados para cubrir algunas de las funcionalidades.

Subsistemas internos para AraBoard Player

Los subsistemas internos propios de la aplicación se dividen en los siguientes módulos:

- *AraBoardP*: Paquete principal de la aplicación de visualización que contiene todos los subsistemas.

- *AraBoardP.AraBoardPlayer*: Módulo principal del paquete *AraBoardP*. Se encarga de dibujar todos los componentes de la interfaz gráfica y de su gestión. Posibilita al usuario visualizar e interactuar con tablero de comunicación.

- *AraBoardP.TratamientoTablero*: Módulo que se encarga de cargar tablero XML externo a la aplicación y de gestionar sus elementos para que el usuario los visualice por pantalla.

- *AraBoardP.Celda*: Módulo correspondiente al subsistema Celda. Se encarga de dibujar y posicionar las celdas en el tablero, y gestiona los eventos asociados a ellas.

Subsistemas externos para AraBoard Constructor y Player

Las bibliotecas externas y que son requeridas para el funcionamiento de la aplicación son:

- ***MinimalComps*** [11]: Esta librería incluye un conjunto de componentes gráficos implementados en lenguaje *ActionScript* 3.0. Estos componentes son compatibles con Flash Profesional en entornos web y en entornos de ejecución Air y Air para iOS y Android (a pesar de los ajustes de tamaño, que se tuvieron que adaptar manualmente para las plataformas móviles).

- ***Thanksmister*** [12]: Esta librería implementa una lista táctil programada en lenguaje *ActionScript* 3.0, que puede ser ejecutada sobre varios dispositivos bajo entornos de ejecución Air (para iOS y Android). La librería incluye el módulo *TouchList* que contiene las clases que permiten gestionar los eventos y dibujar la lista y los componentes gráficamente que la forman. La lista posibilita al usuario realizar *scroll* táctil, el dedo del usuario realiza el papel del ratón permitiéndole desplazarse por la lista y seleccionar sus elementos. Además, la lista posibilita las siguientes acciones: crea la lista, añade y borra elementos, permite seleccionar elementos y se ocupa de eventos táctiles enviados por los procesadores de los dispositivos.

- Una **API XML** basada en la especificación de ECMAScript [37] para XML (E4X) (ECMA-357 edición 2). E4X es una extensión del lenguaje ECMAScript que añade XML como un tipo de datos nativo del lenguaje.

D.2.2 Diseño de Objetos

El Diseño de Objetos muestra las entidades u objetos principales en la aplicación. A lo largo de esta sección se implementarán los objetos identificados en fases anteriores.

Diagrama Entidad-Relación y Modelo Relacional

A continuación se muestran el diagrama Entidad/Relación (Figura D.10) y el modelo relacional que reflejan los datos y las relaciones que establecen. El diagrama E/R es un modelo de datos conceptual de alto nivel que describe los tipos de datos, sus relaciones y restricciones. Mientras que el modelo relacional (Figura D.11) representa el diseño de los tipos de entidades y muestra las relaciones haciendo uso de las claves primarias y las claves ajenas de cada entidad.

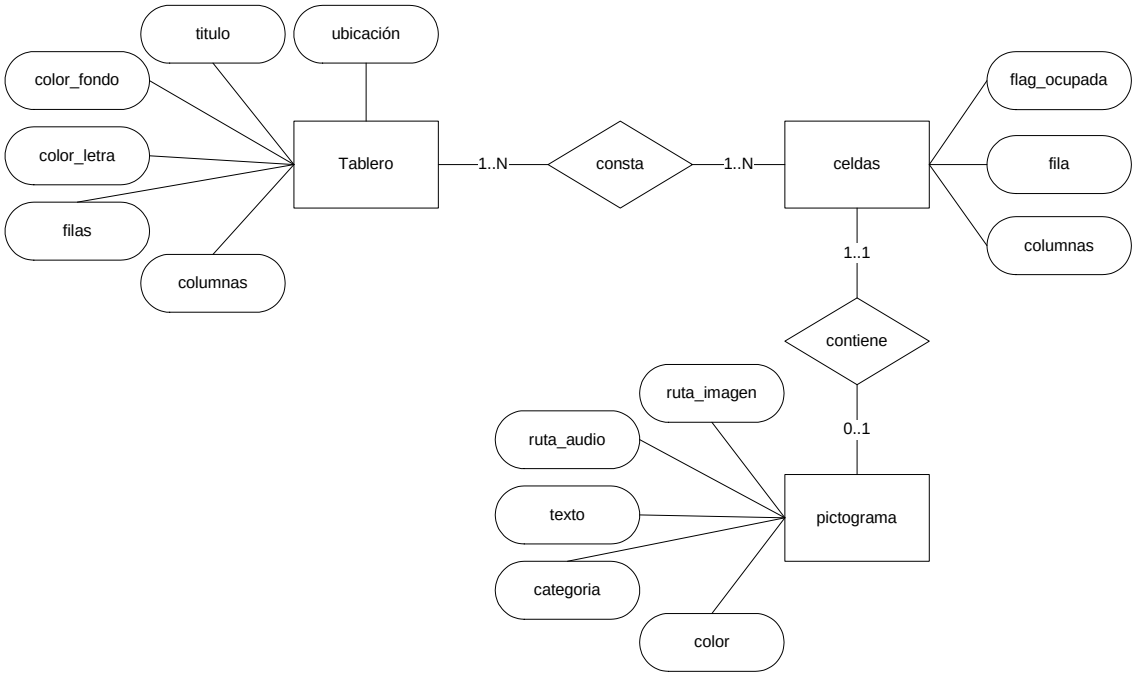


Figura D.10 Diagrama Entidad - Relación

Tablero (ubicación_tablero, título, color_fondo, color_letra, filas, columnas)

Pictograma (ruta_imagen, ruta_audio, texto, categoría, color)

Celda (fila, columna, ocupada)

Contiene (fila, columna, ruta_imagen, ruta_audio)

Consta (fila, columna, ubicación_tablero)

RESTRICCIONES:

- ubicación_tablero es clave ajena de tablero. *NO NULO.*
- (ruta_imagen, ruta_audio) es clave ajena de Pictograma. *NO NULO.*
- (fila, columna) es clave ajena de Celda. *NO NULO.*

Figura D.11 Modelo Relacional

Diseño de la aplicación de Construcción

Durante las fases de análisis y de diseño se generaron distintos diagramas que facilitarían la fase posterior de implementación. En esta sección recoge únicamente el diseño elaborado para la aplicación de construcción por considerarse de mayor interés.

A continuación se muestran todas las clases y las relaciones que existen entre ellas (Figura D.12). Para simplificar el diagrama, se han incluido únicamente los atributos de las clases más importantes.

En los puntos que se muestran a continuación se describen y explican las clases y métodos más significativos de la aplicación de construcción desde perspectiva cercana a la implementación.

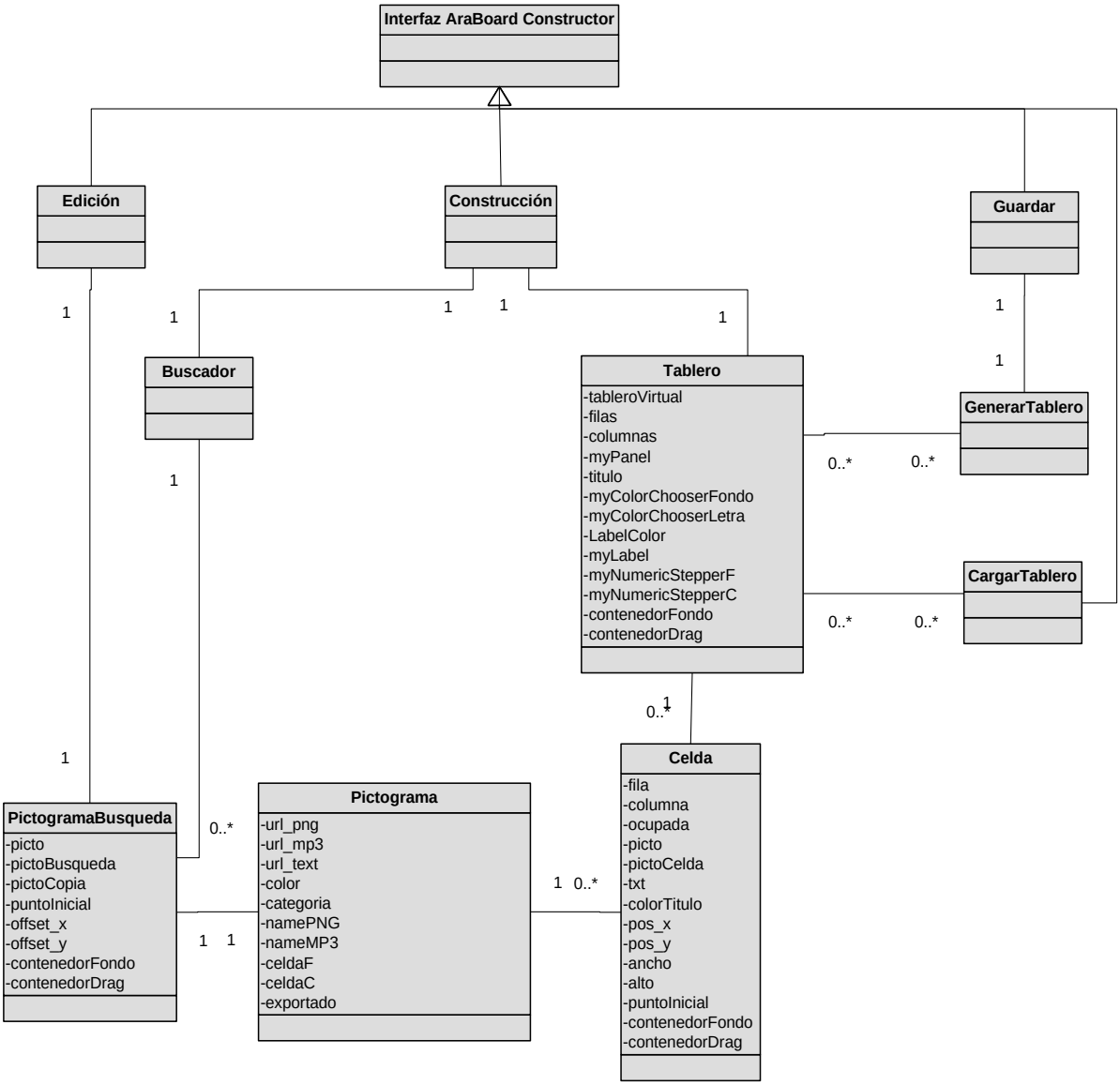


Figura D.12 Diagrama de clases

▪ DISEÑO DE LA INTERFAZ GRÁFICA

El diseño de la interfaz gráfica comienza por el estudio previo de varias librerías de componentes gráficos compatibles con el lenguaje ActionScript 3.0 y el entorno de ejecución AIR. Tras realizar el estudio, se optó por emplear la librería gráfica *Minimal Componentes*, desarrollada por Keith Peters, principalmente por su extensa documentación. Una vez seleccionada la librería, se estudiaron en detalle los componentes de la biblioteca gráfica *MinimalComps* así como la documentación disponible de la misma [11] [38].

A continuación se diseñaron los prototipos de las pantallas de las aplicaciones (*AraBoard Constructor y Player*), sección D.2.3 de este anexo. Una vez diseñadas todas las ventanas, se procedió a la construcción de la interfaz gráfica. Debido a que, en ocasiones, los componentes de la biblioteca no se ajustaban al comportamiento deseado en la aplicación, se modificaron algunos de sus componentes, y se crearon aquellos que no existían y resultaban necesarios para conseguir las funcionalidades deseadas.

En la Figura D.13 se muestran las clases de la interfaz que fueron creadas como complemento a la librería *MinimalComps*.

En la Figura D.14 (página siguiente) se puede observar el diagrama de clases la librería *MinimalComps* (las clases sombreadas en gris indican que se han hecho modificaciones en ellas). En el diagrama se observa que la clase del elemento *component* se encuentra en la parte superior. Esto representa que todos los demás elementos extienden de la clase *componet*. Se indican mediante líneas de relación aquellos elementos que extienden de alguna otra clase de la librería.

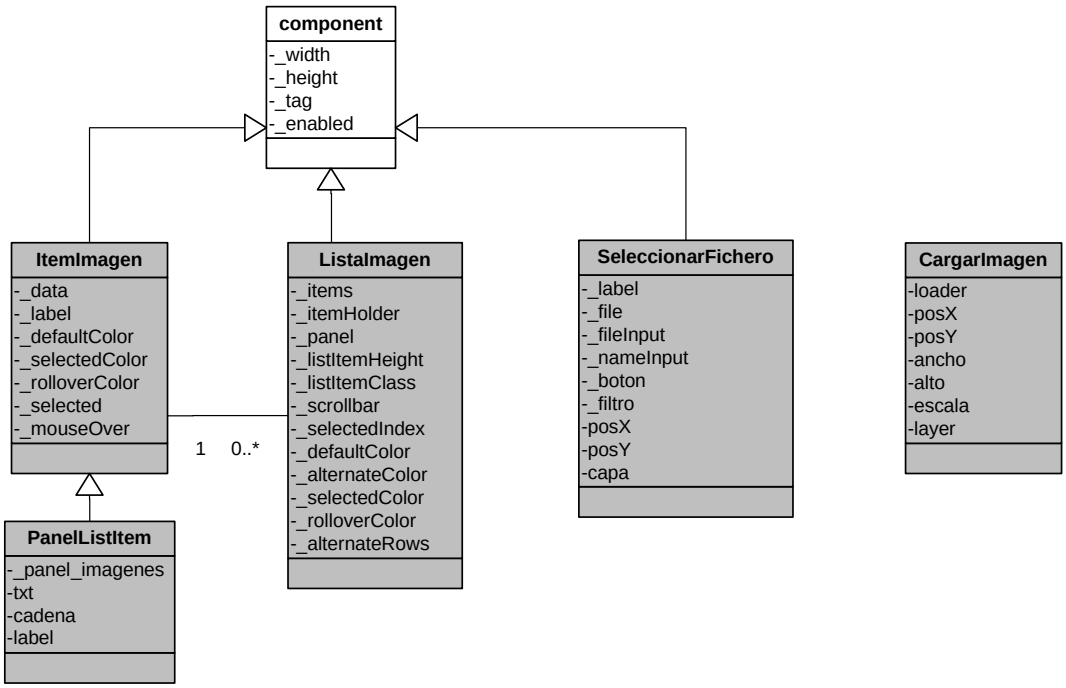


Figura D.13 Clases implementadas como extensión de la librería MinimalComps

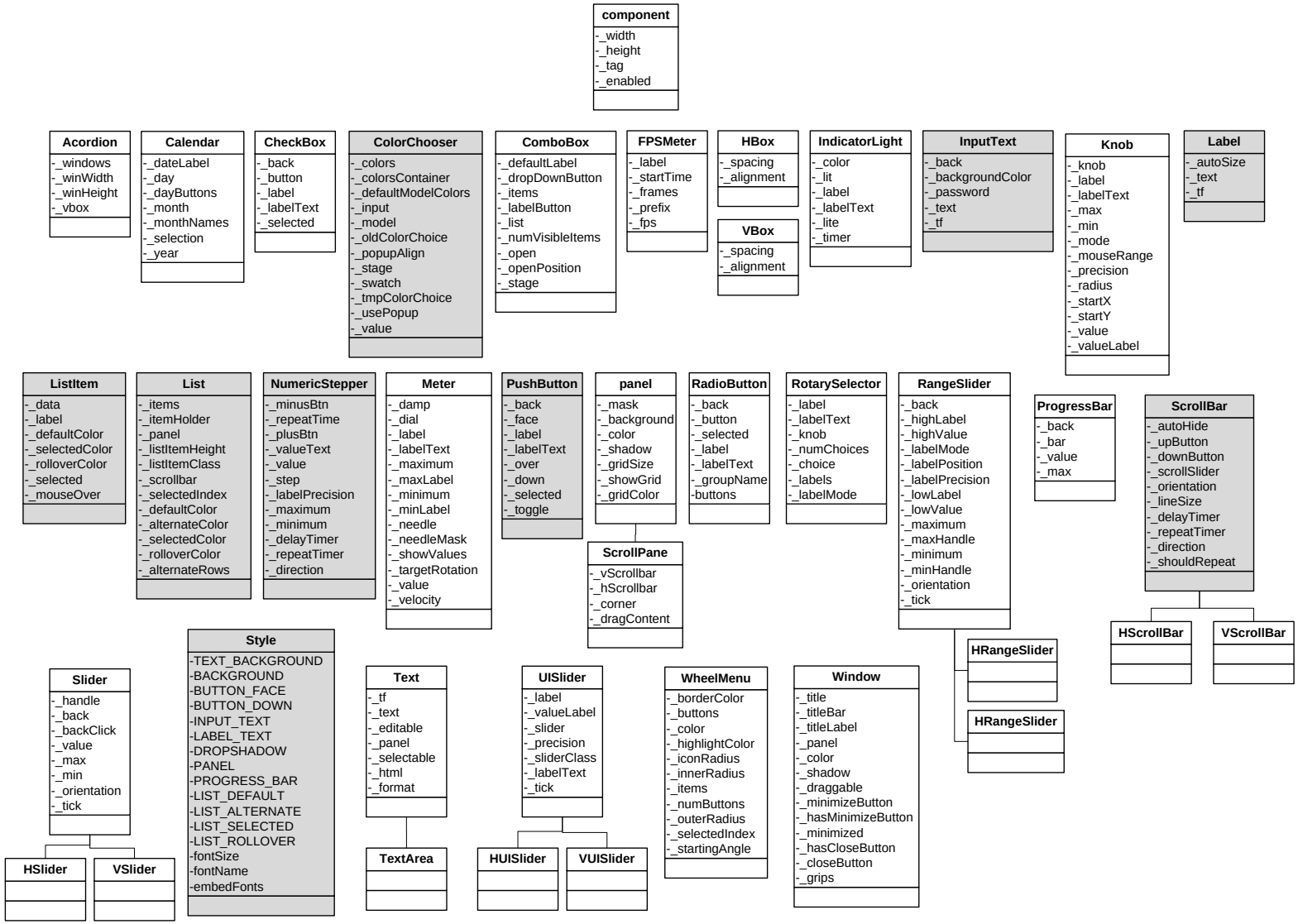


Figura D.14 Diagrama de clases de la librería MinimalComps

Las figuras D.15, D.16, y D.17 representan la relación existente entre las interfaces de la aplicación de construcción: "*Guardar*", "*Edición*" y "*Construcción*"; y los componentes de la librería gráfica *MinimalComps,* así como los componentes modificados y creados como extensión a dicha librería gráfica (Figuras D.13 y D.14).

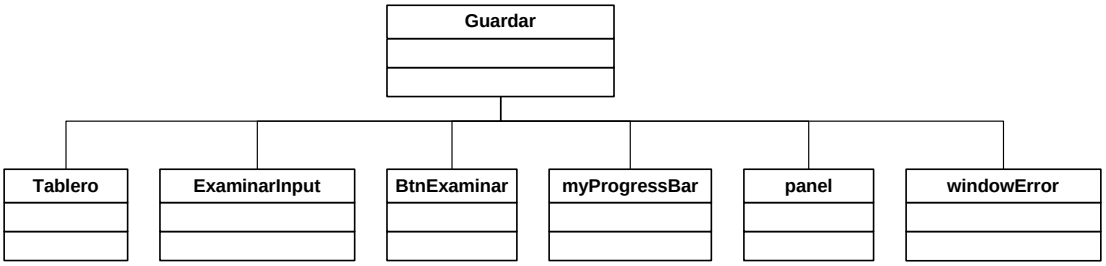


Figura D.15 Relación de la clase de almacenamiento con los elementos de la interfaz

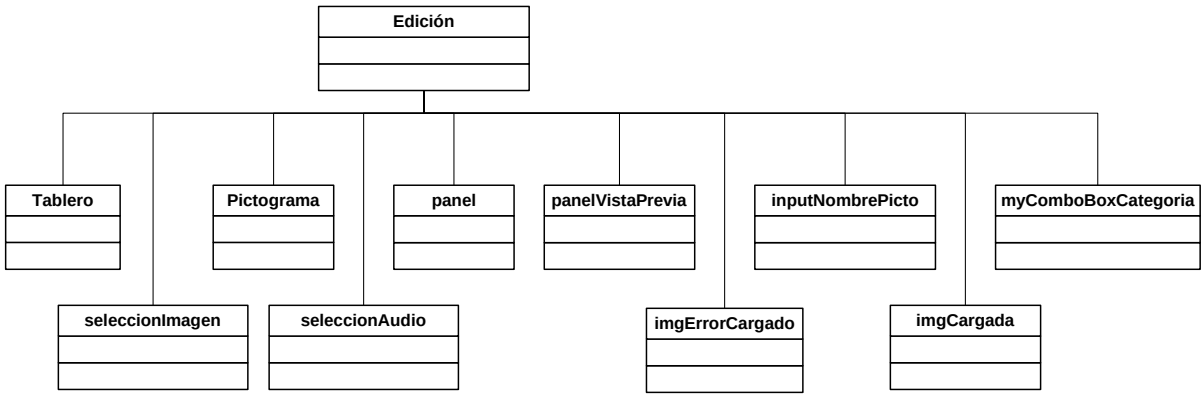


Figura D.16 Relación de la clase de edición con los elementos de la interfaz

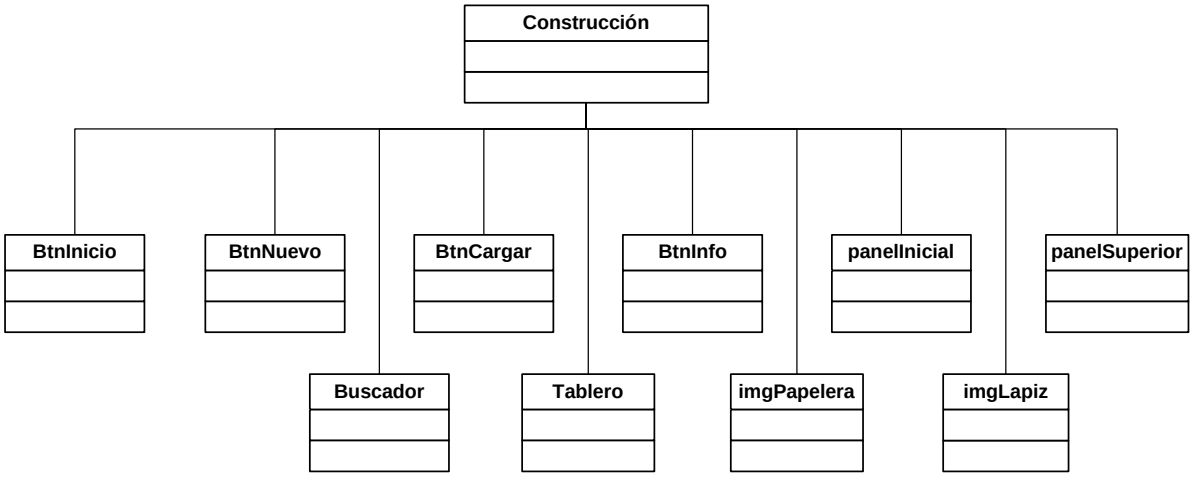


Figura D.17 Relación de la clase de construcción con los elementos de la interfaz

La mayor dificultad encontrada en esta fase fue construir la interfaz gráfica manualmente a partir del estudio del código de las clases de *MinimalComps*. Los ejemplos de construcción de componentes no eran muy numerosos, debido a que la librería de componentes había sido desarrollada recientemente (Junio de 2011, 3 meses antes de que comenzase este PFC).

Por otro lado, hay que considerar que, aunque la estructura de los componentes en pantalla es muy similar en la interfaz para PC y en la interfaz para Android, todos los componentes se tuvieron que redimensionar y ajustar manualmente. Además, las dimensiones de la pantalla de los Smartphone y tablet tienen proporciones fijas, que son diferentes a las de los ordenadores, por lo que resultó necesario cambiar la disposición de los componentes en la interfaz de ambas versiones (para PC, y para dispositivos móviles y tablet). A esto se debe añadir, que aspectos propios del dispositivo, como la rotación automática, también deben ser considerados. En el caso de la rotación automática, la interfaz se tiene que redibujar cada vez que se detecte la rotación. Este problema no afecta tanto a la interfaz de construcción, puesto que ésta debe aparecer siempre ocupando todo el ancho de la pantalla, pero es un aspecto muy importante en la interfaz de reproducción, ya que estará expuesta a un conjunto de usuarios muy heterogéneos (que en ocasiones preferirán la disposición vertical u horizontal de la pantalla).

▪ LA CLASE BUSCADOR

La Clase buscador es una de las clases fundamentales en la aplicación *AraBoard* Constructor. Esta clase ha sido implementada para permitir realizar búsquedas en la colección de pictogramas de ARASAAC, mediante la conexión con su API.

En la Figura D.18 se muestra la clase Buscador junto sus atributos y métodos más significativos.

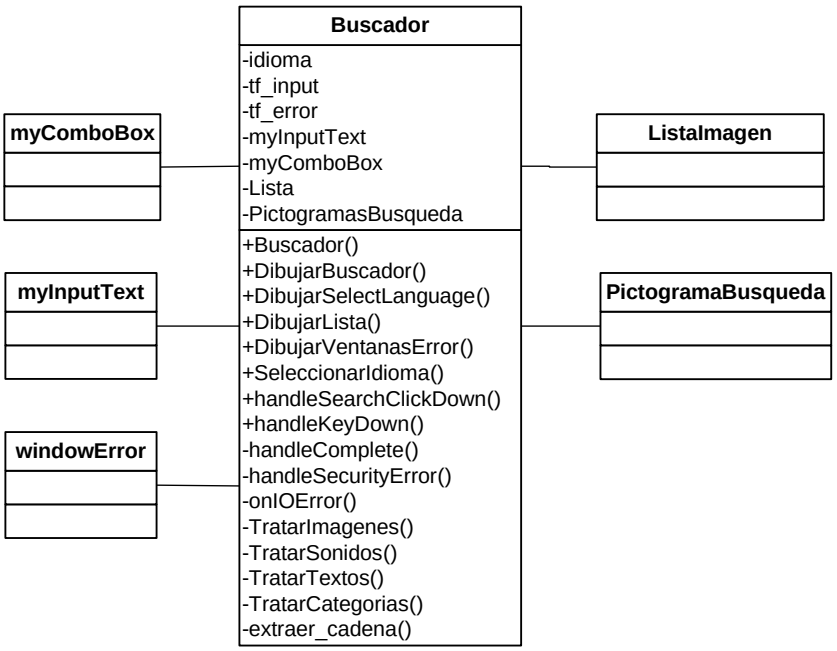


Figura D.18. Clase Buscador

A continuación se agrupan los métodos según la funcionalidad que desempeñan dentro de la clase y posteriormente, se explican sus aspectos más destacados.

- **Método constructor:**

```
public function Buscador (c:Stage, capaFondo:Sprite,
                          capaDrag:Sprite, t:Tablero)
```

- **Métodos para dibujar los componentes gráficos que forman el buscador:**

```
public function DibujarBuscador():void
public function DibujarSelectLanguage():void
public function DibujarLista():void
public function DibujarVentanasError():void
```

- **Métodos de gestión de eventos:**

```
public function SeleccionarIdioma(event:Event):void
public function handleSearchClickDown(event:MouseEvent):void
public function handleKeyDown(event:KeyboardEvent):void

private function handleComplete( event:Event ):void
private function handleSecurityError( event:Event ):void
private function onIOError(event:IOErrorEvent):void
```

- **Métodos para tratar los resultados de búsqueda (parseador):**

```
private function TratarImagenes(cadena:String):Array
private function TratarSonidos(cadena:String,
                               numElementos:int):Array
private function TratarTextos(cadena:String):Array
private function TratarCategorias(cadena:String):Array
private function extraer_cadena(url:String, patron:String,
                                esAudio:Boolean):String
```

En primer lugar, se muestra el grupo de los métodos que permiten dibujar en pantalla los distintos elementos que componen el buscador: Seleccionador de idiomas, el campo de búsqueda y la lista de resultados de búsqueda. Estos métodos hacen uso de las librerías de componentes gráficos, *MinimalComps* y los implementados en la librería propia (citados en la sección anterior), y son los encargados de dibujar en pantalla los elementos con los que el usuario puede interactuar para realizar una búsqueda.

El segundo grupo incluye aquellos métodos que se encargan de obtener la palabra de búsqueda que el usuario introduce a través del campo buscador, y de enviar la consulta http a la API de ARASAAC.

La primera función de este grupo permite seleccionar el idioma. La segunda función es llamada cuando el usuario presiona el botón de la lupa que hay al lado del buscador. La tercera se produce cuando el usuario presiona la tecla *Enter*. Ambas funciones tiene el mismo comportamiento asociado.

```
public function handleSearchClickDown(event:MouseEvent):void
public function handleKeyDown(event:KeyboardEvent):void
```

La **conexión con la API** se realiza a través de los siguientes pasos:

1. Se crea una **instancia de la clase *URLLoader*** para poder realizar la consulta y obtener los datos de la API:

```
var loader:URLLoader = new URLLoader();
```

2. **Se añaden los *listeners* a la instancia** `URLLoader`**. Éstos se ejecutarán cuando la petición se complete y se obtengan los resultados:**

```
/* Definimos las rutinas de eventos que se ejecutaran cuando se
   complete el loader (handleComplete), o en caso de que ocurra
   un error (handleSecurityError) o error de conexion (IOéRROR)
*/

loader.addEventListener( Event.COMPLETE, handleComplete );
loader.addEventListener(IOErrorEvent.IOéRROR, onIOError);
loader.addEventListener(SecurityErrorEvent.SECURITYéRROR,
                        handleSecurityError );

/* Configuracion del loader para cargar las variables
   codificadas en la URL */

loader.dataFormat = URLLoaderDataFormat.TEXT;
```

3. **Se realiza la consulta http especificando las variables de la API. A continuación se detallan los parámetros de la API**

```
VARIABLES DE LA API DE ARASAAC:

callback=json
language=ES             // Ver formatos en:
                        // http://es.wikipedia.org/wiki/ISO_639-1
word= cadena de texto a buscar
catalog=colorpictos     // b&wpictos para pictogramas en
                        // Blanco y Negro
nresults=10             // Numero de resultado que quiero obtener
                        // como máximo
thumbnailsize=150       // Tamaño de la minuatura de la que
                        // quiero me genere URL
TXTlocate=1             // Tipo de búsqueda:
                                1-Comienza por
                                2-Contiene
                                3-Termina por
                                4-Es igual a.
```

El usuario puede personalizar los campos *word* (a través de la palabra introducida en el buscador), y el campo *language* (a través del seleccionador de idioma). El código de la consulta es el siguiente:

```
loader.load( new URLRequest
((http://arasaac.org/api/index.php?callback=json&language= +
idioma + "&word=" + tf_input.text + Resto de datos de la API));
```

El tercer grupo incluye aquellos métodos que se encargan de tratar los resultados de búsqueda que devuelve la API (`loader.data`), y elaborar un listado de pictogramas. Los pasos son los siguientes:

1. El resultado de búsqueda es tratado ("parseado") para obtener las imágenes, los audios, las categorías y los textos de cada uno de los pictogramas almacenados en la cadena `loader.data`:

```
/* Funciones para almacenar las imágenes, los sonidos, los textos
   y las categorias de los resultados de busqueda almacenados en
   loader.data */

listaI=TratarImagenes(loader.data);
listaS=TratarSonidos(loader.data, listaI.length );
listaT=TratarTextos(loader.data);
listaC=TratarCategorias(loader.data);
```

2. A partir de los datos almacenados en las listas anteriormente creadas se crean los pictogramas resultados de búsqueda:

```
// Extraemos el nombre de la imagen a descargar
var namePNG:String=extraer_cadena(listaI[i],"originales/", false);

// Extraemos el nombre del audio a descargar
var nameMP3:String=extraer_cadena(listaS[i], "locuciones/", true);

// Creamos el pictograma de búsqueda
var pictoB:PictogramaBusqueda = newPictogramaBusqueda
        (layerFondo,layerDrag,listaI[i],listaS[i],listaT[i],
         listaC[i],namePNG,nameMP3, tablero);
```

3. Se crea una lista de pictogramas que se muestran al usuario:

```
Lista.addItem(PictogramasBusqueda[i]);
```

▪ LA CLASE TABLERO

La clase tablero es una de las clases fundamentales de la aplicación de construcción. En ella se definen todos aquellos métodos y atributos necesarios para dibujar los componentes gráficos que lo constituyen y las herramientas de modificación: contadores de número de filas, columnas, título, etc. Por otro lado, se encuentran todos aquellos métodos y atributos que permiten gestionar su comportamiento. En este aspecto un tablero se relaciona con otras clases, concretamente con las celdas y pictogramas resultados de búsqueda.

En el diagrama de clases mostrado en la Figura D.19 se aprecian todas las relaciones de la clase tablero con otras clases (por ejemplo, con la clase celda, edición, etc.), bien se traten de clases de componentes gráficos o clases para la gestión del tablero.

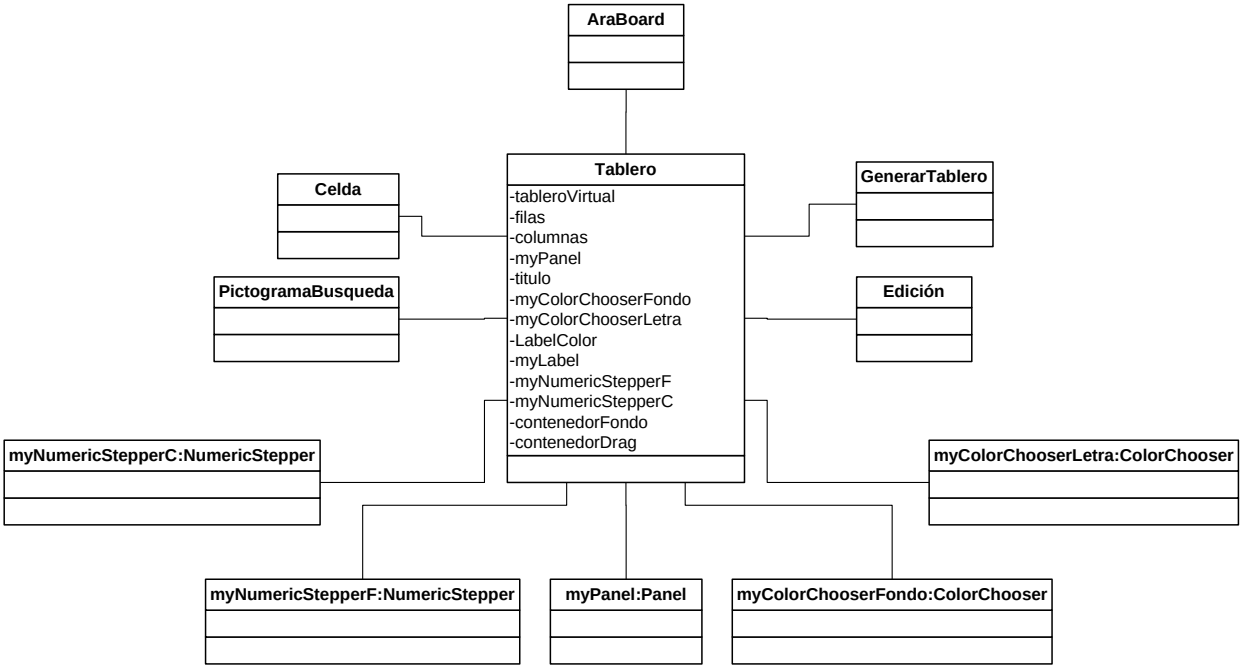


Figura D.19. Clase Tablero

Los principales métodos de la clase tablero son los siguientes:

- **Constructor:**

```
public function Tablero (capaF: Sprite, capaD: Sprite, w:Number,
h:Number)
```

- **Métodos de dibujo en la interfaz**

```
public function DibujarTituloTablero():void
public function DibujarColorChooser():void
public function ColorearTablero(event:Event):void
```

```
public function ColorearLetras(event:Event):void
public function DibujarContadores():void
public function MostrarPanelEdicion (event:MouseEvent):void
```

- **Métodos de gestión del tablero gráficamente y funcionalmente**

```
public function ActualizaFilas(event:MouseEvent):void
public function ActualizaColumnas(event:MouseEvent):
public function ActualizaTablero():void
public function ActualizarCeldasExistentes(anchura:int, altura:int):void
public function ActualizarTitulosCeldas():void
public function BorrarSubTableros():void
public function ActualizaTableroVirtual(actualizacion:int,
                                        cambioEnFilas:Boolean):void
public function CargarTableroVirtual():void
public function MostrarTableroVirtual():
```

- **Métodos de gestión del las celdas del tablero**

```
public function EnTablero (PosX:int, PosY:int):Boolean
public function Colocar (picto:Pictograma, PosX:int, PosY:int):Boolean
public function BuscarCelda (fil:int, col:int):int
```

- **Métodos de generación del tablero**

```
public function GenerarContenidoXML (fXML:GenerarTableroXML, f:File):void
```

Entre los métodos más destacados de la clase tablero, se encuentran la función *Colocar*, *BuscarCelda* y la función *actualizarTableroVirtual* descritas a continuación.

La función ***Colocar*** se llama cada vez que el usuario mueve un pictograma sobre el tablero y lo suelta para colocarlo. Si el pictograma se encuentra sobre una celda libre del tablero, será colocado con éxito. Si el pictograma se encuentra sobre una celda ocupada o sobre cualquier otro lugar del tablero será devuelto a su posición inicial.

```
//-----------------------------------------------------------------
public function Colocar(picto:Pictograma, PosX:int PosY:int):Boolean {
/* -----------------------------------------------------------------
   Pre: El pictograma esta sobre el tablero.
   Post: Coloca el pictograma en la posición adecuada del tablero. */
```

El algoritmo realiza los siguientes pasos.

1. Recorremos el vector de celdas del tablero.
2. Comprobamos si las coordenadas (PosX, PosY) del ratón están dentro de la celda, y además la celda no está ocupada.
    a. Si las coordenadas están dentro y la celda está libre →
        - Marcamos la celda como ocupada.
        - Dibujamos el título de la celda.
        - Posicionamos el pictograma.
        - Comprobamos si el pictograma procedía de un resultado de búsqueda o de otra celda. Para ello empleamos la función *BuscarCelda* que nos dice en qué celda se encuentra el pictograma. Si procedía de una celda lo eliminamos su contenido de la celda anterior. Si no, no hacemos nada.
    b. O bien las coordenadas no se corresponde a una posición sobre el tablero o bien la celda no está libre → No hacemos nada.
3. Devolvemos cierto si se ha colocado, y falso en caso contrario.

La función ***Buscar*** se llama desde la función Colocar previamente descrita. Esta función se encarga de buscar en el vector de celdas del tablero, la posición que ocupa una celda dada su posición en el tablero: fila y columna. De esta manera se puede comprobar si la celda procedía de una celda del tablero (si se encontraba en el vector del tablero), o procedía de un resultado de búsqueda (no se encontraba en el vector del tablero).

```
// ---------------------------------------------------------------------
public function BuscarCelda (fil:int, col:int):int {
/* ---------------------------------------------------------------------
Pre: "fil" y "col" son las filas y columnas de la celda que buscamos
Post: Devuelve la posición en el tablero virtual de la celda
      correspondientes a (fil,col).
*/
```

El **algoritmo** es el siguiente:

1. Recorre el vector de celdas del tablero
2. Si la celda contiene un pictograma →
    a. Comprobamos si sus filas y columnas coinciden con “fil” y “col”.
3. Devuelve el índice del vector si lo encuentra, y -1 en caso contrario.

La función ***ActualizaTableroVirtual*** se encarga de mantener actualizado y ordenado el vector de celdas del tablero. Esta función es llamada siempre que se produce alguna variación en el tablero: cambio de filas, de columnas, etc.

```
// ------------------------------------------------------------------
public function ActualizaTableroVirtual(actualizacion:int,
                                        cambioEnFilas:Boolean):void {
/* ------------------------------------------------------------------
  Pre: Actualizacion:
       1 si se ha producido un incremento en las filas o columnas del ta-
       blero
       0 si se ha producido un decremento en las filas o columnas del ta-
       blero
       -1 si el valor no se ha actualizado
  Post: Actualiza el tablero virtual de celdas y lo ordena
*/
```

El **algoritmo** realiza los siguientes pasos.

1. Comprueba si el valor se ha actualizado.
   1.1. Si se ha actualizado y es incremento:
   - Si es incremento en filas → Creamos una nueva fila de celdas (tantas celdas como columnas haya)
   - Si es incremento en columnas → Creamos una nueva columna de celdas (tantas celdas como filas haya)

   1.2. Si se ha actualizado y es decremento:
   - Si es decremento en filas → Eliminamos del vector de celdas del tablero todas las celdas que se encuentren en la fila que se ha decrementado. En el caso de que las celdas estuviesen ocupadas, eliminamos su contenido del tablero.
   - Si es decremento en columnas → Eliminamos del vector de celdas del tablero todas las celdas que se encuentren en la columna que se ha decrementado. Si las celdas estuviesen ocupadas, eliminamos su contenido del tablero.
2. Si no se ha actualizado, vamos a 3.
3. Ordena el vector de celdas del tablero.

▪ LAS CLASES CELDA Y PICTOGRAMA DE BÚSQUEDA

Ambas clases son muy similares y tienen una gran importancia en la aplicación, ya que los tableros de comunicación se componen de ellas. Ambas clases se encargan de almacenar un pictograma y toda la información referente a él: en el caso de los pictogramas resultados de búsquedas, se indicará que procede de una búsqueda. Y en el caso de la celda, que pertenece a una fila y columna determinadas del tablero. Además, estas dos clases poseen todas aquellas funciones necesarias para una buena gestión del pictograma que contienen.

En la Figura D.20 podemos ver las clases ***celda*** y ***PictogramaBusqueda***, con sus métodos y atributos más significativos.

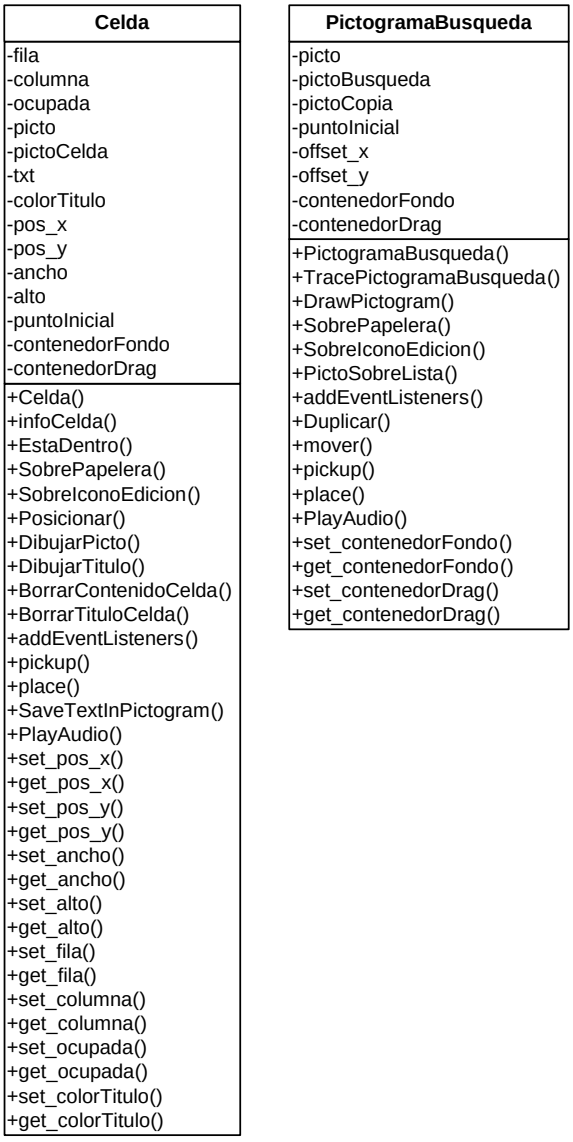


Figura D.20. Clases Celda y PictogramaBúsqueda

El primer grupo de tres funciones se encarga de dibujar y eliminar el contenido de una celda. La cuarta función se encarga de dibujar el pictograma resultado de búsqueda.

```
public function DibujarTitulo():void
public function BorrarContenidoCelda():void
public function BorrarTituloCelda():void

public function DrawPictogram(capa:Sprite, offsetX:int,
                              offsetY:int):void
```

Las siguientes funciones se encargan de comprobar sobre qué zona de la pantalla se encuentra un pictograma:

```
public function EstaDentro (PosX:int, PosY:int):Boolean
public function SobrePapelera (PosX:int, PosY:int):Boolean
public function SobreIconoEdicion (PosX:int, PosY:int):Boolean
```

Las siguientes funciones se encargan de la gestión de los eventos que pueden actuar sobre una Celda/PictogramaBusqueda, como por ejemplo: Hacemos "*Click*" encima de él, lo movemos por la pantalla, lo dejamos de arrastrar, etc.

```
//---------------------------------------------------------------------
  public function addEventListeners():void{
/*---------------------------------------------------------------------
      Pre: True
      Post: Añade los eventListener al pictograma de la celda.
*/
      pictoCelda.addEventListener( MouseEvent.MOUSE_DOWN, pickup );
      pictoCelda.addEventListener( MouseEvent.MOUSE_UP, place );
      pictoCelda.doubleClickEnabled=true;
      pictoCelda.addEventListener(MouseEvent.DOUBLE_CLICK, PlayAudio);
}
```

- **Pickup** es el método encargado de coger y mover el pictograma por las mismas coordenadas en pantalla que el ratón. El comportamiento asociado a este evento es la principal diferencia entre las clases celda y pictograma de búsqueda. En el primer caso, el pictograma sobre el que se hace `MOUSE_DOWN`, es el pictograma que se arrastra por pantalla. En el segundo caso es sobre una copia del pictograma resultado de búsqueda, ya que el pictograma original debe permanecer en la lista de resultados.

- **Place** es el método encargado de colocar adecuadamente el pictograma cuando el ratón se deja de presionar. Pueden darse las siguientes situaciones (Tabla D.2) al dejar de presionar el ratón (evento `MOUSE_UP`):

Situación	Acción
Se suelta el ratón cuando el pictograma está sobre una celda vacía	Se coloca en la celda sobre la que están las coordenadas del ratón
Se suelta el ratón cuando el pictograma está sobre una celda ocupada	El pictograma vuelve a su posición inicial
Se suelta el ratón cuando el pictograma está sobre el icono de edición	El pictograma se carga en la pantalla de edición de pictogramas
Se suelta el ratón cuando el pictograma está sobre la papelera	El pictograma se elimina
Se suelta el ratón cuando el pictograma está sobre cualquier otro lugar del tablero.	El pictograma vuelve a su posición inicial

Tabla D.2. Acciones al dejar de presionar el ratón cuando se arrastra un pictograma

- ***PlayAudio*** es el método encargado de reproducir el audio del pictograma cuando se produce el evento `DOUBLE_CLICK` sobre un pictograma de una celda. En el caso de los pictogramas resultados de búsqueda se procede de distinta manera, ya que al lado de cada pictograma hay un icono que indica si el resultado de búsqueda tiene audio o no.

▪ LA CLASE PICTOGRAMA

La clase pictograma (Figura D.21) es la clase encargada de almacenar los datos que caracterizan a un pictograma. En la siguiente figura se pueden observar los atributos de clase y los métodos de obtención y escritura de cada uno de ellos.

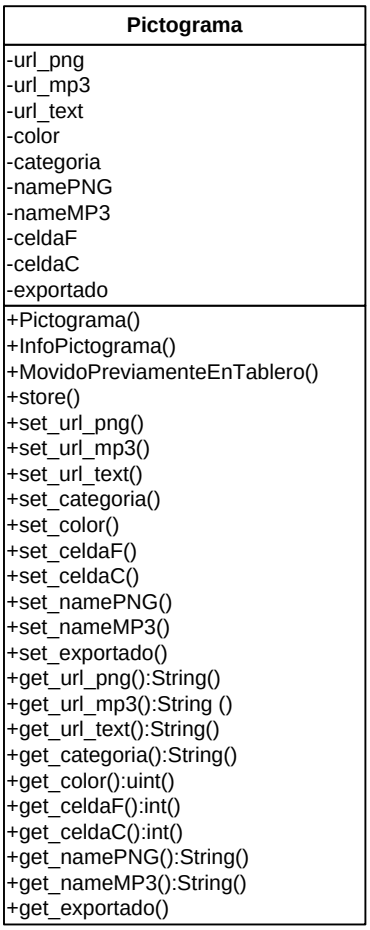


Figura D.21. Clase Pictograma

A su vez, la clase pictograma tiene las funciones que permiten descargar y almacenar en la memoria del ordenador/dispositivo los archivos que forman un pictograma (audio e imagen).

```
public function store(f_raiz:File)
private function StoreOnDisk(url:String,file_name:String,f:File)
```

D.2.3 Diseño de Ventanas

En esta sección se incluyen los diseños de ventanas desarrollados que forman la interfaz de *AraBoard* para PC y de Android. Los prototipos para la versión Android se desarrollaron a partir de estos prototipos de la versión PC, adaptando las opciones de menú, ya que éstas pasarán a ser accedidas vía menú pop-up que parece al presionar la tecla "Menú" del dispositivo.

Los prototipos de esta sección se basan en los prototipos que fueron presentados en las primeras reuniones con los profesores del CPEE Alborada, y modificados a lo largo de éstas.

En primer lugar se muestran las ventanas diseñadas para la aplicación PC de construcción *AraBoard Constructor*, y a continuación la aplicación de visualización/ reproducción, *AraBoard Player* para PC. El diseño de las ventanas de ambas interfaces tiene el mismo estilo. Por último, se mostrarán los prototipos de la versión Android.

Ventanas para AraBoard Constructor PC

Durante el desarrollo de la aplicación de visualización de tableros, *AraBoard Constructor*, se partió de los primeros prototipos de ventanas desarrollados durante las reuniones con los profesores. Los prototipos finales se muestran en las siguientes figuras.

La Figura D.22 se corresponde con la primera pantalla de la aplicación, la cual muestra el logo de la aplicación y un botón para comenzar la construcción de tableros.

Figura D.22. Ventana Inicial de AraBoard Constructor

La Figura D.23 se corresponde con la interfaz de construcción, la cual ofrece la posibilidad de crear un nuevo tablero, cargar un tablero existente, guardar el tablero creado, consultar la información acerca de la aplicación, buscar pictogramas en ARASAAC, y todas aquellas herramientas que permiten personalizar un tablero de comunicación: El tablero en construcción, seleccionador de numero de filas, de columnas, color de fondo, etc.

Figura D.23. Ventana de construcción y edición del tablero

En la Figura D.24 se muestra la ventana de "Guardar Tablero", en ella el usuario puede seleccionar dónde se guarda el tablero (esto sólo en la versión PC, ya que en Android se guarda en una ubicación predeterminada) y el nombre con el que se guardará.

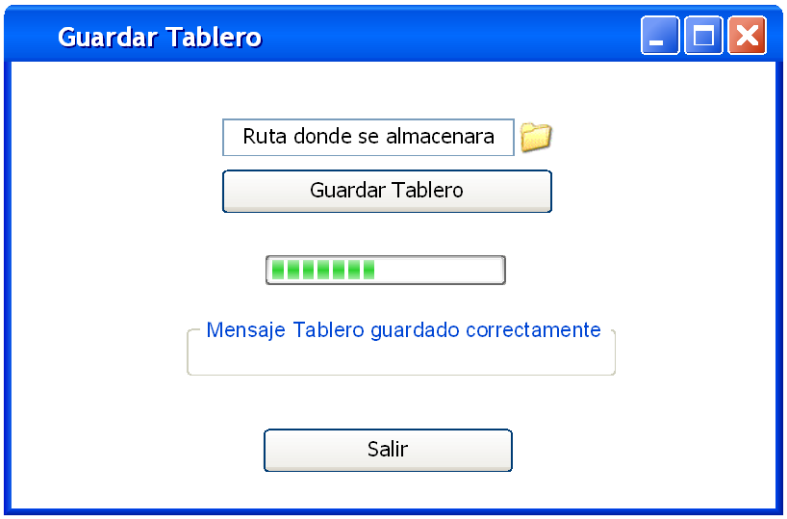


Figura D.24. Ventana Guardar Tablero

En la Figura D.25 se muestra la ventana de "Edición de Pictogramas", en ella el usuario puede editar pictograma existentes sobre el tablero, o crear pictogramas desde cero.

Figura D.25. Ventana de Edición de Pictogramas

Ventanas para AraBoard Player PC

Durante el desarrollo de la aplicación de visualización de tableros, *AraBoard Player*, se partió de la interfaz desarrollada para la aplicación de construcción, únicamente fue necesario modificar la parte central de la ventana principal, en la que se incluyó el tablero. Los prototipos finales se muestran en las siguientes figuras.

La Figura D.26 se corresponde con la primera pantalla de la aplicación, la cual muestra el logo de la aplicación y un botón para comenzar la reproducción de tableros.

Figura D.26. Ventana Inicial de la aplicación

La Figura D.27 se corresponde con la pantalla principal de la aplicación, la cual muestra los pasos a seguir para cargar un tablero, posibilita cargar sucesivos tableros de comunicación, y permite consultar la información acerca de la aplicación. Una vez seleccionado un tablero, éste se carga en la pantalla principal como se muestra en la Figura D.28.

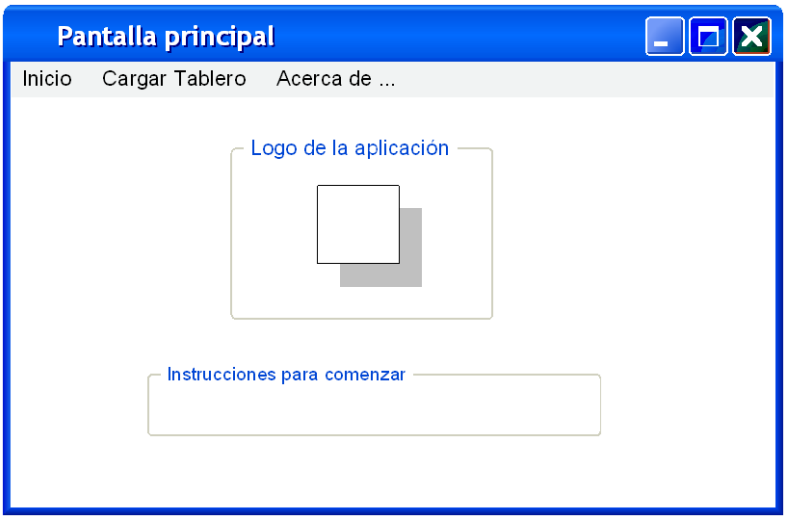


Figura D.27. Ventana principal de la aplicación antes de cargar un tablero

Figura D.28. Ventana principal de la aplicación después de cargar un tablero

Ventanas para AraBoard Constructor Android

La Figura D.29 se corresponde con la pantalla inicial de la aplicación de construcción, la cual ofrece la posibilidad de crear un nuevo tablero o de cargar un tablero existente. La Figura D.30 muestra el listado de tableros a cargar (si se ha presionado anteriormente la opción "Cargar Tablero").

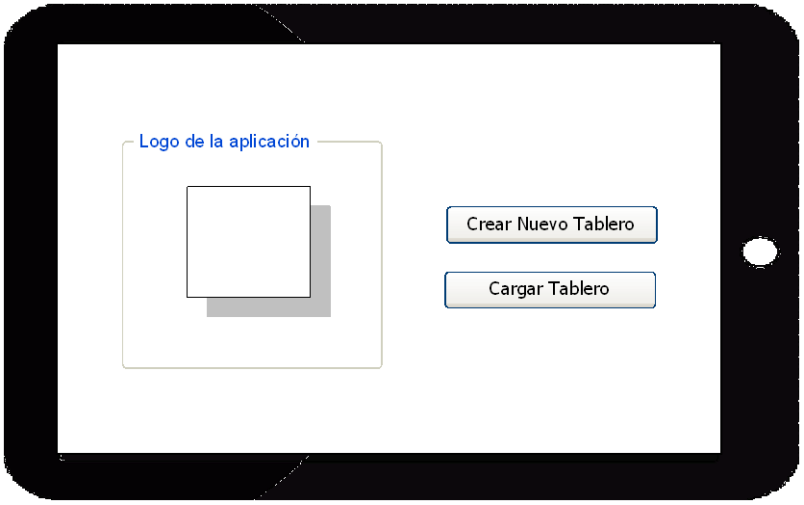


Figura D.29. Ventana Inicial de la aplicación

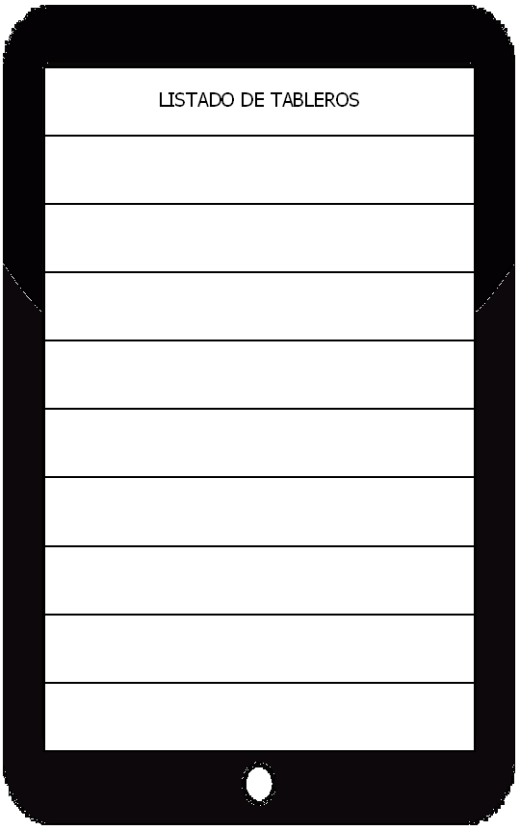


Figura D.30. Ventana de selección de tableros a cargar

A continuación, en la Figura D.31 se encuentra la ventana de construcción y edición de tableros, en la cual el usuario puede personalizarlo de la misma manera que lo hacía en la versión PC.

Figura D.31. Ventana de construcción y edición del tablero

En la Figura D.32 se muestra la ventana de "Edición de Pictogramas", en ella el usuario puede editar pictograma existentes sobre el tablero, o crear pictogramas desde cero.

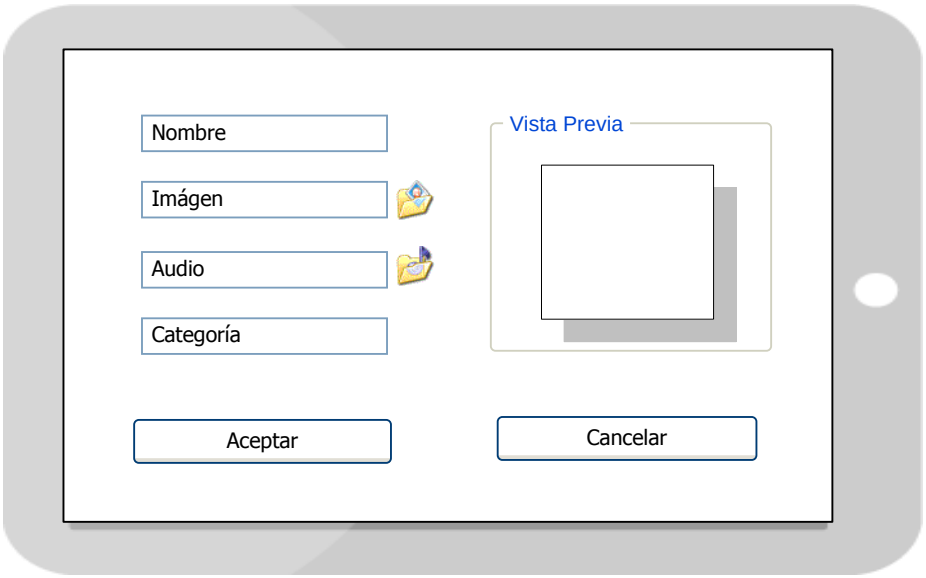


Figura D.32. Ventana de edición de pictogramas

Ventanas para AraBoard Player Android

La Figura D.33 se corresponde con la pantalla inicial de la aplicación de visualización, la cual ofrece la posibilidad cargar un tablero para visualizarlo. La pantalla que muestra el listado de tableros a cargar coincide con la Figura D.30 de la aplicación constructora. Cuando se selecciona uno de los tableros de la lista, el tablero se carga ocupando toda la pantalla como se puede observar en la Figura D.34.

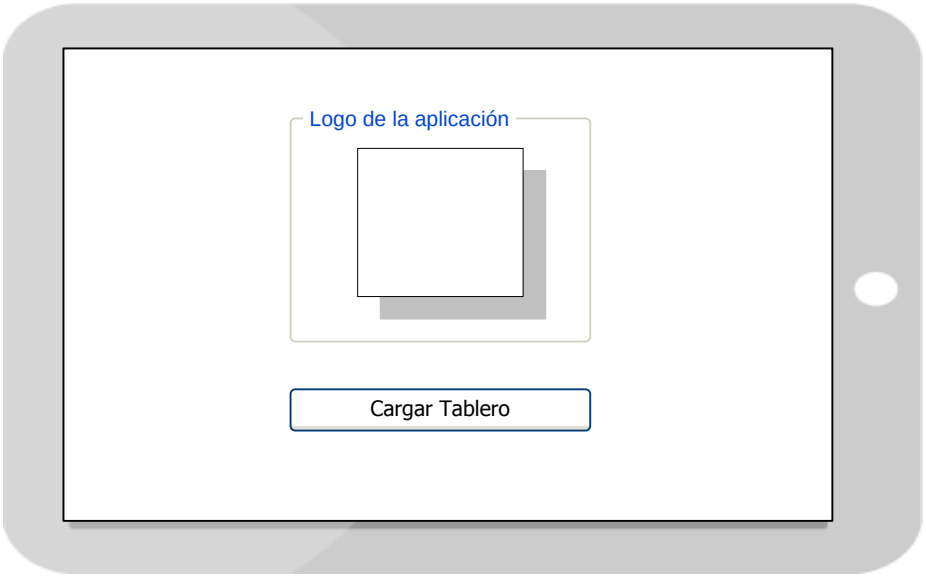


Figura D.33. Ventana Inicial de la aplicación

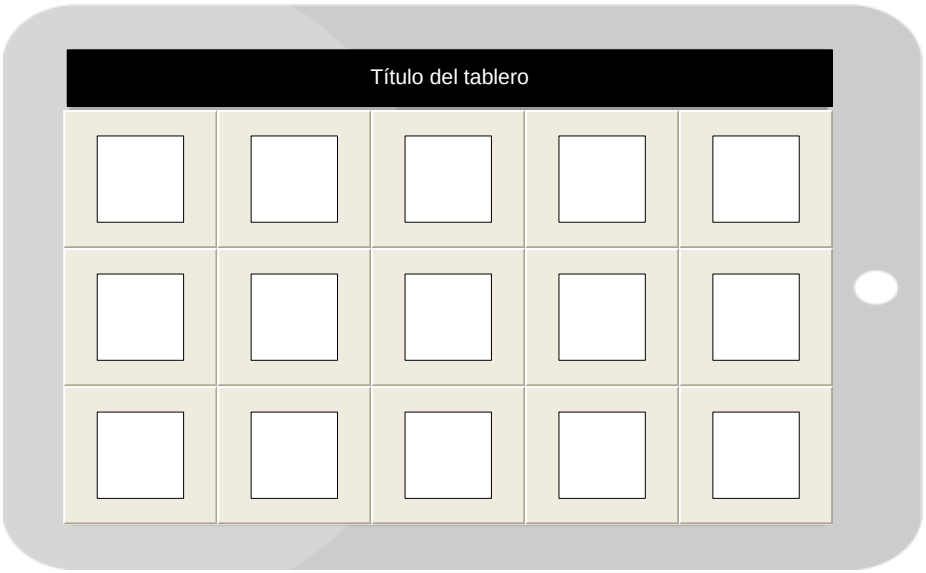


Figura D.34. Ventana con el tablero seleccionado cargado

D.3 Tablero de comunicación AraBoard

Los tableros creados con *AraBoard Constructor* se almacenan en un directorio cuya ubicación y nombre es especificado por el usuario a través de la Interfaz de construcción en el caso de la aplicación para PC, y sobre el directorio sd/data en el caso de la aplicación para Android. En este último caso, el usuario puede especificar el nombre del tablero, pero no puede especificar la ruta del directorio donde se almacenará ya que no es posible la navegación a través de los directorios del sistema.

D.3.1 Estructura de un Tablero AraBoard

Un tablero de comunicación de *AraBoard* se almacena en un directorio creado por la aplicación al guardar un tablero. En el directorio creado se almacena un fichero XML, denominado *tablero_comunicacion.xml*, que especificará la estructura y características del tablero y sus componentes; además, el fichero hará referencia a los ficheros de imagen y audio que formen parte de sus componentes. Estos ficheros se guardarán dentro de dos carpetas ubicadas en el directorio del tablero, en el mismo nivel que el fichero XML, los nombres de estos directorios son: *IMÁGENES* y *AUDIOS*, respectivamente para el directorio que almacena los ficheros de imagen, y para el directorio que almacena los ficheros de audio. La Figura D.35 muestra con un ejemplo la estructura de un *tablero AraBoard*.

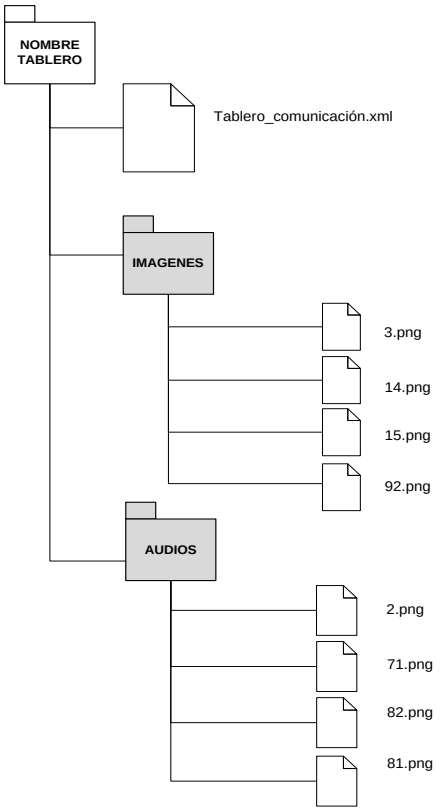


Figura D.35. Ejemplo de estructura de un tablero AraBoard

D.3.2 Estructura del fichero XML del tablero

En esta sección se especifica y explica la estructura del fichero XML que almacena el tablero de comunicación *AraBoard*.

Para una mejor comprensión de la descripción de los tableros almacenados en *tablero_comunicacion.xml*, se definió el formato de los mismos mediante la Descripción de Tipo de Documento (DTD). Su función básica es definir el formato de datos, de manera que sea común y mantenga la consistencia entre todos los documentos que utilicen la misma DTD.

Objeto cabecera

La estructura del nodo cabecera:

```
<cabecera>
   <fondo color=”rrggbb”\>
   <letra color=”rrggbb”\>
   <nombre titulo=“titulo del tablero”\>
   <dimensiones ancho=num alto=num\>
   <componentes filas=num columnas=num\>
</cabecera>
```

Descripción de los atributos del nodo celda:

- **Fondo**: “color” define el color de fondo del tablero, definido en rgb.
- **Letra**: “color” define el color de la letra de las celdas del tablero, definido en rgb.
- **Nombre**: “titulo” indica el titulo del tablero.
- **Componentes:** Indica el número de filas y de columnas del tablero.

Objeto celda

Celdas del tablero: Las celdas del tablero son los elementos que lo conforman, y se encargan de almacenar un pictograma, en el caso de que estén ocupadas. Un tablero de comunicación puede estar formado por una o varias cerdas.

La estructura del nodo celda del fichero XML es la siguiente:

```
<celda>
     <coordenadas fila=num columna=num/>
     <ocupada flag=booleano>
```

```
        <pictograma
        imagen=”carpeta/nombrefichero/”
        audio=”carpeta/nombrefichero”
        texto=”texto asociado al pictograma”
        categoría=”categoría del pictograma”
        color=”rrggbb”/>
    </celda>
```

Descripción de los atributos del nodo celda:

- **Coordenadas**: Indica la fila y columna donde se sitúa la celda en el tablero.
- **Ocupada**: “flag” indica si la celda está ocupada o no por un pictograma.
- **Pictograma:** Existe si ocupada→flag = true.
- **Imagen**: Indica la ruta del archivo PNG del pictograma de la celda.
- **Audio**: Indica la ruta del archivo MP3 del pictograma de la celda.
- **Texto**: Indica el nombre del pictograma de la celda.
- **Categoría**: Indica la categoría del pictograma de la celda.
- **Color**: Indica el color correspondiente a la categoría.

En la Figura D.36 se muestra un ejemplo de *tablero_comunicacion.xml*.

```
<xml version="1.0" encoding="ISO-8859-1">
  <cabecera>
    <fondo color="8475136"/>
    <letra color="1315860"/>
    <nombre titulo="Los animales"/>
    <componentes filas="2" columnas="3"/>
  </cabecera>
  <celda>
    <coordenadas fila="1" columna="1"/>
    <ocupada flag="true"/>
    <pictograma imagen="IMAGENES/2351.png" audio="AUDIOS/0/1189.mp3"
                texto="conejo" categoria="2" color="16737792"/>
  </celda>
  <celda>
    <coordenadas fila="1" columna="2"/>
    <ocupada flag="true"/>
    <pictograma imagen="IMAGENES/2517.png" audio="AUDIOS/0/1227.mp3"
                texto="perro" categoria="2" color="16737792"/>
  </celda>
  <celda>
    <coordenadas fila="1" columna="3"/>
    <ocupada flag="true"/>
    <pictograma imagen="IMAGENES/9881.png" audio="AUDIOS/0/1197.mp3"
                texto="gato" categoria="2" color="16737792"/>
  </celda>
  <celda>
    <coordenadas fila="2" columna="1"/>
    <ocupada flag="true"/>
    <pictograma imagen="IMAGENES/3346.png" audio="AUDIOS/0/1203.mp3"
                texto="hámster" categoria="2" color="16737792"/>
  </celda>
  <celda>
    <coordenadas fila="2" columna="2"/>
    <ocupada flag="true"/>
    <pictograma imagen="IMAGENES/3004.png" audio="AUDIOS/0/5939.mp3"
                texto="canario" categoria="2" color="16737792"/>
  </celda>
  <celda>
    <coordenadas fila="2" columna="3"/>
    <ocupada flag="true"/>
    <pictograma imagen="IMAGENES/2520.png" audio="AUDIOS/0/3081.mp3"
                texto="pez" categoria="2" color="16737792"/>
  </celda>
</xml>
```

Figura D.36. Ejemplo de tablero_comunicacion.xml

D.4 Pruebas realizadas en AraBoard

Las pruebas de un producto *software* son los procesos que permiten verificar su calidad, y se han utilizado para identificar posibles fallos de implementación o de usabilidad. El conjunto de pruebas se han realizado a lo largo de todo el proceso de desarrollo de *AraBoard*. Puesto que el modelo de proceso seguido a lo largo del desarrollo es el modelo incremental [9], descrito en el Anexo C de la memoria, la finalidad de las pruebas ha sido poder solucionar el mayor número de errores en cada etapa, y no ir arrastrándolos con el avance del proyecto.

En esta sección se explican los diferentes tipos de *test* empleados y la finalidad de cada uno de ellos.

D.4.1 Pruebas unitarias

Este tipo de pruebas se utilizan para probar cada módulo de código desarrollado y permiten asegurar que cada uno de ellos funciona correctamente por separado. Una de las principales ventajas de este tipo de pruebas es que simplifican la integración de los módulos, ya que proporcionan un alto nivel de garantía de que el código funciona de la manera esperada.

Estas pruebas se han realizado para cada una de las aplicaciones que componen *AraBoard* por separado. Por un lado, se probaron cada una de las funcionalidades de la aplicación de construcción, de manera independiente a la aplicación de reproducción. Y por otro lado, y de igual forma, se probaron todas y cada una de las funcionalidades de la aplicación de visualización/reproducción.

En primer lugar se elaboraron pruebas para cada una de las principales funcionalidades de la aplicación de construcción: conexión con la API, funcionamiento del buscador de pictogramas, incorporación de celdas al tablero, etc. Y conforme éstas se iban desarrollando, se iban añadiendo y probando junto a las demás funcionalidades ya probadas. En segundo lugar, se procedió de la misma manera para las funcionalidades de la aplicación de reproducción: Carga de tableros, visualización, reproducción, etc.

Ambas pruebas se han realizado para las versiones PC y Android, y han sido testeadas sobre distintas máquinas en función del sistema operativo base. En el caso de la versión PC, las pruebas se realizaron sobre equipos con distintas versiones de sistema operativo Windows: XP, Vista, 7. En el caso de la versión para Android, las pruebas se realizaron en el emulador proporcionado por el Android SDK, debido a que no se tuvo un dispositivo tablet hasta mediados del mes de Mayo.

Las figuras D.37 y D.38 muestran las pruebas realizadas para la funcionalidad "Incorporación de resultados de búsqueda al tablero", de una versión inicial de la aplicación de construcción ejecutada sobre el emulador de Android, y sobre un dispositivo Tablet Motorola.

Figura D.37. Pruebas de una de las primeras versiones de la aplicación de construcción, sobre el Emulador de Android SDK.

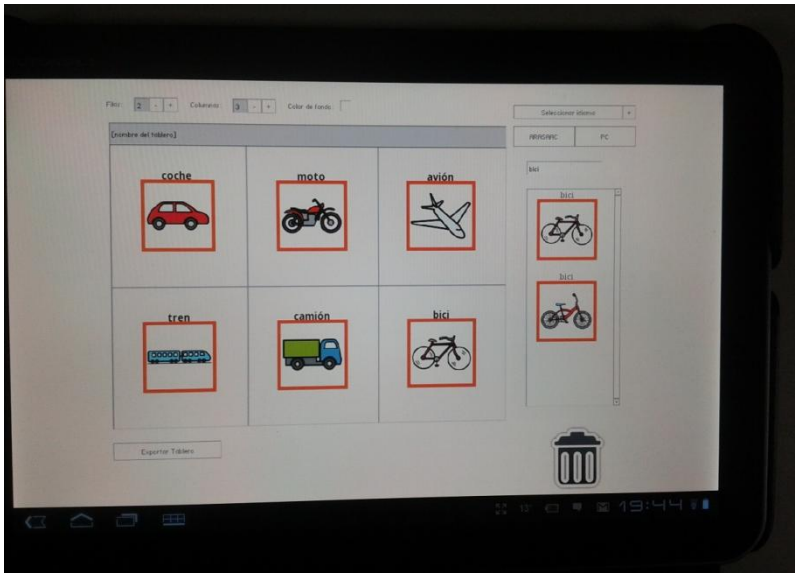


Figura D.38. Pruebas de una de las primeras versiones de la aplicación de construcción, sobre el dispositivo Tablet Motorola

Las figuras D.39 y D.40, muestran una versión posterior (con muchas más funcionalidades incorporadas) de la misma aplicación sobre el emulador de Android y sobre el entorno de ejecución que incorpora el entrono de programación *Adobe AIR for* Android.

Figura D.39. Pruebas de una versión posterior de la aplicación de construcción, sobre el dispositivo Tablet Motorola

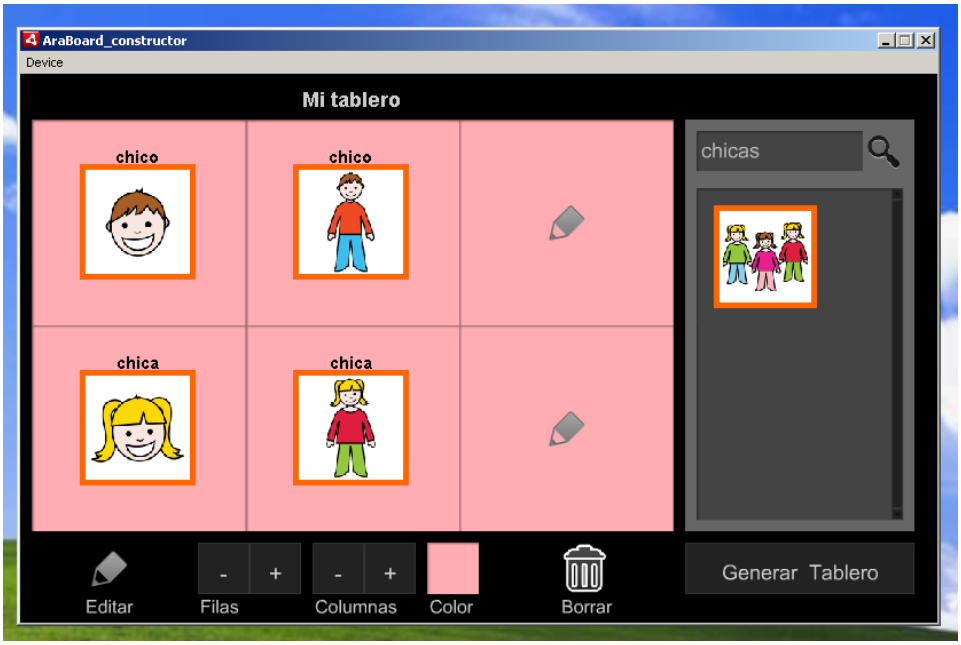


Figura D.40. Pruebas de una versión posterior de la aplicación de construcción, sobre el entorno de ejecución Adobe AIR for Android

D.4.2 Pruebas de integración

Después de realizar las pruebas unitarias de cada uno de los componentes se realizaron las pruebas de integración. Estas pruebas sirven para verificar que las distintas partes del *software* funcionan correctamente cuando se integran entre sí. De esta forma, se prueba que cada parte desarrollada y probada de forma individual se comporta de igual manera al integrarse con las demás funcionalidades.

Este tipo de pruebas se han realizado para verificar, por un lado, la correcta integración de las dos aplicaciones, es decir, que los tableros creados por la aplicación de construcción (*AraBoard Constructor*) pueden ser reproducidos por la aplicación de visualización/reproducción (*AraBoard Player*); y por otro lado, para verificar la compatibilidad de los tableros de comunicación creados con las distintas versiones de la aplicación, es decir, la independencia de los tableros creados respecto a la plataforma sobre la que se han creado.

Para asegurar el buen funcionamiento de la aplicación de forma global, se implementaron un conjunto de tableros de distintas características con la aplicación de construcción para PC y para Android, y se ejecutaron en ambas versiones de la aplicación Player. La siguiente tabla resume las combinaciones de pruebas realizadas.

Tablero reproducido con / **Tablero construido con**	*AraBoard Constructor* Versión Android	*AraBoard Player* Versión Android
AraBoard Constructor Versión PC	✓	✓
AraBoard Player Versión PC	✓	✓

Tabla D.3. Pruebas de integración para verificar la compatibilidad de tableros entre distintas plataformas

D.4.3 Pruebas de sistema

Este tipo de pruebas permite comprobar el funcionamiento y el rendimiento global de la aplicación. El software ya validado con las pruebas de integración, se probó con el resto del sistema para verificar los siguientes aspectos:

- **Rendimiento**: Se han realizado diferentes pruebas para determinar el tiempo de respuesta de determinadas operaciones que ofrece la aplicación. Algunas de ellas han sido las operaciones de búsquedas de pictogramas en las que había varios resultados de búsqueda, otras han consistido en crear y exportar tableros de distintas dimensiones para comprobar el tiempo de espera empleado para almacenar el tablero y descargar los pictogramas que contiene.

- **Resistencia**: Estas pruebas sirven para determinar hasta dónde puede soportar el programa determinadas condiciones extremas. Sobre todo se ha probado la carga

de tableros con recursos propios de mayor tamaño que los pictogramas de ARASAAC, la creación de tableros con las dimensiones máximas establecidas, y con un número elevado de imágenes y pictogramas.

- **Robustez**: Las pruebas realizadas tenían por objetivo determinar la capacidad del programa para soportar entradas incorrectas y asegurar que la aplicación era capaz de responder sin fallos.

D.4.4 Pruebas de usuario

Con este tipo de pruebas se mide el nivel de adaptación del sistema a las necesidades del usuario. Es importante tener en cuenta la facilidad de uso durante todo el desarrollo del proyecto para conseguir unos resultados satisfactorios para la mayoría de los futuros usuarios.

Estas pruebas son esenciales para el tipo de programa desarrollado, debido a los diferentes tipos de usuarios a los que va dirigida. La aplicación de construcción (*AraBoard constructor*) ha sido probada principalmente por los profesores y logopedas del colegio Aborada (como aclaración, decir que los conocimientos informáticos no son los mismos en todos los profesores, sino que varían mucho de unos a otros). Asimismo, la aplicación también ha sido por padres de alumnos y por expertos en Comunicación Alternativa y Aumentativa, todos ellos relacionados con el entorno del CPEE Alborada. Por otro lado, la aplicación de reproducción ha sido probada exclusivamente por los alumnos del colegio Alborada bajo la supervisión de los profesores y/o logopedas.

Algunas de los aspectos evaluados con este tipo de pruebas son los siguientes:

- **Facilidad de uso y aprendizaje:** Se ha buscado en todo momento que la aplicación sea sencilla y manejable por los usuarios, realizando para ello interfaces gráficas intuitivas y similares entre sí. Esto facilita al usuario el aprendizaje de la aplicación y que pueda manejarla sin necesidad de tener unos conocimientos de informática elevados.

- **Fácil de recordar:** Un aspecto importante en el uso de la aplicación es que su manejo sea fácil de recordar aunque haya pasado bastante tiempo sin utilizarla. Para lograrlo se aseguró que la interfaz fuera intuitiva y sencilla.

- **Satisfacción del usuario:** Se determina la calidad de la experiencia de un usuario en su interacción con el programa. Para evaluar este aspecto se han realizado distintas pruebas con profesores y alumnos del colegio CPEE Alborada. La satisfacción en el uso de la aplicación es consecuencia de la facilidad de uso del programa y el cumplimiento de los objetivos previamente establecidos.

D.5 Herramientas utilizadas

A continuación se detallan las principales herramientas y tecnologías utilizadas en la implementación de este proyecto:

Sistemas operativos

- Microsoft WindowsXP, WindowsVista, Windows 7
- Android 2.2 en adelante.

Tecnologías

- XML (eXtensible Markup Language)

Propósito general

- Navegador web: Mozilla Firefox
- Navegador web: Chrome
- Adobe Photoshop CS5.5
- Paint

Análisis y diseño

- Generador de diagramas: ArgoUML 0.24
- Microsoft Office Visio 2007
- Microsoft Office Project 2007

Desarrollo y pruebas

- Entorno de desarrollo: Adobe Flash Professional CS5
- Entorno de desarrollo: Adobe Flash Professional CS5.5
- Entorno de ejecución: Adobe AIR
- Android SDK – Manager
- Android SDK – AVD

Generación de ejecutables e instaladores

- Adobe Flash Profesional CS5.5 con extensión AIR.

Documentación

- Microsoft Office Word 2007 y 2010
- Microsoft Office Visio 2007
- Gantt Project 2.5.1

Dispositivos hardware

- Tablet ASUS Eee Pad Transformer
- Tablet Motorola XOOM 16GB
- Tablet Samsung Galaxy Tab 10.1
- Tablet bq Pascal
- Tablet bq Kepler

Anexo E. Documentación del desarrollo software de Acoti

Este anexo complementa al capítulo 5 con información referente a las distintas fases del desarrollo software de la herramienta *Acoti* de creación y ejecución de actividades didácticas. La realización del software de *Acoti* ha seguido una metodología de trabajo que ha comenzado con una fase de análisis, en la que se ha definido el problema a resolver. Posteriormente se ha continuado con las fases de diseño e implementación. Por último, durante el desarrollo, y tras su finalización, se han realizado distintos tipos de pruebas para garantizar un buen funcionamiento de la aplicación.

E.1 Metodología de análisis

En esta sección se explica la metodología de análisis, en la que se definen los requisitos y los diagramas utilizados, y los pasos seguidos para desarrollar cada modelo de descripción del sistema. La metodología de análisis seleccionada es OMT [10]. Según esta metodología la fase de análisis se realiza en tres pasos:

- **Modelo de objetos:** Describe la estructura estática de los objetos del sistema (relaciones, atributos y operaciones), el cual se representa mediante diagramas de objetos.
- **Modelo dinámico:** Describe los aspectos de un sistema estudiando la organización de estados y la secuencia de operaciones mediante diagramas de estado.
- **Modelo funcional:** Describe las transformaciones que pueden sufrir los datos dentro del sistema, representado gráficamente mediante diagramas de flujo de datos.

Se ha creído oportuno añadir a estos tres pasos la fase de análisis de requisitos, que es propio de la metodología UML pero que en nuestro caso permite estudiar más a fondo las necesidades de la aplicación.

E.1.1 Análisis de requisitos

La primera etapa del desarrollo consistió en establecer los requisitos que debía cumplir la aplicación así conseguir una mayor comprensión del problema a resolver. Para ello, antes de comenzar el desarrollo, se realizaron reuniones previamente con los directores del proyecto. Posteriormente se realizaron reuniones en el colegio en las que también estaban los profesores del CPEE Alborada. En estas reuniones los profesores proporcionaron ideas de los tipos de juegos que tendrían que poder ser definidos con *Acoti*, y se establecieron los objetivos y funcionalidades que debía cubrir la aplicación.

A continuación se listan los requisitos de forma conjunta, tal y como quedaron en su versión final, agrupados en requisitos funcionales y no funcionales. Los requisitos funcionales definen el comportamiento del software, así como las características de sus funcionalidades. Los requisitos no funcionales describen las restricciones tanto software como hardware de la aplicación y las facilidades que debe proporcionar.

Requisitos Funcionales

RF- 1: El sistema permitirá a los profesores configurar actividades de abstracción de cualidades de objetos.

RF- 2: El sistema permitirá definir cualquier actividad que consista en definir una o varias áreas sobre la mesa en una determinada posición, y un conjunto de objetos correctos asociados a cada área.

RF- 3: El sistema deberá permitir definir secuencias de tareas.

RF- 4: El sistema permitirá definir tantas áreas como el usuario prefiera.

RF- 5: El sistema permitirá definir un área de juego que ocupe toda la mesa.

RF- 6: El sistema deberá facilitar la inclusión de recursos propios: imágenes, pictogramas y sonidos.

RF- 7: El sistema deberá resolver los problemas de re-escalado asociados a la inclusión de imágenes.

RF- 8: El sistema permitirá asociar un sonido a un área de juego. Este sonido podrá repetirse cada equis segundos, o bien podrá escucharse cuando se produzca un evento sobre la mesa.

RF- 9: El sistema proporcionará *feedback* al alumno, tanto gráficamente como mediante audios, según la acción que el alumno realice sobre la mesa.

RF- 10: El sistema debe identificar tres tipos de acciones sobre la mesa:

- Poner un juguete
- Mover un juguete
- Quitar un juguete

Requisitos No Funcionales

RNF- 1: La interfaz del sistema estará adaptada a las características de la mesa tangible o *tabletop* NIKVISION

RNF- 2: La aplicación será implementada en lenguaje en el *ActionScript* 3.0, bajo el entorno de ejecución Adobe Air.

RNF- 3: La aplicación empleará funciones de la Interfaz *TuioListener* para detectar cuando un objeto se pone, se mueve o se quita de la mesa.

RNF- 4: Para la gestión y almacenamiento de las actividades se empleará la tecnología XML. El fichero XML referenciará a los ficheros de audio y de imagen necesarios para la actividad.

RNF- 5: Junto al fichero XML se guardarán las carpetas que almacenarán los ficheros de imágenes y los ficheros de audio.

RNF- 6: Los formatos de imagen que serán soportados por la aplicación son PNG y JPEG.

RNF- 7: El formato de audio que serán soportados por la aplicación es MP3.

E.1.2 Modelo de objetos

Como se ha explicado anteriormente el modelo de objetos define la estructura estática de los objetos del sistema y proporciona el entorno en el que situar el modelo dinámico y funcional. En esta sección se muestra el diagrama de clases que componen la aplicación y el diccionario de datos. Primero se hace un estudio de las principales clases mostrando las relaciones que existen entre todas ellas, y una vez construido el esquema de clases, se elabora el diccionario de datos.

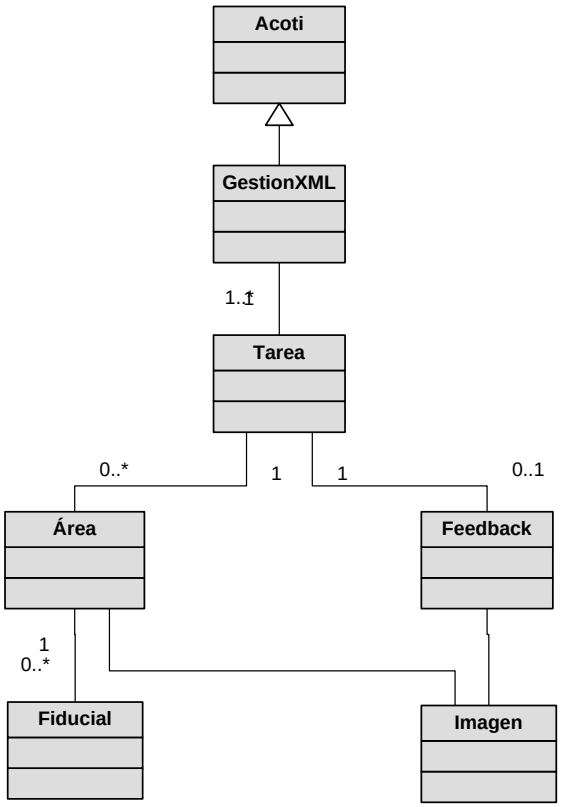


Figura E.1 Diagrama de clases de la aplicación Acoti

E.1.3 Diccionario de Datos

Como se ha explicado anteriormente, el diccionario de datos contiene la descripción de las clases extraídas en el diagrama anterior, haciendo referencia al alcance de cada una de ellas dentro del problema y cualquier suposición o restricción relativa a su uso.

AplicaciónAcoti	Sistema que carga actividades y las ejecuta sobre la mesa.
InterfazAcoti	Diálogo para la comunicación persona/ordenador en el que se solicita al usuario que cargue un fichero XML que almacena la información referente a la actividad.
TratarXML	Módulo que tomando las características de una actividad se encarga de construir la lista de tareas a completar, junto con sus respectivas áreas, *fiduciales*, y *feedback*.
Tarea	Define un conjunto de áreas de juego junto con un *feedback* que proporcionará al usuario información acerca del transcurso de la actividad.
Feedback	Define un tipo de *feedback* que se asociará a una tarea de la actividad. Un objeto de tipo *feedback* se construirá mediante unas dimensiones y coordenadas en pantalla. Además llevará asociadas unas urls correspondientes a una imagen de acierto, una de fallo, una neutra, y una de actividad completada; a su vez, también tendrá asociado un sonido de acierto, de fallo y de actividad completada.
Área	Define un área de la actividad mediante una url de una imagen, una url de un audio, un texto, una categoría y un color.
Fiducial	Define un *fiducial* mediante un identificador, una url de un audio y un flag que indica si es un *fiducial* correcto o no en el área en la que estará incluido.

Tabla E.1. Diccionario de datos de Acoti

7.3.1...E.1.4 Modelo dinámico

El modelo dinámico describe las operaciones y el orden en que se llevan a cabo. Para ello se cuenta con los diagramas de casos de uso.

Diagrama de casos de uso

Los diagramas de casos de uso se crean para representar las operaciones que realiza cada usuario dentro del sistema. Los posibles roles que pueden tener los usuarios en *Acoti* son:

- **Profesor:** Es el usuario que se encarga de crear las actividades pedagógicas.
- **Alumno:** Es el usuario que juega a la actividad creada por el profesor.
- **Tabletop NIKVision - TUIO:** Es la superficie donde se ejecutará *Acoti*. Se cuenta como actor ya interactúa con *Acoti* enviando información a la aplicación cuando un usuario pone/quita/mueve un objeto sobre la mesa.

La Figura E.2 muestra el diagrama de casos de uso que se ha modelado para la aplicación *Acoti*.

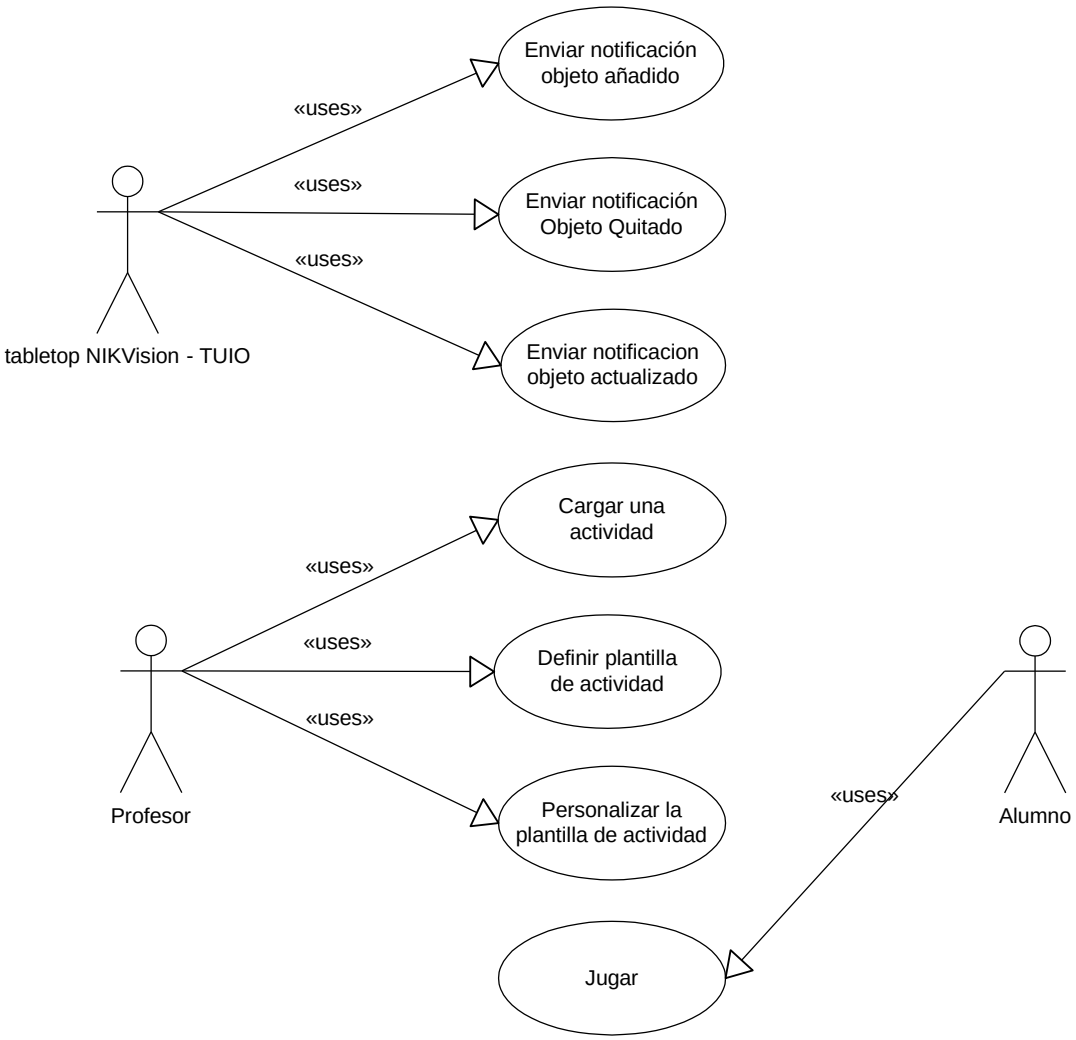


Figura E.2 Casos de uso de Acoti

E.1.5 Modelo funcional

El modelo funcional se emplea para especificar el significado de las operaciones en el modelo de objetos y las acciones o actividades en el modelo dinámico. Para representar estas actividades se han utilizado los Diagramas de Flujos de Datos, en adelante DFD. Los DFD son grafos que están compuestos de arcos y nodos, donde los arcos pueden ser valores de entrada o salida; o flujos de datos (valores intermedios). Los nodos representan actores, que son los que producen o consumen datos; procesos, que son los que transforman los datos; o almacenes de datos, elementos pasivos que guardan datos.

En esta sección se especifica el funcionamiento global del sistema para la operación de carga de una actividad. En la Figura E.3 se muestra el diagrama de flujo de datos de nivel 0. En este nivel, la aplicación interactúa con el usuario que está cargando la actividad y con el fichero XML y sus recursos asociados, ambos almacenados en la máquina del usuario. El usuario selecciona, a través de la interfaz de la pantalla de selección de ficheros de Windows, un fichero de actividad. La aplicación se encarga de tratar el fichero XML y gestiona los elementos en él definidos y su funcionamiento, para que el usuario final (alumno) pueda interactuar con la actividad.

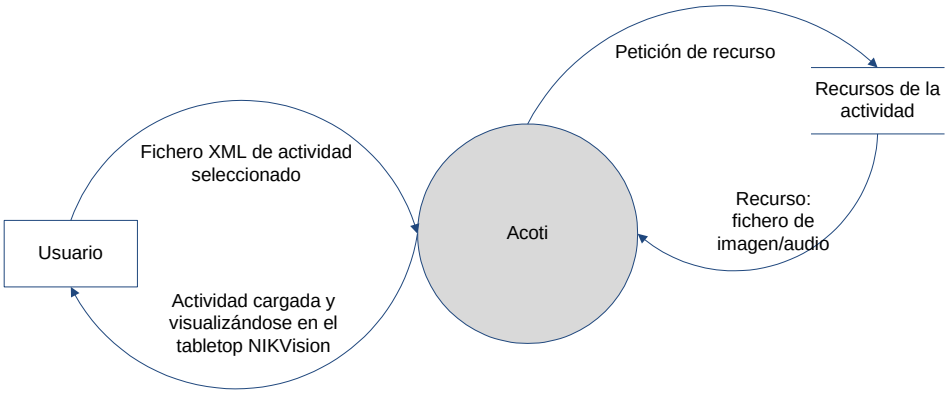


Figura E.3 DFD de nivel 0 para el de carga de una actividad

En la Figura E.4 se muestra el DFD de nivel 1 en el que se ve el funcionamiento del sistema en detalle.

En el DFD de nivel 1 se observa en primer lugar el proceso 1, "*TratarXML*", que es el encargado de leer los elementos definidos en el fichero XML y de crear la actividad en pantalla. En primer lugar, el módulo lee los datos almacenados en el fichero y gestiona adecuadamente las tareas que van apareciendo por pantalla según las acciones que el usuario vaya realizando. Para cargar cada uno de los elementos que forman una actividad, el proceso 1 envía los datos leídos en el fichero XML de actividad a los demás procesos. Envía al proceso 2, "*Área*", los datos leídos del nodo Área del fichero XML, y este proceso se encarga de construir cada una de ellas y de mostrarlas por pantalla, además se encargará de construir la lista de *fiduciales* que tiene asociada cada una d. De la misma manera, el proceso 1 envía al proceso 4, "*Feedback*", los datos para construir el *feedback* de la actividad.

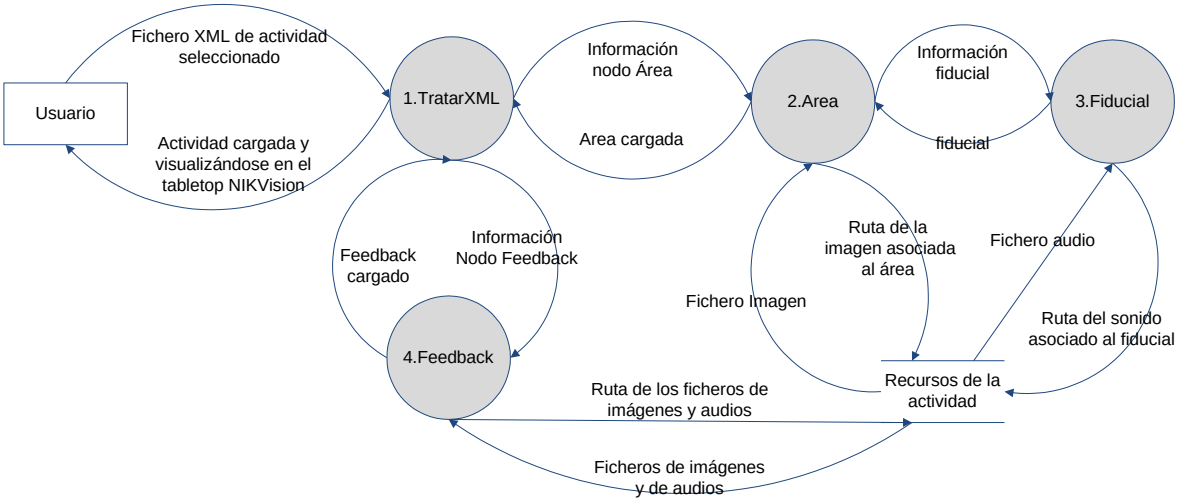


Figura E.4 DFD de nivel 1 para el de carga de una actividad

La Figura E.5 muestra el Diagrama de Flujo de Datos de nivel 2. El proceso 1.1 "*Gestión de Tareas*" se encarga de recorrer las tareas definidas en el fichero XML y enviar a los procesos 1.2, "*Tratar Áreas*", y 1.3, "*Tratar Feedback*", los nodos de cada tarea para que extraigan las áreas y los *feedback* respectivamente. Una vez extraída toda la información, los procesos 1.2 y 1.3 los comunican al proceso "*Gestión de tareas*", quien se encarga de mostrar los distintos elementos al usuario y los gestiona en función de las acciones que el usuario realice sobre la mesa. Cuando el usuario supere la tarea, repetirá el proceso cargando las sucesivas tareas hasta que termine el fichero XML.

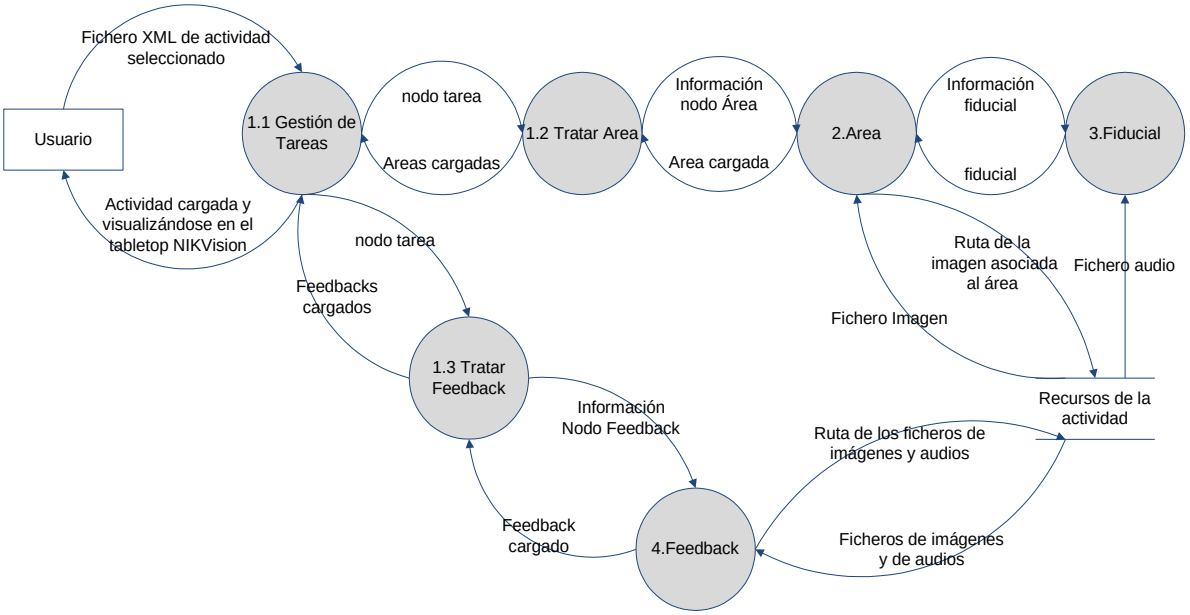


Figura E.5 DFD de nivel 2 para el de carga de una actividad

E.2 Metodología de diseño e implementación

La etapa de diseño dentro de la metodología OMT [10]. Consta de dos fases principales.

- **La división en subsistemas,** para dividir la aplicación en componentes o módulos más pequeños**, y**
- **El diseño de objetos,** donde se muestran en detalle las clases que componen cada subsistema.

En esta sección se exponen en primer lugar los aspectos más relevantes del diseño, y en algunos casos se ofrece una perspectiva cercana a la implementación. En primer lugar se detallan los distintos subsistemas tanto internos como externos que conforman la aplicación.

E.2.1 Subsistemas de Acoti

El primer paso en el diseño consiste en dividir la aplicación a desarrollar en subsistemas agrupando aquellas clases o funciones con comportamiento similar.

En el sistema *Acoti* se observan diferentes módulos, los cuales se muestran en la Figura E.6. En el sistema se observan tres niveles diferentes:

- **Nivel inferior:** En este nivel se encuentran las actividades pedagógicas definidas bajo la estructura del fichero XML que se define posteriormente en la sección B.3.2. Las actividades de esta capa habrán sido creados por el profesor o tutor, y podrán ser cargados desde la aplicación para ejecutar las secuencias de tareas en ellos definidas.
- **Nivel medio:** Este nivel gestiona todos los datos que forman las tareas de una actividad. La "inteligencia" del sistema se encuentra en este nivel puesto que en él se implementan todas las funcionalidades de la aplicación, desde cargar el fondo, las áreas, los *feedback,* y todos los componentes que forman una tarea, hasta interactuar con las funciones *TuioListener* que permiten detectar interacciones sobre el *tabletop*.
- **Nivel superior:** En este nivel se encuentra la clase *TuioClient*, que es el centro del protocolo TUIO [18] de detección de objetos sobre el *tabletop* NIKVision. La instancia de la clase *TuioClient* genera eventos TUIO, que se transmiten a todas las clases que implementan la interfaz *TuioListener*, en nuestro caso, a la clase *Acoti* del nivel medio. *TuioListener* se encarga de esperar a que ocurran eventos sobre la mesa y transmitirlos a la clase *Acoti*, para que los gestione adecuadamente.

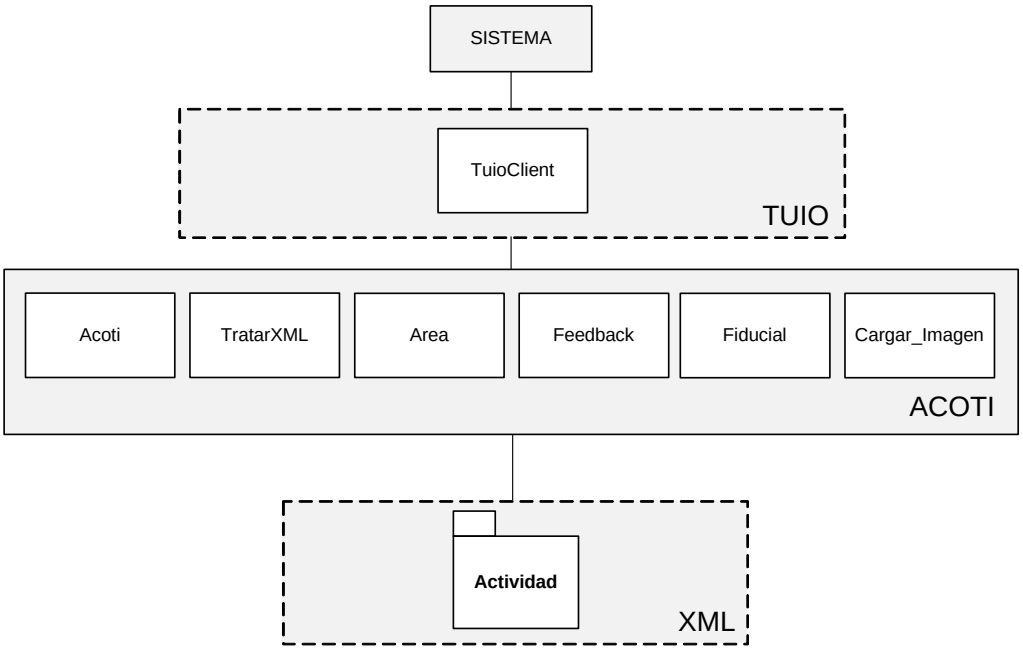


Figura E.6 Subsistemas de Acoti

Subsistemas internos para Acoti

Los subsistemas internos propios de la aplicación se dividen en los siguientes módulos:

- *Acoti*: Paquete principal de la aplicación que contiene todos los subsistemas.
- *Acoti.Acoti*: Módulo principal del paquete *Acoti*. Este módulo se encarga de gestionar y establecer los comportamientos asociados a los distintos eventos que se producen sobre la mesa: poner/quitar/mover objetos. La clase *Acoti* incorpora una instancia de la clase *TuioClient*, encargada de generar eventos TUIO según las acciones que se produzcan sobre la mesa. El comportamiento asociado a cada acción podrá ser programado mediante llamadas a las funciones de la interfaz *TuioListener*.
- *Acoti.TratarXML*: Módulo correspondiente al subsistema *TratarXML*. Se encarga de leer, gestionar y cargar los elementos que constituyen una actividad: tareas, fondos, áreas, *feedbacks* y *fiduciales*, que se encuentran definidos en los nodos del fichero XML.
- *Acoti.Área*: Módulo correspondiente al subsistema *Área*. Se encarga de gestionar la información almacenada en un nodo área. Carga el área en pantalla y gestiona su funcionamiento asociado.
- *Acoti.Feedback*: Módulo correspondiente al subsistema *Feedback*. Se encarga de gestionar la información almacenada en un nodo *feedback*. Carga el *feedback* en pantalla y gestiona su funcionamiento asociado.
- *Acoti.Fiducial*: Módulo correspondiente al subsistema *Fiducial*. Se encarga de la obtención de la información referente a los *fiduciales* definidos dentro de un área.
- *Acoti.Cargar_Imagen*: Módulo correspondiente al subsistema *Cargar_Imagen*. Se encarga del manejo y la carga de las imágenes definidas en las áreas y los *feedbacks*.

Subsistemas externos para Acoti

Las bibliotecas externas y que se requieren para el funcionamiento de la aplicación son:

- ***TUIO AS3 library*:** La librería TUIO [39] para entornos *ActionScript 3.0* (Flash Project en la Figura E.7) ha sido diseñada principalmente como modo de abstracción entre las superficies interactivas y el entorno de programación empleado.

- ***TuioClient*:** La clase *TuioClient* [40] proporciona una infraestructura de devolución de llamada sencilla, utilizando la interfaz *TuioListener*. Con el fin de recibir y decodificar los mensajes TUIO, se debe crear una instancia de la clase *TuioClient*. Esta instancia genera eventos TUIO que se transmitirán a todas las clases registradas que implementen la interfaz *TuioListener*.

- ***Interfaz TuioListener:*** La interfaz de *TuioListener* [40] proporciona una infraestructura de devolución de llamada simple, que es utilizada por *TuioClient* para distribuir eventos TUIO a todas las instancias registradas de clases que implementan la interfaz *TuioListener*. Cualquier clase que implemente la interfaz *TuioListener* puede emplear sus funciones y modificar su comportamiento.

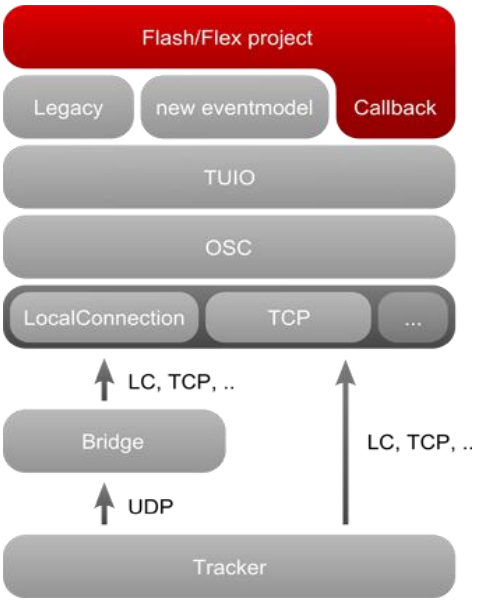


Figura E.7 Comunicación del proyecto Flash con las librerías externas del protocoloTUIO

E.2.2 Diseño de Objetos

El Diseño de Objetos muestra las entidades u objetos principales en la aplicación. A lo largo de esta sección se diseñarán e implementarán los objetos identificados en fases anteriores.

Diagrama Entidad-Relación y Modelo Relacional

A continuación se muestran el diagrama Entidad/Relación (Figura E.8) y el modelo relacional (Figura E.9) que reflejan los datos y las relaciones que establecen. El diagrama E/R es un modelo de datos conceptual de alto nivel que describe los tipos de datos, sus relaciones y restricciones. Mientras que el modelo relacional representa el diseño de los tipos de entidades y muestra las relaciones haciendo uso de las claves primarias y las claves ajenas de cada entidad.

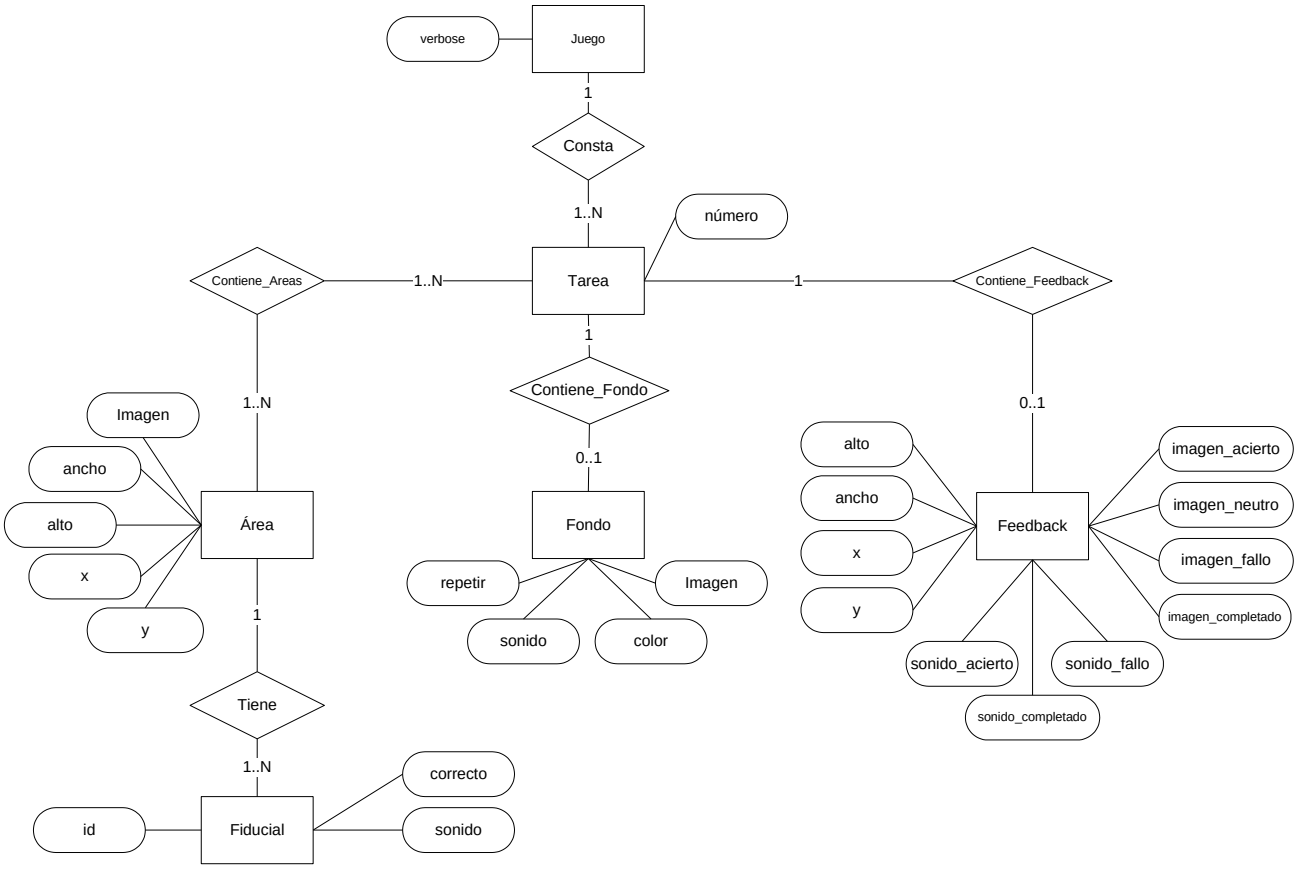


Figura E.8 Diagrama Entidad Relación

Juego (verbose)

Tarea (número)

Fondo (color, imagen, sonido, repetir, número) {Entidad débil respecto a tarea}

Feedback (x, y, ancho, alto, imagenácierto, imagen_neutro, imagen_fallo,
imagen_completado, sonido_correcto, sonido_fallo,
sonido_completado, número) {Entidad débil respecto a tarea}

Área (x, y, ancho, alto, imagen)

Fiducial (id, correcto, sonido)

Contieneáreas (número, x, y)

Tiene (id, x, y)

RESTRICCIONES:

- número es clave ajena de tarea. *NO NULO.*
- (x, y) es clave ajena de *Área. NO NULO.*
- (id) es clave ajena de *Fiducial. NO NULO.*

- La clave de *Feedback* es número (clave primaria de *Tarea*), ya que *Feedback* es entidad débil de Tarea. De este modo queda definida la relación "Contiene_feedback".
- La clave de *Fondo* es número (clave primaria de *Tarea*), ya *Fondo* que es entidad débil de *Tarea*. De este modo queda definida la relación "Contiene_fondo".

Figura E.9 Modelo Relacional

Diseño de la aplicación

Durante las fases de análisis y de diseño se generaron distintos diagramas que facilitarían la fase posterior de implementación.

A continuación se muestran todas las clases y las relaciones que existen entre ellas (Figura E.10). Para simplificar el diagrama, se han excluido los métodos propios de algunas de las clases que se detallarán más adelante.

En los puntos que se muestran a continuación se describen y explican las clases y métodos más significativos de la aplicación de construcción desde perspectiva cercana a la implementación.

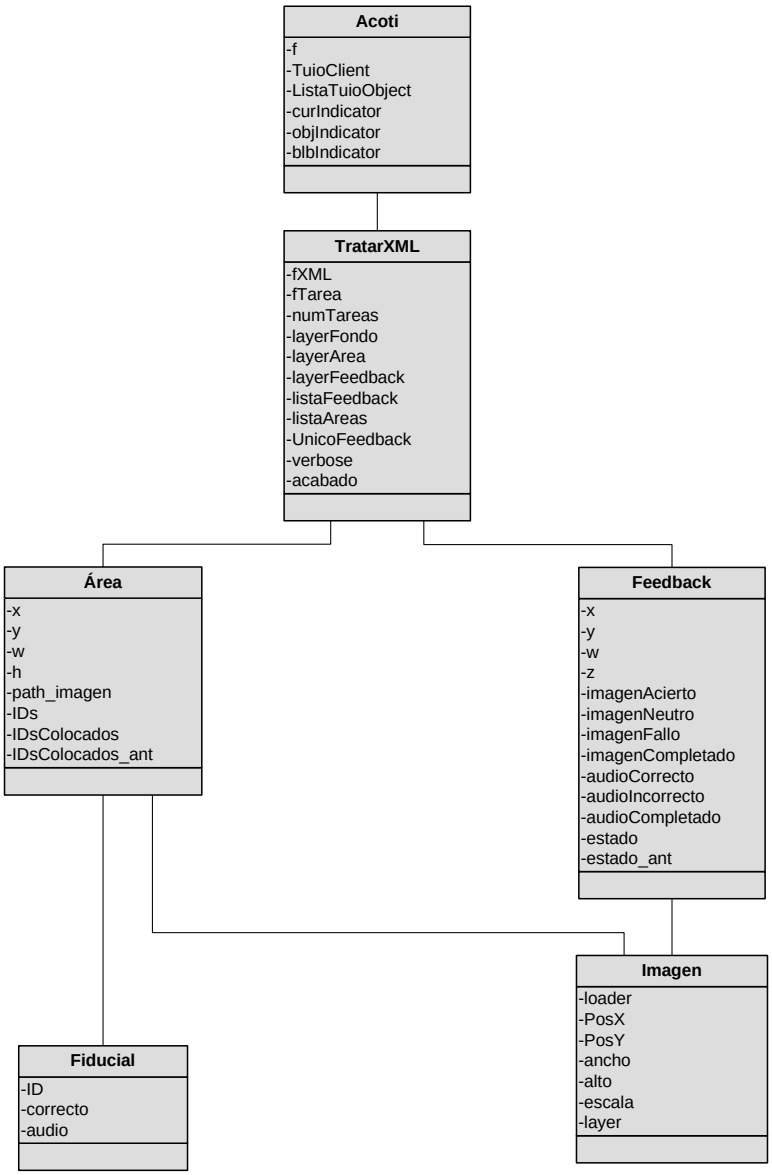


Figura E.10 Diagrama de clases

▪ MÓDULO PRINCIPAL DE ACOTI

La Clase *Acoti* es la clase principal de la aplicación, ya que es la encargada de solicitar la carga de un fichero de actividad al usuario, y establecer la conexión con las demás clases para cargar y ejecutar la actividad definida en el fichero cargado.

Además, esta clase es la encargada de comunicarse con el *tabletop* (o mesa) NIKVision a través de la Interfaz *TuioListener*, que le permite detectar cuando un juguete/objeto ha sido añadido/eliminado/modificado en la superficie de la mesa, y realizar el comportamiento correspondiente definido en la actividad.

En la Figura E.11 se muestra la clase *Acoti* junto sus atributos y métodos más significativos.

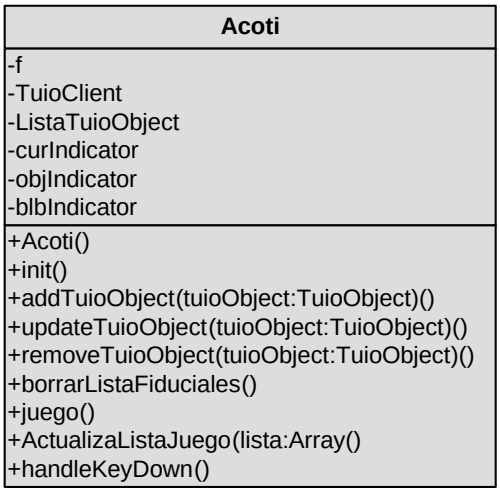


Figura E.11 Clase Acoti

A continuación se agrupan los métodos según la funcionalidad que desempeñan dentro de la clase y se explica su funcionamiento.

- **Método constructor:**

```
public function Acoti();
```

El cual, inmediatamente llama a:

```
private function init(e:Event = null):void
```

- **Método de la Interfaz TuioListener:**

```
public function addTuioObject(tuioObject:TuioObject):void
public function updateTuioObject(tuioObject:TuioObject):void
public function removeTuioObject(tuioObject:TuioObject):void
```

- **Métodos para gestionar el juego:**

```
public function juego()
private function borrarListaFiduciales()
public function actualizaListaJuego(lista:Array)
```

- **Métodos de gestión de eventos:**

```
handleKeyDown(event:KeyboardEvent)
```

En primer lugar, se muestra el método constructor de la aplicación que inmediatamente llama a la función *Init*, que inicializa la aplicación. La función *Init* se encarga de inicializar el *TuioClient* (el *tabletop* NIKVision), y de conectar con la clase que se encarga de tratar el fichero XML para obtener todos los elementos y comportamientos de la actividad definida.

El segundo grupo incluye aquellos métodos que se encargan de programar el comportamiento asociado a los eventos de poner/quitar/modificar un objeto sobre la mesa. Estas funciones pertenecen a la interfaz *TuioListener*, que nos facilita la detección de interacciones sobre la mesa a través de la clase *TuioObject*.

La primera función de este grupo se ejecuta cuando un objeto se añade a la mesa. Esta función añade el objeto que se ha puesto sobre la mesa a la lista de objetos *ListaTuioObject*, que almacena el listado de objetos sobre la superficie de la mesa (*tabletop)*.

La segunda función del grupo se ejecuta cuando las posiciones de un objeto sobre la mesa se han visto modificadas, debido a que el usuario ha realizado un movimiento con alguno de los objeto sobre la mesa. Esta función detecta el objeto modificado, y actualiza sus coordenadas en pantalla:

```
if (item.name==tuioObject.sessionID.toString()) {
    // Actualizamos las coordenadas en pantalla del objeto sobre la mesa
    item.actualiza(coord.convertir_X(tuioObject.x,tuioObject.y),
    coord.convertir_Y(tuioObject.x,tuioObject.y),
    coord.convertir_a(tuioObject.a));
}
```

Y la tercera se ejecuta cuando un objeto se quita de la mesa. Los pasos que realiza está función son los siguientes:

1. Recorre la lista *ListaTuioObject* para encontrar el objeto que se ha quitado.
2. Comprueba si el objeto i-ésimo está en la lista.
3. Si no lo encuentra, lo borra.

El tercer grupo agrupa aquellas funciones que permiten gestionar el juego en función de la interacción del usuario con la mesa. La descripción de cada función es la siguiente:

- **`juego()`:** Se encarga de gestionar la actividad y todos los elementos que la componen. Es la función que permite ejecutar la actividad definida en el fichero. Recorre de forma secuencial las tareas e informa al usuario del éxito o fracaso de sus acciones y de cuando cada tarea es completada.

- **`borrarListaFiduciales()`:** Elimina todos los elementos de la lista "ListaTuioObject", es decir, todos los *fiduciales* de los objetos que estén sobre la mesa.

- **`actualizaListaJuego(lista:Array)`:** Recorre las áreas para ver que juguetes hay colocados en cada una de ellas, y actualiza la lista de juego.

▪ MÓDULO DE TRATAMIENTO DEL XML

La Clase *TratarXML* es una de las clases más importantes de la aplicación, ya que es la encargada de solicitar el fichero que define la actividad al usuario, y leer y tratar todos los elementos en él definidos.

En la Figura E.12 se muestra la clase *TratarXML* junto sus atributos y métodos más significativos.

Figura E.12 Clase TratarXML

A continuación se agrupan los métodos según la funcionalidad que desempeñan dentro de la clase y se explica su funcionamiento.

- **Método constructor:**

```
public function TratarXML(capa_fondo: Sprite, capa_area: Sprite,
                          capa_feedback: Sprite)
```

El constructor de la clase *TratarXML* llama inmediatamente a la función *loadXML*, que se encarga de cargar de posibilitar al usuario que cargue un fichero XML de actividad.

- **Métodos de solicitud y carga del fichero al usuario:**

```
private function selectHandler(event:Event):void
private function loadXML(cadena:String) :void
private function loadedCompleteHandler(e:Event):void
```

Estas funciones permiten al usuario cargar la actividad o juego en la aplicación *Acoti*. La primera de ellas, `selectHandler`, se encarga de obtener el nombre y la ruta del fichero XML seleccionado por el usuario. La segunda, `loadXML`, crea un nuevo objeto *loader* para cargar el XML y le añade un *listener* para saber cuándo se ha cargado por completo el fichero XML. Una vez cargado, se ejecuta `loadedCompleteHandler`. Esta función se encarga de eliminar el *listener*, y recorrer el fichero secuencialmente para extraer sus tareas y poder tratar los 3 elementos distintos que puede contener cada una: *Fondo*, *Área* y *Feedback*.

- **Métodos de tratamiento de los elementos del fichero:**

  - **`public function TratarFondo(f:XML):void`** → Se encarga de obtener y tratar el nodo *fondo* definido en una tarea (`nodo f` que se le pasa como parámetro) y de cargar todos sus componentes y su comportamiento.

  - **`public function TratarArea(f:XML):void`** → Se encarga de obtener y tratar el nodo *área* definido en una tarea (`nodo f` que se le pasa como parámetro) y de cargar todos sus componentes y su comportamiento.

  - **`public function TratarFeedback(f:XML):void`** → Se encarga de obtener y tratar el nodo *feedback* definido en una tarea (`nodo f` que se le pasa como parámetro) y de cargar todos sus componentes y su comportamiento.

▪ CLASES PARA LAS ÁREAS Y FIDUCIALES

La clase *Área* es la clase que se encarga de gestionar la información asociada a un área, y de dibujar cada una de las áreas que componen una tarea. Mientras que la clase *Fiducial* es la clase que se encarga de gestionar la información asociada a los *fiduciales* asociados a cada área.

En la Figura E.13 se muestran las clases *Área y Fiducial* junto sus atributos y métodos más significativos.

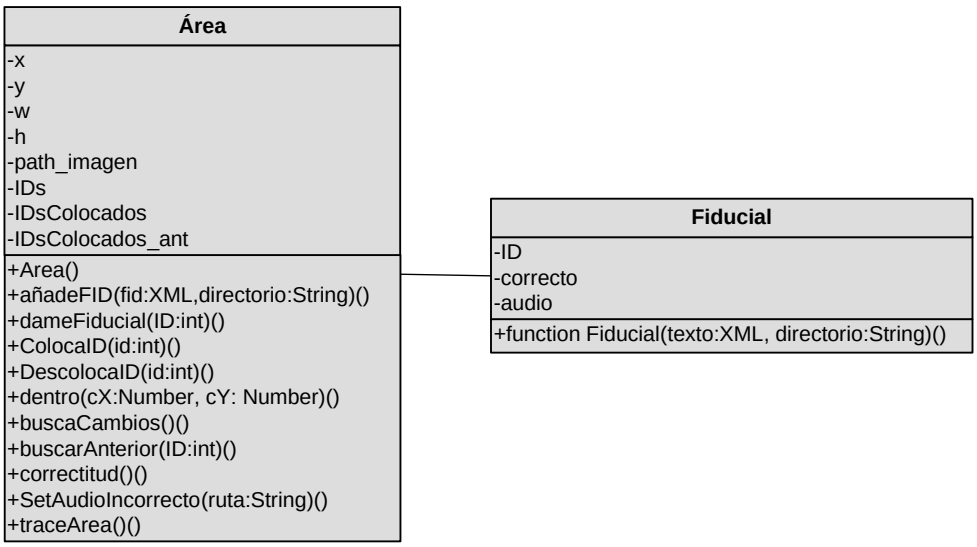


Figura E.13 Clases Área y Fiducial

A continuación se muestran y explican los métodos más significativos de la clase Área según la funcionalidad que desempeñan dentro de la clase.

- **Método constructor:**

```
public function Area (cX:Number, cY:Number, ancho:Number,
              alto:Number, tam:Number, img:XML, directorio:String,
              capa:Sprite, verbose:Boolean)
```

- **Método de gestión de la clase área:**

```
public function ColocaID(id:int):void
public function DescolocaID(id:int):void
```

Los dos métodos anteriores permiten añadir y eliminar respectivamente un *fiducial* a la lista de *IDsColocados* de la clase área. Estos dos métodos se llaman cuando un usuario pone o quita un objeto de la mesa.

```
public function añadeFID(fid:XML,directorio:String):void
private function dameFiducial(ID:int):Fiducial
```

La función `añadeFID` permite crear un objeto *Fiducial* a partir del nodo *Fiducial* definido dentro del nodo *Área* del fichero XML. Y la función `dameFiducial`, devuelve el objeto *fiducial* a partir de su identificador *ID*.

```
public function correctitud():int
public function dentro(cX:Number, cY: Number):Boolean
```

La función `correctitud` () se encarga de comprobar el estado en el que se encuentra un área. Devuelve -1, cuando el área se encuentra en estado incorrecto (alguno de los *fiduciales* colocados sobre ella no es correcto), 0 cuando el estado es neutro (todos los *fiduciales* colocados sobre ella son correctos, pero falta algún *fiducial* más por colocar), y 1 cuando el estado es correcto (todos los *fiduciales* colocados sobre ella son correctos y no falta ninguno por colocar).

La función `dentro(cX:Number, cY: Number)` se encarga de comprobar si las coordenadas (cX, cY) se encuentran dentro de un área. Esta función se usa para comprobar si un objeto está dentro o no de un área.

▪ CLASE FEEDBACK

La clase *Feedback* es la clase que se encarga de gestionar la información asociada a un nodo *feedback* del fichero XML, y de dibujarlo y cargarlo en pantalla.

En la Figura E.14 se muestra la clase *Feedback* junto sus atributos y métodos más significativos.

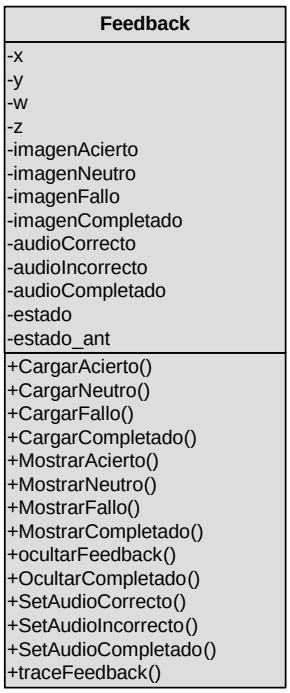


Figura E.14 Clase Feedback

A continuación se muestran y explican los métodos más significativos de la clase Área según la funcionalidad que desempeñan dentro de la clase.

- **Método constructor:**

```
public function Feedback(cX:Number, cY:Number, ancho:Number,
                 alto:Number, im_acierto:String, im_fallo:String,
                 im_neutro:String,im_completado:String,capa:Sprite)
```

- **Métodos de carga y gestión de *feedback* visual y auditivo:**

Para cargar el feedback visual:

```
public function CargarAcierto():void
public function CargarNeutro():void
public function CargarFallo():void
public function CargarCompletado():void
```

Para cargar el feedback auditivo:

```
public function SetAudioCorrecto(ruta:String):void
public function SetAudioIncorrecto(ruta:String):void
public function SetAudioCompletado(ruta:String):void
```

Para la gestión del feedback:

```
public function MostrarAcierto():void
public function MostrarNeutro():void
public function MostrarFallo():void
public function MostrarCompletado():void
public function OcultarCompletado():void
public function ocultarFeedback():void
```

El primer y segundo grupo de funciones se encargan de cargar en pantalla las imágenes y los audios correspondientes al *feedback* definido en la tarea. El último grupo de funciones permiten ocultar y mostrar el *feedback* cuando sea necesario en la actividad, de manera que el usuario recibirá un tipo y otro de *feedback* según haga acciones correctas, parcialmente correctas (neutras), o incorrectas.

E.3 Actividades Pedagógicas con Acoti

El conjunto de actividades pedagógicas que se pueden definir según la estructura y componentes especificado por *Acoti* es muy amplio. En esta sección se describe la estructura común de todas ellas, así como los elementos que se puede incluir.

E.3.1 Estructura de una actividad Acoti

Las actividades pedagógicas creadas con *Acoti* se definen en un fichero XML creado por el usuario con el nombre y la ubicación que él elija. En este fichero, el usuario referencia a todos aquellos recursos gráficos y de audio que compongan una actividad.

El fichero especifica la estructura y características de la actividad y sus componentes; además, el fichero hace referencia a los ficheros de imagen y audio que formen parte de cada uno de sus componentes. Estos ficheros podrán estar guardados en una ruta distinta a la ruta en la que se encuentre el fichero XML, pero tendrán que estar correctamente referenciados en el contenido de dicho fichero.

La Figura E.15 muestra un ejemplo de una posible organización de distintos juegos creados con *Acoti*.

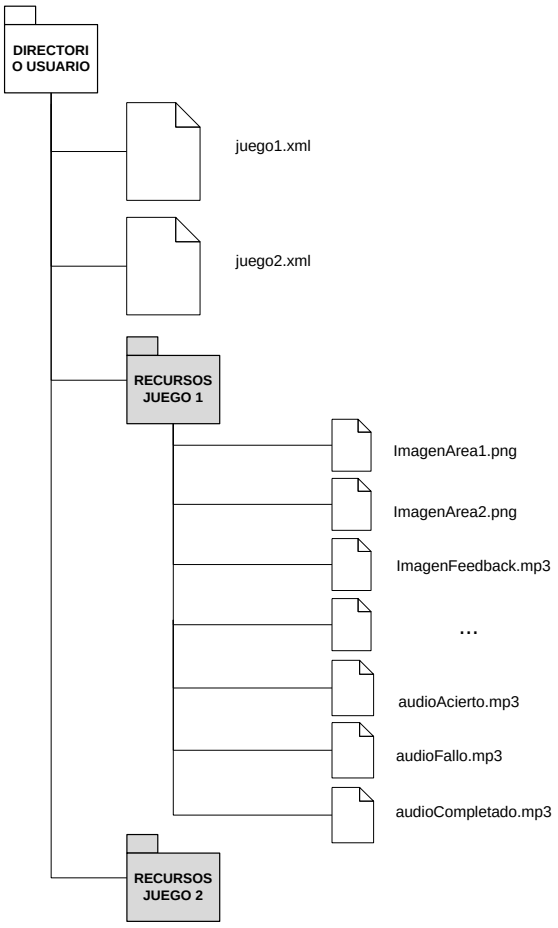


Figura E.15 . Ejemplo de estructura de una actividad de Acoti

E.3.2 Estructura del fichero XML de una actividad

En esta sección se especifica y explica la estructura del fichero XML que almacena una actividad *Acoti*, junto con todos sus componentes.

Para una mejor comprensión de la descripción de las actividades que almacenadas en los ficheros XML, se definió el formato de las mismas mediante la Descripción de Tipo de Documento (DTD). Su función básica es definir el formato de datos, de manera que sea común y mantenga la consistencia entre todos los documentos que utilicen la misma DTD.

1. Objeto juego

   <juego>

   Explicación: El objeto de juego principal, el cual se compone de una o varias tareas.

2. Objeto verbose

   <verbose valor="si/no"/>

   Explicación: Si el valor de *verbose* es "si", aparecerán marcadas en rojo las áreas interactivas.

3. Objeto tarea

   <tarea>

   Explicación: Una tarea es una actividad o juego completo. Un XML puede incluir varias tareas, las cuales se ejecutarán en secuencia conforme se vayan completando. Un bloque tarea contendrá los siguientes objetos:

   a. Objeto fondo

   ```
   <fondo>
   <color rgb="rrggbb"\>
   <imagen path="carpeta/nombrefichero"\>
   <sonido path="carpeta/nombrefichero" repetir="si/no"
   tiempo="num_segundos"\>
   </fondo>
   ```

   Explicación: El fondo es opcional. Puede no haber ningún fondo definido, pero si hay, sólo hay uno.

- **Color**: definido en formato rgb. (opcional)
- **Imagen**: La imagen que se pone de fondo. No se define ni su posición ni tamaño. Se re-escala para ocupar toda la pantalla y se coloca en la posición (0,0) (Opcional).
- **Sonido**: Se asocia un sonido a reproducir que puede ser de dos tipos: Un sonido que sonará al inicio del juego, o sonido que se irá repitiendo durante el juego. (Opcional)
- **Repetir true:** Si Repetir=true, se usa para música, o un sonido de ambiente que se repetirá continuamente. En este caso, *tiempo* indica los segundos de pausa entre cada repetición.
- **Repetir false:** Si Repetir =false. El audio sólo se reproduce al comienzo del juego una sola vez (Por ejemplo una explicación del juego). En este caso, *tiempo* indica los segundos que tarda este audio en reproducirse.

b. Objeto área

```
<area>
<posicion x=num y=num ancho=num alto=num/>
<imagen path="carpeta/nombrefichero/" x="num" y="num" ancho=num
alto=num />
<fid id="[num/*]" correcto=[si/no]sonido="carpeta/fichero.mp3"/>
  …
</area>
```

Explicación: Áreas de juego, donde se ponen los juguetes. Puede haber una o varias áreas de juego.

- **Posición**: Indica las posiciones y dimensiones del área.
- **Imagen**: Imagen que se pone en el área, en la posición x, y (opcional), pero podría tener otro tamaño distinto a su original (ancho, alto). (Opcional)
- **Fid**: Lista de *fiduciales* que se pueden poner en el área. ID es el numero fid, o * para todos los que no estén especificados. Correcto, para decir si colocar allí ese *fiducial* es correcto o no, y sonido es el sonido que se dispara al poner allí el *fiducial*

En la Figura E.16 se muestra un ejemplo de un fichero de juego según el formato descrito previamente.

```
<juego>
<verbose valor="no"/>
<tarea>
	<fondo>
		<color rgb="0x000000"/>
		<imagen path="supermercado\fondo_super.png"/>
		<sonido path="supermercado\intro_1.mp3" repetir="no" tiempo="3"/>
	</fondo>
	<area>
		<posicion x="0" y="0" ancho="600" alto="700"/>
		<imagen path="supermercado\cebolla.png" x="700" y="70" ancho="100" alto="100"/>
		<fid id="0" correcto="si" sonido="supermercado\cebolla.mp3"/>
	</area>
	<area>
		<posicion x="0" y="0" ancho="600" alto="700"/>
		<imagen path="supermercado\guisantes.png" x="700" y="220" ancho="100" alto="100"/>
		<fid id="1" correcto="si" sonido="supermercado\guisantes.mp3"/>
	</area>
	<area>
		<posicion x="0" y="0" ancho="600" alto="700"/>
		<imagen path="supermercado\pimiento.png" x="700" y="370" ancho="100" alto="100"/>
		<fid id="2" correcto="si" sonido="supermercado\pimiento.mp3"/>
	</area>

	<feedback>
		<pos_rel x="-100" y="0" ancho="100" alto="100"/>
		<imagen_acierto path="ARCHIVOS\sonrisa_cal.png"/>
		<imagen_fallo path="ARCHIVOS\triste_cal.png"/>
		<imagen_neutro path="ARCHIVOS\neutral_cal.png"/>
		<imagen_completado path="ARCHIVOS\corona.png"/>
		<sonido_acierto path="ARCHIVOS\bien.mp3"/>
		<sonido_completado path="ARCHIVOS\aplausos.mp3"/>
	</feedback>
</tarea>

<tarea>
	<fondo>
		<color rgb="0x000000"/>
		<imagen path="supermercado\fondo_super.png"/>
		<sonido path="supermercado\intro_2.mp3" repetir="no" tiempo="3"/>
	</fondo>
	<area>
		<posicion x="0" y="0" ancho="600" alto="700"/>
		<imagen path="supermercado\naranja.png" x="700" y="70" ancho="100" alto="100"/>
		<fid id="3" correcto="si" sonido="supermercado\naranja.mp3"/>
	</area>
	<area>
		<posicion x="0" y="0" ancho="600" alto="700"/>
		<imagen path="supermercado\tomate.png" x="700" y="220" ancho="100" alto="100"/>
		<fid id="4" correcto="si" sonido="supermercado\tomate.mp3"/>
	</area>
	<area>
		<posicion x="0" y="0" ancho="600" alto="700"/>
		<imagen path="supermercado\zanahoria.png" x="700" y="370" ancho="100" alto="100"/>
		<fid id="5" correcto="si" sonido="supermercado\zanahoria.mp3"/>
	</area>
	<feedback>
		<pos_rel x="-100" y="0" ancho="100" alto="100"/>
		<imagen_acierto path="ARCHIVOS\sonrisa_cal.png"/>
		<imagen_fallo path="ARCHIVOS\triste_cal.png"/>
		<imagen_neutro path="ARCHIVOS\neutral_cal.png"/>
		<imagen_completado path="ARCHIVOS\corona.png"/>
		<sonido_acierto path="ARCHIVOS\bien.mp3"/>
		<sonido_completado path="ARCHIVOS\aplausos.mp3"/>
	</feedback>
</tarea>


</juego>
```

Figura E.16 . Ejemplo de estructura de una actividad de Acoti

E.4 Pruebas realizadas en Acoti

Las pruebas de un producto *software* son los procesos que permiten verificar su calidad, y se han utilizado para identificar posibles fallos de implementación o de usabilidad. El conjunto de pruebas se han realizado a lo largo de todo el proceso de desarrollo de *Acoti*. Puesto que el modelo de proceso seguido a lo largo del desarrollo es el modelo incremental [9], descrito en el Anexo C de la memoria, la finalidad de las pruebas ha sido poder solucionar el mayor número de errores en cada etapa, y no ir arrastrándolos con el avance del proyecto.

En esta sección se explican los diferentes tipos de *test* empleados y la finalidad de cada uno de ellos.

E.4.1 Pruebas unitarias

Este tipo de pruebas se utilizan para probar cada módulo de código desarrollado y permiten asegurar que cada uno de ellos funciona correctamente por separado. Una de las principales ventajas de este tipo de pruebas es que simplifican la integración de los módulos, ya que proporcionan un alto nivel de garantía de que el código funciona de la manera esperada.

Las pruebas se han realizado para cada uno de los módulos de la aplicación *Acoti* por separado. En primer lugar se elaboraron pruebas para verificar la correcta lectura y carga de los distintos tipos de elementos que componen una actividad. En concreto se probó la correcta lectura y carga del fondo, de las áreas, de los *feedback* y de los *fiduciales*. Y en segundo lugar, una vez todos estos componentes se cargasen y visualizasen correctamente de forma individual, se probaron de manera conjunta.

Estas pruebas han sido realizadas con el simulador *TuioSimulator*, que permite emular el comportamiento del *tabletop* NIKVision: Poner juguetes, quitarlos y modificarlos sobre la zona de juego. En la Figura E.17 se muestra una imagen del simulador *TuioSimulator*.

E.4.2 Pruebas de integración

Después de realizar las pruebas unitarias de cada uno de los componentes se realizaron las pruebas de integración. Estas pruebas sirven para verificar que las distintas partes del *software* funcionan correctamente cuando se integran entre sí. De esta forma, se prueba que cada parte desarrollada y probada de forma individual se comporta de igual manera al integrarse con las demás funcionalidades.

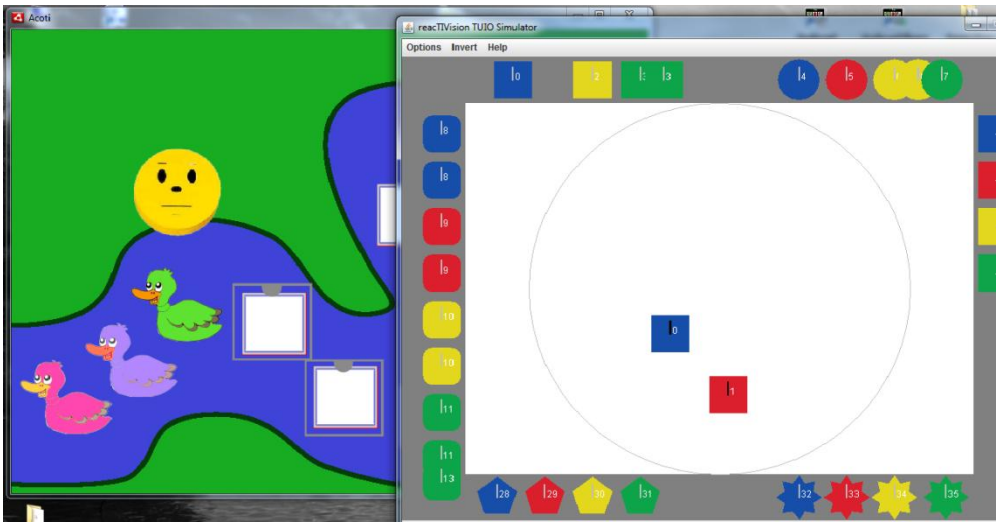


Figura E. 17 . Prueba sobre el simulador TuioSimulator

Este tipo de pruebas se han realizado para verificar, por un lado, la correcta integración de los distintos elementos verificados en las pruebas unitarias y su incorporación dentro de tareas que forman una misma actividad. Y por otro lado, para verificar la compatibilidad de la ejecución de *Acoti* sobre las distintas versiones de *tabletop* disponibles tanto en el laboratorio como en el CPEE Alborada.

Para asegurar el buen funcionamiento de la aplicación de forma global, se implementaron un conjunto de actividades y se ejecutaron tanto en el *TuioSimulator* como en los dos *tabletops* disponibles en el laboratorio y en el colegio.

E.4.3 Pruebas de sistema

Este tipo de pruebas permiten comprobar el funcionamiento y el rendimiento global de la aplicación. El software ya validado con las pruebas de integración, se probó con el resto del sistema para verificar los siguientes aspectos:

- **Rendimiento**: Se han realizado diferentes pruebas para determinar el tiempo de respuesta de determinadas operaciones que ofrece la aplicación. Algunas de ellas han sido las operaciones de poner/quitar/modificar objetos sobre la superficie de la mesa NIKVision para observar la rapidez con la que el programa responde al usuario con *feedbacks*. Otro aspecto evaluado ha sido el tiempo de respuesta de carga de las secuencias de tareas una vez se ha completado la anterior.

- **Resistencia**: Estas pruebas sirven para determinar hasta dónde puede soportar el programa determinadas condiciones extremas. Sobre todo se ha probado la inclusión de numerosos objetos en una misma tarea, para comprobar cómo reacciona el sistema ante la gestión de varios objetos colocados sobre la superficie de la mesa simultáneamente.

- **Robustez**: Las pruebas realizadas tenían por objetivo determinar la capacidad del programa para soportar entradas incorrectas y asegurar que la aplicación era capaz de responder sin fallos.

E.4.4 Pruebas de usuario

Con este tipo de pruebas se mide el nivel de adaptación del sistema a las necesidades del usuario. Es importante tener en cuenta la facilidad de uso durante todo el desarrollo del proyecto para conseguir unos resultados satisfactorios para la mayoría de los futuros usuarios.

Son esenciales para el tipo de programa desarrollado debido a los usuarios a los que va dirigida la aplicación. Algunas de los aspectos evaluados con este tipo de pruebas son los siguientes:

- **Facilidad de aprendizaje:** Se ha buscado en todo momento que la aplicación sea sencilla para los usuarios proporcionándoles *feedback,* tanto visual como auditivo, siempre que interactúen con el *tabletop*. Esta manera de enfocar las actividades proporciona una guía al usuario para aprender de sus errores y ayudarle a completar las actividades.

- **Satisfacción del usuario:** Se determina la calidad de la experiencia de un usuario en su interacción con el programa. Para evaluar este aspecto se han realizado distintas pruebas con alumnos del colegio CPEE Alborada, y durante las jornadas de La semana de la Ingeniería, con alumnos de distintos colegios. La satisfacción en el uso de la aplicación es consecuencia de la facilidad de uso del programa y el cumplimiento de los objetivos previamente establecidos.

E.5 Herramientas utilizadas

A continuación se detallan las principales herramientas y tecnologías utilizadas a lo largo del desarrollo de este proyecto:

Sistema operativo

- Microsoft WindowsXP, WindowsVista, Windows 7

Tecnologías

- XML (eXtensible Markup Language)

Propósito general

- Navegador web Mozilla Firefox
- Navegador web Chrome
- Paint

Análisis y diseño

- Generador de diagramas: ArgoUML 0.24
- Microsoft Office Visio 2007

Desarrollo y pruebas

- Entorno de desarrollo: Adobe Flash Professional CS5
- Entorno de desarrollo: Adobe Flash Professional CS5.5
- Entorno de ejecución: Adobe AIR
- Simulador *TuioSimulator*

Generación de ejecutables e instaladores

- Adobe Flash Profesional CS5 y CS5.5 con extensión AIR.

Documentación

- Microsoft Office Word 2007 y 2010
- Microsoft Office Visio 2007
- Gantt Project 2.5.1

Dispositivos hardware

- *Tabletop* disponible en el CPEE Alborada.
- *Tabletops* (v0 y v1) disponibles en el Laboratorio Affective Lab del grupo GIGA.

Anexo F. Manual de usuario de AraBoard

Este anexo presenta el manual de usuario elaborado para la aplicación *AraBoard*, publicada bajo licencia `GNU General Public License version 3.0`. En él se presenta una guía para instalar y ejecutar las aplicaciones de construcción y reproducción de tableros de comunicación: *AraBoard Constructor* y *AraBoard Player*.

F.1 Introducción

Este es el manual de usuario de *AraBoard versión 1.0*. En él se explica el funcionamiento de las aplicaciones que lo componen, su aspecto y las formas de realizar cada una de las posibles acciones.

F.2 Distribución, instalación y ejecución de la aplicación

La aplicación *AraBoard* se distribuye a través de la página web del GIGA Affective Lab (1), y a través de su página de la aplicación en Source Forge (2):

1. http://giga.cps.unizar.es/affectivelab/araboard.html
2. http://sourceforge.net/projects/ara-board/

En el primer caso, la aplicación se obtiene a través de la página web del GIGA Affect, tras hacer *click* sobre el enlace "*Descarga AraBoard para MS Windows en SourceForge*" (Figura F.1). Automáticamente, la página redirige a la web de *SourceForge*, lugar donde se encuentra disponible la versión de *AraBoard* para Windows.

Figura F.1. AraBoard en la página web del GIGA Affective Lab

Una vez en la página de *AraBoard* en *SourceForge*, se presiona el botón "Download" para obtener la aplicación (Figura F.2).

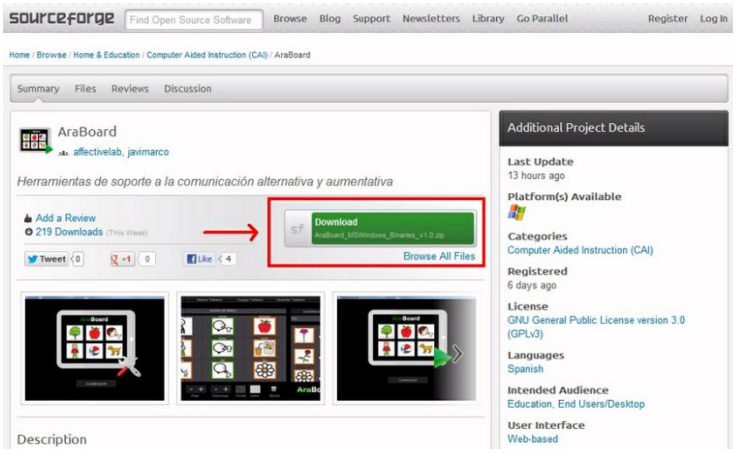
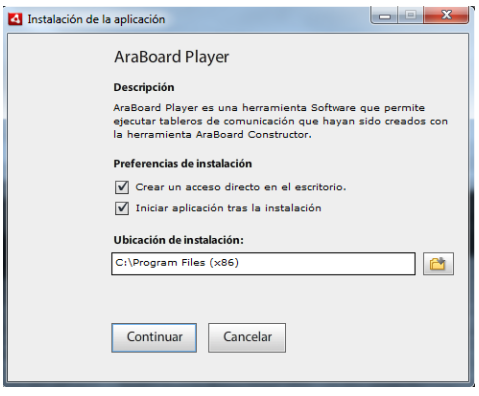


Figura F.2. AraBoard en SourceForge

Para instalar *AraBoard* hay que descomprimir el archivo descargado: *AraBoard_MSWindows_Binaries_v1.0.zip*, en la carpeta elegida para ello. Una vez se ha descomprimido, se obtienen dos instaladores: `AraBoardConstructor.exe` y `AraBoardPlayer.exe`. En la figura F.3 se muestran las ventanas de instalación de la aplicación.



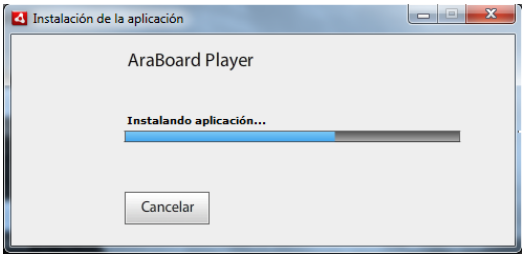


Figura F.3. Instalación de AraBoard Player

Los instaladores crean automáticamente dos ejecutables, uno para la aplicación *AraBoard Constructor* y otro para la aplicación *AraBoard Player*. También se encargan de crear los accesos directos correspondientes en el Escritorio y el Menú Inicio de Windows (Figura F.4).

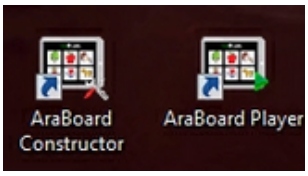


Figura F.4. Accesos directos de AraBoard

F.2.1 Aplicaciones requeridas

Para que la aplicación se pueda ejecutar es necesario tener instalado el entorno de ejecución multiplataforma Adobe AIR en el ordenador en el que se va a utilizar la aplicación. En caso de que el entorno no esté instalado, se descarga automáticamente al realizar la instalación de la aplicación.

F.3 Formatos soportados

A continuación se detallan los distintos formatos de imagen y sonido que son reconocidos por las aplicaciones.

F.3.1 Formatos de imagen

Los formatos de imagen reconocidos por las aplicaciones son: JPEG, JPG y PNG.

F.3.2 Formatos de audio

Los formatos de audio reconocidos por la aplicaciones son: MP3.

F.4 Manual de usuario de AraBoard Constructor

En esta parte del manual, se explican las funcionalidades de la aplicación de construcción y edición de tableros de comunicación: *AraBoard Constructor*.

F.4.1 Ventana de inicio de la aplicación

Esta ventana aparece al comenzar la aplicación, y muestra el logo de *AraBoard Constructor*. Si se presiona el botón "comenzar", la aplicación muestra la ventana principal de la aplicación (Figura F.5).

Figura F.5. Ventana inicial de AraBoard Constructor

F.4.2 Ventana principal de la aplicación

La Figura F.6 muestra la ventana principal de la aplicación *AraBoard Constructor* (Figura F.6). Esta ventana permite construir, editar y personalizar tableros de comunicación. A continuación se describen todos los elementos que forman la ventana principal.

Figura F.6. Ventana principal de AraBoard Constructor

Los elementos de esta ventana son los siguientes:

1. **Barra de menú superior:** Mediante la barra de menú superior se pueden realizar las siguientes acciones:

    - **Botón Inicio:** Permite volver a la pantalla inicial (Figura F.5).

    - **Botón Cargar Tablero:** Permite crear un nuevo tablero. En el caso de que hubiera un tablero cargado previamente, elimina todos los elementos de la interfaz (tableros previamente cargados y resultados de búsqueda).

    - **Botón Cargar Tablero:** Permite cargar un tablero de comunicación para editarlo.

2. **Botón Acerca de…:** Permite consultar la información acerca de la aplicación.

3. **Título del tablero**: Campo editable que permite definir el título del tablero.

4. **Idioma de búsqueda**: Permite seleccionar el idioma en que se realizarán las búsquedas de pictogramas en la colección de ARASAAC.

5. **Campo de búsqueda**: Permite escribir la palabra que se desea buscar en la colección de pictogramas de ARASAAC.

6. **Resultados de búsqueda**: Muestra el listado de pictogramas resultados de búsqueda, como resultado de la búsqueda de la palabra introducida en el campo de búsqueda.

    - Cada uno de los resultados puede ser arrastrado y colocado en las celdas del tablero.

    - El audio asociado a cada resultado de búsqueda puede escucharse al presionar el icono del altavoz de su derecha.

7. **Celdas del tablero**: Si la celda contiene un pictograma, éste puede ser arrastrado y recolocado en otra celda. Si no lo contiene, al hacer *click* sobre el icono del lápiz, editamos el contenido de la celda, y a partir de recursos propios se puede crear un nuevo pictograma.

8. **Icono de edición**: Muestra la pantalla de edición de pictogramas cuando un pictograma es colocado sobre él.

9. **Herramientas de personalización**: Mediante la barra de herramientas de personalización del tablero se pueden realizar las siguientes acciones:

    a. Modificar el número de filas: Incrementar el número de filas (al presionar el botón "+"), o decrementarlo (al presionar el botón "-").

    b. Modificar el número de columnas: Incrementar el número de columnas (al presionar el botón "+"), o decrementarlo (al presionar el botón "-").

    c. Modificar el color de fondo: Presionando sobre el seleccionador de color "Fondo".

    d. Modificar el color de la fuente: Presionando sobre el seleccionador de color "Letra".

10. **Icono de la papelera**: Elimina un pictograma cuando éste es colocado sobre él.

F.4.3 Ventana de carga de tableros

Esta ventana permite seleccionar el tablero de comunicación que se desea modificar, y cargarlo en la ventana principal de construcción y edición de tableros. Para ello se debe pulsar en el botón "Cargar Tablero" del menú superior de opciones, y buscar la carpeta donde se haya almacenado el tablero. Y a continuación pulsar sobre el archivo con extensión XML (Figura F.7 y F.8).

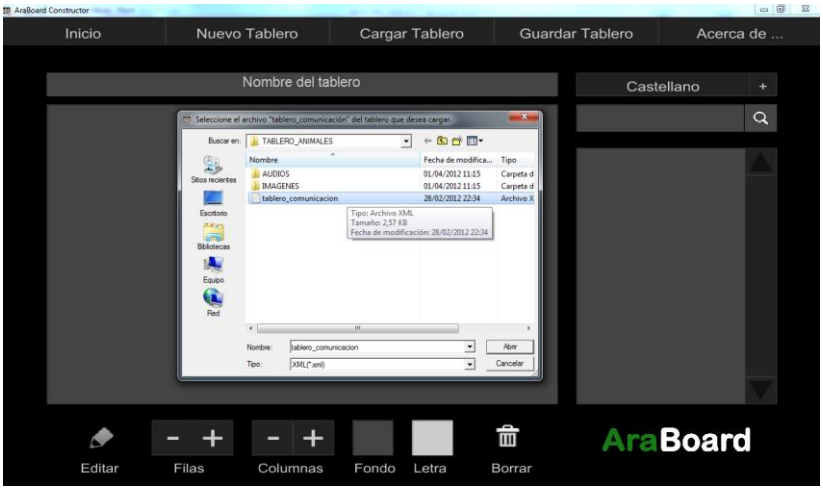


Figura F.7. Ventana de carga de tablero

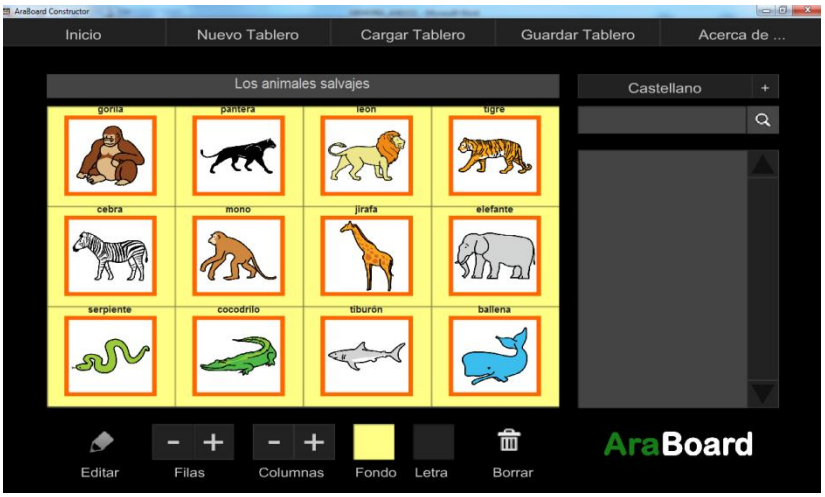


Figura F.8. Tablero cargado en la ventana principal de construcción

F.4.4 Ventana de almacenamiento de tableros

Esta ventana permite almacenar el tablero de comunicación creado en la ubicación del ordenador que se prefiera. Para ello se debe pulsar en el botón "Guardar Tablero" del menú superior de opciones, y seleccionar la ubicación a través de la ventana de exploración de Windows (Figura F.9).

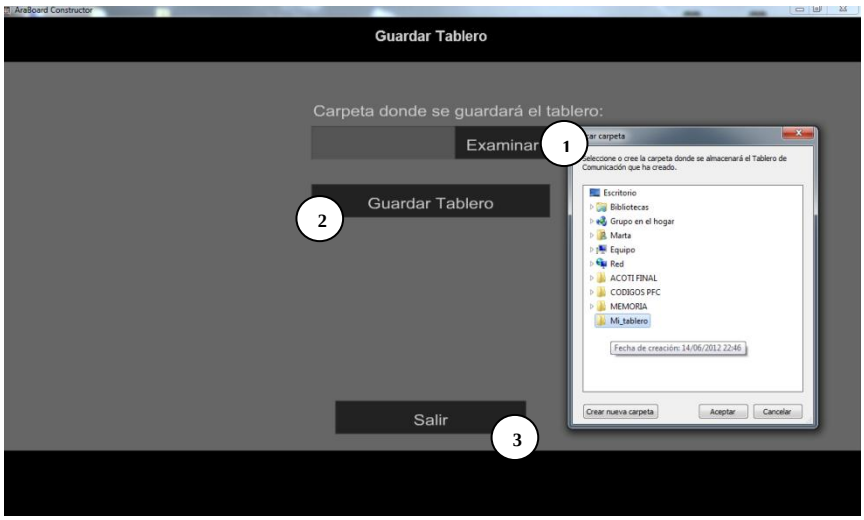


Figura F.9. Ventana de almacenamiento de un tablero

Los elementos de esta ventana son los siguientes:

1. **Botón Examinar:** Permite seleccionar la ubicación donde se almacenará el tablero de comunicación creado.
2. **Botón Guardar Tablero:** Permite guardar el tablero de comunicación en la ubicación seleccionada.
3. **Botón Salir:** Permite salir de la ventana de almacenamiento.

F.4.5 Ventana de edición de pictogramas

Para poder editar un pictograma es necesario arrastrarlo con el ratón sobre el icono de edición (Figura F.6, elemento 8). Una vez se deja de presionar el ratón, aparece la ventana de edición de pictogramas. Esta ventana permite editar el contenido de un pictograma que forma parte de una celda del tablero (Figura F.10). Esta ventana también es accesible si se hace *click* sobre una celda vacía del tablero. De esta manera, se puede crear un pictograma desde cero, que se situará en la celda vacía sobre la que se ha hecho *click* (Figura F.11).

Figura F.10. Ventana de edición de un pictograma

Figura F.11. Ventana de edición de un pictograma a partir de una celda vacía

Los elementos de esta ventana son los siguientes:

1. **Nombre:** Permite cambiar el nombre de un pictograma.
2. **Botón para examinar imagen:** Permite examinar un fichero de imagen para incorporarlo al pictograma y reemplazar la imagen existente.
3. **Botón para examinar audio:** Permite examinar un fichero de audio para incorporarlo al pictograma y reemplazar el audio existente.
4. **Botón para seleccionar la categoría del pictograma:** Permite seleccionar la categoría de la palabra que representa el pictograma (nombre propio, nombre común, acción, descriptivo, contenido social, miscelánea)
5. **Vista previa:** Muestra el pictograma que se está editando. Si pulsamos encima del pictograma se puede escuchar su audio asociado.
6. **Botón Aceptar:** Permite guardar los cambios realizados sobre el pictograma y volver a la pantalla principal de la aplicación.
7. **Botón Cancelar:** Permite salir de la ventana de edición sin guardar los cambios.

F.5 Manual de usuario de AraBoard Player

En esta parte del manual, se explican las funcionalidades de la aplicación de visualización y reproducción de tableros de comunicación: *AraBoard Player*.

F.5.1 Ventana de inicio de la aplicación

Esta ventana aparece al comenzar la aplicación, y muestra el logo de *AraBoard Player*. Si se presiona el botón "comenzar", la aplicación muestra la ventana principal de la aplicación (Figura F.12).

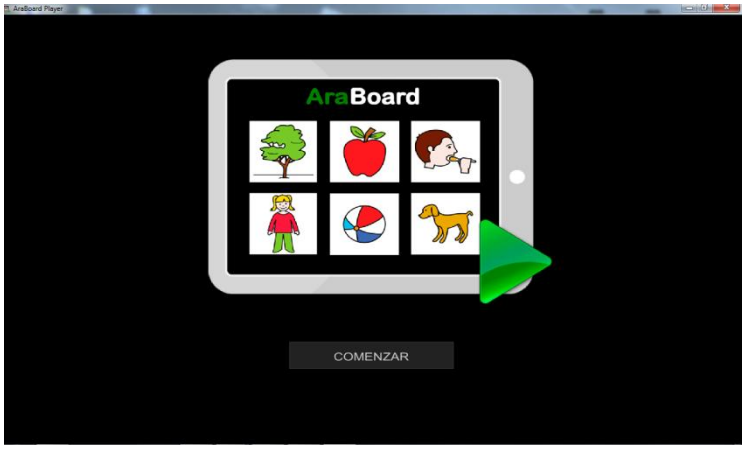


Figura F.12. Ventana inicial de AraBoard Player

F.5.2 Ventana principal de la aplicación

Esta ventana describe los pasos que se deben seguir para cargar y visualizar un tablero de comunicación previamente creado con la aplicación de construcción *AraBoard Constructor*.

Mediante la barra de menú superior se pueden realizar las siguientes acciones:

- **Botón Inicio:** Permite volver a la pantalla inicial (Figura F.13).
- **Botón Cargar Tablero:** Permite cargar un tablero de comunicación para reproducirlo
- **Botón Acerca de…:** Permite consultar la información acerca de la aplicación.

Figura F.13. Ventana inicial de AraBoard Player

F.5.3 Ventana para cargar y reproducir un tablero

Esta ventana permite seleccionar el tablero de comunicación que se desee visualizar. Para ello se debe pulsar en el botón "Cargar Tablero", a continuación buscar la carpeta donde haya almacenado su tablero y pulsar sobre el archivo con extensión XML (Figura F.14 y F.15).

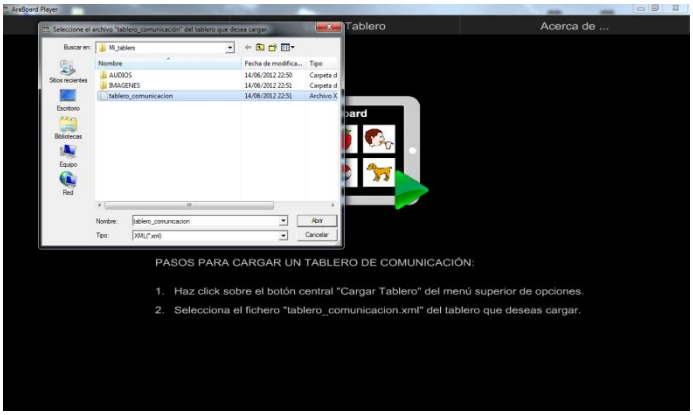


Figura F.14. Ventana de carga de un tablero

Figura F.15. Tablero cargado en la ventana principal de la aplicación

Índice de Figuras

Índice de Tablas

Bibliografía

[1] Colegio Público de Educación Especial. Alborada. C.P.E.E. Alborada. http://centros6.pntic.mec.es/cpee.alborada/ (último acceso: 15/06/2012).

[2] Augmentative and alternative communication. Wikipedia. http://en.wikipedia.org/wiki/Augmentativeándálternative_communication (último acceso: 15/06/2012).

[3] American Speech-Language-Hearing Association (ASHA). http://www.asha.org/ (último acceso: 15/06/2012).

[4] Abadín, C. I. Delgado Santos, A. Vigara Cerrato. *Comunicación aumentativa y Alternativa*. Guía de referencia. Edición CEPAT, 2009.

[5] R. Sánchez Montoya. *Ordenador y discapacidad*. Madrid. Ciencias de la educación preescolar y especial. ISBN: 8478694021, 2002.

[6] Alcantud, F. *Las Tecnologias de la Información y de la Comunicación y los Trastornos Generalizados del Desarrollo*. Universitat de Valencia, 2004.

[7] J.Marco, E.Cerezo, S. Baldassarri, E. Mazzone, J. Read. *Bringing Tabletop Technologies to Kindergarten Children*. 23rd BCS Conference on Human Computer Interaction. pp. 103-111, 2009.

[8] Portal Aragonés de la Comunicación Alternativa y Aumentativa http://catedu.es/arasaac/index.php (último acceso: 15/06/2012).

[9] Ian Sommerville. *Ingeniería del Software*. Addison-Wesley, 7a edition, 2005.

[10] W. Premerlani F. Eddy W. Lorensen J. Rumbaugh, M. Blaha. Modelado y diseño orientado a objetos. Metodología OMT. Prentice Hall, 1999.

[11] MinimalComps. A set of ActionScript 3.0 User Interface Components for Flash. http://www.minimalcomps.com/ (último acceso: 14/06/2012)

[12] ThanksMister. Flex, AIR, & Android Development Blog. http://thanksmister.com/category/air/ (último acceso: 14/06/2012)

[13] Veronique Brossier. *Developing Android Applications with Adobe AIR*. O'Reilly. ISBN: 978-1-449-39482-0, 2011.

[14] World Wide Web Consortium: W3C. Web XML: Extensible Markup Language. http://www.w3.org/XML/ (último acceso: 13/06/2012)

[15] Kaltenbrunner, M., Bencina, R.: *"ReacTIVision: A Computer-Vision Framework for Table-Based Tangible Interaction"*. Proceedings of the first international conference on *"Tangible and Embedded Interaction"* (TEI07). Baton Rouge, Louisiana, 2007.

[16] Framework de ReacTIVision. http://reactivision.sourceforge.net/ (último acceso: 12/06/2012)

[17] Kaltenbrunner, M., Bovermann, T., Bencina, R., Costanza, E.: *"TUIO - A Protocol for Table-Top Tangible User Interfaces"*. Proceedings of the 6th International Workshop on Gesture in Human-Computer Interaction and Simulation (GW 2005), Vannes, France, 2005.

[18] Especificación del protocolo TUIO. http://www.tuio.org/?specification (último acceso: 12/06/2012)

[19] Blog de ARASAAC. http://blog.arasaac.org/ (último acceso: 16/06/2012).

[20] Blog: Informática para la educación Especial. http://informaticaparaeducacionespecial.blogspot.com.es/ (último acceso: 16/06/2012).

[21] AraBoard en la página web del GIGA Affective Lab. http://giga.cps.unizar.es/affectivelab/araboard.html (último acceso: 16/06/2012).

[22] Proyecto AraBoard en SourceForge. http://sourceforge.net/projects/ara-board/ (último acceso: 16/06/2012).

[23] Blog Pordereito: Asociación para la integración de personas con autismo. http://pordereito.blogspot.com.es/2012/03/1-xornada-de-por-dereito-ferramentas.html (último acceso: 16/06/2012).

[24] IV Semana de la Ingeniería y la Arquitectura.
http://www.semanadelaingenieria.com/index.php?option=com_content&view=article&id=15&Itemid=31 (último acceso: 16/06/2012).

[25] TICO (Tableros Interactivos de Comunicación). Página web del Proyecto Tico.
http://www.proyectotico.com/wiki/index.php/Inicio (último acceso: 28/05/2012).

[26] In-TIC (Integración de las Tecnologías de la Información y las Comunicaciones en los colectivos de personas con diversidad funcional). Página web de In-TIC.
http://www.proyectosfundacionorange.es/intic/queés_in-tic/ (último acceso: 28/05/2012).

[27] e-Mintza (en Euskera, *habla electrónica*). Página web de la aplicación e-Mintza.
http://fundacionorange.es/emintza.html (último acceso: 28/05/2012).

[28] PictoDroid Lite. Página web de la aplicación PictoDroid Lite.
http://www.accegal.org/pictodroid-lite/ (último acceso: 28/05/2012).

[29] Baluh. Página web de la aplicación Baluh.
http://blog.baluh.org/ (último acceso: 28/05/2012).

[30] iAutism. Página web de IAutismo.
http://www.iautism.info/ (último acceso: 28/05/2012).

[31] CPA (Comunicador Personal Adaptable). Página web de la aplicación CPA.
http://www.comunicadorcpa.com/ (último acceso: 28/05/2012).

[32] Adobe ActionScript. Wikipedia.
http://es.wikipedia.org/wiki/ActionScript (último acceso: 12/06/2012).

[33] Adobe Flash. Wikipedia.
http://es.wikipedia.org/wiki/Adobe_Flash (último acceso: 12/06/2012).

[34] Adobe Flash Player. Wikipedia.
http://es.wikipedia.org/wiki/Adobe_Flash_Player (último acceso: 12/06/2012).

[35] Adobe Integrated Runtime (Adobe AIR). Wikipedia.
http://es.wikipedia.org/wiki/Adobe_Integrated_Runtime (último acceso: 12/06/2012).

[36] Creación de aplicaciones de ADOBE ® AIR.
http://help.adobe.com/eséS/air/build/air_buildingapps.pdf

[37] Especificación ECMAScript. Wikipedia.
http://es.wikipedia.org/wiki/ECMAScript (último acceso: 17/06/2012).

[38] Blog Bit101 (librería de componentes gráficos Minimal Componentes)
http://www.bit-101.com (último acceso: 17/06/2012).

[39] Documentación TUIO para Adobe Flash
http://bubblebird.at/tuioflash/doc/ (último acceso: 17/06/2012).

[40] TUIO. Processing *Tuio Client API*.
http://bubblebird.at/tuioflash/guides/using-the-tuioclient/ (último acceso: 17/06/2012).