

Dedicatoria

Maia Grauer

“...quiero dedicar este proyecto a mi familia y amigos que me acompañaron durante estos cinco años de carrera. En especial a mis padres y hermanos, a mis tíos Ernesto, Mariana, Marcelo y Anita, a mis primos, a mis abuelos y a mi amiga María. Gracias por compartir junto a mí tantos momentos de alegría, tensión y cansancio. Hoy culmina una etapa muy importante de mi vida y quiero agradecerles por el apoyo brindado.

También quiero agradecer a mi hermano Alex, estudiante de Ingeniería, que se interesó por este proyecto y me ayudó a concretar algunas ideas importantes.

Finalmente quiero dedicarle este proyecto a aquellos profesores y funcionarios de la Universidad de Montevideo que me guiaron durante este arduo camino, escucharon mis inquietudes y me motivaron para seguir adelante...”

Andrés País

“...en este momento de mi vida donde me encuentro finalizando mi etapa universitaria, quiero acordarme de aquellas personas que siempre estuvieron a mi lado apoyándome y motivándome para seguir adelante. Decirles que sin ustedes hubiera sido imposible recorrer este difícil pero maravilloso camino.

En primer lugar quiero dedicar este proyecto a mi familia, que siempre estuvo a mi lado dándome apoyo y afecto tanto en los buenos momentos como en los malos. Me hicieron ver que lo importante no es simplemente el objetivo sino la forma en que se logra. La importancia que tienen las decisiones correctas y la manera de encarar las dificultades hacen que el objetivo final sea cada vez menos utópico. Su constante apoyo hizo que hoy pueda cosechar el fruto del esfuerzo realizado durante todos estos años.

Por otro lado a mis amigos, que por más que en muchas ocasiones sacrificué mí tiempo con ellos, siempre lo entendieron y me impulsaron a seguir adelante.

A mis compañeros, que a lo largo de todos estos años hemos atravesado momentos de alegría y tristeza, de incertidumbre, de nervios, de esperanza, hemos compartido desayunos, almuerzos, y cenas, días y noches, inviernos y veranos pero siempre con un objetivo final que nos mantenía unidos y deseosos de lograrlo, sin importar cuantas piedras nos pusieran en el camino.

Y por último pero no menos importante a todos los profesores y funcionarios de la universidad que me hicieron ser lo que soy, no sólo como profesional pero sino también como persona.

Gracias a todos y espero que todas las esperanzas que han puesto en mí tengan sus frutos en el transcurso de mi vida como ingeniero y como persona...”

Reconocimientos

A nuestros tutores de PFC Thomas Hobbins y Marcelo Abreu por ayudarnos a llevar a cabo este proyecto. Gracias a su exigencia y dedicación pudimos ordenar nuestras ideas y concretar los objetivos planteados.

Agradecemos a Rafael Sotelo, tutor y coordinador de la carrera, quien nos aconsejó en el transcurso de la misma, y supo atender nuestras inquietudes.

Resumen Ejecutivo

Introducción

Actualmente, la televisión es el medio masivo de comunicación que más usuarios ha captado. Si bien esto se debe al grado de entretenimiento brindado, también ha sido muy difundida gracias a su bajo costo.

Hoy en día la misma hace uso de tecnología analógica, pero próximamente, con el fenómeno llamado “apagón analógico”, dicha tecnología será reemplazada por la digital, permitiendo así la introducción de interactividad. La interactividad permitirá a los televidentes no sólo utilizar la televisión de la manera tradicional, sino que dará la posibilidad de acceder a un amplio conjunto de servicios públicos o privados a través del televisor, con un único terminal y un comando a distancia.

Bajo este marco, el presente proyecto tiene como principal objetivo contribuir con el desarrollo de aplicaciones interactivas para TDT (Televisión Digital Terrestre) destinadas a proporcionar servicios ciudadanos. Para ello se analizarán las tecnologías involucradas y se desarrollará una aplicación estándar de carácter comunitario.

En cuanto a las tecnologías involucradas, se destaca el estándar MHP (Multimedia Home Platform). Una plataforma que intenta estandarizar las aplicaciones interactivas de TV permitiendo una elevada portabilidad e interoperabilidad entre decodificadores y plataformas.

Antecedentes

El mercado mundial de aplicaciones interactivas de TV se encuentra en pleno auge. Pero, dado que es una tecnología muy reciente, en Uruguay todavía no se han hecho grandes esfuerzos en desarrollar este tipo de aplicaciones, aunque se maneja la idea de instalar un Laboratorio Tecnológico de TV Digital Interactiva (iLab) con miras a la instalación de un Polo Tecnológico en nuestro país.

Motivación

TvDTI es un proyecto que intenta adentrarse en las tecnologías que hacen a la televisión digital terrestre interactiva, enfocándose en brindar a través de ellas servicios ciudadanos.

Para el año 2015 se estima que se producirá el conocido “apagón analógico” en el Uruguay. Ante este panorama, es válido decir que no resta mucho tiempo para que se abra paso a las aplicaciones interactivas en el Uruguay, y por lo tanto el proyecto en cuestión sería uno de los pioneros en abordar las áreas de conocimiento relacionadas a la televisión interactiva.

Adicionalmente, el estudio de las tecnologías relacionadas a la TDTi es valioso en el sentido de que habilita la creación de aplicaciones interactivas relevantes en el plano social, profundizando en los campos del e-learning (Información educativa), e-health (Noticias y educación para la salud), e-government (Información local en conjunto con noticias nacionales y globales, alertas de emergencias locales, información sobre trámites burocráticos), entre otros.

Objetivos

El objetivo general del proyecto consiste en la contribución al desarrollo de aplicaciones interactivas para TDT (Televisión Digital Terrestre) destinadas a proporcionar un servicio

ciudadano. Para ello se analizan las tecnologías involucradas y se desarrolla una aplicación estándar de carácter comunitario.

Dentro de los objetivos específicos del proyecto se encuentran los siguientes:

- Establecimiento de un ambiente de desarrollo (herramienta de desarrollo Java, emulador MHP) para aplicaciones interactivas de televisión digital.
- Implementación de una aplicación representativa del poder del estándar MHP dentro del sector de servicios ciudadanos, sin la necesidad de utilizar un canal de retorno que envíe información desde el televíidente a la emisora correspondiente.
- Determinación de la configuración requerida del lado de la emisora para un correcto funcionamiento de la aplicación.
- Estudio de las diferentes posibilidades que existen para implementar un canal de retorno en el contexto de la TDT, escogiendo uno para la realización de encuestas a través de este medio, aplicación de índole comunitario.
- Análisis de la posibilidad de testear la aplicación en el contexto real de funcionamiento, multiplexando la aplicación y el flujo de video/audio de una emisora, y recibiendo el mismo en un decodificador MHP (STB, set-top box) conectado a un televisor analógico.

Metodología

El proyecto en cuestión abarca tanto tareas técnicas como de gestión de proyectos. La planificación es una de las tareas principales en el proyecto y permite cumplir con los plazos especificados, hacer un seguimiento del cronograma, y detectar desvíos y riesgos.

En lo que concierne al área técnica del proyecto, las tareas se dividen en dos grupos, las tareas de investigación y las tareas de implementación. El primer grupo incluye todas aquellas tareas relacionadas al estudio de normas, manuales, tutoriales, foros, artículos, etc., que permiten comprender los aspectos medulares de la TDTi y generar así el conocimiento requerido para las tareas de implementación. Estas últimas tienen como principal objetivo el desarrollo de una aplicación de carácter comunitario.

Resultados

El proyecto TvDTi generó los siguientes resultados:

- Documento explicativo sobre las tecnologías involucradas en la TDTi (“Guía MHP”).
- Aplicaciones interactivas para TDT de carácter comunitario (“Cartelera de Espectáculos” y “Encuesta”).

En cuanto al alcance de objetivos, el grado de cumplimiento de los mismos es muy satisfactorio.

Conclusiones

A lo largo del proyecto TvDTi se encontraron algunas limitaciones tales como:

- Fuentes de información insuficientes.
- Falta de recursos apropiados para el desarrollo y testeo de aplicaciones MHP.

Por otro lado, durante el transcurso del proyecto se identificaron aquellos factores que se consideran necesarios para implantar la TDTi de manera exitosa en el Uruguay:

- Desarrollo de políticas públicas.
- Desarrollo de una “Killer application”.
- Elección de un estándar abierto de interactividad.
- Capacitación de recursos humanos.

Abstract

Introduction

Nowadays, television is the mean of communication that has gained more users since it was conceived. Even though we can attribute this to the grade of entertainment it provides, its low cost is also a special feature that attracts consumers.

Today's television technology is defined as analog, but this technology is about to change with the "analog blackout", and the digital television will be introduced with a whole variety of new services such as interactivity. Interactivity will not only allow viewers to use the TV in the traditional way, but they will have the possibility to access to a whole new world of services through the TV, using a unique terminal and a remote control.

In this context, the project's main goal is to contribute with the development of interactive applications for DTT (Digital Terrestrial Television) oriented to public services. To achieve this goal, technologies involved in the development of standard interactive public applications will be studied.

Between the technologies involved, special attention will be put on MHP (Multimedia Home Platform). MHP is a platform that tries to standardize all the aspects associated to interactive applications, highly portable and interoperable between different decoders and platforms are some of the main characteristics.

Background

TV interactive application's market is a boom market. However, as it is a very recent technology here in Uruguay interactivity does not seem to be the next boom market yet, and there have not been many investment in this area. Besides, some ideas are being managed, such as installing an Interactive Digital TV Laboratory (iLab) with the aim of the creation of a Technological Pole in Uruguay.

Motivation

TvDTI project does a research on the technologies related with interactive digital terrestrial television, making focus on public applications that provide general interest information.

Uruguay's "analog blackout" is estimated for the year 2015, so there is not much time left to start working in those subjects treated in the project. Besides, this project will be a pioneer in the research of technologies involved in developing interactive applications for digital television in Uruguay.

In addition, TvDTI project will focus on the technologies required for the development of public services, with the aim of improving the life quality of Uruguayan citizens. Applications like t-learning (educative information), t-health (health news and information), t-government (local information, national news, emergency warnings, different government procedures), between others, are enabled.

Objectives

The overall objective of the project consists in contributing with the development of interactive applications for DTT (Digital Terrestrial Television) in the context of public services. To accomplish this objective relevant technologies in this area will be studied.

Specific objectives are also set up:

- Establish a development environment (Java development tool, MHP emulator) for interactive digital TV applications.
- Implement a representative MHP application in the context of public services, without using a return channel that sends information from the viewer to the source of broadcasting.
- Determinate the required configuration at the broadcaster's place in order to enable the transmission of interactive applications.
- Study the different possibilities that exist to implement a return channel in the context of the DTT, choosing one of these to develop a survey application for television (public service application).
- Analyze the possibility of testing the application in the real broadcasting scenario, multiplexing the application with the audio and video stream and then receiving it in a MHP decoder (STB, set-top-box) connected to an analog TV.

Methodology

This project not only covers technical areas, but also project management areas. The project's planning is one of the main tasks, letting achieve deadlines, follow the schedule and detect risks and deviations.

The technical tasks are divided into two groups, the research tasks and the implementation tasks. The first group includes all tasks related to the study of standards, manuals, tutorials, forums, articles, etc., in order to achieve a better comprehension of crucial aspects of the interactive DTT, acquiring the knowhow required for the implementation tasks. These implementation tasks have as main goal the development of an interactive application for DTT in the context of public services.

Results

The TvDTi project achieved the following results:

- “MHP Guide”, a document that explains the different technologies related with the interactive DTT.
- Two interactive applications for DTT in the context of public services. (“Cartelera de Espectáculos”, “Encuesta”).

Conclusions

During the development of the project some limitations were found:

- Not enough sources of information
- Lack of appropriated resources for the development and testing of MHP applications.

On the other hand some factors were identified that may be necessary to implant the interactive DTT in a proper way, these are:

- Strong public policies
- The development of an interactive “killer application”

- The election of open interactive standard
- Human resources training

Tabla de Contenido

Índice de tablas	11
Índice de figuras.....	13
Capítulo 1: Introducción	16
1.1. Proyecto TvDTI.....	16
1.2. Objetivos	16
1.3. Antecedentes	18
1.4. Necesidad del producto.....	19
1.5. Justificación de impacto	20
1.6. Grupos de interés.....	20
Capítulo 2: Marco Teórico	22
2.1 Introducción.....	22
2.1.1 Televisión Digital Terrestre - TDT.....	22
2.1.2 TV Digital Interactiva.....	23
2.1.3 MHP	33
2.2 MHP - Conceptos básicos	37
2.2.1 Contexto MHP	37
2.2.2 Xlets	51
2.3 MHP - Guía para el desarrollo de aplicaciones.....	62
2.3.1 Ambiente de desarrollo.....	62
2.3.2 MHP- Recursos escasos de los STB	70
2.3.3 Usabilidad	71
2.3.4 Gráficos en MHP	75
2.3.5 Presentación de texto	88
2.3.6 Interacción con el usuario.....	91
2.3.7 Referenciar contenidos en MHP.....	93
2.3.8 Java Media Framework	97
2.3.9 La API de filtros de sección de DAVIC	102
2.3.10 Seleccionando un nuevo servicio.....	104
2.3.11 Almacenamiento de archivos en un receptor MHP	107
2.3.12 Acceso a los archivos de una aplicación MHP	109
2.4 MHP - Guía para la emisión de aplicaciones	117
2.4.1 Transporte de archivos MHP	117
2.4.2 Señalización de aplicaciones MHP	119
Capítulo 3: Marco Metodológico	129
3.1 Planificación	129
3.2 Metodología de estudio	131
Capítulo 4: Desarrollo	133
4.1 Tarea 1.1 Búsqueda y estudio de tutoriales, manuales, publicaciones, etc.	133
4.2 Tarea 2.1 Selección de posibles aplicaciones	133
4.3 Tarea 2.2 Formulación de una encuesta que permitirá determinar los intereses de la sociedad, con el fin de seleccionar la aplicación.	135
4.4 Tarea 2.3 Realización de la encuesta.....	137
4.5 Tarea 2.4 Procesamiento de los resultados de la encuesta.....	137

4.6	Tarea 2.5 Definición de la aplicación a desarrollar	138
4.7	Tarea 3.1 Diseño de Interfases	139
4.8	Tarea 3.2 Diagramas de secuencia	145
4.9	Tarea 4.1 Elección de la herramienta de desarrollo Java	148
4.10	Tarea 4.4 Diagrama de clases de la aplicación	150
4.11	Tarea 4.6 Desarrollo de código	151
4.12	Tarea 5.2 Descarga e instalación del emulador MHP	172
4.13	Tarea 5.3 Configuración del emulador MHP	173
4.14	Tarea 5.4 Testeo de la aplicación y corrección de errores.....	184
4.15	Tarea 6.1 Búsqueda de información acerca del canal de retorno TDT	187
4.16	Tarea 6.2: Análisis de la posibilidad de introducir un canal de retorno en aplicaciones interactivas para TDT con el objetivo de realizar encuestas.....	196
4.17	Tarea 7.1 y 7.2 Estudio del funcionamiento de un “carrusel de objetos” Establecimiento de los parámetros a tener en cuenta por parte de la emisora para un correcto desempeño de la aplicación ya desarrollada	207
4.18	Tarea 8.1 Contactar autoridades para negociar una posible prueba de la aplicación cuando ésta es transmitida conjuntamente con el flujo de video....	216
4.19	Tarea 8.2 Testeo	219
	Capítulo 5: Resultados	227
	Capítulo 6: Conclusiones	230
	Capítulo 7: Recomendaciones	233
	Glosario y abreviaciones	235
	Bibliografía	239
	Anexos	242
	Anexo 1.	Plan de proyecto
	Anexo 2.	Ánalysis FODA
	Anexo 3.	Informe de justificación de avance de 50%
	Anexo 4.	Informe de justificación de avance de 100%
	Anexo 5.	Actas de reunión
	Anexo 6.	Código desarrollado
	Anexo 7.	Generador de contenidos – Cartelera de Espectáculos
	Anexo 8.	Manual de uso del iLAB
	Anexo 9.	Proyecto Sala+. Acta de reunión nº 1, 18 de diciembre de 2008
	Anexo 10.	Productos para la instalación de un ambiente de desarrollo MHP

Índice de tablas

Tabla 2.1:	Aplicaciones y sus respectivos niveles de interactividad	26
Tabla 2.2:	Penetración de MHP en Europa.....	30
Tabla 2.3:	Paquetes de la API JavaTV	34
Tabla 2.4:	Modulaciones DVB	48
Tabla 2.5:	Señalización del estado del Xlet a través de la AIT	59
Tabla 2.6:	Contenido de un TS y cómo generararlo	67
Tabla 2.7:	Eventos relacionados al control remoto.....	91
Tabla 2.8:	Eventos de un player.....	101
Tabla 2.9:	Estructura del Transport Protocol Descriptor.....	118
Tabla 2.10:	Estructura del “Selector”	118
Tabla 2.11:	Estructura de una AIT.....	120
Tabla 2.12:	Descriptors de la AIT	121
Tabla 2.13:	Estructura del Application Descriptor	122
Tabla 2.14:	Estructura del Application name descriptor.....	123
Tabla 2.15:	Estructura del Application icons descriptor.....	123
Tabla 2.16:	Estructura del DVB-J application descriptor.....	124
Tabla 2.17:	Estructura del DVB-J application location descriptor	125
Tabla 2.18:	Estructura del transport protocol descriptor	125
Tabla 2.19:	Estructura del External application authorization descriptor..	126
Tabla 2.20:	Estructura del Prefetch descriptor	127
Tabla 2.21:	Estructura del DII location descriptor	127
Tabla 4.1:	Esquema de la encuesta	136
Tabla 4.2:	Resultados de la encuesta	137
Tabla 4.3:	archivo /settings.txt del emulador XleTView	175

Tabla 4.4:	Parámetros de configuración la interfase del XleTView.....	176
Tabla 4.5:	Control remoto y teclado de la PC	180
Tabla 4.6:	Casos de testeo de la aplicación Cartelera	186

Índice de figuras

Figura 1.1: Contexto de trabajo del proyecto TvDTI	18
Figura 2.1: Mercado vertical de TvDTI	27
Figura 2.2: Mercado horizontal de TvDTI	28
Figura 2.3: Codificador MPEG-2	37
Figura 2.4: Transport Stream y servicios.....	39
Figura 2.5: Nomenclatura DTV.....	41
Figura 2.6: Componentes para la emisión de aplicaciones interactivas.....	43
Figura 2.7: Arquitectura extremo-extremo de la cadena MHP	44
Figura 2.8: Equipamiento para la transmisión de la señal de broadcast....	46
Figura 2.9: Estados de un Xlet.....	54
Figura 2.10: Función del Xlet context A.....	55
Figura 2.11: Función del Xlet context B	56
Figura 2.12: Función del Xlet context C	56
Figura 2.13: Emisión de un TS.....	63
Figura 2.14: Proceso de generación de un TS	65
Figura 2.15: Equipamiento de broadcasting MHP.....	67
Figura 2.16: Testeo en un STB de desarrollo	69
Figura 2.17: Testeo local a través de un emulador MHP	69
Figura 2.18: Comando a distancia o control remoto.	75
Figura 2.19: Relación de aspecto de una pantalla	76
Figura 2.20: Capas de imagen en los dispositivos MHP	78
Figura 2.21: HScreen, HDevices	79
Figura 2.22: Coordenadas de posición en una pantalla	84
Figura 2.23: Frame, HScene, HContainer, HComponent	85

Figura 2.24: Gama de colores soportada por los receptores MHP.....	88
Figura 2.25: Despliegue de una misma fuente en diferentes STBs	91
Figura 2.26: Servicios de TV digital.....	104
Figura 2.27: Classloaders.....	116
Figura 4.1: Gráfica correspondiente a los resultados de la encuesta.....	138
Figura 4.2: “Cartelera de Espectáculos”, Pantalla 1	140
Figura 4.3: Cartelera de Espectáculos”, Pantalla 1.1.....	141
Figura 4.4: “Cartelera de Espectáculos”, Pantalla 1.1.1	142
Figura 4.5: “Cartelera de Espectáculos”, Pantalla 1.1.2.....	143
Figura 4.6: “Cartelera de Espectáculos”, Pantalla 1.2.....	144
Figura 4.7: “Cartelera de Espectáculos”, Estructura de navegación	145
Figura 4.8: Diagrama de flujo de ejecución de la aplicación Cartelera	147
Figura 4.9: Tiempo de arranque de una aplicación.....	148
Figura 4.10: Diagrama de clases de la aplicación Cartelera	150
Figura 4.11: Pantalla “Cartelera de Espectáculos”	152
Figura 4.12: Pantalla “Cartelera de cines”	153
Figura 4.13: Pantalla “Cines – Salas y Horarios”	154
Figura 4.14: Pantalla “Cine – Resúmenes de Películas”	155
Figura 4.15: Pantalla “Cartelera de Teatros”	156
Figura 4.16: Estructura de directorios de la aplicación.....	162
Figura 4.17: Ejecución de aplicaciones en el XleTView	181
Figura 4.18: Registro de aplicaciones en el XleTView	182
Figura 4.19: Diagrama de flujo_ establecimiento de un canal de retorno....	192
Figura 4.20: Pantalla “Encuesta”	197
Figura 4.21: Pantalla “Encuesta Completada”	198
Figura 4.22: Pantalla “Error de conexión”	199

Figura 4.23: Estructura de directorios de la aplicación Encuesta.....	201
Figura 4.24: Representación lógica del carrusel de objetos.....	208
Figura 4.25: Optimización del carrusel por agrupación en módulos	213
Figura 4.26: Agrupación en módulos del proyecto “Cartelera”.....	215
Figura 4.27: Agrupación en módulos del proyecto “Encuesta”	215
Figura 4.28: Arquitectura del iLAB	221
Figura 4.29: Reporte entregado por el iLAB	223
Figura 4.30: Batería de pruebas del iLAB para los receptores MHP	223
Figura 4.31: Batería de pruebas del iLAB para las aplicaciones MHP	224
Figura 4.32: Interfase gráfica del iLAB.....	225

Capítulo 1: Introducción

1.1. Proyecto TvDTi

El presente proyecto, “Implementación de una aplicación interactiva para Televisión Digital Terrestre” (TvDTi), consiste en el entendimiento del contexto en el que se desarrolla la televisión digital, más específicamente la terrestre, para luego desarrollar una aplicación interactiva, y establecer las principales condiciones, necesarias para el correcto funcionamiento de la misma. En otras palabras, profundiza en el rol del desarrollador de aplicaciones, sin perder de vista aquellas consideraciones que deben hacerse por parte de la emisora.

El producto final consiste en una aplicación interactiva de televisión digital terrestre. La misma tiene como objetivo fundamental brindar información de carácter comunitario (nuevas legislaciones, servicios públicos, intereses sociales, etc.) a los consumidores de TDT. Se llevó a cabo una encuesta en la que se identificaron los servicios de carácter comunitario atractivos para la sociedad. La misma reflejó el interés de los encuestados en conocer la cartelera de cines y teatros, que podrá ser consultada a través del televisor.

Para que dicho producto sea aplicable a la futura TDT en Uruguay, la aplicación fue desarrollada según el estándar MHP (Multimedia Home Platform) del consorcio DVB. Siendo DVB-T la norma adoptada por Uruguay en el 2007.

Con la idea de continuar con el desarrollo de aplicaciones interactivas, este proyecto servirá como base para futuros desarrollos, no sólo de aplicaciones interactivas del tipo “servicio ciudadano”, sino también de entretenimiento, comerciales, etc.

1.2. Objetivos

Objetivo general del proyecto

Contribuir con el desarrollo de aplicaciones interactivas para TDT destinadas a proporcionar un servicio ciudadano. Para ello se analizarán las tecnologías involucradas y se desarrollará una aplicación estándar de carácter comunitario.

Objetivo específico del proyecto

- Establecer un ambiente de desarrollo para aplicaciones interactivas de televisión digital. El mismo consiste en:
 - Herramienta de desarrollo Java: Existen herramientas Java específicas para el desarrollo de aplicaciones MHP que permiten generar el código a través del diseño gráfico de interfaces, menús, etc. Sin embargo, también es posible hacer uso de herramientas Java genéricas, pero éstas requieren un mayor conocimiento del estándar MHP y de las clases asociadas al mismo, ya que se trabaja directamente sobre código Java.
 - Emulador MHP: éste tiene como principal objetivo testear el desempeño de la aplicación desarrollada de manera local en una PC.
- Implementación de una aplicación representativa de las potencialidades del estándar MHP dentro del sector de servicios ciudadanos, sin la necesidad de utilizar un canal de retorno que envíe información desde el televidente a la emisora correspondiente. La misma será testeada con un emulador y no tendrá en cuenta aquellos aspectos a ser considerados por la emisora de TDT al transmitirla junto al flujo de video/audio.
- Determinación de la configuración requerida del lado de la emisora para un correcto funcionamiento de la aplicación. Con el objetivo de transmitir aplicaciones junto al tradicional flujo de video las emisoras deben realizar algunos cambios en su equipamiento de transmisión. Esto implica la adquisición de un software llamado “generador de carrusel de objetos”. El mismo tiene las siguientes funciones: físicamente contiene la información de la aplicación, permite al administrador controlar los parámetros de transmisión de la misma (ancho de banda, frecuencia de envío, prioridad, etc.), provee una interfase al multiplexor, entre otras.
- Estudio de las diferentes posibilidades que existen para implementar un canal de retorno en el contexto de la TDT. Finalmente se seleccionará el más apropiado para la realización de encuestas a través de este medio, aplicación de índole comunitario.

- Análisis de la posibilidad de testear la aplicación en el contexto real de funcionamiento. Para esto se deberá multiplexar la aplicación y el flujo de video/audio de una emisora, y el mismo debe ser recibido por un decodificador MHP (STB, set-top box) conectado a un televisor analógico.

En la figura 1.1 se representa el contexto de trabajo del proyecto. A grandes rasgos éste consistirá en desarrollar la aplicación que será introducida por la emisora en lo que se denomina “Carrusel de objetos”, generado por un software bajo completo control de la misma y cuyas características principales serán estudiadas a lo largo del proyecto. Luego será plena responsabilidad de la emisora la multiplexación de la aplicación con el flujo de video/audio, y la transmisión de la señal digital. Al llegar al televidente, el decodificador MHP traducirá la información recibida para que ésta pueda ser desplegada en pantalla. Finalmente, existe la posibilidad de que el televidente envíe información a la emisora de TV a través del denominado “canal de retorno”, éste no será implementado pero se estudiarán los diferentes tipos existentes.

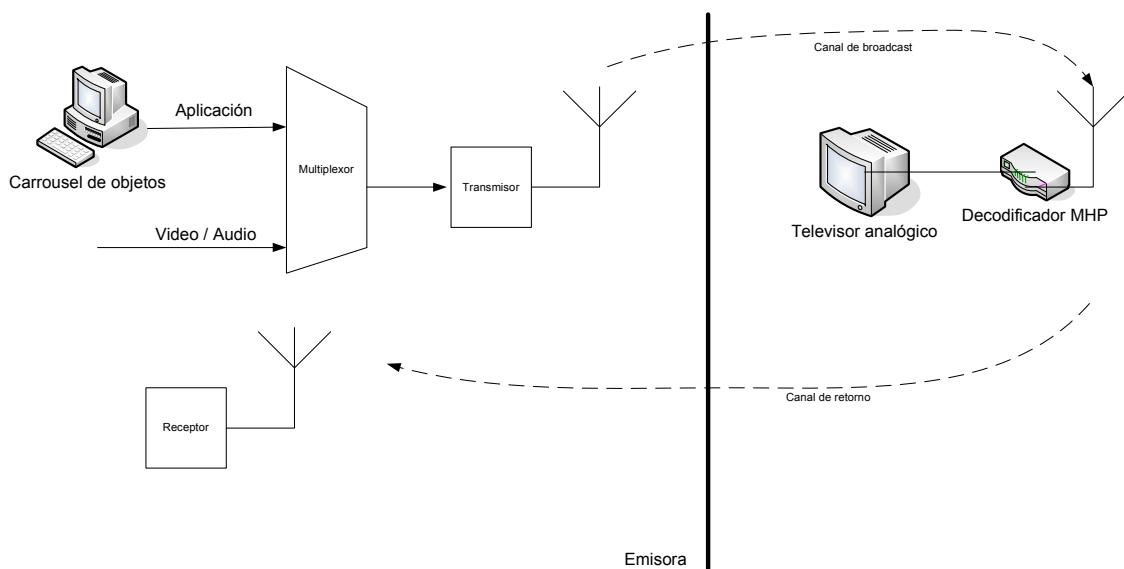


Figura 1.1: Contexto de trabajo del proyecto TvDTI

1.3. Antecedentes

A finales de agosto de 2007, el gobierno uruguayo decidió aprobar la norma europea de televisión digital, DVB, y así dar inicio a la transición desde el actual sistema analógico al digital. Uruguay fue uno de los primeros países en América Latina en tomar esta decisión, aunque luego Argentina y Brasil optaron por la norma estadounidense y japonesa, respectivamente.

La preferencia de Uruguay por esta norma no sólo se debió a las buenas relaciones generadas entre el gobierno uruguayo y el consorcio DVB, sino que también esta norma tiene una ventaja en su aplicación: los operadores de cable ya han adoptado la norma europea para digitalización de su señal, y esto significa que ya se tiene un mínimo conocimiento del sistema aplicable a la TV abierta, similar al de TV por cable.

Al día de hoy, la norma DVB fue adoptada por más de cien países de Europa, Asia, África y Oceanía, lo que indica que se está convirtiendo en una norma mundial, como sucedió con la norma GSM en el ámbito de las comunicaciones móviles.

El conjunto de normas DVB abarca todos los aspectos de la TV digital, desde la transmisión hasta la estructuración de interfaces, el acceso condicional y la interactividad para datos, audio y video digitales, comprendiendo todas las modalidades de la TV Digital: terrestre (DVB-T), satelital (DVB-S), por cable (DVB-C) y móvil (DVB-H).

Tras la adopción de la norma europea por parte del gobierno uruguayo, en varios espacios públicos de Montevideo se realizaron jornadas para demostrar las virtudes de la nueva tecnología, entre ellas la interactividad. Y en el 2008 se comenzó a hablar de la instalación de un Polo Tecnológico de Televisión Digital Interactiva en el país.

Actualmente, un gran número de empresas/firmas del sector tecnológico y el gobierno español participaron de reuniones con el fin de lograr acuerdos de colaboración para el desarrollo de proyectos conjuntos de investigación, desarrollo e innovación. El polo tecnológico estaría compuesto principalmente por:

- Universidades (órgano impulsor de la infraestructura de conocimiento a través de cursos, intercambio de docentes, proyectos, etc.)
- Industria de Software
- Industria Audiovisual
- Regulador (encargado de homologar equipamiento)
- Radiodifusores (órgano que aporta entornos de prueba de laboratorio y pre-comerciales)

1.4. Necesidad del producto

Actualmente la televisión terrestre se define como analógica. En el Uruguay, alrededor del 2015, ocurrirá el llamado “apagón analógico”, y la televisión abierta será netamente digital, dando lugar a las aplicaciones interactivas de televisión.

Dado que es una tecnología muy reciente, el Uruguay todavía no se ha embarcado en proyectos de interactividad de gran envergadura, pero sin lugar a dudas, representa grandes oportunidades para la industria de desarrollo de aplicaciones interactivas. En particular se podrán desarrollar aplicaciones que tengan como objetivo mantener informada a la población respecto a nuevas legislaciones, servicios públicos, intereses sociales, etc.

1.5. Justificación de impacto

Actualmente la televisión es un medio masivo de comunicación fácilmente accesible por todos los estratos de la sociedad mundial. Hoy en día en el Uruguay la misma es analógica. Próximamente la tecnología analógica será obsoleta y la televisión digital pasará a ser la nueva protagonista, produciéndose el llamado “apagón analógico”, que implicará la migración de la televisión por aire a la tecnología digital. Este cambio habilitará la interactividad, a modo de diálogo, entre el televisor y el usuario. La principal ventaja social de esta manera de acceder a la información es que será masiva, y no hará falta saber acerca del manejo de nuevo hardware como teclados, mouse, etc. El control remoto permitirá la interacción de manera simple, debido a que la sociedad ya se encuentra familiarizada con este dispositivo. Esto habilita la aparición de nuevos servicios ciudadanos a través de la creación de aplicaciones interactivas.

1.6. Grupos de interés

- Televidentes en general
- Operadores de TV
- Gobierno
- Industria electrónica
- Proveedores de contenido
- Empresas desarrolladoras de software

- Universidad de Montevideo

Capítulo 2: Marco Teórico

2.1 Introducción

2.1.1 Televisión Digital Terrestre - TDT

La TV Digital se basa en la conversión de la señal analógica de la televisión convencional, en una secuencia de bits que representa la unidad mínima de información. Por tanto, la Televisión Digital Terrestre (TDT) es el resultado de la aplicación de la tecnología digital a la señal de televisión, para luego transmitirla por medio de ondas hercianas terrestres, es decir, aquellas que se transmiten por la atmósfera sin necesidad de cable o satélite y se reciben por medio de antenas UHF convencionales.

La televisión es un medio masivo de comunicación hoy en día fácilmente accesible por todos los estratos de la sociedad. Actualmente la misma es analógica debido a que ésta es la tecnología utilizada para transmitir la señal que será desplegada en pantalla. En un futuro cercano esta tecnología será obsoleta y la televisión digital será la nueva protagonista, presentando las siguientes ventajas:

- Existencia de más canales, debido al eficiente uso del espectro.
- Alta calidad de imagen y sonido.
- Nuevos servicios, por ejemplo la interactividad que se explicará más adelante.

Esta nueva tecnología comenzará a introducirse poco a poco en el mercado mundial, pero existe una fecha límite establecida por cada país en la que se producirá el “apagón analógico”. Este hecho implicará el pasaje de la televisión vía aire a tecnología digital, y los clientes se verán obligados a adaptar sus receptores ya sea a través de un dispositivo externo conectado directamente al televisor analógico denominado decodificador, o a través de un televisor que disponga de un receptor integrado.

2.1.2 TV Digital Interactiva

Anteriormente, con el uso de la televisión analógica, para introducir nuevas funcionalidades a la misma era necesario adquirir hardware específico. Pero hoy en día, la televisión digital permite lanzar nuevas funcionalidades al mercado de manera rápida. Esto se debe a que sólo requiere de la adquisición de nuevo software que será ejecutado en un “motor” ya existente en el televisor digital.

La funcionalidad más destacada que se introduce en la nueva y emergente televisión digital es la interactividad. Este concepto refiere a cualquiera acción que permita a los usuarios interactuar con el contenido que se visualiza en pantalla del televisor. Para que la interactividad sea considerada pura, el usuario debe ser capaz de alterar el contenido audiovisual, o enviar información a la fuente de difusión del mismo, a través de un canal de retorno, pudiendo ser telefónico, SMS o cable, entre otros. Las aplicaciones interactivas, de esta forma, aumentan las funcionalidades del servicio de radiodifusión televisivo ofrecido al usuario, que percibe una mejora evidente en la personalización de la información a la que puede acceder.

La gran ventaja de la televisión digital interactiva reside en el hecho de que los servicios corren en un ambiente completamente controlado, y vía broadcast llega a todos los televidentes, sin necesidad de aumentar las capacidades de la red o de los servidores.

2.1.2.1 **Tipos de aplicaciones interactivas**

De acuerdo a la relación de las aplicaciones con la programación de la emisora, existen dos tipos:

- Dependientes de la programación de la emisora

Este grupo de aplicaciones puede estar destinado a promocionar un programa de televisión, a fomentar la participación de los televidentes durante el curso del mismo a través de cuestionarios, o simplemente brindar información extra sobre lo que se está visualizando.

- Independientes de la programación de la emisora

Estas aplicaciones ofrecen servicios de información general, comunicados, entretenimiento, selección de video-on-demand, T-commerce (análogo a E-commerce

pero en el contexto de la TV), T-Government (comunicados de gobierno), T-Learning (educación vía TV) y T-health (servicios de salud).

2.1.2.2 Algunos ejemplos de aplicaciones interactivas

A continuación se describen brevemente algunas de las posibles aplicaciones interactivas a desarrollar:

- *EPG (Electronic Programme Guide - Guía electrónica de programación)*

Ésta es una guía cuyo principal objetivo es listar la programación de los canales, y generalmente cubre los siguientes 7 días.

- *Pronóstico del tiempo*

Como lo refleja su nombre, esta aplicación está destinada a mostrar el pronóstico del tiempo. La interactividad de la misma recae sobre el hecho de que es posible consultar el estado del tiempo para diferentes regiones.

- *Servicio de tránsito*

Brinda información acerca del estado del tráfico en ciertos puntos del país cuando éste se encuentra obstruido por embotellamientos, construcciones, bloqueos, etc.

Por otro lado puede incluir información acerca del servicio de transporte urbano, mostrando el recorrido de las diferentes líneas de transporte, paradas de ómnibus, estaciones de subtes, etc.

- *T-chat*

Este tipo de aplicación consiste en un chat implementado directamente en el STB. Comandado por el control remoto, envía y recibe información. Este tipo de aplicación es del tipo bi-direccional, recibe información personalizada para cada usuario y tiene un canal de retorno desde el televíidente hacia los servidores de la aplicación.

- *T-Games*

Este tipo de aplicación consiste en juegos que pueden utilizar o no un canal de retorno. Por ejemplo el “Tetris” y el “Solitario” no exigen de un canal de retorno para su ejecución.

Generalmente vienen implementados en el STB, por lo que dependen del fabricante del mismo. Sin embargo, aplicaciones como Quizes, hacen uso de un canal de retorno, a través del cual se envía la respuesta escogida y en base a ella se informa al jugador sobre su acierto o no.

- *T-commerce*

Éstas pueden simplemente promocionar un producto o permitir la realización de compras a través de la TV.

- *T-Government*

Permite publicar información de interés general a través de un medio tan masivo como es la televisión. También permite la realización de encuestas relevantes en el ámbito político o gubernamental.

- *T-Care*

Así se denomina a aquellas aplicaciones que de alguna manera actúan como servicio de acompañantes, monitoreando constantemente a pacientes, pidiendo información al mismo de su presión, temperatura, etc. Estas aplicaciones son claramente bidireccionales.

2.1.2.3 Niveles de interactividad

La interactividad puede ser clasificada en tres niveles:

- Broadcast

No requiere de un canal de retorno, la interactividad se da de manera local, permitiendo al consumidor consultar información que ya se encuentra almacenada en su STB.

- Unidireccional

Este tipo de interactividad aparece mayoritariamente en aplicaciones de votación o cuestionarios en donde la información viaja en una sola dirección. Requiere de un canal de retorno desde el televíidente hasta el servidor de la aplicación, pero no esperan una respuesta del mismo.

- Bi-Direccional

Aplicaciones como T-chat, T-mail, o navegación en Internet son del tipo bi-direccional. Requieren del tránsito de información en ambas direcciones. El usuario envía una petición y luego le llega información respondiendo a la misma.

Aplicación	Nivel de interactividad		
	Broadcast	Unidireccional	Bi-direccional
GEP	x		
Pronóstico del tiempo	x		
Servicio de tránsito	x		
T-chat			x
T-game	x	x	x
T-commerce	x	x	x
T-Government	x	x	
T-Care		x	x

Tabla 2.1: Aplicaciones y sus respectivos niveles de interactividad

2.1.2.4 La necesidad de estandarizar la interactividad

El consorcio DVB (Digital Video Broadcasting) se encarga de la estandarización de los diferentes aspectos que involucran al broadcasting de la televisión digital. Han tratado temas tales como:

- Transmisión de señales de TV digital sobre cable.
- Transmisión de señales de TV digital satelital.
- Transmisión de señales de TV digital terrestre.
- Codificación para llevar a cabo la digitalización.

Años atrás DVB se embarcó en el proyecto de obtener un estándar abierto para la TV digital interactiva y de esta forma competir con los sistemas propietarios que en ese momento dominaban el mercado (OpenTV, NDS, Microsoft, entre otras), y cuyo nivel de independencia era tal que no permitía la evolución de la nueva tecnología.

Sistemas propietarios - Mercado Vertical

Si los sistemas propietarios dominaran el mercado de la TV digital, la estructura del mismo sería netamente vertical como se muestra en la siguiente figura:

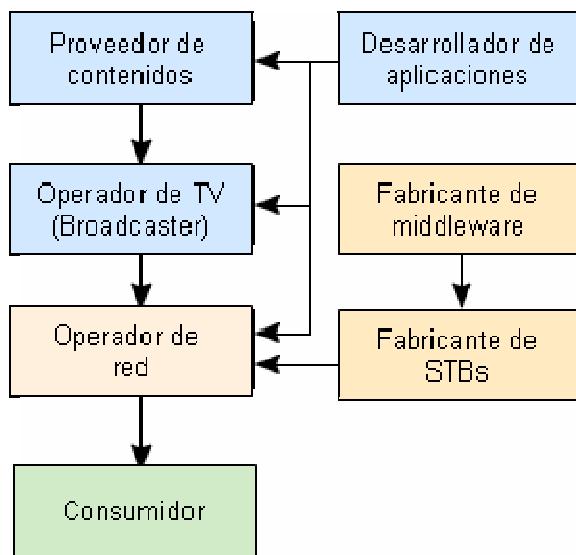


Figura 2.1: Mercado vertical de TvDTI

En un esquema de mercado vertical los Operadores de Televisión digital (Broadcasters) proveen de manera exclusiva a sus usuarios los decodificadores. Estos últimos deben adaptarse al servicio que se desea brindar, por lo que los fabricantes se ven obligados a desarrollar receptores con las características impuestas por los operadores de TV. A su vez los operadores controlan qué tipo de aplicaciones serán ejecutadas en el receptor del usuario por lo que deciden el tipo de middleware que se va a utilizar en los mismos. Por todo esto, la mayor carga recae sobre los operadores. A cambio de un control total, éstos deben suministrar los receptores a sus clientes, darles soporte y mantenimiento.

En resumen, un mercado vertical no es la mejor opción, no permite una libre expansión del mismo, y sobrecarga al operador de TV.

Estandarización - Mercado horizontal

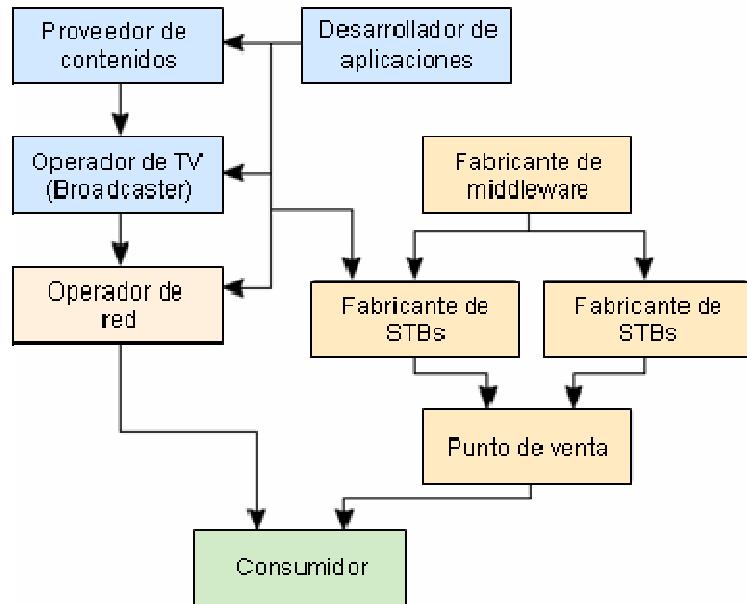


Figura 2.2: Mercado horizontal de TvDTI

En el caso del mercado horizontal el consumidor tiene la posibilidad de comprar el receptor en cualquier tienda de electrónica sin depender del operador al que desee suscribirse. Adicionalmente, dado que el operador no controla las especificaciones para los fabricantes, éstos se encuentran en una mejor posición, pudiendo brindar valor agregado a sus productos, generando así una justa competencia, y proporcionando beneficios para el consumidor. Por otro lado, gracias a la estandarización del middleware, los desarrolladores de contenidos pueden proporcionar sus contenidos a todos los operadores sin incurrir en personalizaciones.

2.1.2.5 MHP

Luego de la implantación de la televisión digital, cuyos esfuerzos estuvieron dirigidos al desarrollo de técnicas de codificación de video, modulación, transmisión, etc., aparece el concepto de interactividad. Es así que comenzaron a surgir en el mercado diferentes APIs para la implementación de aplicaciones, todas propietarias e incompatibles entre sí. Por esta razón, en 1997 el proyecto DVB (Digital Video Broadcasting) comienza a fomentar la estandarización de aplicaciones interactivas a través de la creación de una única API abierta.

Si bien en los últimos años han habido intentos de difundir soluciones abiertas de middleware para TV digital, el único que ha tenido aceptación ha sido MHP. Esto se debe a que su incorporación en el mercado no intentó ser un reemplazo a las soluciones propietarias ya existentes, sino que su propósito es proporcionar una plataforma para la TV digital y detener la fragmentación del mercado. No es una lucha entre abierto y propietario, o nuevo y viejo, sólo se trata de crear un gran mercado y no muchos pequeños.

MHP proporciona entonces las siguientes características con el fin de establecerse como una plataforma común:

- Estándar abierto para la implementación de middleware.
- La especificación puede descargarse de manera gratuita, los únicos costos corresponden a pequeños impuestos por compilaciones, testeos y algunos licenciamientos necesarios.
- Propone la implementación de aplicaciones interactivas en lenguaje Java o HTML, por lo que no depende de ninguna plataforma de hardware ni de ningún sistema operativo en especial. Por otro lado facilita la tarea de los desarrolladores que ya se encuentran familiarizados con estos lenguajes de programación.

En este contexto es que entra MHP con la intención de cambiar el panorama. En su forma más simple MHP es un conjunto de APIs de Java que permite a los programadores escribir aplicaciones interoperables. Estas aplicaciones serán incluidas en el flujo MPEG-2 que compone la señal digital de TV y serán interpretadas por los receptores MHP que se encargarán de ejecutar las aplicaciones.

2.1.2.6 GEM

MHP ha definido GEM (Globally Executable MHP) para permitir a otros estándares de TV digital construir aplicaciones de iTV compatibles con MHP.

MHP es un estándar de DVB, y por lo tanto ha sido utilizado en redes DVB (generalmente en Europa y muchos países de Asia) pero todavía no ha sido muy difundido en redes que siguen el estándar americano para la transmisión digital. Por esta razón fue que CableLabs propuso a DVB remover aquellos aspectos que fueran específicos de MHP y

que funcionan únicamente con redes DVB. Luego de mucho trabajo, DVB publicó la especificación GEM, basada en la versión 1.0.2 de MHP que removía aquellos elementos que eran específicos de los estándares DVB. GEM no es una especificación por sí sola sino que está pensada para trabajar en conjunto con otros estándares como ATSC, ARIB y CableLabs.

Con el uso de GEM, MHP se irá convirtiendo en una plataforma global para las aplicaciones de televisión interactiva.

2.1.2.7 La TV digital en el mundo

La introducción de la TV digital no es una simple modernización del sistema de televisión actual, sino que implica la generación de nuevos servicios. Es así que la implantación de la TV digital irá de la mano, sin lugar a dudas, de la TV interactiva.

TV digital en Europa

Tal como se mencionó anteriormente, la introducción de la TV digital causará la aparición de aplicaciones interactivas. La tabla a continuación muestra el despliegue de interactividad MHP satelital, por cable y terrestre, en algunos países, la mayoría de ellos europeos, en marzo de 2006 (“Analysis of the current MHP situation - Version 3”, MHP Knowledge Database, Marzo 2006):

País	DVB-S	DVB-C	DVB-T
Alemania	●	●	●
Bélgica		●	
Austria	●	●	●
Dinamarca	●		●
Noruega	●		●
Finlandia	●	●	●
Suecia	●	●	●
Italia	●		●
Francia			●
España			●
Hungría			●
Australia			●
Corea	●		

- Aplicaciones MHP en funcionamiento
- Aplicaciones MHP en desarrollo

Tabla 2.2: Penetración de MHP en Europa

Se estima que para el 2012 la TDT tendrá una penetración casi total en Europa debido a la implantación de la red digital y a la reducción de precios de los receptores digitales.

Vale la pena destacar el gran interés de varias industrias del mercado por desarrollar esta tecnología. Algunos colaboradores son:

- Panasonic: Fabricó el primer decodificador certificado bajo la norma MHP.
- tComLabs: Ofrece servicios de testeo y consultoría en MHP.
- Philips: Miembro de DVB, interesado en la estandarización e interoperabilidad.

TV digital en América Latina

En América Latina la TV digital también está comenzando a tomar importancia. Diversos países latinoamericanos ya han escogido la norma y próximamente comenzarán a implementar la TDT. La situación de la TV digital puede resumirse en los siguientes puntos:

- **Argentina:** Optó por el estándar ATSC para la TDT abierta. Sin embargo desde el 2006 existe un sistema de TV Digital Terrestre por suscripción, que utiliza el estándar DVB-T, en la Ciudad de Buenos Aires y el Gran Buenos Aires.
- **Brasil:** En el 2006 optó por la versión modificada de la norma japonesa ISDB y ha comenzado rápidamente a reemplazar la TV analógica por la TDT. Actualmente ya cuenta con servicios comerciales funcionando.
- **Colombia:** Optó por la norma europea. La primera señal digital al aire será de televisión abierta y fue mostrada a la prensa en febrero de 2009. Se prevé iniciar el proceso de implantación de la TDT en el año 2009. Se estima que en 24 meses el 25% de los colombianos accederá a la señal digital de los canales privados. El apagón analógico está programado para el año 2019.
- **México:** En el 2004 optó por la norma ATSC. En algunas ciudades como Ciudad de México, Monterrey, Guadalajara y Tijuana 3 cadenas televisivas (Multimedios, Televisa y TV Azteca) ya han comenzado a emitir señales digitales.
- **Uruguay:** En el 2007 optó por el estándar europeo DVB-T. No se ha definido con exactitud la fecha de inicio de las transmisiones ni la del apagón. Actualmente algunas emisoras de manera independiente han hecho pruebas. Uruguay, por su

parte, ha lanzado este año un servicio piloto de televisión digital para teléfonos móviles, proyecto pionero latinoamericano en la materia.

La TV digital está causando tal impacto que en el 2008, la CIESPAL (Centro Internacional de Estudios Superiores de Comunicación para América Latina) propuso trabajar sobre el tema de televisión digital, tendencias y realidades, a fin de proponer nuevos enfoques y prácticas digitales. A partir de esto, en noviembre de 2008, se desarrolló un seminario en Quito – Ecuador denominado “Del apagón analógico a la televisión digital”. En el mismo se analizaron las experiencias positivas y las limitaciones de países latinoamericanos y caribeños que han presentado la tecnología digital de televisión, como Argentina, Brasil, México y Uruguay. Los objetivos del seminario fueron:

- Conocer las tendencias y sistemas de adopción de la televisión digital en el continente.
- Analizar las experiencias positivas y las limitaciones identificadas en los países de América Latina y el Caribe que han incorporado la tecnología digital en televisión.
- Crear espacios de discusión que articulen a los actores en los sectores públicos y privados del continente, para comprender el impacto de la televisión digital en los ámbitos social, económico, cultural y político de la sociedad.

Obstáculos para la implantación de la TV digital abierta

El proceso de implantación de la TDT abierta y gratuita en Latinoamérica es lento. Esta lentitud se debe a que el Estado en cada país no está impulsando de manera adecuada esta nueva tecnología. Dado que los televidentes de la TV gratuita son los de menor poder adquisitivo, los anunciantes (financiadores del servicio) no presentan mucho interés en realizar la migración. Por esta razón es que debe ser el Estado quien subvencione la TDT abierta, y para esto, éste debe ser capaz de percibir la importancia de contar con medios de comunicación electrónica actualizados.

2.1.3 **MHP**

DVB MHP es un estándar enfocado en el desarrollo de aplicaciones interactivas para televisión digital. Sobre esta especificación se implementan las aplicaciones de TV interactiva. MHP es una interfase abierta de la organización DVB y define las características básicas con las que deben cumplir los terminales MHP y cómo las aplicaciones pueden hacer uso de ellas.

2.1.3.1 **Variantes del estándar de interactividad DVB - MHP**

Las aplicaciones MHP pueden ser clasificadas en dos grandes grupos: DVB-J y DVB-HTML, siendo hoy en día DVB-J la variante más difundida.

DVB-J

Estas aplicaciones están escritas en Java y hacen uso de la API MHP. El estándar DVB adoptó Java como el lenguaje de las aplicaciones interactivas debido a la madurez del mismo, la calidad y la variedad de herramientas existentes para desarrollar en él.

DVB-HTML

Define la existencia de navegadores integrados en los terminales MHP. DVB-HTML es un lenguaje implementado sobre XML. Sin embargo DVB-HTML no es soportado por todos los terminales MHP existentes en la actualidad.

2.1.3.2 **JavaTV**

Por muchos años Java fue utilizado para añadir funcionalidades avanzadas a las tecnologías de TV digital, pero no como un modelo de aplicación. Finalmente Sun decidió realizar una solución puramente Java, denominada JavaTV. Ésta describe una variedad de conceptos de TV digital como: acceso a información de servicios, selección de un nuevo servicio, carga de archivos insertados en el flujo emitido por la emisora, entre otros, pero no describe una plataforma completa de TV digital como para ser un estándar en sí mismo. Es así que surge el estándar de interactividad MHP, haciendo uso del modelo de aplicación de JavaTV y todas sus APIs.

MHP utiliza el modelo de aplicación definido por JavaTV y a su vez define qué información adicional es imprescindible para que el receptor pueda saber las aplicaciones que tiene disponibles. Notificar al receptor que hay aplicaciones para ser desplegadas es

sólo una fracción de las funciones MHP, una aplicación debe también ser capaz de realizar una serie de operaciones cuando se está ejecutando, y es por esto que MHP incluye APIs de Java para acceder a la información de servicio, dibujar objetos en la pantalla, controlar el audio y el video, comunicarse con servidores remotos, entre otras cosas.

La ventaja de contar con JavaTV es que éste no está particularmente asociado a ningún estándar. MHP de DVB hace uso de JavaTV, pero existen otras familias de estándares, y éstas podrán hacer uso de JavaTV como parte de su especificación. Por lo tanto las aplicaciones desarrolladas sobre esta API podrán ser utilizadas sobre cualquier plataforma de TV digital, lo que posibilita la portabilidad.

API JavaTV

Consiste en un conjunto de paquetes Java y subpaquetes, todos dentro de javax.tv. A continuación se muestra una lista con los paquetes y una pequeña descripción de los mismos.

Paquete	Descripción
javax.tv.xlet	Modelo del ciclo de vida de las aplicaciones.
javax.tv.locator	Provee un mecanismo similar a URL para referenciar los servicios de broadcast.
javax.tv.net	Provee un mecanismo para acceder a los datagramas IP contenidos en el stream de broadcast, y da soporte a los canales de retorno “always-on”.
javax.tv.util	Clases útiles entre las que se incluye la administración de timers y eventos programados.
javax.tv.media	Da soporte al JMF para funcionalidades especiales de TV.
javax.tv.media.protocol	Da soporte al JMF para incorporar los protocolos de streaming.
javax.tv.service	Describe los servicios de la TV digital. Provee un mecanismo para solicitar información de servicios.
javax.tv.service.guide	Da soporte a aplicaciones como EPG, planificación de la programación e información de rating.
javax.tv.service.navigation	Permite navegar entre los servicios.
javax.tv.service.transport	Describe los conceptos relacionados al mecanismo de transporte de la TV digital como TS, redes de broadcast, etc.
javax.tv.service.selection	Describe conceptos relacionados a cómo los servicios son presentados al usuario y cómo se pueden seleccionar nuevos servicios.

Tabla 2.3: Paquetes de la API JavaTV

JavaTV intenta ser una librería de alto nivel. Por esta razón es que el control de recursos debe ser especificado por cada estándar en particular.

2.1.3.3 Nuevas funcionalidades específicas de la TDTI DVB

Dado que JavaTV no conforma un estándar en sí mismo, MHP agregó nuevas funcionalidades relacionadas a diferentes áreas de la TV digital. Éstas se organizan en diferentes APIs que pueden ser categorizadas de la siguiente manera:

- Acceso de bajo nivel a la trama MPEG, para así acceder a los archivos asociados a una aplicación.
- Acceso a la información de transmisión, incluyendo información de cada servicio emitido.
- Control de los diferentes medios de comunicación (audio, video, subtítulos, etc.).
- Interfases gráficas.
- Comunicación con los servidores de back end, a través de un canal de retorno, y otras aplicaciones.
- Acceso al HW de los receptores y a periféricos como smart cards.
- Seguridad.

No todas las aplicaciones hacen uso de todas las interfaces propuestas, pero sí es necesario contemplar todos los casos posibles, para así generar aplicaciones confiables y que puedan interactuar con cualquier tipo de STB.

2.1.3.4 Perfiles MHP

MHP propone tres tipos de perfiles que pueden ser adoptados libremente por los fabricantes de STBs para desarrollar sus productos. Estos perfiles son:

- Enhanced Broadcast Profile: Habilita la interactividad pero de manera local.
- Interactive Broadcast Profile: Habilita la utilización de una canal de retorno.
- Internet Access Profile: Permite el acceso a servicios de Internet.

Para que los fabricantes de dispositivos STB sean capaces de probar su interoperabilidad con otros dispositivos MHP, y demostrar que soportan las especificaciones MHP, existe lo que se denomina “MHP Test Suite”, y la aprobación de este test certifica a los dispositivos y habilita a los mismos a llevar el logo MHP.

2.1.3.5 Xlets

El modelo convencional de aplicaciones de Java no se adecúa de manera perfecta al contexto de la TV digital. Mientras que Java tradicionalmente asume que una única aplicación se está ejecutando en la JVM (teniendo control total sobre su ciclo de vida), los receptores de TV digital están pensados para ejecutar varias aplicaciones a la vez, tal como lo hacen los applets en un navegador web. Así surgen los Xlets. Al igual que los applets, éstos también cuentan con una interfase que permite a una fuente externa, un administrador de aplicaciones por ejemplo, comenzar y terminar la aplicación. Sin embargo la mayor diferencia entre los applets y Xlets, es que los Xlets pueden ser pausados y reiniciados. Estas dos nuevas funciones surgen como consecuencia de las restricciones de hardware, permitiendo liberar recursos mientras una aplicación no se encuentra visible.

2.2 MHP - Conceptos básicos

2.2.1 Contexto MHP

DVB y MHP - Protocolos de transmisión de Broadcast

Existen diversos protocolos involucrados en la transmisión de datos a través de ondas hercianas y así lograr la conocida TDT. A continuación se definen una serie de conceptos básicos que se consideran relevantes:

MPEG-2

MPEG-2 es el grupo de estándares de codificación de audio y vídeo acordado por MPEG más utilizado a nivel internacional. Por lo general es utilizado con el fin de codificar audio y vídeo en señales de transmisión, para televisión digital terrestre, satelital o por cable.

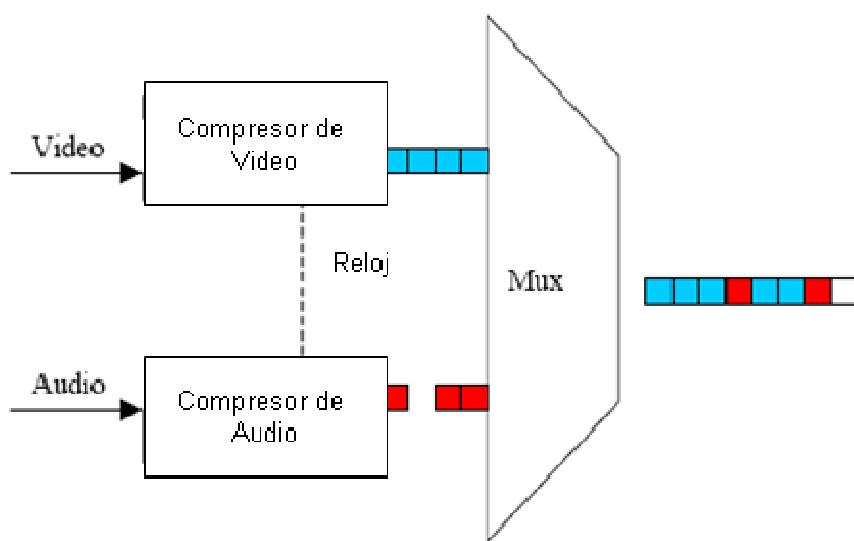


Figura 2.3: Codificador MPEG-2

Se le llama codificador MPEG-2 al dispositivo que recibe por una entrada el video sin comprimir y por otra el audio sin comprimir para luego comprimirlos y multiplexarlos.

Anatomía de la trama MPEG-2

Una señal de TV digital se transmite como una trama MPEG-2, cuya información se encuentra encapsulada en lo que se denomina trama de transporte (TS – Transport

Stream). Cada TS tiene una tasa de datos de hasta 40Mbps en redes cableadas o satelitales y de hasta 25Mbps en redes terrestres.

Cada TS está a su vez constituido por un conjunto de sub tramas denominadas tramas elementales (ES – Elementary Stream). Estos ES pueden contener audio, video u otro tipo de información encapsulada dentro de la misma trama MPEG-2. Para poder identificar los ES se utiliza un identificador de paquete (PID), que es único dentro de cada TS. Cada ES representa un conjunto continuo de cuadros de video o audio y es necesario dividirlos en paquetes para hacer de la multiplexación un proceso más sencillo, por esta razón se generan las tramas elementales paquetizadas (PES).

Ahora, para crear la trama de transporte a partir de las PES es necesario paquetizarlas nuevamente guardando la información en los denominados paquetes de transporte, que tienen un tamaño de 188 bytes. Esta paquetización permite que la trama sea más sensible a técnicas de corrección de errores.

Por lo tanto, es posible concluir que cada TS consiste en un número de tramas de audio y video que son multiplexadas de forma conjunta, y así cada servicio contenido en el TS “viaja” junto a su audio y video codificados según el estándar MPEG-2.

DVB TS - Adicionando información a la trama MPEG-2

UN TS DVB es similar a un TS MPEG-2 pero con algunas características adicionales:

- DVB introduce las tablas llamadas SI (Service Information) dentro del TS. Básicamente la tabla SI funciona como una base de datos que indica qué ES está asociado a qué servicio. De esta forma el receptor consultará la tabla para ver qué ES del TS debe decodificar para obtener el audio, video y datos de cierto servicio. Existen dos tablas fundamentales a la hora de describir un servicio y sus tramas:
 - PMT (Program Map Table): Existe una PMT por cada servicio transportado dentro del TS. Esta tabla indica qué ES son los que arman el servicio. Cada PMT se transporta en un ES pero no tiene un PID fijo.
 - PAT (Program Association table): Esta tabla siempre se transmite en el PID = 0, e indica la ubicación (PID) de las tablas PMT asociadas a cada servicio.

- Si el servicio también lleva aplicaciones MHP se agrega una nueva tabla denominada AIT (Application Information Table). Tabla que será descrita más adelante.

En la siguiente imagen se ilustra la estructura de un TS con sus servicios asociados, en este caso 2.

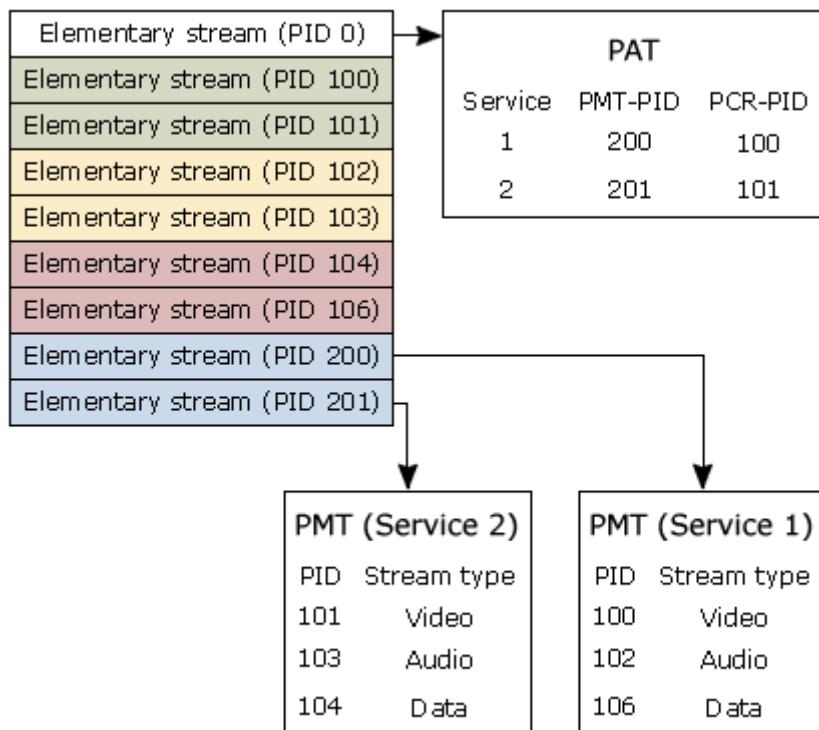


Figura 2.4: Transport Stream y servicios

Nomenclatura DTV

- Multiplex:

En la jerga de la televisión digital los TS se denominan multiplex, ya que están constituidos por un número de servicios multiplexados.

- Servicio:

Cada grupo de ES que componen un canal de TV se denomina servicio. No existe ninguna restricción respecto a la cantidad de ES que conforman un servicio, ya que esto depende de los programas televisivos que proporcione el canal. Por ejemplo, aquellos

programas que se emiten en diferentes lenguajes requieren de un aumento en el número de ES debido a que introducen mayor cantidad de audio.

- Evento:

Cada programa televisivo se conoce como evento, y por lo general un servicio está compuesto de varios programas, o sea varios eventos, transmitidos uno a continuación del otro.

- Bouquet:

Dentro del TS los servicios pueden ser agrupados físicamente o pueden ser agrupados de manera lógica en lo que se denomina un bouquet. Esto resulta útil cuando el número de canales que se quiere transmitir es elevado. Por ejemplo si se tienen muchos canales para transmitir, una de las opciones para que los usuarios accedan a ellos es segmentarlos en paquetes, donde se vende un paquete básico, un paquete de deportes, un paquete de películas, etc. Cada uno de estos paquetes tiene una serie de canales y generalmente no entran en una TS, por lo que resulta práctico asociarles un bouquet a cada paquete.

- Red:

Los sistemas de TV digital también manejan el concepto de red refiriéndose a un conjunto de TS que comparten la información de servicio mencionada anteriormente. Estos TS serán transmitidos por la misma compañía emisora.

Resumiendo:

- Red: Consiste en uno o más TS que son transmitidos por la misma entidad emisora.
- Trama de transporte (TS): Es una trama MPEG-2 que contiene una serie de servicios.
- Servicio: Cada servicio es un canal de TV, y está compuesto por una serie de eventos consecutivos.
- Evento: Cada evento es un único programa televisivo, y tienen asociado una serie de tramas elementales.

- Trama Elemental (ES): Es una trama MPEG-2 paquetizada que contiene audio, video y datos codificados según el estándar MPEG-2.
- Bouquet: Conjunto de servicios (que contienen una serie de TS) que pueden ser agrupados de manera lógica.

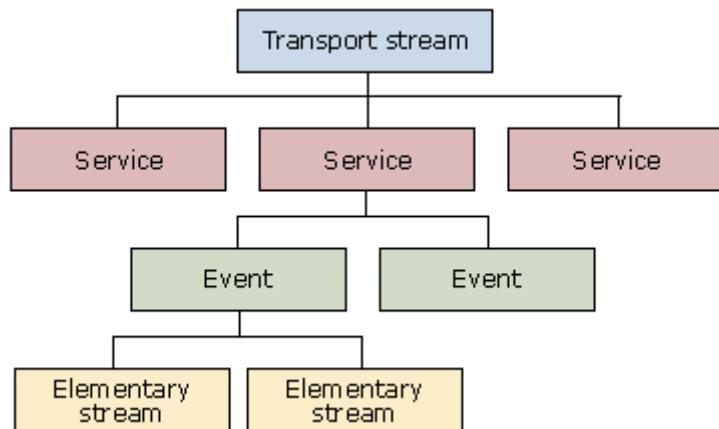


Figura 2.5: Nomenclatura DTV

DSM-CC

DSM-CC o Digital Media Command and Control, parte de la especificación MPEG-2, es un formato para la transmisión de contenidos e información de control de una trama MPEG-2. A partir del mismo surge el DSM-CC MHP que transporta las aplicaciones MHP bajo la modalidad de “Carrusel de objetos” que se explica a continuación.

Carrusel de objetos

Dado que el sistema broadcast es en un único sentido, el cliente no puede solicitar la información que desea recibir. Por esta razón es que las aplicaciones se ordenan de modo que parecen un carrusel, una se transmite a continuación de la otra, y se retransmiten una vez completada la vuelta. Si se desea que cierta información sea accesible la mayor parte del tiempo, ésta debe existir varias veces en el carrusel. El período que demora en reproducirse depende de la repetición del archivo, del ancho de banda del carrusel y el tamaño del mismo.

En otras palabras, el DSM-CC object carrusel es similar a un sistema de archivos, en donde los mismos son enviados continuamente en el mismo orden.

Optimización del carrusel de objetos

El tiempo que demora el carrusel en enviar el mismo archivo es lo que determina sustancialmente el tiempo de espera para la ejecución de una aplicación, por lo que aquí entra en juego el tamaño del carrusel. Por ejemplo, el tiempo que demora en dar “una vuelta” un carrusel de 256 KB con un ancho de banda de 128Kb/s es aproximadamente 16 segundos. Por lo tanto, cada 16 segundos la aplicación comenzará a transmitirse nuevamente, y su transmisión demorará cierto tiempo dependiendo de su tamaño. Generalmente, una aplicación y su contenido ocupan docenas de Kbytes.

Cuando se desarrollan aplicaciones MHP se desea que éstas se perciban como rápidas y fluidas. Esto se puede lograr a través de técnicas de optimización del carrusel. A continuación se exponen dos de ellas:

- Organizando adecuadamente los archivos. Este método es difícil de aplicar ya que exige un conocimiento profundo de las aplicaciones a enviar.
- Repitiendo archivos críticos a lo largo del carrusel. Esto tiene sus desventajas, produce un aumento en el tamaño del mismo, y por lo tanto el tiempo de acceso a archivos que no son repetidos también aumenta. Se debe negociar entre el desempeño individual de una aplicación (repeticIÓN) y el desempeño del resto de las aplicaciones (tamaño del carrusel).

Sincronismo

El codificador pone marcas de tiempo en los paquetes para poder sincronizar audio y video. Las aplicaciones MHP también pueden ser sincronizadas haciendo uso de estas marcas.

Aquellas aplicaciones MHP que requieren de sincronización con el flujo de video y audio, son denominadas “Stream events”. Un ejemplo claro es el de aquellos programas de preguntas y respuestas en los que se les permite a los televidentes participar.

Existen dos tipos de eventos, aquellos que deben ser ejecutados inmediatamente cuando son recibidos (“Do it now”), y aquellos que son programados, almacenados y en un momento dado deben ejecutarse (“Scheduled”).

Resolución del video

El video es codificado en MPEG-2 tanto para las resoluciones PAL o NTSC dependiendo del país en que se esté realizando la transmisión, pudiendo contar con una relación de aspecto 4:3, 16:9, entre otros. Sin embargo nada asegura que el receptor de TV sea compatible con la relación de aspecto con la que fue codificada la trama en origen. Por esta razón existe la “Descripción de Formato Activo” que brinda información de este tipo y es transmitida en la trama MPEG. Los receptores tienen incorporada la función de conversión de formatos para poder representar la imagen de manera correcta.

2.2.1.2 DVB y MHP – Arquitectura extremo-extremo

Para poder acceder a una aplicación desde un receptor de televisión es necesario que ésta esté embebida en el TS junto con la señalización que permita una ejecución adecuada. A la señalización se la conoce como PSI/SI, y la introducción de la aplicación en el flujo de audio/video se hace a través de un generador de carrusel de objetos.

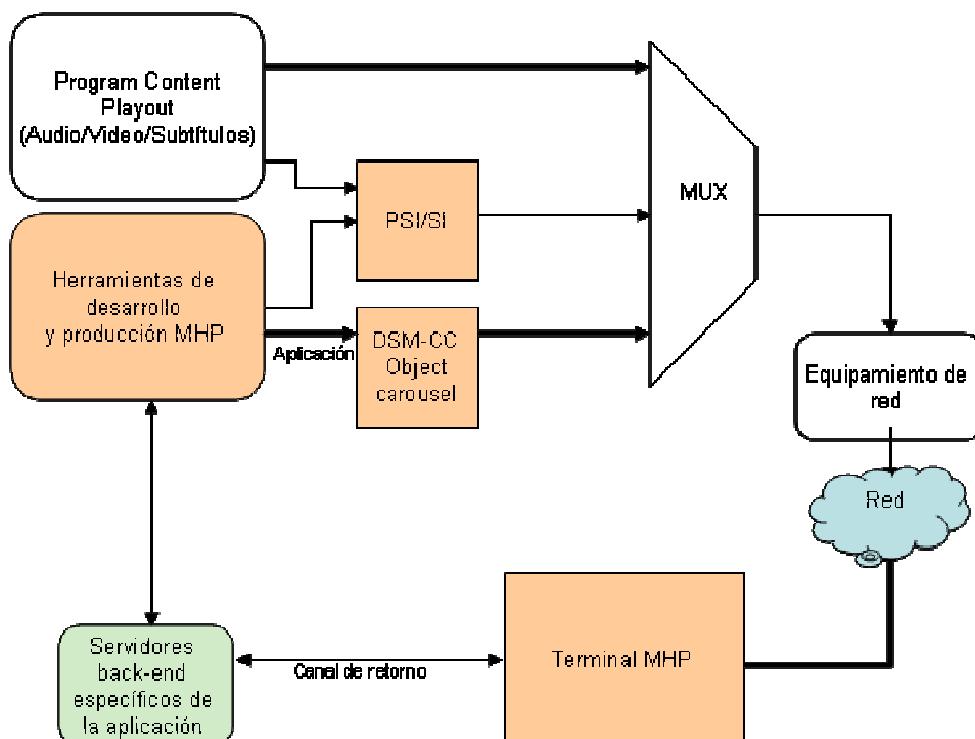


Figura 2.6: Componentes para la emisión de aplicaciones interactivas

A continuación se detalla la arquitectura extremo-extremo de la cadena MHP:

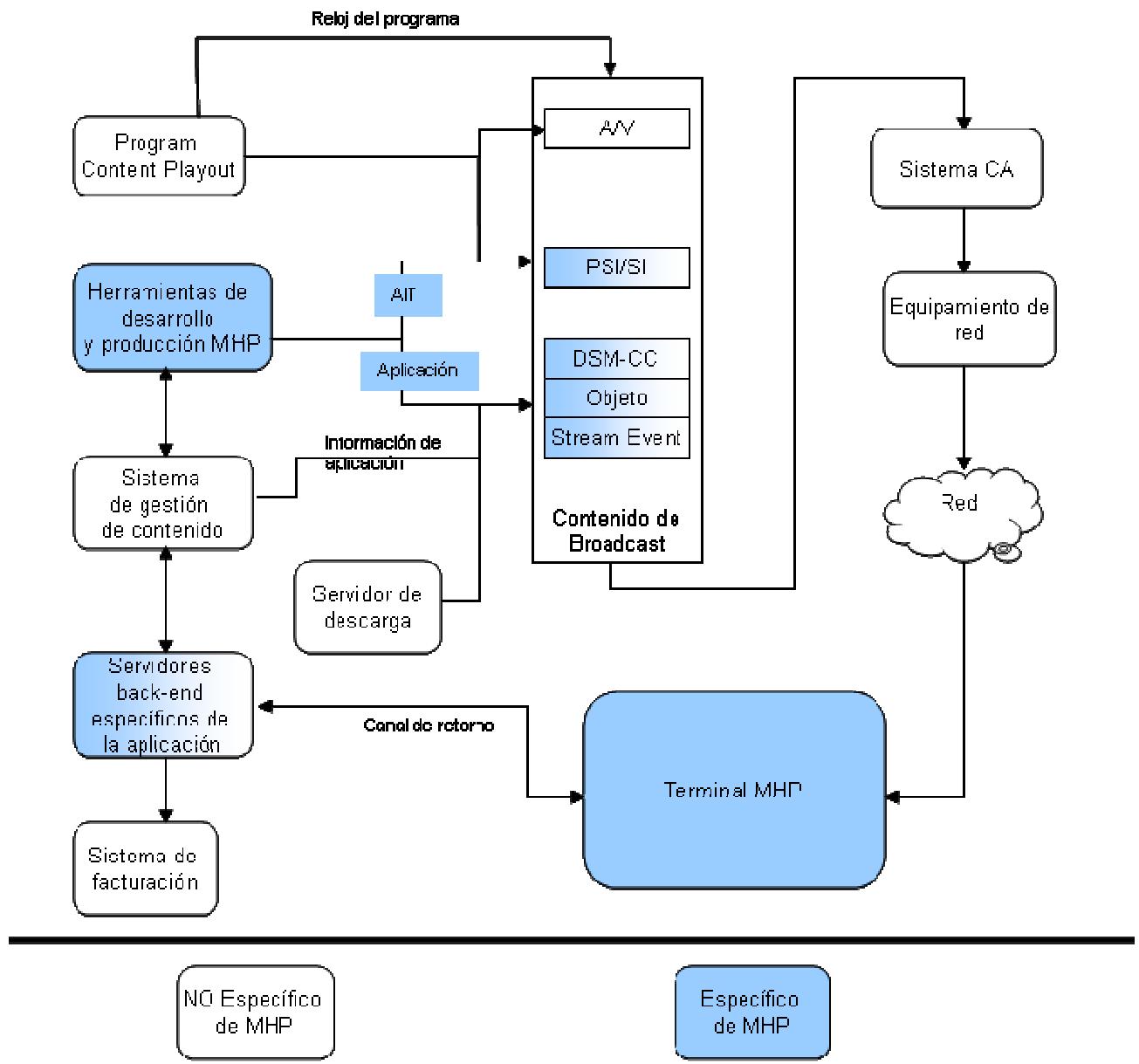


Figura 2.7: Arquitectura extremo-extremo de la cadena MHP

Program Content Playout

Este componente no es específico de los sistemas MHP, y su función es introducir el contenido de la programación (audio, video) en el equipamiento de red encargado de hacer el broadcast.

Herramientas de desarrollo y producción MHP

Componente encargado de la creación y firma de aplicaciones interactivas, y de la generación del contenido.

Sistema de gestión de contenido

Separa el contenido de la lógica y el diseño. Soporta la creación, configuración, publicación y distribución de información combinada.

Public Key Infraestructure MHP

Consiste en el uso de certificados para verificar la identidad que ha firmado una aplicación. Los certificados “root” se encuentran embebidos en cada terminal MHP.

PSI/SI (Service Information)

No específico de MHP. El video, audio y otro tipo de información son multiplexados para ser transmitidos en un único TS (Transport Stream). Cada TS está identificado por un PID y para navegar entre ellos es necesario recurrir a la tabla SI que describe el contenido disponible.

DSM-CC

El Digital Storage Media Command and Control es necesario para transmitir aplicaciones, datos, eventos, etc. Lo hace de modo tal que se le llama carrusel de objetos, ordenando las aplicaciones de cierta manera que cada determinado período de tiempo las vuelve a enviar, salvo que las mismas hayan sido eliminadas de la estructura.

Sistema CA (Conditional Access)

No es específico de MHP, y es utilizado para controlar el acceso al contenido, como por ejemplo video “on-demand”. Los sistemas de control de acceso son propietarios de cada terminal MHP, pero conceptualmente todos se basan en lo mismo. CA refiere a la forma de controlar el acceso al contenido distribuido por el operador de TV. Para esto el contenido viaja encriptado y al llegar al receptor podrá ser descifrado si el usuario tiene los permisos (clave de descifrado) para hacerlo.

El sistema de CA está estrechamente relacionado a una base de datos que guarda los permisos de cada abonado. Del lado del abonado se tiene una tarjeta (smart card) que recibe por el canal de broadcast información y de alguna manera la decodifica si corresponde.

Los algoritmos de cifrado actualmente no son públicos, y se basan en el concepto de “security through obscurity”. La seguridad reside en mantener secreto el algoritmo.

Equipamiento de red

El equipamiento de red consiste en multiplexores, moduladores, y todo lo necesario para la transmisión de la señal.

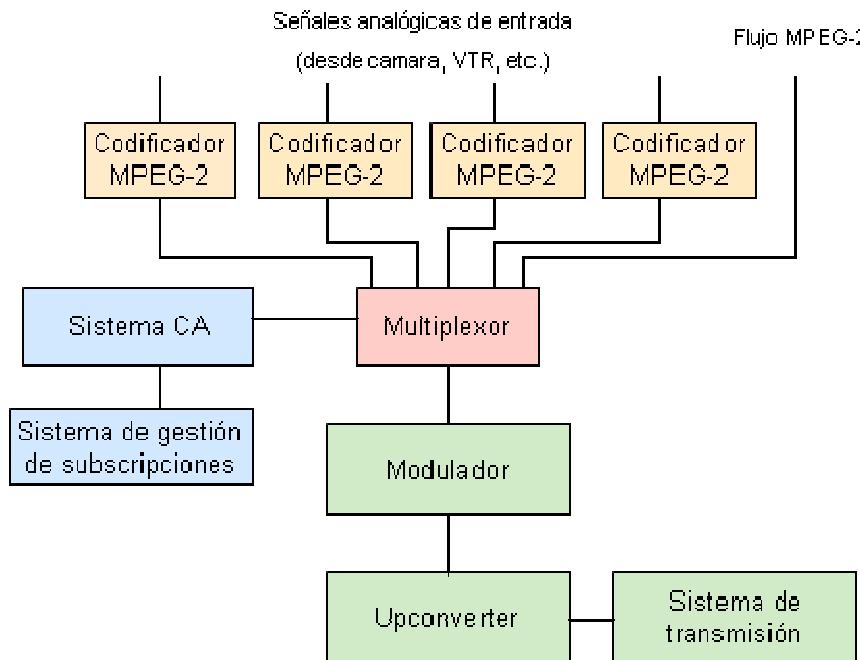


Figura 2.8: Equipamiento para la transmisión de la señal de broadcast

Codificador MPEG-2

El codificador tiene como función convertir las señales analógicas en tramas MPEG-2. Este codificador puede generar dos tipos de tramas:

- Constant Bit Rate:

Genera tasas de bit constantes sin importar la complejidad de la escenas con las que se está trabajando, si la señal es demasiado compleja para que sea codificada a esa tasa de

datos, la calidad de imagen resultará insatisfactoria. En aquellos casos en los que la escena necesite menos datos para codificar, se insertarán paquetes nulos para llegar a la tasa de bits especificada. Si bien este modo tiene la ventaja de que es más fácil realizar predicciones, desaprovecha ancho de banda.

- Variable Bit Rate:

En este caso la tasa de bits se va ajustando de forma dinámica dependiendo de la necesidad de ancho de banda que se requiera para codificar las imágenes con una calidad establecida. La mayoría de los operadores utilizan este tipo de codificación de tasa variable debido a que se obtienen mejores calidades mientras se usa eficientemente el ancho de banda.

Multiplexor

La función del multiplexor es tomar las tramas MPEG y convertirlas en una única trama de transporte.

Típicamente cada trama de transporte tiene un ancho de banda fijo disponible, éste depende del medio de transmisión y de cómo la red de transmisión fue montada. Una de las tareas que tiene asociada el multiplexor es la de colocar el conjunto de servicios en el ancho de banda disponible. La forma más sencilla de hacer esto es utilizar el método mencionado anteriormente “Constant bit rate”, de esta forma se conoce de antemano cuánto ancho de banda va a usar cada trama. Pero como se comentó esto resulta ineficiente, en contrapartida se utiliza la técnica de multiplexación estadística con una tasa de bit variable.

Prevención y corrección de errores

Antes de transmitir la señal debe asegurarse de que la misma será recibida de forma correcta. Por lo tanto se deben poder identificar y corregir errores. Generalmente se exige que la trama MPEG sea “Quasi-Error Free” (QEF), lo que significa que la tasa de bits erróneos debe ser aproximadamente 1×10^{-10} .

Dependiendo del medio de transmisión que se utilice se determina qué tipo de sistema de corrección de errores se manejará. Los sistemas DVB usan “Reed-Solomon” encoding para agregar un primer nivel protección. Para esto se agregan bytes de paridad a cada paquete, típicamente 16 bytes se le agregan a los 188-byte del paquete MPEG lo que

hace que se puedan corregir hasta 8 bytes. Errores más grandes pueden ser detectados pero no corregidos.

Modulación de la señal

Una vez que se tiene la trama digital, la señal está prácticamente pronta para ser transmitida, pero para ello antes debe ser modulada, o sea convertida en analógica para que pueda ser transmitida usando señales de radio y voltajes eléctricos.

Dependiendo del mecanismo de transmisión que se utilice se escoge el tipo de modulación. La siguiente tabla muestra qué modulación se utiliza en el ambiente de DVB.

Mecanismos de transmisión	Modulación
Satelital	QPSK
Cable	QAM
Terrestre	OFDM

Tabla 2.4: Modulaciones DVB

La transmisión terrestre utiliza OFDM con el objetivo de proporcionar una mayor resistencia a los errores producidos por la reflexión de las señales.

El proceso de modulación es llevado a cabo en el modualdor, que toma como entrada la trama de transporte (TS) y produce una salida analógica que será introducida en el equipo de transmisión.

Upconverter

Generalmente las señales son moduladas a más bajas frecuencias de lo que efectivamente son transmitidas (frecuencias que oscilan entre 950MHz y los 30GHz). Esto se debe a que es muy difícil lograr modular las señales a frecuencias tan altas. Para solucionar este problema se utiliza el “Upconverter”, que se encarga de convertir una señal desde una frecuencia hacia otra de mayor valor, a la que efectivamente se hará la transmisión.

Luego de que la señal se encuentra modulada a la frecuencia de transmisión se envía al sistema de transmisión que se encargará de transmitirla, éste puede ser una antena o un cable dependiendo del mecanismo utilizado.

Terminal MHP

El Terminal o receptor MHP, también conocido como STB (Set Top Box) es un dispositivo requerido para hacer uso de las aplicaciones interactivas de TV. El receptor MHP es quien descarga los archivos necesarios para ejecutar una aplicación desde el flujo de broadcast y ejecuta la aplicación en la Java Virtual Machine. Este receptor está generalmente separado de la televisión aunque en algunos casos puede encontrarse integrado al receptor de TV.

Los terminales MHP provienen de diferentes fabricantes y si bien todos cumplen con la especificación MHP, se diferencian entre sí por sus propiedades y capacidades tales como: memoria (varía de 16 a 64 Mbytes), velocidad de procesamiento, capacidades gráficas, etc. Sin embargo, el estándar MHP establece un conjunto de requerimientos mínimos que todo terminal certificado debe cumplir.

Canal de retorno

Éste es quien realmente introduce la interactividad bidireccional, ya que permite el envío de información desde el televidente hacia los servidores de la aplicación. Generalmente el canal de retorno es establecido mediante un módem PSTN que generará las llamadas correspondientes a través de la red telefónica, o un puerto Ethernet a través del cual el terminal se conectará al mundo IP mediante cablemodem o un módem xDSL. La mayoría de los terminales MHP cuentan con la posibilidad de establecer un canal de retorno mediante un módem PSTN, pero no siempre disponen de un puerto Ethernet.

Servidores back-end específicos de la aplicación

Esta es la parte menos estandarizada de la red, y es específica de cada aplicación. Generalmente un conjunto de servidores es utilizado por aquellas aplicaciones que necesitan acceder a información solicitada mediante una petición que llega por el canal de retorno.

Éstos son los que reciben la información generada en el terminal MHP, por lo que deben tener conexión con la red IP y tener en cuenta los aspectos de seguridad y escalabilidad. Por otro lado algunos servidores deben contar con una interfase de conexión al sistema de facturación. Este es el caso de aquellos servidores asociados a juegos de apuesta.

Dado que los servidores cuentan con muchos más recursos de procesamiento que los terminales MHP, se recomienda diseñar las aplicaciones de manera tal de introducir la mayor parte de la lógica en los servidores y así quitarle peso a los decodificadores MHP.

Actores en la cadena

- Proveedores de herramientas de desarrollo.
- Desarrolladores de aplicaciones MHP: éstos son quienes desarrollan el código y hacen uso de las herramientas provistas por los actores mencionados previamente.
- Proveedores de servicio MHP: Generalmente son los mismos operadores de TV los que proveen servicios interactivos.
- Operadores de TV: Son los responsables de transmitir el audio, el video, las aplicaciones e incluso de mantener el carrusel de objetos y los multiplexores.
- Operadores de red: Son los encargados de la infraestructura de red (multiplexores, sistemas de derechos de acceso, moduladores, etc.).
- Proveedores de sistemas de “playout” MHP: Proveen lo necesario para contar con generadores de carrusel de objetos, generadores de PSI/SI, etc.
- Proveedores CAS (Conditional Access System): Proveen los sistemas de seguridad que serán utilizados por los operadores de TV o proveedores de servicios interactivos.
- ISP (Internet Service Provider): En caso de existir un canal de retorno IP, el ISP es quien habilita la conectividad entre el terminal MHP, la red IP y los servidores de back-end.
- Operadores de servidores: Son los encargados de mantener los servidores asociados a cada aplicación en caso de que exista un canal de retorno.

- Fabricantes de terminales MHP.
- Certificadores DVB y MHP.
- Usuarios finales.

2.2.2 Xlets

2.2.2.1 Los Xlets y su ciclo de vida

Las aplicaciones DVB-J tienen dos grandes diferencias con las aplicaciones Java tradicionales:

- No siguen el comportamiento del ciclo de vida las aplicaciones Java, sino que se parecen más a los applets, donde el ciclo de vida es controlado en este caso por el middleware MHP en vez de por el usuario.
- Las aplicaciones tradicionales se ejecutan en su propia maquina virtual, que se inicializa cuando se carga la aplicación. Contrariamente las aplicaciones DVB-J pueden ser cargadas en JVMs (Java Virtual Machine) que ya estén siendo ejecutadas y pueden seguir ejecutándose una vez que la aplicación haya finalizado.

Debido a su parecido con los applets las aplicaciones DVB-J son conocidas como Xlets.

Tipos de Xlets

Dependiendo de dónde provengan los Xlets, éstos se pueden clasificar de la siguiente manera:

- Residentes

Son aquellas aplicaciones, no necesariamente MHP, que se almacenan en la memoria no volátil del terminal. Un ejemplo de aplicación residente puede ser un juego. Las aplicaciones residentes no pueden interactuar con las MHP.

- No residentes

Éstas son aplicaciones MHP que se han recibido por el canal de broadcast. Las aplicaciones de este tipo pueden ser cargadas rápidamente si fueron cacheadas, o en caso contrario se cargarán desde el carrusel de objetos. Una aplicación no residente será ejecutada cuando lo indique la AIT que llega por el canal de broadcast. Adicionalmente, este tipo de aplicaciones contienen un “Application Storage Descriptor” en el que se menciona el tipo de almacenamiento y las versiones de la aplicación. Este descriptor será obtenido desde el carrusel de objetos por el terminal MHP como una estructura XML, y así el receptor MHP habilitará el almacenamiento de la aplicación. No se recomienda almacenar archivos que requieran de actualizaciones constantes, ya que el terminal puede rechazar las actualizaciones.

Ciclo de vida de la aplicación

Una aplicación MHP puede empezar:

- automáticamente
- ante petición explícita del usuario

La vida de una aplicación generalmente termina cuando se cambia de canal, y lo único que queda de ella es la información que se almacenó en memoria.

Los Xlets tienen un ciclo de vida que se describe a partir de 4 estados:

- Loaded: La instancia de la aplicación fue cargada pero no inicializada.
- Paused: En este estado la aplicación minimiza el uso de recursos, pero permanece activa en background, no visible. Lo que debe hacer cada aplicación cuando es pausada no está claro, depende de la implementación de la aplicación, pero lo que es seguro es que el receptor ocultará todos los elementos gráficos asociados a ella.
- Active: La aplicación está en funcionamiento y provee servicios.
- Destroyed: La instancia de la aplicación libera todo los recursos y es finalizada.

La razón por la que se tienen todos estos estados es la siguiente:

- Los ambientes de ejecución son de bajo desempeño, por lo que el uso de recursos debe ser óptimo. Es por esto que resulta beneficioso poder pausar las aplicaciones en lugar de destruirlas por completo.
- El sistema debe permanecer limpio, por lo tanto las aplicaciones deben ser capaces de liberar recursos cuando hayan terminado.

¿Cómo se ejecuta una aplicación?

Existen los llamados XletManagers que son una aplicación más, pero con la diferencia de que éstos residen en el terminal MHP. Se encargan de manipular las aplicaciones a través de su interfase. La clase inicial de la aplicación MHP debe implementar la interfase javax.xlet.Xlet, ya que ésta será utilizada por el XletManager para señalar el estado de la aplicación y sus cambios.

A continuación se detalla la interfase de un Xlet:

```
public interface Xlet {  
  
    public void initXlet(XletContext ctx) throws  
        XletStateChangeException;  
  
    public void startXlet() throws XletStateChangeException;  
  
    public void pauseXlet();  
  
    public void destroyXlet(boolean unconditional) throws  
        XletStateChangeException;  
}
```

Diagrama de estado de un Xlet:

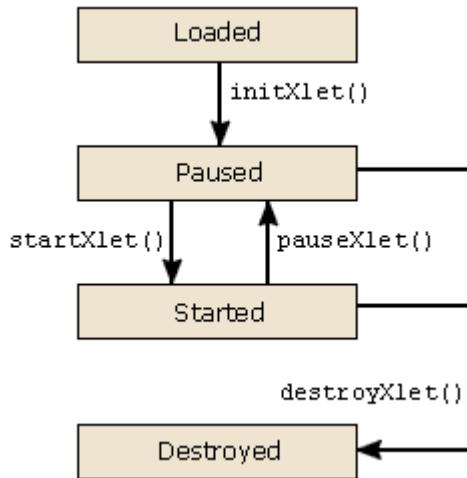


Figura 2.9: Estados de un Xlet

1. El XletManager carga el Xlet y crea una instancia del mismo.
2. Cuando el usuario decide comenzar la aplicación, o alguna señalización lo indica, el XletManager invoca al método initXlet() al que le pasa como parámetro un nuevo XletContext. Este XletContext podrá ser utilizado por la aplicación para inicializarse a sí misma y descargar archivos que requieren de un tiempo de descarga prolongado. Entonces la aplicación queda en estado “Paused”.
3. Una vez que se finalizó la ejecución del método initXlet() el XletManager invoca al startXlet(), y entonces el Xlet pasará a estado “Started”, y podrá así interactuar con el usuario.
4. Durante la ejecución del Xlet, el XletManager puede invocar el método pauseXlet(), y entonces pasará desde el estado “Started” a “Paused” tantas veces como sea necesario durante su ciclo de vida.
5. Al final de la vida útil del Xlet, el XletManager invocará al método destroyXlet(), que dejará a la instancia Xlet sin validez.

XletContext

Cada Xlet tiene un contexto asociado representado por una instancia de la clase javax.tv.xlet.XletContext. El contexto permite a la aplicación obtener más información del entorno en el que se ejecuta e informar de su estado al mismo.

```
public interface XletContext {

    public static final String ARGS = "javax.tv.xlet.args"

    public void notifyDestroyed();

    public void notifyPaused();

    public void resumeRequest();

    public Object getXletProperty(String key);

}
```

Ejemplo del ciclo de vida de una aplicación

Cuando una aplicación pide ser pausada y luego resumida:

1. El Xlet invoca a notifyPaused() del XletContext.
2. El XletContext notifica al XleManager sobre ello para que este último actualice su estado interno.

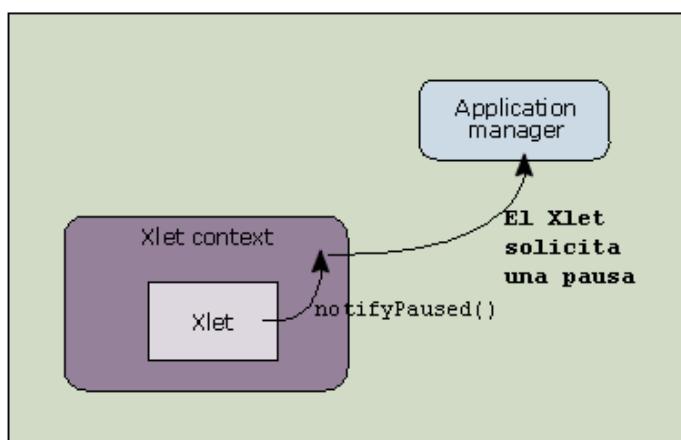


Figura 2.10: Función del Xlet context A

3. El Xlet invoca el método `requestResume()` cuando quiere reiniciarse.
4. El XletContext pasa esta petición al XletManager, quien invocará o no al método `startXlet()` dependiendo de los recursos con los que cuente.

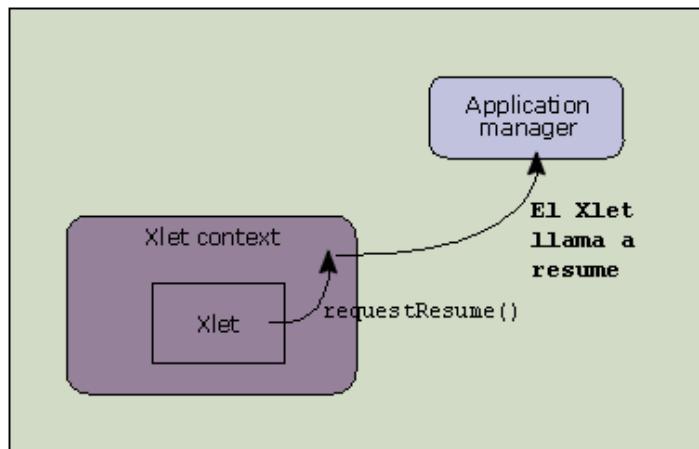


Figura 2.11: Función del Xlet context B

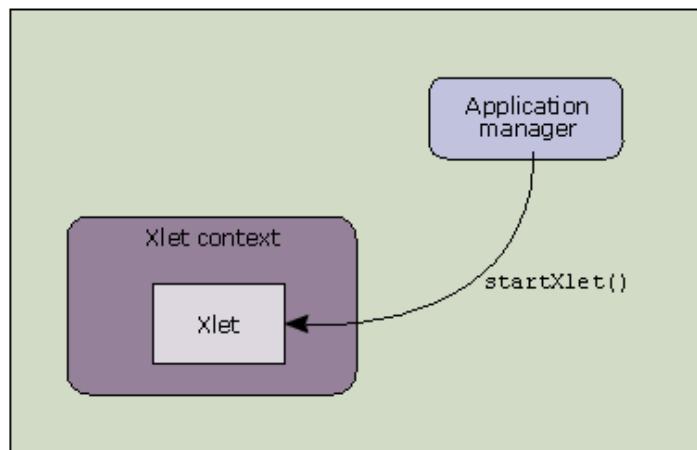


Figura 2.12: Función del Xlet context C

El método `getXletProperty()` permite al Xlet acceder a las propiedades que fueron definidas para él mediante la señalización del broadcaster. La propiedad más utilizada se denomina `XletContext.ARGS`, y habilita a la aplicación a acceder a la señalización de aplicación (AIT).

También existe un conjunto de propiedades del sistema (por ejemplo el `path.separator` que indica el separador de nombres de archivos) a las que se puede acceder a través del `System.getProperty()`.

2.2.2.2 Un Xlet básico

A continuación se muestra el esqueleto básico de un Xlet del tipo “Hello world”.

```
public class MiXlet implements javax.tv.xlet.Xlet
{
    /*Todo Xlet tiene un contexto que se lo pasará el middleware a través del
     *parámetro del método initXlet()*/
    private javax.tv.xlet.XletContext context;

    /*mantiene el estado actual del Xlet, permite distinguir si al invocar un
     *start es un start o un resume*/
    private boolean hasStarted;

    public MiXlet() {}

    public void initXlet(javax.tv.xlet.XletContext context) throws
        javax.tv.xlet.XletStateChangeException {
        this.context = context;

        //el Xlet no fue comenzado, por lo tanto:
        hasStarted = false;

        //imprimimos un mensaje en la salida del debug
        System.out.println("El contexto es"+context);
    }

    public void startXlet() throws
        javax.tv.xlet.XletStateChangeException{
        if (hasStarted) {
            System.out.println("Hola nuevamente!");
        }
        else {
            System.out.println("Hola, gusto en conocerte!");
            hasStarted = true;
        }
    }

    public void pauseXlet() {
```

```
        System.out.println("Se invocó el método pauseXlet()");  
    }  
  
    public void destroyXlet(boolean unconditional) throws  
javax.tv.xlet.XletStateException{  
    if (unconditional) {  
        System.out.println("Se invocó destroyXlet(), Adiós");  
    }  
    else {  
        System.out.println("Se invocó el método destroyXlet(), pero  
no lo aceptaré");  
        throw new XletStateException("Por favor, no me destruya!");  
    }  
}  
}
```

2.2.2.3 Señalización sobre el ciclo de vida de una Aplicación MHP

En todos los casos, cada aplicación está asociada a un servicio, por lo que no existe una aplicación MHP autónoma. Esto significa que el ciclo de vida de una aplicación está estrechamente relacionado con el servicio padre, y cuando el televidente cambia de canal, cualquier aplicación que esté asociada con el canal anterior probablemente finalizará.

Típicamente toda la información que se necesita para la aplicación, será transportada en la trama MPEG junto con el audio y video necesarios. Esta información consiste de dos partes. La primera parte son los archivos que construyen la aplicación, y la segunda parte es la tabla denominada Application Information Table (AIT) que permite identificar las aplicaciones que se encuentran disponibles e indica cómo ejecutarlas.

Basándose en la información contenida en la AIT, el XletManager es responsable de monitorear los servicios existentes, buscar aplicaciones disponibles que sean parte de los servicios, ejecutar y parar aplicaciones de forma apropiada. El XletManager tiene total control de la aplicación y de la interacción de la misma con el ambiente MHP.

Contar con la tabla AIT presenta ventajas para:

- Operadores de TV: les permite mantener el control de cuándo se ejecutarán las aplicaciones. Si una aplicación no se encuentra en la AIT entonces no se ejecutará.
- Usuarios finales: tendrán acceso a aquellas aplicaciones que sean relevantes en ese momento, o en todo caso a aplicaciones genéricas como EPG. Esto implica que la navegación será más simple.

AIT

La AIT no sólo contiene una lista de aplicaciones asociadas al servicio, sino que también mantiene información de control acerca de ellas. Contiene un código de control que indica al receptor MHP qué debe hacer con la aplicación. Los códigos de control disponibles son los siguientes:

Código	Valor
AUTOSTART	0x01
PRESENT	0x02
DESTROY	0x03
KILL	0x04
PREFETCH	0x05
REMOTE	0x06

Tabla 2.5: Señalización del estado del Xlet a través de la AIT

AUTOSTART

Con este código de control la aplicación será ejecutada automáticamente cuando el receptor sintonice ese servicio.

PRESENT

Con este código la aplicación no será ejecutada automáticamente pero será incluida en la lista de aplicaciones disponibles, prontas para ejecutar en caso de que el usuario lo solicite.

KILL/DESTROY

La aplicación será terminada o destruida por el receptor. Por ejemplo, esto puede ser utilizado para terminar con una aplicación que estaba asociada a un programa que ya ha finalizado.

La diferencia entre “Kill” y “Destroy” es que con el código Kill la aplicación tendrá la opción de seguir existiendo, mientras que con “Destroy” no.

PREFETCH

Aplica a aplicaciones DVB-HTML.

REMOTE

Este código indica al receptor que debe cambiar de servicio para ejecutar esta aplicación.

Por otro lado la AIT contiene una entrada que indica qué MHP profile se requiere para ejecutar cada aplicación. Si el MHP profile especificado supera al soportado por el receptor entonces la aplicación será ignorada.

Sintonizando servicios y modificando el ciclo de vida de una aplicación

Dado que una aplicación sólo puede ser ejecutada si se encuentra listada en la AIT del servicio actual, cambiar el servicio implicará un cambio de AIT, y por lo tanto un cambio en las aplicaciones disponibles. Si una aplicación que se encuentra ejecutando figura en la nueva AIT entonces seguirá ejecutándose, de lo contrario será finalizada.

Si se desea que una aplicación siga ejecutándose aunque se cambie de servicio, pero sin permitir que se ejecuten nuevas instancias de ella, entonces la misma será marcada en la AIT como “external”.

El proceso de sintonización es como sigue:

1. El XletManager examina el set de aplicaciones actuales. Toda aplicación cuya señalización indica que se encuentra asociada al servicio actual será finalizada.
2. El XletManager examina las aplicaciones asociadas al nuevo servicio y todas aquellas cuyo código de control sea AUTOSTART serán cargadas y ejecutadas.

3. Las aplicaciones que están ejecutándose y no están señalizadas en la nuevo AIT serán comparadas contra la lista de aplicaciones del tipo “external”.

2.3 MHP - Guía para el desarrollo de aplicaciones

2.3.1 Ambiente de desarrollo

La situación ideal sería contar con un entorno real de aplicaciones, de manera de montar un sistema de transmisión y recepción de la señal digital de televisión con el objetivo de facilitar la tarea de desarrollar y testear aplicaciones interactivas MHP. Para esto, habría que adquirir los siguientes componentes:

- Playout: Se encarga de transportar la trama. Pone los datos de las aplicaciones interactivas en banda base y se encarga de generar un flujo de datos compatible con el audio y video MPEG-2.
- Modulador: Se encarga de modular la señal que le llega del playout y la traslada a la banda que le permita enviar la trama de transporte de forma compatible con los receptores MHP.
- Receptores MHP: Se encargan de decodificar la señal que le llega vía aire en formato MPEG-2, audio, video y datos.

Dado que el escenario anteriormente descripto implica un presupuesto económico muy elevado, en sustitución de estos equipos se pueden utilizar emuladores y STBs de desarrollo.

A partir de ahora se denominará “ambiente de desarrollo real” al contexto real de aplicaciones MHP, y “laboratorio MHP” al de simulación.

2.3.1.1 Herramientas para el desarrollo de aplicaciones

Este componente es común a ambos ambientes de desarrollo, el real y el de simulación. Para el desarrollo de aplicaciones pueden utilizarse herramientas de desarrollo específicas para MHP, herramientas Java genéricas o editores de texto. La ventaja de utilizar herramientas de desarrollo MHP es que éstas incluyen todas las clases necesarias para compilar aplicaciones, y a través de interfaces gráficas permiten ir generando el código. Algunas de estas herramientas se listan a continuación:

- AltiComposer de Alticast
- Cardinal Studio de Cardinal Information Systems
- JAME de Fraunhofer IMK

Si se desea hacer uso de una herramienta Java genérica, se deberá contar con un set de clases para compilar la aplicación. En caso de utilizar un simple editor de textos, se necesitará un compilador java, y el conjunto de clases asociado a MHP. Las “stub classes” permiten compilar las aplicaciones, ya que contienen la API completa MHP.

2.3.1.2 Ambiente de desarrollo real

En ocasiones es adecuado introducir la aplicación en el TS emitido por el operador de TV, para así evaluar cómo se despliega la aplicación junto con el video, y el comportamiento de la misma al seleccionar un nuevo servicio de la trama MPEG, leer información del mismo, etc. Para esto se debe alimentar el STB con un TS. Si una señal real no se encuentra disponible entonces es posible generar un TS propio. Existen dos aspectos principales a tener en cuenta:

- Transmitir el TS a la velocidad correcta.
- Generar un TS que el STB pueda entender.

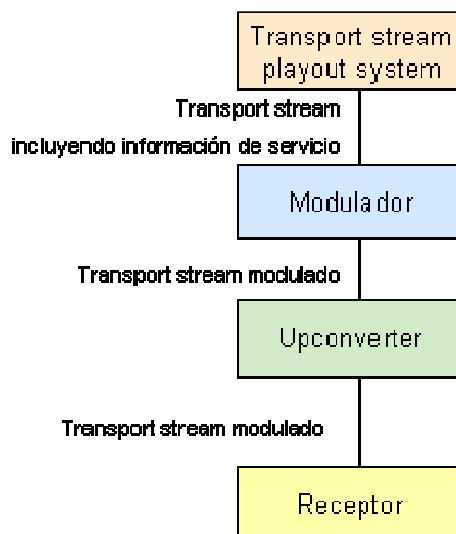


Figura 2.13: Emisión de un TS

TS playout system:

Es un dispositivo que básicamente, luego de multiplexado el audio, video y carrusel de objetos, genera el TS y lo envía sobre una conexión serie asíncrona de alta velocidad (ASI) o una interfase Ethernet.

Modulador:

Los moduladores toman el TS generado anteriormente y lo convierten en una señal que pueda ser entendida por el STB.

Sin embargo antes de hacer uso de todos estos elementos se debe contar con un TS. Si no se cuenta con uno se podrá generar mediante alguna de las opciones que se exponen a continuación.

Generar un TS con una aplicación embebida

Opción 1:

Hacer uso de un TS que se ha obtenido y grabado del aire, y anexarle la aplicación deseada.

Opción 2:

Generar un TS con la aplicación deseada, el carrusel de objetos, la trama de datos y la trama de eventos.

Se profundizará en la opción 2, por lo que se generará un TS con un único servicio, un único video, un único audio, un carrusel de objetos y la señalización adecuada. El diagrama a continuación muestra el proceso básico para generar un TS MPEG-2:

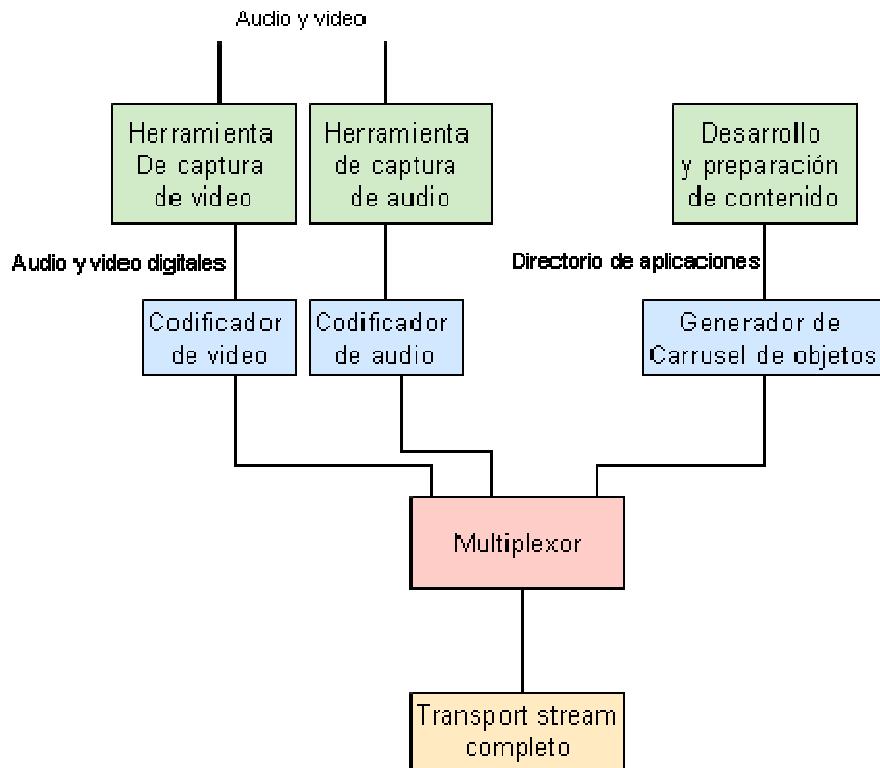


Figura 2.14: Proceso de generación de un TS

- Audio y video

El primer paso consiste en capturar y codificar audio y video. Luego, se debe codificar el audio y el video según los siguientes parámetros:

- formato MPEG-2, ningún otro formato es soportado por MHP
- resolución de video 720*576, 25 fps

Existen codificadores de uso libre como el “MPEG-2 encoder” del MPEG Software Simulation Group. También existen trials de mejor calidad como el “TMPGEnc”.

- Aplicación: archivos y señalización

Se debe generar una trama que contenga el carrusel de objetos en donde los archivos de la aplicación estarán almacenados. Para esto se necesita un generador de carrusel de objetos que generará a grandes rasgos la estructura de archivos correspondientes a la aplicación, y a partir de éstos generará una trama MPEG en la que irá el carrusel de objetos. Adicionalmente generará la señalización de la aplicación que el STB necesita

para ejecutarla de manera apropiada. En conclusión, el generador del carrusel de objetos dará como resultado dos elementary streams: el carrusel de objetos, y la señalización.

Existen generadores de carrusel por SW, que dejan como salida un archivo, un ejemplo es “SoftOC” de Strategy & Technology, o “MediaSphere Lab” de Softel, o “Just-DVB-It” de Cineca.

- Stream events

Si se desea sincronizar las aplicaciones con los contenidos de video entonces se necesitan stream events, que grosso modo se traducen en marcas de tiempo ubicadas en el TS. La mayoría de los generadores de carrusel de objetos generan stream events si es necesario.

- Multiplexor

Video + Audio + carrusel de objetos +señalización de la aplicación + stream events.

Todo esto debe ir en un único TS, con información de servicio (SI) adicional para que el STB pueda encontrar las tramas adecuadamente.

Los multiplexores pueden ser implementaciones de SW y generan un TS a partir de archivos, dando como resultado otro archivo.

Luego del proceso anterior se obtiene un TS listo para introducir en el sistema de Playout. Resumiendo, el principal contenido de un TS es el siguiente:

Tipo de trama	Descripción	Generado Por:
Video	Video asociado a la programación del canal.	Codificador MPEG.
Audio	Audio asociado al video, puede haber más de uno (ej. Diferentes lenguajes)	Codificador MPEG.
Información de servicio (SI)	Describe como está organizado el servicio y le indica al STB donde encontrar la información que necesita.	Multiplexor.
DSM-CC object carousel	Sistema de archivos transmitido contiene la aplicación MHP y los archivos de datos asociados.	Generador de carrusel de objetos.
Señalización de la aplicación	Brinda información acerca de cuando comenzar una aplicación y de donde descargar sus archivos.	Generador de carrusel de objetos.

Tabla 2.6: Contenido de un TS y cómo generararlo

A grandes rasgos, el equipamiento a utilizar por parte de la emisora se dibuja a continuación:

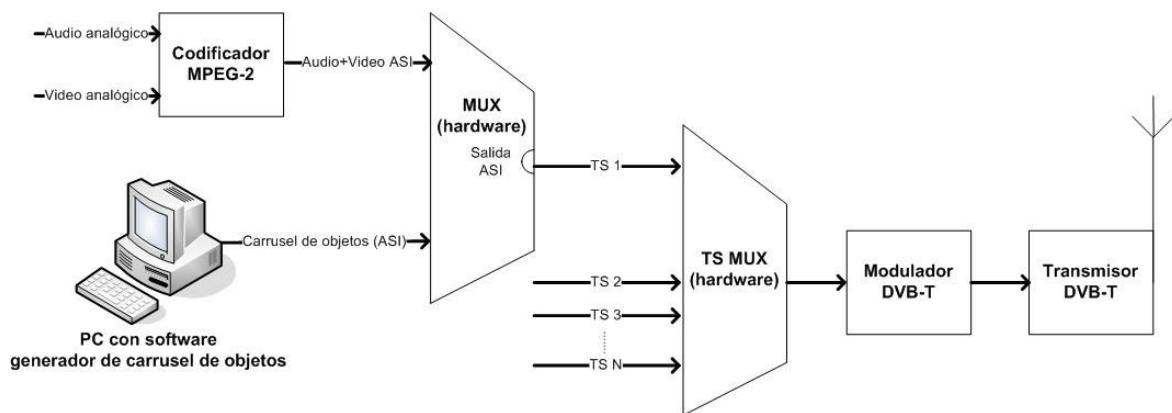


Figura 2.15: Equipamiento de broadcasting MHP

En este contexto, el equipamiento requerido para la instalación de un ambiente de desarrollo real consiste de:

- Herramientas de desarrollo Java
- Sistema de playout MHP (generador de carrusel de objetos, señalización, etc.)

- Cabecera de transmisión digital (codificadores, multiplexores, generadores de TS, moduladores, transmisores)
- Terminal MHP (STB)
- Receptor de TV analógico

En el Anexo 10 (“Productos para la instalación de un ambiente de desarrollo MHP”) figuran productos actualmente disponibles en el mercado, sus características, precios y hojas de datos.

2.3.1.3 Laboratorio MHP

Ésta es una alternativa al contexto real de desarrollo y testo de aplicaciones MHP. El mismo consiste en:

- STB de desarrollo.
- Emulador MHP.

STB de desarrollo

A diferencia de los STBs comerciales, los de desarrollo permiten descargar la aplicación sin tener que recurrir a la trama de transporte. Simplemente acceden a los archivos de la aplicación, por ejemplo a través de una conexión serial, o FTP, y la ejecutan. Adicionalmente, la mayoría de los STBs de desarrollo cuenta con una salida de debugging hacia una consola en la que imprimen mensajes de log.

En otras palabras, los STBs de desarrollo permiten la carga de aplicaciones desde una PC vía el puerto RS-232 o Ethernet del receptor. Generalmente incluyen un software para facilitar esta tarea, y la transferencia de archivos se realiza desde el puerto serial de la PC. Vale la pena aclarar que en este contexto no es necesario contar con un generador de carrusel de objetos, ya que al receptor simplemente deben transferirse los archivos propios de la aplicación, y algunos archivos de configuración a modo de señalización. El STB de desarrollo permite montar un entorno de simulación con un simple ordenador y un televisor.

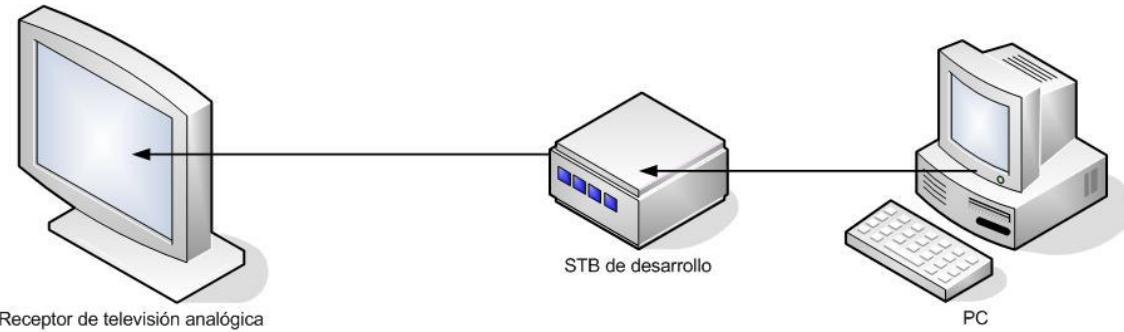


Figura 2.16: Testeo en un STB de desarrollo

Emulador

Un emulador es adecuado para principiantes en el desarrollo de aplicaciones, ya que permite abstraerse de un montón de complejidades, realizando el testeo de manera local. Existen emuladores comerciales como el “MHP4Win” de Osmosys que al estar desarrollado sobre el mismo código que un receptor MHP brinda guías de cómo se comportará la aplicación en el contexto real. Por otro lado están los emuladores libres como “XleTVView” y “OpenMHP” que no son implementaciones completas de MHP, por lo que no cuentan con todas las funcionalidades, sin embargo para comenzar el testeo son las más apropiadas.

Se debe tener en cuenta que un emulador no puede prever cómo se comportará la aplicación en diferentes STBs, debido a que cada uno está implementado de manera diferente. Sin pruebas en el contexto real y en diferentes STBs es difícil estimar: tiempo de carga de una aplicación, retrasos experimentados, tiempo de descarga de archivos, etc.

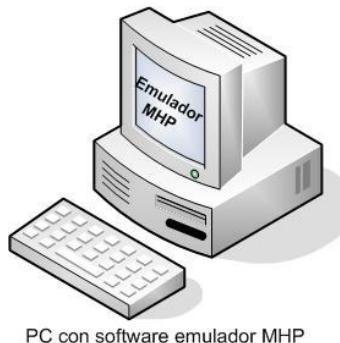


Figura 2.17: Testeo local a través de un emulador MHP

Entonces, para la instalación de un laboratorio MHP se debe contar con los siguientes productos:

- Herramientas de desarrollo Java
- Emulador MHP
- STB de desarrollo
- Receptor de TV analógico

En el Anexo 10 (“Productos para la instalación de un ambiente de desarrollo MHP”), figuran productos actualmente disponibles en el mercado, sus características, precios y hojas de datos.

2.3.2 MHP- Recursos escasos de los STB

Los terminales MHP ofrecen recursos limitados a las aplicaciones interactivas, por lo tanto, éstos deben ser optimizados y compartidos de manera eficiente.

2.3.2.1 Memoria

Para reducir costos, los terminales MHP cuentan con memorias pequeñas, sin embargo el mínimo aceptable es de 4 Mbytes.

Para un uso eficiente de memoria existen algunas reglas:

- Poner las aplicaciones inactivas en estado “paused” para que éstas liberen recursos. Es un estado de hibernación, en el que las aplicaciones se mantienen en background prontas para ser ejecutadas.
- Evitar el uso de archivos de gran tamaño como imágenes, audio, etc.
- Ejecutar una aplicación a la vez.

2.3.2.2 Almacenamiento persistente

En el ambiente MHP se les permite a aquellas aplicaciones autorizadas almacenar archivos en un espacio de almacenamiento persistente. Sin embargo, éste no asegura a la aplicación que sus archivos serán almacenados por siempre.

2.3.2.3 Interfase de tuning (selección de servicio)

Para televisión digital terrestre, satelital o por cable cada emisora transmite una señal (transport stream) según el estándar MPEG-2. Cada TS lleva consigo varios servicios interactivos. Tuning es el término utilizado para describir la acción de seleccionar un servicio determinado. Si el servicio seleccionado es parte del mismo TS que el servicio actualmente en uso, el acto de tuning consiste únicamente en un filtrado de paquetes. Sin embargo si el servicio se encuentra en otro transport stream entonces se debe sintonizar otra frecuencia y este procedimiento no es tan sencillo y consume recursos.

La interfase de sintonización es un recurso escaso, ya que únicamente una aplicación a la vez será capaz de realizar operaciones de sintonización.

2.3.2.4 Canal de retorno

La calidad/velocidad del canal de retorno depende del tipo del mismo y del servicio contratado.

XDSL o cable-modems proveen una conexión permanente (siempre disponible para su uso), y por lo tanto no son percibidos como recursos escasos. Sin embargo estos tipos de modems no vienen integrados al terminal MHP, generalmente éstos cuentan con una interfase Ethernet. Cuando se hace uso de conectividad a la PSTN el terminal MHP debe establecer una conexión telefónica y luego una conexión IP, y en este caso el canal de retorno sí debe ser considerado como un recurso escaso.

2.3.3 Usabilidad

Un factor crítico para la aceptación de aplicaciones interactivas por parte del usuario y satisfacción respecto a las mismas radica en la facilidad de uso.

Cuando se desarrolla una aplicación interactiva se debe contemplar que el usuario aún sigue enfocándose en la TV como medio de entretenimiento, una actividad pasiva, y el

uso de iTV requiere de un usuario activo. A la hora de desarrollar aplicaciones se deben tener en cuenta tres aspectos:

- legibilidad
- layout y diseño
- navegabilidad

2.3.3.1 Legibilidad

Este es uno de los más grandes desafíos en TV, debido a que los usuarios no están acostumbrados a leer textos desplegados en la pantalla del dispositivo en cuestión, la televisión. Por esta razón la fuente utilizada y el tamaño deben ser cuidadosamente elegidos.

Se recomienda respetar las siguientes reglas:

- Tamaño de fuente de al menos 18 pts.
- Texto claro sobre fondo oscuro.
- Interlineado amplio.
- No más de 90 palabras por pantalla.
- No hacer uso de más de 3 fuentes diferentes.
- Consistencia en el uso de fuentes (títulos, texto, etc.).

2.3.3.2 Layout y diseño

El diseño de la aplicación debe también considerar cómo se ordenarán en pantalla las diferentes informaciones, la posición, el tamaño. La consigna es: “Lo necesario, pero lo menos posible”

Todas las pantallas de una aplicación deben tener el mismo diseño a la vista del usuario, para que éste pueda encontrar más fácilmente la información que desea.

Los usuarios por lo general recorren la pantalla desde la esquina superior izquierda hasta la inferior derecha, por lo tanto estas esquinas son privilegiadas, e ideales para títulos y logos.

El lado izquierdo de la pantalla es recomendado para posicionar texto, y el derecho video si se requiere.

En cuanto a los colores, se debe tener en cuenta que la correcta reproducción y percepción de los mismos no está garantizada. Por esta razón se debe usar colores sólo cuando se desea resaltar información. A mayor cantidad de colores más información debe procesar el usuario.

Por otro lado, debido a las características de la TV, algunos colores no pueden ser combinados ya que causan distorsión, por ejemplo: rojo y azul francia, verde flúo y rojo, verde flúo y azul Francia.

2.3.3.3 Navegabilidad

Una óptima navegación requiere de correspondencia entre la interfase visual y los equipos de usuario, en este caso el control remoto.

Toda interfase de navegación debe cumplir con los siguientes objetivos:

- Dar idea al usuario de donde está “parado”, cómo llegó a ahí y a dónde puede dirigirse.
- Proveer feedback cada vez que el usuario ejecuta un comando.
- Enseñar al usuario cómo hacer uso del servicio en sólo unos segundos.
- Proveer un escape rápido.

Los controles remotos son los dispositivos que permiten al usuario interactuar, y la aplicación debe permitir intuitivamente presionar los botones correctos, aunque se debe tener en cuenta que los controles remotos son diferentes entre sí y no están estandarizados.

Todos los controles remotos al menos cuentan con los siguientes botones:

- Controles tradicionales de TV (volumen, canal, on/off, etc.)
- Números
- Flechas
- Botón de selección (“ok”)
- Botones de color
- Propietarios

Algunos controles cuentan con botones específicos que hacen fácil la navegación, y si existen debe tomarse ventaja de ellos, sin perder de vista que el contenido debe poder accederse desde cualquier plataforma, con o sin botones específicos. Algunos de estos botones especiales son:

- Back: no aplicable en el ámbito de la TDT, pero permite al usuario ir un nivel atrás, guarda un historial.
- Help: generalmente aplicado para TV satelital. Muestra ayuda para hacer uso del servicio.
- Page Up/Down: Navega por aquel contenido de la aplicación que no entra por completo en pantalla

El estándar MHP define un set mínimo de teclas que deben existir:

- 10 teclas numéricas
- 4 flechas
- Tecla “OK”
- Teclado para acceder al Teletext
- 4 teclas de color

Por lo tanto, MHP permite la interacción mediante:

- números
- botones de color
- flechas y botón de selección



Figura 2.18: Comando a distancia o control remoto.

2.3.4 Gráficos en MHP

Toda aplicación interactiva tiene como principal tarea desplegar imágenes, texto, botones, etc. en pantalla. Para esto, MHP hace uso de secciones de AWT (Abstract Window Toolkit) y la API GUI de la especificación HAVI.

El modelo de gráficos es una de las partes más complejas de MHP y los siguientes aspectos deben ser considerados:

- Pixeles

Los pixeles en la TV no son cuadrados como los son en las PCs. Combinar estos dos tipos puede resultar sumamente complicado para los desarrolladores, inclusive algunos receptores no soportan todos los formatos, por lo que lograr contornos perfectos resulta extremadamente complejo.

- Cambios en la relación de aspecto

La señal de TV puede estar destinada a pantallas cuya relación de aspecto sea 4:3, o 16:9. Y por lo tanto el diseño de gráficos para la utilización en uno de estos aspectos no será adecuado para la presentación en el otro. Por esta razón es que se diseña para una relación de aspecto 14:9, que será levemente distorsionada en cualquiera de las anteriores.

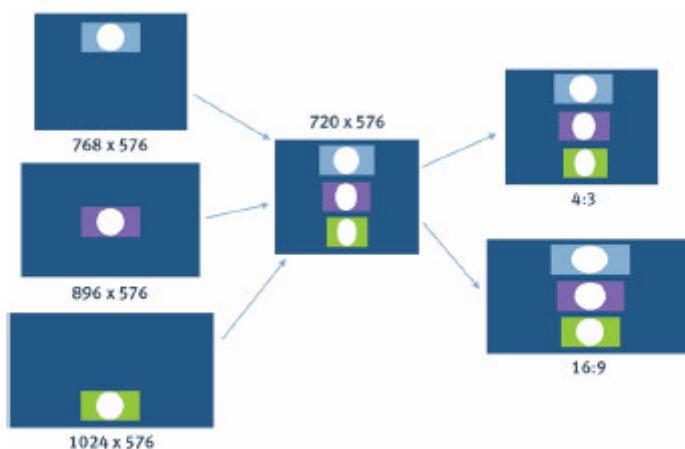


Figura 2.19: Relación de aspecto de una pantalla

Una imagen que desea ser vista de manera correcta en una pantalla 16:9 debe ser creada con un tamaño de 1024x576 pixeles. Si se desea crear una imagen para una TV cuya relación de aspecto es 4:3, entonces su tamaño deberá ser de 768x576. Para crear una imagen que sea tolerada por ambas pantallas el tamaño óptimo es de 896x576.

- Overscan

La TV puede ampliar las imágenes al momento de desplegarlas, por lo tanto al diseñar se debe trabajar sobre una zona segura de la pantalla para que en caso de que ocurra una ampliación la imagen no sobresalga de la pantalla visible. Esta zona segura no incluye el 5% de la pantalla cerca de cada límite de la misma.

- Transparencia

Muchas veces es deseable que el usuario pueda ver lo que se encuentra por debajo de la aplicación, esto hace referencia a la transparencia de la misma, que se despliega sobre el flujo de video.

- Administrador de ventanas

Dada la escasa capacidad de los receptores MHP, no existen administradores de ventanas, pero de todas formas las aplicaciones necesitan de algo que les asigne el área en pantalla que pueden utilizar. Esto se hace a través de la clase HScene.

- Interfases de usuario

Una aplicación no podrá interactuar con el usuario si otra aplicación se está ejecutando.

2.3.4.1 Dispositivos de pantalla

Un dispositivo de despliegue en el ámbito de la TV interactiva puede ser dividido en 3 capas ordenadas de menor a mayor grado de superficialidad:

- capa de fondo: despliega un fondo de color o una imagen fija.
- capa de video: despliega el video asociado a la programación de TV.
- capa de gráficos: en esta capa las aplicaciones desplegarán su contenido.

Es importante aclarar que la capa de gráficos probablemente tenga una resolución diferente a la de video y fondo. Adicionalmente los tamaños de los píxeles también serán diferentes.

Capas

Como se mencionó anteriormente existen tres capas diferentes de imagen en los dispositivos MHP:

- Imagen de tamaño completo, coloreada.

- Imagen de tamaño completo o no.
- Gráficos de la aplicación.

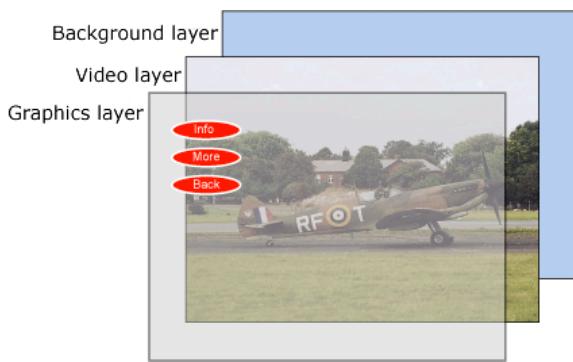


Figura 2.20: Capas de imagen en los dispositivos MHP

Cada capa será configurada por separado, pero no de manera independiente, ya que las configuraciones deben ser compatibles.

2.3.4.2 Clases

HScreen

MHP y HAVI definen la clase llamada **HScreen** que facilita la configuración de las tres capas de manera compatible. Cada receptor MHP tendrá una instancia **HScreen** por cada dispositivo de pantalla al que se encuentre conectado, por lo general uno solo.

Cada **HScreen** incluye una serie de objetos **HScreenDevices** que representan a las diferentes capas:

- **HBackgroundDevice**
- **HVideoDevice**
- **HGraphicsDevice**

A continuación se define la clase HScreen con referencia a los diferentes HScreenDevices:

```
public class HScreen

{
    public static HScreen[ ] getHScreens();

    public static HScreen getDefaultHScreen();

    public HVideoDevice[ ] getHVideoDevices();

    public HGraphicsDevice[ ] getHGraphicsDevices();

    public HVideoDevice getDefaultHVideoDevice();

    public HGraphicsDevice getDefaultHGraphicsDevice();

    public HBackgroundDevice getDefaultHBackgroundDevice();

    public HScreenConfiguration[] getCoherentScreenConfigurations();

    public boolean setCoherentScreenConfigurations(
        HScreenConfiguration[ ] hsca);
}
```

A través de esta clase se puede acceder a los dispositivos por defecto y a todos los existentes. Sin embargo sólo existe un único fondo. En el caso de los dispositivos de video, éstos pueden ser más de uno ya que existe la posibilidad de utilizar por ejemplo picture-in-picture. La figura a continuación muestra esta relación de cantidades:

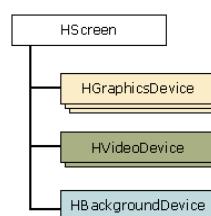


Figura 2.21: HScreen, HDevices

2.3.4.3 Configuración de los ScreenDevices

Para configurar cada ScreenDevice existe la clase HScreenConfiguration. A través de ésta se puede setear la relación de aspecto de los pixeles, la relación de aspecto, la resolución de la pantalla, etc. Para esto se utilizan instancias de la clase HScreenConfigTemplate como se muestra a continuación:

(Este ejemplo es para el GraphicsDevice, pero es aplicable al resto de los dispositivos de pantalla)

```
public class HGraphicsDevice extends HScreenDevice {  
  
    public HGraphicsConfiguration[ ]  getConfigurations();  
  
    public HGraphicsConfiguration getDefaultConfiguration();  
  
    public HGraphicsConfiguration getCurrentConfiguration();  
  
    public boolean setGraphicsConfiguration(HGraphicsConfiguration hgc)  
throws SecurityException, HPermissionDeniedException,  
HConfigurationException  
  
    public HGraphicsConfiguration getBestConfiguration  
(HGraphicsConfigTemplate hgct);  
  
    public HGraphicsConfiguration getBestConfiguration  
(HGraphicsConfigTemplate hgcta[ ]);  
}
```

Los tres primeros métodos permiten obtener las configuraciones disponibles, las que se encuentran establecidas por defecto y las que están actualmente en uso.

El método setGraphicsConfiguration() permite setear una configuración y lanza una excepción si la misma es incompatible con la del resto de las capas.

Antes de setear la configuración se debe obtener dicha configuración mediante el método getBestConfiguration(), que toma como parámetro un array HgraphicsConfigTemplate y devuelve la configuración aceptable más aproximada a la deseada. Para esto el receptor chequea las preferencias especificadas y si puede cumplirlas retorna la clase mencionada, en caso contrario retorna NULL.

HGraphicsConfigTemplate

Esta clase permite setear las preferencias de configuración. Cada preferencia está caracterizada por un valor y una prioridad, lo que le da libertad a la aplicación para elegir la configuración que desea y al receptor de cumplirlas en el grado que le sea posible.

Algunas de las preferencias son:

- ZERO_GRAPHICS_IMPACT: implica que la configuración escogida NO tendrá influencia sobre los gráficos de las aplicaciones que ya se encuentran en ejecución.
- ZERO_VIDEO_IMPACT : implica que la configuración escogida no tendrá influencia sobre los videos que se encuentran desplegados en pantalla.
- VIDEO_GRAPHICS_PIXEL_ALIGNED: indica que los pixeles de video y gráficos deben estar perfectamente alineados.

Las prioridades que pueden asignarse son las siguientes:

- REQUIRED: preferencia que debe cumplirse de manera obligatoria.
- PREFERRED : preferencia deseada pero puede ser ignorada si es necesario.
- UNNECESSARY: No existe preferencia al respecto.
- PREFERRED_NOT: la preferencia no debe tomar el valor especificado en caso de que sea posible.
- REQUIRED_NOT: la preferencia no debe tomar el valor especificado.

Aspectos de configuración de la capa de Fondo (Background)

Como se mencionó anteriormente, el fondo puede ser un color sólido, o en el mejor de los casos una foto en formato MPEG-1.

Si el HbackgroundConfigTemplate especifica que se desea una imagen entonces la configuración retornada será una clase HstillImageBackgroundConfiguration que tendrá los siguientes métodos para establecer la foto a utilizar:

```
public void displayImage(HBackgroundImage image);  
  
public void displayImage(HBackgroundImage image, HScreenRectangle r);
```

La clase HBackgroundImage está destinada al manejo de imágenes y su interfase es la siguiente:

```
public class HBackgroundImage  
{  
  
    public HBackgroundImage(String filename);  
  
    public void load(HBackgroundImageListener l);  
  
    public void flush();  
  
    public int getHeight();  
  
    public int getWidth();  
  
}
```

Cambio de configuraciones

Bajo este contexto es muy probable que durante la ejecución de una aplicación, la configuración de las 3 capas sea modificada por otra aplicación.

Si una aplicación desea conocer el estado de la configuración entonces debe recurrir al HScreenConfigurationEvent. Para esto debe convertirse en un Listener de ésta a través del método addScreenConfigurationListener(). Luego de invocar este método, las aplicaciones serán notificadas sólo en aquellos casos en que la configuración se haya modificado y esta modificación no sea compatible con la configuración del Template pasado como parámetro al método addScreenConfigurationListener().

Ejemplo: Lo que debe ejecutar una aplicación

```
//Se obtiene el HScreen
HScreen screen = HScreen.getDefaultHScreen();

// Se obtiene el HGraphicsDevice
HGraphicsDevice device= screen.getDefaultHGraphicsDevice();
//Se crea un nuevo template para setear las preferencias
HGraphicsConfigTemplate template= new HGraphicsConfigTemplate();

//Seteo de preferencias: image scaling, no afectar el resto de las
aplicaciones,
template.setPreference(template.IMAGE_SCALING_SUPPORT,
template.PREFERRED);
template.setPreference(template.ZERO_GRAPHICS_IMPACT, template.REQUIRED);

//Se obtiene la configuración posible que considera nuestras
//preferencias
HGraphicsConfiguration configuration=
device.getBestConfiguration(template);

//Se setea la configuración y se verifica que lo obtenido no sea Null
if (configuration != null) {
    try {
        device.setGraphicsConfiguration(configuration)
    }
    catch (Exception e) {
        // Ignoro
    }
}
```

2.3.4.4 Coordenadas de posicionamiento

Para definir la posición de un elemento en pantalla es posible utilizar diferentes sistemas de coordenadas.

El origen de todo sistema de coordenadas es la esquina superior izquierda, y alcanza el máximo valor en la esquina inferior derecha.

Coordenadas normalizadas

Origen: (0,0)

Máximo valor: (1,1)

La ventaja de este sistema es que permite posicionar los elementos conociendo la posición relativa entre ellos, sin necesidad de conocer la resolución de la pantalla.

Coordenadas de pantalla

Origen: (0,0)

Máximo valor: (X,Y)

Este sistema de coordenadas depende de la resolución de la pantalla. MHP especifica que (X,Y) será al menos (720,576).

Coordenadas AWT

Este sistema está basado en el posicionamiento de pixeles, y no considera la pantalla completa, sino que toma una porción de la misma a la que se le denomina HScene, y allí despliega la aplicación.

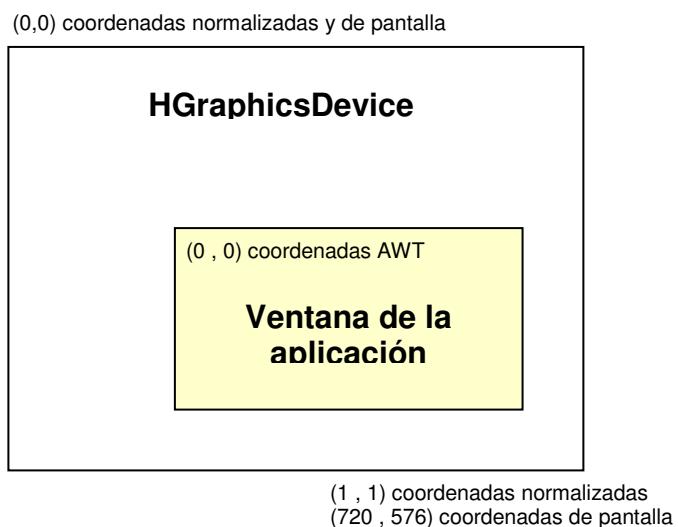


Figura 2.22: Coordenadas de posición en una pantalla

2.3.4.5 HScene

Dado que los STBs tienen recursos limitados (CPU, memoria, etc.), no es viable que se haga uso de un administrador de ventanas. En su lugar se define la clase HScene, y existirá una por aplicación. La cualidad más destacada de este modo de administración de ventanas es que una aplicación no puede ver más que su propio HScene, por lo que las actividades gráficas de cada aplicación son completamente aisladas.

HScene hace uso de la jerarquía AWT (Abstract Window Toolkit) y la ordena de la siguiente manera:

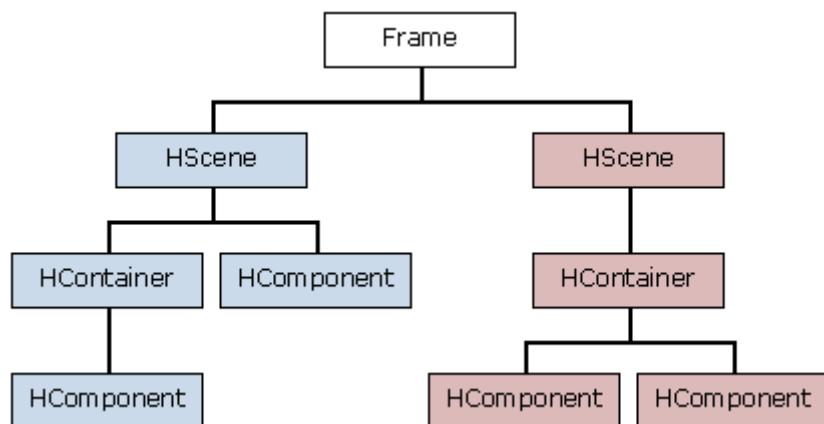


Figura 2.23: Frame, HScene, HContainer, HComponent

El HScene permitirá a una aplicación tener control sobre la relación de aspecto de la pantalla de TV, tamaño de la misma, ubicación, transparencia entre capas, etc.

Para crear un HScene se hace uso de la clase org.havi.ui.HSceneFactory a la que se le ingresan los parámetros deseados y se obtiene el HScene soportado por la plataforma. Tal como en los ScreenDevices, en este ámbito también existe un Template que permite el seteo de parámetros, y a cada uno se le asocia la prioridad correspondiente.

```

public class HSceneTemplate extends Object {

    // prioridades

    public static final int REQUIRED;

    public static final int PREFERRED;
  
```

```
public static final int UNNECESSARY;

// preferencias a setear

public static final Dimension LARGEST_DIMENSION;

public static final int GRAPHICS_CONFIGURATION;

public static final int SCENE_PIXEL_RESOLUTION;

public static final int SCENE_PIXEL_RECTANGLE;

public static final int SCENE_SCREEN_RECTANGLE;

// métodos para setear y obtener prioridades
public void setPreference(int preference, Object object, int priority);

public Object getPreferenceObject(int preference);

public int getPreferencePriority(int preference);

}
```

Ahora, a partir del Template se obtendrá el HScene más apropiado:

```
public class HSceneFactory extends Object{

    public static HSceneFactory getInstance();

    public HSceneTemplate getBestSceneTemplate( HSceneTemplate hst );

    public HScene getBestScene(HSceneTemplate hst);

    public void dispose(HScene scene);

    public HSceneTemplate resizeScene( HScene hs,HSceneTemplate hst )
throws java.lang.IllegalStateException;

    public HScene getSelectedScene( HGraphicsConfiguration selection[
],HScreenRectangle screenRectangle, Dimension resolution);
```

```
    public HScene getFullScreenScene( HGraphicsDevice device, Dimension
resolution);
}
```

Es importante destacar que el método `dispose()` debe ser utilizado para liberar los recursos obtenidos por el `HScene` que ya no se utilizarán.

2.3.4.6 Gráficos de alto nivel

Los formatos de imagen soportados por MHP son los siguientes:

- GIF
- JPEG
- PNG
- MPEG I-frame

2.3.4.7 Color

Los gráficos de Java hacen uso de la tripleta de colores RGB, mientras que las señales de TV hacen uso de la tripleta de colores diferencia YUV. Sin embargo esto no presenta un problema para el desarrollador, debido a que MHP maneja de manera transparente la conversión de un sistema al otro.

Vale la pena aclarar que el color de una imagen será afectado por la cantidad de colores manejados por el receptor MHP. Generalmente éstos soportan colores de 24-bits, pero MHP sólo exige un mínimo de 8 bits. Esto debe ser tenido en cuenta por los desarrolladores y diseñadores a la hora de elegir los colores a utilizar en la aplicación, ya que si se desea desplegar una imagen de color cuya cantidad de bits sea mayor a la soportada por el receptor entonces cada píxel será modificado hacia el color más cercano disponible, y esto puede afectar la estética de la imagen.

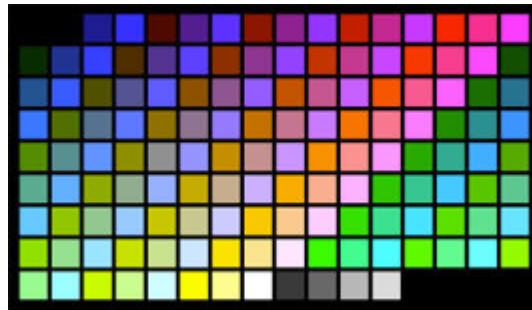


Figura 2.24: Gama de colores soportada por los receptores MHP

2.3.4.8 **Objetos de menú**

MHP es capaz de manejar los siguientes elementos:

- Botones
- Check boxes
- Radio buttons
- Icons
- Scrollable lists
- Dialog boxes
- Text input

Es posible modificar la estética de estos elementos a través de la redefinición del método paint() de la clase HLook.

2.3.5 **Presentación de texto**

La mayoría de las aplicaciones normalmente despliegan algún tipo de texto en pantalla, y para esto los métodos estándares de Java pueden ser utilizados. Sin embargo deben considerarse los siguientes puntos al desplegar texto en una pantalla de TV:

- No desplegar grandes cantidades de texto en la pantalla, en su lugar usar scrollable text o mejor aún jump scrolling.

- No hacer scrolling horizontal.
- La TV tiene baja resolución y contraste, lo que dificulta la lectura.
- La TV es visualizada generalmente a una distancia que dificulta la lectura.
- Utilizar fuentes claras.
- Utilizar fuentes de tamaños apropiados (18 puntos mínimo).
- Establecer un espacio entre líneas que permita la fácil lectura.

2.3.5.1 **Diseño de texto**

Es tarea de la aplicación chequear que el texto quepa en el espacio asignado para el mismo. Para esto los desarrolladores pueden hacer uso de la clase `HTextLayoutManager`, que desplegará el texto teniendo cuidado que no exceda el espacio asignado, y en caso de que el texto exceda este umbral se agregará el símbolo “(...)”.

2.3.5.2 **HTextLayoutManager**

Todo objeto `HVisible` puede tener asociado un `HTextLayoutManager` y puede setearse a través del método `Hvisible.setTextLayoutManager()`. Se debe tener en cuenta que si bien con el Layout manager se puede controlar el texto, éste no puede ser formateado en caso de que exceda sus límites de posicionamiento.

El ejemplo a continuación manipula un objeto `HText`:

```
java.lang.String miString;
org.havi.ui.HText miTexto;
org.dvb.ui.DVBTextLayoutManager mgr;

// texto a desplegar
miString = "Este es mi texto. Es bastante largo" +
"por lo que en la mayoría de los receptores" +
"debería ser cortado.\nMúltiples líneas serán desplegadas" +
"por el layout manager si estan separadas" +
"por un \\n";
```

```
// seteo de la fuente y tamaño
miFuente = new java.awt.Font("Tiresias", java.awt.Font.PLAIN, 18);

// Crea un nuevo text layout manager
mgr = new org.dvb.ui.DVBTTextLayoutManager();

// Seteo de parámetros para controlar el tamaño del texto
mgr.setLineSpacing(23);
mgr.setHorizontalAlign(org.dvb.ui.DVBTTextLayoutManager.HORIZONTAL_CENTER);
;
mgr.setVerticalAlign(org.dvb.ui.DVBTTextLayoutManager.VERTICAL_CENTER);
mgr.setTextWrapping(true);

/*Crea un Htext object seteando fuente, colores de foreground y
background y el layout manager*/
miTexto = new HText( miString, miFuente, java.awt.Color.white,
java.awt.Color.blue, mgr);

// Posicionamiento del HText
miTexto.setBounds(10, 10, 400, 300);
```

2.3.5.3 Uso de HTML

Los servicios de información generalmente están basados en el despliegue de texto. Éste será actualizado por personal encargado de dicha función y no por el desarrollador, y será almacenado en archivos que luego serán utilizados por la aplicación. Para realizar esto de manera simple se puede hacer uso de HTML. Aunque no todos los receptores MHP lo soportan, es posible introducir navegadores HTML en las aplicaciones transmitidas ya que existen muchas compañías que venden componentes HTML escritos en Java y pueden ser incorporados de manera muy sencilla.

2.3.5.4 Fuentes

Se debe tener en cuenta que no todos los receptores despliegan una misma fuente de igual manera, los siguientes parámetros pueden ser distintos:

- espacio entre letras
- alto de las letras

- ancho de las mismas

Por esta razón es que resulta apropiado una vez desarrollada la aplicación testearla en diferentes receptores.



Figura 2.25: Despliegue de una misma fuente en diferentes STBs

2.3.6 Interacción con el usuario

La interacción con el receptor MHP se realiza a través del control remoto. Para esto MHP tiene definidos los siguientes eventos que se corresponden con la presión de ciertas teclas de dicho control:

Nombre del evento	Botón asociado	Código (si aplica la estandarización)
VK_UP	Flecha arriba	No estandarizado
VK_DOWN	Flecha abajo	No estandarizado
VK_LEFT	Flecha izquierda	No estandarizado
VK_RIGHT	Flecha derecha	No estandarizado
VK_ENTER	Enter/OK	No estandarizado
VK_0 to VK_9	Números	48 - 57
VK_TELETEXT	Teletext	459
VK_COLORED_KEY_0	1º tecla de color	403
VK_COLORED_KEY_1	2º tecla de color	404
VK_COLORED_KEY_2	3º tecla de color	405
VK_COLORED_KEY_3	4º tecla de color	406

Tabla 2.7: Eventos relacionados al control remoto

2.3.6.1 Detección de key events

Usualmente las aplicaciones se despliegan cuando el usuario presiona un botón. Para que la aplicación pueda detectar esta interacción se utiliza la API org.dvb.events que

incluye el método addUserEventListener(), a través del cual se permite a una aplicación estar pendiente de las peticiones del usuario.

Para que una aplicación pueda establecer este Listener debe antes definir un UserEventRepository en el cual declare a qué eventos prestará atención.

2.3.6.2 UserEventRepository

```
public class UserEventRepository {  
  
    public UserEventRepository (String name);  
  
    public void addUserEvent (UserEvent event);  
  
    public UserEvent[] getUserEvent ();  
  
    public void removeUserEvent (UserEvent event);  
  
    public void addKey (int keycode);  
  
    public void removeKey (int keycode);  
  
    public void addAllNumericKeys();  
  
    public void addAllColourKeys();  
  
    public void addAllArrowKeys();  
  
    public void removeAllNumericKeys();  
  
    public void removeAllColourKeys();  
  
    public void removeAllArrowKeys();  
  
}
```

2.3.6.3 EventManager

Luego de haber definido el UserEventRepository, la aplicación deberá ejecutar el método addUserEventListener(), para esto debe obtener la instancia de la clase EventManager a través del método EventManager.getInstance(). El acceso a los eventos es un recurso

limitado que puede ser retirado de las aplicaciones en cualquier momento. Para verificar el estado de los recursos se puede recurrir a ResourceStatusEvents.

2.3.6.4 Acceso exclusivo a key events

Cuando una aplicación solicita al usuario el ingreso de información sensible (contraseñas, números de cuenta, etc.) es necesario que la aplicación tenga acceso exclusivo a los eventos, para esto existen dos métodos a utilizar:

- addExclusiveAccessToAWTEvent(UserEventRepository u, ResourceCliente r)
- removeExclusiveAccessToAWTEvent()

El ResourceClient es a quién se debe notificar cuando el acceso exclusivo a ciertos eventos es dado de baja.

Sólo una aplicación puede tener acceso exclusivo a cierto recurso en un momento dado, por lo que los eventos pueden ser declarados como recursos escasos.

2.3.7 Referenciar contenidos en MHP

MHP maneja dos APIs para controlar la presentación de las señales de video y determinar cuáles van a ser presentadas al usuario. La primera es el Java Media Framework (JMF), que es utilizado principalmente para controlar de forma individual los clips de video. La segunda opción es la API de selección de servicios de JavaTV, que fue diseñada con el objetivo de permitir a las aplicaciones cambiar un servicio completo, por lo que utilizarla para manipular tramas de video y audio es demasiado riesgoso ya que la aplicación podría ser abortada.

Antes de profundizar en cómo manipular y controlar los diferentes tipos de información en MHP, se debe conocer cómo se hace referencia a los mismos. Es aquí donde entran los locators.

2.3.7.1 Locators

Los locators se utilizan para referirse a una parte específica de información. O sea, todos los archivos a los que se quiera acceder deben estar bajo un locator. Se dice que los locators son piezas opacas ya que no exponen directamente la ruta de los archivos, sino

que los referencian a través de punteros, números o cualquier otro tipo de identificador que la aplicación desee utilizar. Sin embargo los locators tienen un formato externo similar al de una dirección URL, definido especialmente para referirse a servicios DVB y que más adelante se detallará.

La necesidad de utilizar locators en vez de utilizar simplemente una dirección URL para referirse a la información es que en Java la clase `java.net.URL` es declarada como final, o sea que no puede ser extendida por otras clases, y por lo tanto diferentes grupos han tenido la necesidad de proveer sus propias representaciones de algo tan básico como la URL. Es por eso que existen una serie de familias de locators, las mismas se pueden dividir en tres jerarquías, estas son: `javax.media.MediaLocator`, `javax.tv.locator.Locator` y `org.davic.net.Locator`.

La lista completa de locators es la siguiente:

- `javax.media.MediaLocator`
- `org.davic.media.MediaLocator`
- `javax.tv.locator.Locator`
- `org.davic.net.Locator`
- `org.davic.net.dvb.DvbLocator`
- `org.davic.net.dvb.DvbNetworkBoundLocator`
- `org.dvb.locator.FrequencyLocator` (agregado en MHP 1.1.2)
- `org.dvb.locator.NetworkInterfaceBoundMediaLocator` (agregado en MHP 1.1.2)

Cada uno fue diseñado para diferentes aplicaciones y dependiendo de la aplicación que se utilice se escogerá el locator a utilizar.

2.3.7.2 Formatos de contenido en MHP

Las APIs MHP que mencionan contenido multimedia manejan tres tipos de formatos. Es importante destacar que muchas de las APIs no serán capaces de aceptar locators que

manején algunos de estos formatos. Solamente la Java Media Framework será capaz de manipular los formatos de contenido definidos a continuación.

1. Servicios DVB: Al trabajar con el formato de servicios DVB generalmente se hace uso de los locator DVB (org.davic.net.dvb.DvbLocator), este locator hace referencia a los servicios DVB que se crean a partir del siguiente formato de URL.

dvb://<onID>.<tsID>.<sID>[.<ctag>[&<ctag>]][;<evID>][<path>]

- onID: Original Network ID, identifica al broadcaster o a la red de origen del contenido.
- tsID: Transport Stream ID, identifica la trama de transporte que está siendo transmitida.
- sID: Service ID, hace referencia al servicio que se encuentra en la trama de transporte.
- ctag: Component Tag, hacer referencia a una trama elemental.
- evID: Event ID, identifica un evento específico dentro de un servicio.
- path: Representa la ruta de un archivo dentro del sistema de archivos del broadcaster que está siendo transmitido en esa trama elemental.

Todos estos componentes, excepto el path, son representados de forma hexadecimal. El path utiliza el formato estándar de URL.

Ejemplos:

dvb://123.456.789 (identifica un servicio DVB)
dvb://123.456.789;42 (identifica un evento DVB)
dvb://123.456.789/images/logo.gif (identifica un archivo dentro de un carrusel de objetos)
dvb://123.456.789.66 (identifica una trama elemental dentro de un servicio usando la etiqueta correspondiente.)

2. Clips de Audio MPEG: Este tipo de contenido es simplemente una pieza de audio MPEG-1 que se carga desde un archivo. Este archivo será guardado típicamente en el carrusel de objetos DSM-CC.

Encontrar el path completo del archivo no es fácil, ya que el montaje del objeto en el carrusel es dependiente de la plataforma. Para determinar correctamente la dirección URL del archivo es necesario acceder al archivo en sí.

```
/* Crear un objeto DSMCC que hace referencia a un archivo del carrusel.  
Note que este es un path relativo(relativo al directorio raíz de  
aplicación). En este caso un archivo de audio MPEG */  
  
DSMCCObject myDsmccObject;  
  
myDsmccObject = new DSMCCObject("my/file/path/filename.mpg");  
  
/*Podemos invocar al getURL() dentro del objeto para obtener el file URL  
al que apunta*/  
URL url = myDsmccObject.getURL();  
  
//Ahora creamos el locator  
try {  
    LocatorFactory locatorFactory = LocatorFactory.getInstance();  
    Locator myLocator;  
    myLocator = locatorFactory.createLocator(  
        url.toString());  
}  
catch (MalformedLocatorException e)  
{  
    // Capuramos la exception  
}
```

3. “DRIPS” de Video: Este es un nuevo formato de contenido que básicamente es usado en el mundo de la TV digital. El principal objetivo de este formato es proporcionar una forma de desplegar una secuencia de imágenes similares (alta correlación entre ellas) de manera eficiente desde el punto de vista del consumo de memoria. Se basa en los conceptos de MPEG-2, en donde primero se decodifica una trama MPEG-2 I y luego se decodifican una o más tramas

MPEG-2 P que se basan en la predecesora trama MPEG-2 I. Esto hace que se ahore significativamente en el uso de memoria.

En este formato la información es pasada al JMF como un array de bytes, es por eso que el contenido es cargado directamente desde un archivo o una fuente de información. Debido a esto es que no se necesita un Locator que identifique de dónde cargar los datos. En cambio se utiliza una URL fijo para crear el Locator.

La URL completa para un “drip” de video es: dripfeed://

Este le dice al locator qué tipo de formato de contenido es, sin tener que referirse a una pieza específica de información.

2.3.8 Java Media Framework

JavaTV y MHP confían en el uso del Java Media Framework (JMF) en lo que refiere al control de contenidos tanto de audio como de video. Es utilizado para controlar qué tipo de contenido será desplegado y determinar cómo se presentarán los flujos de video que fueron transmitidos. Algunas aplicaciones también utilizan el JMF para reproducir archivos de audio y visualizar formatos especiales de imágenes.

JMF fue diseñado para la utilización en PCs, en donde la información se almacena en diferentes medios y no llega al estilo broadcast. A pesar de esto fue adoptado por la norma DVB-MHP, ya que los requerimientos de las aplicaciones son similares y no tiene sentido “reinventar la rueda”. Además muchos de los programadores ya cuentan con el conocimiento para trabajar con JMF lo que representa una fortaleza importante a considerar.

JMF maneja tres conceptos que son necesarios conocer para utilizarlo de manera eficiente.

- El elemento más importante es el Reproductor JMF (Players). Esta es una clase que se encarga de decodificar y reproducir la información transmitida. Cada reproductor tiene un conjunto de controles asociados a él.
- Como segundo elemento se define el Control JMF que es una extensión de los objetos Players que permite agregar diferentes funcionalidades sin la necesidad

de crear una subclase. Por ejemplo los objetos Control son típicamente usados para proporcionar la funcionalidad de frame-freeze, o por ejemplo habilitar diferentes opciones de lenguaje.

- El tercer elemento es el DataSource (fuente de información). Este objeto se encarga de obtener la información multimedia que el reproductor efectivamente decodificará.

Al separar el Player del DataSource, uno se encarga de decodificar y presentar información multimedia, y el otro de obtener dicha información. Así se hace más fácil decodificar y reproducir al mismo tiempo información de una misma fuente.

Una aplicación MHP puede manipular el objeto DataSource y obtener la información de una sola interfase sin tener en cuenta dónde efectivamente se encuentra la misma.

2.3.8.1 El proceso de creación del Player

Para crear un Player existen varias opciones, en el siguiente ejemplo se creará un Player a partir de una URL.

Como primer paso la aplicación debe invocar el método estático Manager.createPlayer() con la siguiente URL:

```
http://www.example.net/media/SomeContent.mp3
```

El Manager se encarga de examinar el protocolo que se va utilizar para acceder a los datos, en este caso HTTP, y luego utiliza estos datos para crear el nombre de la clase del objeto DataSource como se muestra a continuación.

```
<class prefix>.<protocol name>.DataSource
```

Continuando con el ejemplo se forma:

```
Com.steven.media.protocol.http.DataSource
```

Donde com.steven.media.protocol es el prefijo de la clase.

El objeto DataSource se utilizará para obtener la información multimedia que será presentada. Una vez que el DataSource fue cargado e instanciado, el Manager lo utiliza para conectarse a la ubicación determinada en la dirección URL.

Luego de que la conexión fue establecida, el Manager invocará el método `getContentType()` para encontrar el tipo de contenido MIME (Multi-Purpose Internet Mail Extensions). Este contenido MIME es usado para construir el nombre de la clase del Player que será cargado. El nombre de la clase tiene la siguiente forma:

```
<class prefix>.<MIME type>.<MIME subtype>.Player
```

Entonces si el tipo de contenido MIME es audio/mp3 el nombre de la clase resultante del Player será:

```
Com.steven.media.players.audio.mp3.Player
```

Una vez que el Manager haya construido el nombre de la clase del objeto Player, la misma es cargada e instanciada. Luego de que fue instanciada el Manager invoca el método `Player.setSource()` para asociar el Player con el origen de los datos. Con esto termina el proceso de creación y el Player es retornado a la aplicación.

Estados del Player

Existen diferentes estados en los que un Player se puede encontrar dependiendo de su función en cada momento. Se definen los siguientes estados principales:

- Unrealized
- Realized
- Prefetched
- Started.

Cuando un Player es creado su estado es “Unrealized”, invocando el método `realize()`, el Player pasa al estado “Realized”, en donde cuenta con toda la información que necesita para obtener los recursos necesarios para la reproducción. Esto no significa que tendrá

inmediatamente todo los recursos, probablemente obtenga algunos excepto aquellos para la decodificación MPEG.

Invocando el método prefetch(), se pasa al estado con ese mismo nombre. En este punto el Player tiene todos los recursos necesarios para la correcta reproducción de la información.

El estado final es “Started”, al que se llega cuando el método start() es invocado. Cuando un Player se encuentra en este estado, es que efectivamente está reproduciendo algún contenido.

Cualquiera de los métodos realize(), prefetch() o start() pueden ser invocados en cualquiera de los estados antes mencionados, ya que en el caso de que se necesite pasar por un estado intermedio los métodos faltantes se invocarán automáticamente.

El método stop() lleva al Player desde el estado “Started” al estado “Prefetched” o “Realized”, dependiendo del tipo de información que esté reproduciendo.

El método deallocate() es usado para liberar recursos que están siendo utilizados por el Player. Al invocar este método desde el estado “Realized” o superior, el Player volverá al estado “Realized”, si no se encuentra en estado “Realized” se pasará al estado Unrealized.

El método close() liberará los recursos y destruirá al Player para que éste no sea utilizado nuevamente.

2.3.8.2 Los eventos del player

JMF utiliza diferentes eventos para notificar a las aplicaciones interesadas sobre los cambios que se dan respecto al estado de los Players. Mediante la invocación del método Player.addControllerListener() se permite a las aplicaciones registrarse para recibir eventos de los Players. También es posible notificar acerca de otros eventos. La siguiente tabla describe algunos de los eventos más comunes:

Evento	Descripción
DataStartedEvent	Indica que el Player se ha detenido debido a que la información no ha sido liberada lo suficientemente rápido desde la fuente de información.
EndOfMediaEvent	Indica que el clip multimedia ha llegado a su fin.
RestartingEvent	Indica que el Player se ha detenido.
StopAtTimeEvent	Indica que el Player ha alcanzado el tiempo estipulado que se le había determinado y se ha detenido automáticamente.
StopByRequestEvent	Indica que una aplicación ha invocado el método stop() del Player.
MediaPresentedEvent	Generado cuando el Player empieza a presentar contenido multimedia al usuario. Este momento puede no coincidir con la invocación del método start() y por temas de sincronismo es necesario saber cuándo el usuario empieza a visualizar el contenido.

Tabla 2.8: Eventos de un player

Asimismo, existen otros eventos que se encargan de notificar los posibles problemas que se pueden dar con la conexión a la fuente de información. También se definen los eventos ResourceWithdrawnEvent y ResourceReturnedEvent que permiten a las aplicaciones saber sobre los diferentes recursos que conciernen al Player. El ResourceAvailableEvent indica que el Player ha perdido recursos imprescindibles para él y no puede continuar su ejecución sin los mismos. Este evento se puede generar en el transcurso del ciclo de vida de los Player y no sólo cuando el mismo está en ejecución.

2.3.8.3 Restricciones del JMF Player, sintonización

La restricción primordial del JMF Player es que a no ser que sea iniciado explícitamente, ninguna API sintonizará el receptor MHP al TS deseado. La aplicación debe explícitamente sintonizar el TS requerido, usando el sistema de sintonización provisto por la API o algún otro mecanismo que se encargue de dicha tarea. Cualquier servicio que se encuentre en el TS actual estará disponible para ser consumido, pero no podrán consumirse servicios que se encuentren en otros TS a no ser que explícitamente se haga la sintonización.

2.3.8.4 Controladores

JMF utiliza el concepto de controladores para agregar funcionalidades extra a los Players. Esto permite a una clase de control manejar los diferentes Players, lo que significa que el

Player tiene que agregar poca implementación para proporcionar funcionalidades adicionales.

Mediante la invocación del método Player.getControls() se pueden conocer los diferentes controles disponibles que tiene el Player. Los controladores no tienen una API estándar y usualmente son diseñados para aprovechar el hecho de que las interfaces no son públicas. Es por esta razón que es difícil para una aplicación manipular el Player de otra forma que no sea a través de un controlador.

2.3.8.5 Formato especial MHP: video “drip”

Este formato de contenido es diferente al resto, ya que es el único formato donde los contenidos multimedia no se cargan automáticamente por el implementador JMF. Este formato es útil cuando se desea transmitir imágenes progresivas que son similares entre ellas, por ejemplo agregar botones de acción entre una imagen y la otra, entonces lo que se codifica en la trama MPEG-2 es el I-frame y luego es P-frame, que es la diferencia entre las dos imágenes. Esto ahorra sustancialmente la memoria requerida por el STB.

Como se describió anteriormente el Player JMF saca la información del objeto desde el DataSource. Esta información es usualmente cargada de forma automática por dicho objeto. Pero en el caso de la reproducción de video drip, la fuente de datos es una instancia de la clase org.dvb.media.DripFeedDataSource. Esta clase tiene el método feed(), que recibe como argumento un array de datos.

2.3.9 La API de filtros de sección de DAVIC

Las secciones privadas de la trama MPEG-2 no están limitadas al contenido de la información de servicios, ya que pueden contener cualquier tipo de datos que el broadcaster quiera insertar. Los objetos del carrusel DSM-CC son transmitidos en las secciones privadas. También se incluye información propietaria, formatos de filesystem o cualquier otro tipo de información privada.

Claro que si los broadcasters hacen uso de las secciones privadas, deben asegurarse de que las aplicaciones tengan acceso a este tipo de información. En este contexto es que entran los filtros, para justamente filtrar las secciones privadas de la trama de transporte

que son de interés. La librería que contiene dichos filtros se encuentra dentro del paquete org.davic.mpeg.sections.

El filtrado de secciones es una operación asíncrona. Las aplicaciones setean los filtros, y levantan un Listener que se encarga de capturar los eventos que se disparan cuando se produce un matcheo en los filtros. O sea, cuando se matchea el filtro propuesto por la aplicación, se captura el evento y luego se ejecutan las instrucciones posteriores a la captura. Estos filtros pueden ser seteados con diferentes criterios, uno de los más utilizados es por PID. Los filtros pueden ser positivos o negativos, por lo que se pueden aplicar tanto filtros de exclusión como de matcheo específico.

Uno de los aspectos a tener en cuenta es que los filtros son un recurso escaso, y es por eso que las aplicaciones deben tener precaución al momento de manejar los filtros, y sólo utilizarlos cuando sea necesario. Los filtros pueden ser implementados por software o por hardware ya que la API maneja ambas opciones, pero se recomienda, por cuestiones de performance, utilizar filtros de hardware.

2.3.9.1 Descripción de la API

La API establece cuatro clases principales, éstas son:

- Section: Representa un sección MPEG-2 y permite el acceso a los datos de la misma.
- SectionFilter: Representa un filtro real que puede ser implementado por el receptor, tanto por hardware como por software.
- SimpleSectionFilter: Representa un filtro simple que filtra una sección única antes de detenerse. Luego puede ser inicializado nuevamente.
- SectionFilterGroup: Permite que el receptor y la aplicación manejen los recursos de los filtros.

Por último se define la interfase SectionFilterListener que debe ser implementada por las aplicaciones para que éstas puedan recibir los eventos asociados a los filtros cuando se realiza el matcheo.

2.3.10 Seleccionando un nuevo servicio

En televisión analógica al cambiar de canal la TV desplegará audio y video diferentes a los que se estaban mostrando hasta ahora. Pero en TV digital es posible cambiar tanto el audio como el video que se está presentando sin tener que cambiar de canal. Cambiar de canal en un sistema de televisión digital (seleccionar un nuevo servicio técnicamente hablando) puede llegar a terminar la aplicación interactiva que se estaba ejecutando.

JavaTV permite extender este concepto para decir que un servicio es una colección de información que incluye video, audio, y aplicaciones que pueden ser representadas en conjunto como una entidad coherente, como se muestra en el siguiente gráfico:

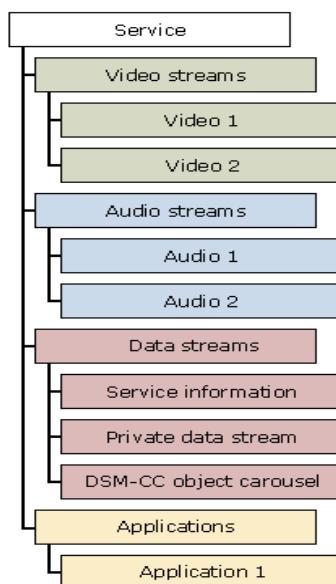


Figura 2.26: Servicios de TV digital

Dado que las aplicaciones están estrechamente ligadas a los servicios, el hecho de seleccionar un nuevo servicio es una operación de riesgo y hará que la aplicación se termine a no ser que sea tenida en cuenta por el nuevo servicio.

2.3.10.1 Sintonizando nuevos servicios

El hecho de querer seleccionar otro servicio que se encuentra en una nueva trama de transporte no es algo trivial y es por eso que existe una API que se encarga de proporcionar herramientas para realizar de forma correcta esta sintonización.

En el caso de tratarse de la misma trama simplemente se aplica un filtro para obtener los paquetes asociados al servicio que sean de interés para la aplicación, pero al tratarse de tramas de transporte distintas la situación es más compleja. Cada trama de transporte (TS) es transmitida a una frecuencia diferente, por lo que cambiar de una trama a otra resulta una tarea compleja. Esto se debe en primer lugar a que el receptor tiene que saber cuál va a ser la nueva frecuencia donde se transmitirá la trama de transporte, y en segundo lugar porque el receptor debe indicarle al sintonizador que efectivamente cambie a la nueva frecuencia para obtener el servicio correctamente. Por último el receptor debe examinar esta nueva trama de transporte, específicamente la información de servicio.

2.3.10.2 Locators y tramas de transporte

Como se vio anteriormente los locators se utilizan para referirse tanto a las tramas de transporte como a los servicios. Sin embargo la API de control de sintonización introduce nuevos conceptos que no se deben pasar por alto.

Las clases que se encuentran dentro de esta API, detallan el acceso a bajo nivel, a nivel de tramas de transporte, servicios y tramas elementales. El objeto más importante es el TransportStream. Al igual que los locators, éstos representan una referencia a las tramas de transporte pero con una diferencia muy sutil. Mientras que los locators proporcionan una referencia abstracta de una trama de transporte (como una dirección URL), el objeto TransportStream referencia la trama de transporte accedida a través de una interfase de red específica. Esto resulta importante ya que MHP soporta tres tipos de interfaces de broadcast: satelital, cable o terrestre. Cada una de estas interfaces tiene diferentes características y es bueno que las aplicaciones sepan cuál se está utilizando.

2.3.10.3 Servicios y Contextos de Servicios

Cada servicio es presentado dentro de un contexto de servicio, esto resulta útil para las aplicaciones ya que les permite identificar a qué servicio pertenecen y de esta forma poder seleccionar nuevos.

Un receptor tiene uno o más contextos de servicios, existiendo un número fijo máximo de éstos, pudiendo cada contexto de servicio presentar un servicio diferente.

Cada contexto de servicio contendrá dos sets de objetos. Uno de estos sets estará compuesto por los objetos player JMF que presentarán el audio y video del servicio, el otro set corresponde al de aplicaciones que correrán con dicho servicio.

2.3.10.4 Utilizando la API de selección de servicios

Los contextos de servicio son creados por la clase `javax.tv.service.selection.ServiceContextFactory`. Este es un objeto singleton y una referencia al mismo se obtiene a través del método `getInstance()`. El resto de los métodos permite a las aplicaciones acceder al contexto de servicio.

```
public abstract class ServiceContextFactory extends java.lang.Object {  
  
    public abstract ServiceContext createServiceContext();  
  
    public static ServiceContextFactory getInstance();  
  
    public abstract ServiceContext getServiceContext(XletContext ctx);  
  
    public abstract ServiceContext[] getServiceContexts();  
  
}
```

El método `getServiceContext()` devuelve un array con los contextos de servicio a los que la aplicación puede acceder, típicamente a los que pertenece.

```
public interface ServiceContext {  
  
    public void select(Locator[] components) throws  
InvalidLocatorException, InvalidServiceComponentException,  
java.lang.SecurityException;  
  
    public void select(Service selection) throws  
java.lang.SecurityException;  
  
    public void stop() throws java.lang.SecurityException;  
  
    public void destroy() throws java.lang.SecurityException;  
  
    public Service getService();  
  
    public ServiceContentHandler[] getServiceContentHandlers() throws  
java.lang.SecurityException;  
  
    public void addListener( ServiceContextListener listener);
```

```
public removeListener( ServiceContextListener listener);  
}
```

El método select() permite seleccionar un nuevo servicio dentro del contexto actual.

El método getService() devolverá el servicio que se está presentando actualmente.

getServiceContentHandlers() devuelve un set de objetos que son responsables de presentar el contenido del servicio, por ejemplo un Player.

Estados del service context

Cuando un service context es creado su estado es “Not Presenting”. Cuando es seleccionado entonces pasa a “Presentation Pending”, mientras tanto el receptor lleva a cabo todo lo necesario para obtener el contenido del nuevo servicio. Cuando el servicio comienza a presentarse en pantalla entonces el contexto pasa a estado “Presentating”. Desde este estado el contexto puede ser destruido si es que se necesitan sus recursos.

¿Cuando utilizar el API de Selección de Servicios?

Si se desea correr una aplicación sobre un nuevo servicio se recomienda utilizar la API de Selección de Servicios. Si únicamente se desea cambiar algunos elementos (audio, video, aplicación) de la presentación utilizar el JMF.

2.3.11 Almacenamiento de archivos en un receptor MHP

2.3.11.1 Funcionamiento del Caching

Por lo que se ha visto hasta ahora, los receptores pueden poner en el caché los carruseles de objetos con el objetivo de mejorar la performance de las aplicaciones, y cargar más rápidamente sus archivos. Esta es una cuestión crucial para los desarrolladores ya que determina el desempeño de sus aplicaciones, pero se debe tener en cuenta que cada receptor es diferente, y por lo tanto nunca se tiene seguridad total en cuanto a la cantidad de memoria disponible.

Los fabricantes de receptores son los encargados de decidir los siguientes aspectos:

- ¿Cuánta información puede llegar a manejar el caché?

Algunos receptores pueden cachear el carrusel de objetos entero, otros prácticamente nada, algunos parte de él. Esta libertad se debe a que la norma MHP no tiene nada estipulado al respecto. Este primer punto sólo afecta la performance y la velocidad de carga.

- ¿Cómo maneja el caché las actualizaciones de los objetos del carrusel?

Pueden ser ignoradas o no. MHP proporciona una serie de lineamientos debido a la gran incidencia que puede tener este aspecto en el comportamiento de las aplicaciones.

Mediante el seteo de un parámetro en el carrusel de objetos, el broadcaster puede elegir una de las siguientes tres opciones de caching.

1. Caching Transparente: Hace referencia a que el receptor sólo puede utilizar el objeto que se encuentra en el caché si en los últimos 0.5 segundos verificó el número de versión del módulo DSM-CC, si no lo hizo debe esperar. Esto asegura a las aplicaciones que utilizarán la última versión del archivo.
2. Caching Semitransparente: Funciona de manera similar al anterior pero con margen de 30 segundos para realizar la verificación del número de versión. Esto permite a las aplicaciones estar al día con los contenidos.
3. Caching Estático: todas las actualizaciones serán ignoradas, las aplicaciones deben solicitar la actualización de manera explícita.

2.3.11.2 Almacenamiento Persistente

Un receptor MHP maneja la posibilidad de guardar información relevante para el funcionamiento de las aplicaciones en su memoria persistente, la misma puede ser un disco duro o del tipo NVRAM, y el tamaño disponible es limitado.

El directorio raíz de un filesystem persistente está dado por las propiedades del sistema bajo dvb.persistent.root que puede ser accedido por el método getSystemProperty().

Toda aplicación tiene su directorio raíz, que es creado automáticamente por el receptor bajo la ruta <persistent_root>/<organization_id>/<application_id>.

Como en todo filesystem se definen diferentes tipos de permisos y usuarios para que manipulen los archivos.

Debido a que la cantidad de información que se puede guardar en este tipo de almacenamiento es limitada, los receptores MHP no garantizan la permanencia de la información, y esto de alguna manera evita que una aplicación monopolice el uso de la memoria disponible en el receptor. Por otro lado, todos los archivos tienen fecha de vencimiento, una vez expirado un archivo, éste será eliminado de la memoria. Para esto es que se define la prioridad de cada archivo asociado a la aplicación.

No es una buena práctica otorgarle a todos los archivos la misma prioridad y una fecha utópica de expiración, ya que a la hora de eliminar archivos, podrían resultar eliminados aquellos que son de suma importancia para la aplicación.

2.3.12 Acceso a los archivos de una aplicación MHP

Usualmente los filesystem son utilizados para cargar diferentes tipos de contenido como por ejemplo gráficos, clips de audio, video, data, texto, entre otros. En MHP el único tipo de filesystem al que las aplicaciones pueden acceder es el filesystem de broadcast, o sea el mismo en el que fue transmitida la aplicación. Sin embargo, en la especificación MHP 1.1 los receptores también pueden descargar los archivos a través de HTTP por el canal de retorno, aunque este tipo de dispositivos todavía no son muy populares en el mercado.

A la hora de manipular archivos en el contexto MHP (considerando la cantidad de memoria FLASH y la de broadcast) es importante recordar que los nombre relativos no se mantienen, o sea son relativos al directorio base de la aplicación que se indica en la Application Information Table.

2.3.12.1 Acceso al filesystem de broadcast

El acceso a este filesystem es complejo, y tiene una serie de problemas asociados que deben ser tenidos en cuenta por los desarrolladores a la hora de decidir qué tipo de aplicaciones desarrollarán. Uno de los puntos más importantes a considerar es la latencia. En un filesystem de broadcast, como puede ser el carrusel de objetos DSM-CC,

se debe esperar a que el archivo sea transmitido y además se deben tolerar demoras en la transmisión, haciendo que la latencia sea un aspecto crucial en la ejecución de aplicaciones.

Se puede hacer frente a este problema de dos maneras:

- Organizando el carrusel de objetos de manera tal que los archivos de acceso recurrente sean transmitidos la cantidad de veces necesarias para reducir la latencia.
- Asegurando que cualquier contenido que sea cargado desde el broadcast filesystem sea requerido con anterioridad, para que así la latencia no sea percibida por el usuario.

No siempre se pueden cumplir estos puntos pero es una buena práctica considerarlos.

Otra de las diferencias entre el DSM-CC y los filesystem convencionales es el caching. Dado que la latencia puede llegar a ser muy elevada, muchos receptores utilizan caché para mejorar la performance. Sin embargo existe la posibilidad de que los broadcasters actualicen los contenidos de un archivo del carrusel mientras se está realizando la transmisión, haciendo que la información que se maneja en el caché quede desactualizada.

Si bien el DSM-CC funciona como cualquier otro filesystem, MHP especifica una API para la manipulación de información, ofreciendo nuevas funcionalidades para tratar aspectos únicos del DSM-CC. Además se provee soporte a otros aspectos del DSM-CC que no refieren al tratamiento de archivos, ya que el mismo también es utilizado para transmitir objetos en general. Con objetos se hace referencia a archivos, flujos de datos y eventos.

2.3.12.2 La clase DSMCCObject

Esta clase es la encargada de representar los objetos del carrusel DSM-CC. Puede ser creada con una ruta absoluta o relativa como cualquier archivo de acceso a datos estándar. La única restricción es que si se utiliza para representar un directorio, la misma debe ser manipulada por los métodos provistos por la clase `java.io.File`. En cambio, al trabajar con archivos se utilizan las clases `java.io.InputStream` o `java.io.RandomAccess` para acceder al contenido de los mismos.

El acceso a los archivos no es muy flexible, en particular el acceso es síncrono o sea que el hilo (Thread) que hace la consulta permanecerá bloqueado hasta que la operación termine. Los problemas de latencia asociados al DSM-CC pueden ser catastróficos especialmente en el caso que se quieran cargar de antemano los contenidos que se pretenden ejecutar, para desplegarlos de forma rápida cuando el usuario lo solicite. Para aquellas APIs que funcionan de manera síncrona se puede cargar cada archivo en un hilo diferente, o esperar a que un archivo termine de cargar antes de comenzar con el próximo. Ninguna de estas opciones es realmente aceptable cuando se maneja un gran número de archivos. Para resolver este problema la clase DSMCCObject soporta la carga asíncrona. El método asynchronousLoad() permite a una aplicación cargar archivos sin los retrasos que pueda llegar a tener el hilo de ejecución. Este método maneja un Listener que se encargará de capturar el evento de finalización de carga de archivos. De manera similar existe el método synchronousLoad() para cargar objetos de manera síncrona. La ventaja que tienen estos métodos es que se puede cancelar la carga de archivos desde otro hilo mediante el método abort().

La clase DSMCCObject provee dos métodos adicionales para el control de la carga de archivos: prefetch() y unload(). El método prefetch() permite a las aplicaciones poner contenido en el caché del receptor, para que puedan ser accedidas en un futuro cercano, ganando tiempo. Por otro lado el método unload() indica al receptor que un objeto ya no se usará por lo que puede ser eliminado del caché.

Dado que el carrusel de objetos puede contener otros objetos a parte de archivos, la clase DSMCCObject permite a las aplicaciones determinar qué tipo de objetos son los que se encuentran referenciado o usando. Los métodos isObjectKindKnow(), isStream() y isStreamEvent() se utilizan con este propósito.

También podemos obtener la URL que refiere a un archivo en particular invocando el método getURL(). Esto puede ser útil cuando se trabaja con APIs como JMF que utilizan URLs para referirse a los contenidos.

2.3.12.3 Dominio de Servicio y carrusel de objetos

No existe restricción alguna respecto al hecho de que un servicio DVB contenga más de un carrusel de objetos, es por eso que se mencionarán algunas de las formas que tiene una aplicación de determinar a qué carrusel de objetos debe acceder. Su funcionamiento es similar al de los filesystem de Unix donde se hace un “mount” (montaje) para poder acceder a un determinado grupo de directorios.

Primero es necesario definir lo que se denomina “dominio de servicio”, grupo de objetos DSM-CC que están relacionados de alguna manera. Estos objetos están contenidos dentro de un carrusel de objetos y son transmitidos a los clientes. La API MHP DSM-CC representa un dominio de servicio mediante la clase ServiceDomain, pero antes de que estos servicios de dominio sean utilizados deben ser adjuntados, para ello se utiliza el método attach() y existen tres maneras de invocarlo:

- public void attach(Locator l): Tiene como parámetro un locator que hace referencia a un carrusel de objetos.
- public void attach(Locator service, int carousell): Tiene un locator referido a un servicio DVB, y un carousell para que el receptor sepa qué carrusel utilizar.
- public void attach(byte[] NSAPAddress): Tiene un NSAP Address de 20 bytes para identificar el carrusel.

La mayoría de las aplicaciones utilizan alguna de las dos primeras opciones.

Cabe destacar que ninguno de los tres métodos permite especificar en qué parte de la jerarquía del directorio se está adjuntando el dominio de servicios. Esto es así debido a que el receptor tiene la libertad de insertarlo donde más convenga y las aplicaciones pueden acceder a dicha ruta mediante el método getMountPoint(), que devuelve el objeto DSM-CC que representa el “mount point” del dominio de servicio.

El hecho de que las aplicaciones puedan adjuntar dominios de servicio no significa que éste podrá ser siempre accedido. Si el receptor sintoniza fuera de la trama de transporte que contiene el carrusel, los archivos no podrán ser accedidos por más que el dominio de servicio haya sido montado.

El siguiente ejemplo ilustra cómo una aplicación puede usar la API DSM-CC para acceder a un determinado archivo dentro de un filesystem de broadcast.

```
// Creamos un Nuevo dominio de servicio para representar el carrusel
// que vamos a utilizar
ServiceDomain carousel = new ServiceDomain();
//Ahora creamos un Locator que haga referencia al servicio que
//contiene nuestro carrusel
org.davic.net.Locator locator;
```

```
locator = new org.davic.net.Locator
("dvb://123.456.789");

// Finalmente adjuntamos el carrusel al objeto ServiceDomain
// para que podamos activamente acceder al carrusel.
carousel.attach(locator, 1);

// Tenemos que crear un nuestro DSMCCObject con una ruta absoluta
// lo que significa que debemos tener el mount point
// del service domain
DSMCCObject dsmccObj;
dsmccObj = new DSMCCObject(carousel.getMountPoint(),
"graphics/image1.jpg");

// Ahora creamos la instancia FileInputStream para acceder a nuestro
// DSM-CC object.
FileInputStream inputStream;
inputStream = new FileInputStream(dsmccObj);

// Ahora podemos usar el FileInputStream como cualquier otro
```

2.3.12.4 Ventajas de la carga asíncrona

```
// Podemos crear una instancia DSMCCObject sin adjuntarla a un carrusel.
DSMCCObject dsmccObj;
dsmccObj = new DSMCCObject("graphics/image1.jpg");

/*Creamos un objeto ServiceDomain para representar el carrusel que vamos
a utilizar */
ServiceDomain carousel = new ServiceDomain();

/*Ahora creamos un Locator que hace referencia al servicio que contiene
al carrusel*/
org.davic.net.Locator locator;
locator = new org.davic.net.Locator
("dvb://123.456.789");
```

```
// Por ultimo adjuntamos el carrusel al objeto ServiceDomain
// para que podamos acceder al carrusel

/* En este caso no especificamos la trama que contiene al carrusel del
Locator, por eso pasamos como parametro el carrusel ID*/
carousel.attach(locator, 1);

/* tenemos que crear nuestro DSMCCObject con un path absoluto, lo que
significa que tenemos que tener un mount point para el service domain*/
DSMCCObject dsmccObj;
dsmccObj = new DSMCCObject(carousel.getMountPoint(),
    "graphics/image1.jpg");

// En esta ocasión cargamos el archivo de manera asíncrona.

/* La variable myListener (que no fue definida todavía) es el event
listener que recibirá la notificación cuando el objeto este totalmente
cargado*/
dsmccObj.asynchronousLoad(myListener);

// Ahora podemos cargar otro archivo en el mismo hilo
DSMCCObject dsmccObj2;
dsmccObj2 = new DSMCCObject(carousel.getMountPoint(),
    "graphics/image2.jpg");

/* Este también utilizará el mismo event listener para recibir
notificaciones que el objeto que fue cargado completamente*/
dsmccObj2.asynchronousLoad(myListener);
```

En este caso se cargan 2 objetos desde el mismo thread sin tener que esperar a que el primero termine. El objeto myListener recibirá la notificación cuando la carga esté completa y a partir de ese momento podrá ser utilizada.

2.3.12.5 Actualizando Objetos

Una de las ventajas más importantes del DSM-CC es que cada objeto dentro del carrusel tiene un número de versión, y los objetos pueden ser actualizados mientras el carrusel está siendo transmitido. Esto puede ser útil para aplicaciones del estilo Home Shopping o aplicaciones tipo Quiz. Las aplicaciones pueden detectar estas actualizaciones y luego decidir cargar de nuevo el archivo que fue modificado o continuar utilizando el mismo.

Para detectar este tipo de actualizaciones las aplicaciones deben implementar la interfase org.dvb.dsmcc.ObjectChangeEventLister, que maneja eventos que son capturados mediante un Listener (addObjectChangeEventLister).

2.3.12.6 Definiendo classloaders

El modelo de seguridad de MHP especifica que cada aplicación debe ser cargada utilizando un classloader diferente. Los classloaders proporcionan una manera de cargar clases de diferentes tipos de fuentes, por ejemplo se pueden cargar archivos que se encuentren en filesystem local, una conexión HTTP o un carrusel de objetos DSM-CC.

Desafortunadamente para los desarrolladores, a las aplicaciones DVB-J no les está permitido definir sus propios classloaders por temas de seguridad. En cambio MHP provee su propio classloader que ofrece la funcionalidad de los mismos sin los problemas de seguridad.

Java especifica que si dos clases son cargadas de diferentes classloader entonces son diferentes clases por más que su contenido sea el mismo. MHP especifica que cada aplicación debe tener un classloader dedicado y que todas las clases de la aplicación deben ser cargadas por ese classloader. Esto revindica el hecho de que las clases que pertenecen exclusivamente a una aplicación son totalmente independientes de las clases pertenecientes a otra aplicación.

En el caso de que una aplicación “A” quiera acceder a un objeto de la aplicación “B”, deberá cargar la clase desde la cual el objeto fue instanciado. Sin embargo esto significa que se estará cargando con el classloader de la clase “A”, por lo que será una clase distinta a la clase de la aplicación “B”. Dado este mecanismo, es que no se pueden pasar referencias a objetos de una aplicación a otra.

A su vez existen dos classloaders más que deben ser tenidos en cuenta, el system classloader que es utilizado por la JVM para cargar las clases proporcionadas por Java como pueden ser las clases del middleware; y un classloader de más bajo nivel denominado classloader primordial, utilizado por la JVM cuando empieza a cargar las clases más básicas de Java como por ejemplo java.lang.Object, las mismas deben cargarse antes de que la JVM pueda crear su system classloader.

El siguiente diagrama muestra las relaciones entre los diferentes classloaders:

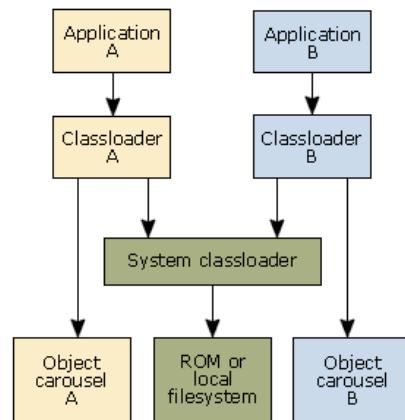


Figura 2.27: Classloaders

2.4 MHP - Guía para la emisión de aplicaciones

2.4.1 Transporte de archivos MHP

Cada aplicación DVB-J consistirá en un conjunto de archivos .class y archivos de datos. Éstos deberán ser obtenidos del flujo de broadcast transmitido por la emisora.

Mediante la tabla de señalización AIT se indicará al receptor el classpath, el directorio base, y los archivos correspondientes a la aplicación, sin embargo esto no da la suficiente información como para cargar los archivos ni determinar dónde se encuentran. Para completar la información se debe indicar el método mediante el cual se transportan los archivos de cada aplicación. Estos métodos pueden ser:

- Mediante un carrusel de objetos DSM-CC en la trama MPEG. La ventaja de este método radica en que el carrusel de objetos desde Java es tratado como un sistema de archivos y las operaciones son las utilizadas normalmente en Java. Otra ventaja es que los cáruseles de objetos son soportados por cualquier receptor, y existe mucho equipamiento disponible para generarlos.
- Mediante una conexión IP multicast en un DSM-CC utilizando un protocolo de encapsulamiento. Sin embargo en la práctica esto no está muy difundido.
- Mediante el canal de retorno usando HTTP. Este método es aplicable a perfiles interactivos o de acceso a Internet, permitiendo una carga rápida de archivos a demanda.

2.4.1.1 Transport protocol descriptor

Para reflejar el método utilizado para transportar los archivos se recurre al transport protocol descriptor.

	No. of Bits
transport_protocol_descriptor() {	
descriptor_tag	8
descriptor_length	8
protocol_id	16
transport_protocol_label	8
for (i = 0; i < N; i++) {	
selector_byte	8
}	
}	

Tabla 2.9: Estructura del Transport Protocol Descriptor

La entrada “protocol_id” comunica al receptor el protocolo utilizado para cargar los archivos asociados a la aplicación, los valores posibles son:

0x0001: object carrusel

0x0002: multicast IP stream haciendo uso de un DSM-CC multi-protocol encapsulation.

0x0003: canal de retorno

El campo “selector” depende del protocolo, y contiene información asociada al mismo.

Para un carrusel de objetos, el “selector” tiene la siguiente estructura:

	Bits
remote_connection	1
reserved_future_use	7
if(remote_connection == "1") {	
original_network_id	16
transport_stream_id	16
service_id	16
}	
component_tag	8

Tabla 2.10: Estructura del “Selector”

El campo “remote_connection” indica si la trama que contiene el carrusel de objetos es la misma que para el servicio en cuestión. Si no lo es, el “selector” indicará el network ID, transport stream ID y service ID.

El “component_tag” permite elegir la trama correcta en aquellos casos en los que la aplicación cuenta con más de un carrusel.

2.4.2 Señalización de aplicaciones MHP

Para que el XletManager del STB sea capaz de ejecutar una aplicación, éste debe contar con la siguiente información:

- Existencia de la aplicación.
- Usuario con permisos de ejecución en ese instante.
- Ubicación de los archivos y clases necesarias para ejecutar la aplicación.

Para brindar esta información existe la tabla AIT (Application Information Table) que será transmitida por la emisora de TV, siempre y cuando el servicio cuente con una aplicación interactiva asociada.

2.4.2.1 AIT

La AIT tiene la información requerida por el receptor para ejecutar la aplicación, por ejemplo nombre de la aplicación, ubicación de sus archivos, etc.

A continuación se muestra la estructura de la AIT y se explican algunos de sus parámetros más importantes:

Sintaxis	Cantidad de bits
application_information_section () {	
table_id	8
section_syntax_indicator	1
reserved_future_use	1
reserved	2
section_length	12
test_application_flag	1
application_type	15
reserved	2
version_number	5
current_next_indicator	1
section_number	8
last_section_number	8
reserved_future_use	4
common_descriptors_length	12
for (i = 0; i < N; i++) {	
descriptor()	
}	
reserved_future_use	4
application_loop_length	12
for (i = 0; i < N; i++) {	
application_identifier()	
application_control_code	8
reserved_future_use	4

application_descriptors_loop_length	12
for (j = 0; j < N; j++) {	
descriptor()	
}	
}	
CRC_32	32
}	

Tabla 2.11: Estructura de una AIT

Cada aplicación emitida contiene un identificador único que se almacena en la AIT, este identificador consta de dos partes:

- ID de organización: 32 bits que permiten identificar a la organización que emite la aplicación.
- ID de la aplicación: 16 bits que pueden no ser únicos.

Entre 1 y 0x3FFF se utiliza para aplicaciones no firmadas digitalmente por la emisora.

Entre 0x4000 y 0x7FFF para las que sí.

0xFFFF wildcard para identificar todas las aplicaciones con el mismo organization ID.

0xFFFF wildcard para identificar todas las aplicaciones firmadas por el mismo organization ID.

El control code indica si la aplicación debe ser ejecutada automáticamente junto a un servicio o si se ejecutará bajo explícita solicitud del usuario.

Como se puede ver, la AIT cuenta con sub-tablas embebidas en ella, cada una conteniendo aplicaciones de un solo tipo, DVB-J o DVB-HTML. La entrada “application_type” indica el tipo de aplicación asociada a la sub-tabla AIT.

2.4.2.2 ¿Cómo describir una aplicación?

La AIT contiene dos loops de descripción destinados a detallar las aplicaciones: un loop común a todas las aplicaciones, y uno por aplicación si es necesario.

Descriptor	Obligatoriedad	Ubicación
application descriptor	obligatorio	application descriptor loop
application name descriptor	obligatorio	application descriptor loop
application icons descriptor	opcional	application descriptor loop
DVB-J application descriptor	obligatorio para aplicaciones DVB-J	application descriptor loop
DVB-J application location descriptor	obligatorio para aplicaciones DVB-J	application descriptor loop
DVB-HTML application descriptor	obligatorio para aplicaciones DVB-HTML	application descriptor loop
DVB-HTML application location descriptor	obligatorio para aplicaciones DVB-HTML	application descriptor loop
DVB-HTML application boundary descriptor	obligatorio para aplicaciones DVB-HTML	application descriptor loop
external application authorization descriptor	opcional	common descriptor loop
IP routing descriptor	opcional	common descriptor loop
transport protocol descriptor	opcional	ambos
pre-fetch descriptor	opcional	application descriptor loop
DII location descriptor	opcional	ambos

Tabla 2.12: Descriptors de la AIT

2.4.2.3 Algunos descriptors

Application descriptor

Describe la aplicación en cuestión (perfil MHP utilizado, prioridad) y su estado.

Sintaxis	Cantidad de bits
application_descriptor() {	
descriptor_tag	8
descriptor_length	8
application_profiles_length	8
for (i = 0; i < N, i++) {	
application_profile	16
version.major	8
version.minor	8
version.micro	8
}	
service_bound_flag	1
visibility	2
reserved_future_use	5
application_priority	8
for (i = 0; i < N; i++) {	
transport_protocol_label	8

Tabla 2.13: Estructura del Application Descriptor

El campo “visibility” indica al receptor si la aplicación debe estar visible o no para el usuario.

El campo “service_bound_flag” es utilizado para identificar aquellas aplicaciones que se encuentran asociadas a un servicio en particular y que serán terminadas cuando se seleccione otro servicio.

El parámetro “transport_protocol_label” hace referencia al descriptor del transport protocol.

Application name descriptor

Indica el nombre de la aplicación que será mostrado al usuario en uno o más idiomas.

Sintaxis	Cantidad de bits
application_name_descriptor	
{	
descriptor_tag	8
descriptor_length	8
for (i = 0; i < N; i++) {	
ISO_639_language_code	24
application_name_length	8
for (i = 0; i < N; i++) {	
application_name_char	8
}	
}	
}	

Tabla 2.14: Estructura del Application name descriptor

Application icons descriptor

Éste es un descriptor opcional que permite definir qué imagen será ícono de la aplicación. Se debe tener en cuenta que no todos los receptores prestarán atención a este parámetro.

Sintaxis	Cantidad de bits
application_icons_descriptor()	
{	
descriptor_tag	8
descriptor_length	8
icon_locator_length	8
for (i = 0; i < N; i++) {	
icon_locator_byte	8
}	
icon_flags	16
for (i = 0; i < N; i++) {	
reserved_future_use	8
}	
}	

Tabla 2.15: Estructura del Application icons descriptor

DVB-x application descriptor

Describe aquellos elementos propios de cada modo DVB. En el caso de DVB-J le indica al receptor los parámetros que debe pasarle a la aplicación al iniciarla.

Sintaxis	Cantidad de bits
dvb_j_application_descriptor()	
descriptor_tag	8
uimsbf	
descriptor_length	8
uimsbf	
for (i = 0; i < N; i++) {	8
parameter_length	
uimsbf	
for (j = 0; j < parameter_length; j++) {	8
parameter bytes	8
uimsbf	
}	
}	
}	

Tabla 2.16: Estructura del DVB-J application descriptor

DVB-x application location descriptor

Indica al receptor dónde encontrar los archivos asociados a la aplicación. En el caso de DVB-J se indicará la clase main, el classpath y el directorio base de la aplicación.

Sintaxis	Cantidad de bits
dvb_j_application_location_descriptor {	
descriptor_tag	8 bits
uimsbf	
descriptor_length	8 bits
uimsbf	
base_directory_length	8 bits
uimsbf	
for (i = 0; i < base_directory_length; i++) {	
base_directory_byte	8
uimsbf	
}	

classpath_extension_length	8
uimsbf	
for (i = 0; i < classpath_extension_length; i++) {	
classpath_extension_byte	8
uimsbf	
}	
for (i = 0; i < N; i++) {	
initial_class_byte	8
uimsbf	
}	
}	

Tabla 2.17: Estructura del DVB-J application location descriptor

Transport protocol descriptor:

Brinda información específica de cómo cargar los archivos de la aplicación en el receptor.

Para esto se indica dónde se encuentra el filesystem y cómo acceder al carrusel de objetos o al servidor remoto.

Sintaxis	Cantidad de bits
transport_protocol_descriptor() {	
descriptor_tag	8
uimsbf	
descriptor_length	8
uimsbf	
protocol_id	16
uimsbf	
transport_protocol_label	8
uimsbf	
for (i = 0; i < N; i++) {	
selector_byte	8
uimsbf	N1
}	
}	

Tabla 2.18: Estructura del transport protocol descriptor

El “selector_byte” indica al receptor cómo acceder a los archivos de la aplicación. La sintaxis de este descriptor depende exclusivamente del protocolo de transporte identificado por el “protocol_id”.

External application authorization descriptor, DVB-x:

Este parámetro es común a todas las aplicaciones de la AIT, opcional, e indica si alguna aplicación no debería comenzarse, aunque si algunas instancias de ella se encuentran ejecutando entonces no serán finalizadas al cambiar de servicio.

Sintaxis	Cantidad de bits
external_application_authorization_descriptor() {	
descriptor_tag	8
uimsbf	
descriptor_length	8
uimsbf	
for (i = 0; i < N; i++) {	
application_identifier()	
application_priority	8
uimsbf	
}	
}	

Tabla 2.19: Estructura del External application authorization descriptor

Pre-fetch descriptor:

Opcional. Da una noción de qué partes de la aplicación deberían ir cargándose para reducir el tiempo de ejecución. Este descriptor es muy útil cuando se hace uso de un carrusel de objetos, aunque el receptor no siempre tiene en cuenta las recomendaciones.

Sintaxis	Cantidad de bits
prefetch_descriptor() {	
descriptor_tag	8
uimsbf	
descriptor_length	8
uimsbf	
transport_protocol_label	8
uimsbf	
for (i = 0; i < N; i++) {	
label_length	8
uimsbf	
for (i = 0; i < N; i++) {	
label_char	8

uimsbf	
}	
}	
prefetch_priority	8
uimsbf	
}	

Tabla 2.20: Estructura del Prefetch descriptor

El campo “transport_protocol_label” es una referencia al descriptor del mismo. Cada label identifica uno o más módulos del carrusel de objetos, y el “prefetch_priority” indica al receptor qué tan importante es el prefetch de ese módulo para la aplicación (1 bajo, 100 alto)

DII location descriptor:

Para aplicaciones que hacen uso de un carrusel de objetos, este descriptor indica al receptor qué mensajes del DSM-CC DownloadInfoIndication contienen los descriptors de los módulos identificados en la label del descriptor prefetch.

Sintaxis	Cantidad de bits
DII_location_descriptor() {	
descriptor_tag	8
uimsbf	
descriptor_length	8
uimsbf	
transport_protocol_label	8
uimsbf	
for (i = 0; i < N; i++) {	
reserved_future_use	1
bslbf	
DII_identification	15
uimsbf	
association_tag	16
uimsbf	
}	
}	

Tabla 2.21: Estructura del DII location descriptor

El campo “association_tag” identifica el elementary stream que contiene el carrusel de objetos en cuestión.

Existen otros descriptors como el “IP routing descriptor” e “IP signalling descriptor” que aplican si algún servicio hace uso de IP multicast como su mecanismo de transporte.

Capítulo 3: Marco Metodológico

3.1 Planificación

El proyecto TvDTi comenzó el 11 de agosto de 2008, y se extendió hasta el 4 de marzo de 2009, quedando por fuera de estas fechas la preparación de la presentación oral del mismo y la defensa en sí.

Para un desarrollo ordenado del proyecto y permitir un seguimiento del mismo, se generó un documento denominado “Plan de Proyecto” (ver Anexo 1) en el que se definen los siguientes aspectos:

- ✓ Nombre del proyecto
- ✓ Fecha de inicio del proyecto
- ✓ Fecha tentativa de finalización del proyecto
- ✓ Alcance
- ✓ Entregables
- ✓ Restricciones
- ✓ EDT – Estructura de Desglose de Tareas
- ✓ Cronograma de tareas

A partir de este documento se comenzó a ejecutar el proyecto, que incluía tareas técnicas (investigación, programación Java, etc.) y de gestión de proyectos (cumplimiento de plazos, entregables, documentación de tareas, actas de reuniones, etc.)

Las tareas del proyecto podrían resumirse en los siguientes ítems:

1. *Estudio de las tecnologías involucradas: Televisión digital Terrestre (TDT), Interactividad, Estándar MHP.*
2. *Elección de la aplicación a desarrollar .*

2. *Diseño de la aplicación.*
3. *Implementación de la aplicación.*
4. *Testeo de la aplicación.*
5. *Investigación del canal de retorno para TDT.*
6. *Investigación de los requerimientos de lado de la emisora.*
7. *Testeo de la aplicación en el contexto real.*
8. *Desarrollo de documentación y entregables.*

Las tareas anteriores fueron desglosadas conformando así el EDT. Luego se estimó la duración de cada tarea, las dependencias entre ellas, y se organizaron las mismas en el tiempo. Para documentar esta planificación se hizo uso del Microsoft Project, un programa informático destinado a planificar los proyectos en el tiempo, que permite reflejar en un diagrama de Gantt la duración de las tareas, el estado de avance de las mismas, las dependencias, etc.

La documentación del proyecto es una tarea esencial durante la ejecución del mismo, y debe ser considerada de manera tal de dedicarle el suficiente tiempo como para generar expedientes claros, concretos y completos. Durante el proyecto TvDTi se tuvo en cuenta la generación de los siguientes documentos y entregables:

- ✓ Definición del tema de Proyecto (27 – Ago – 2008)
- ✓ Plan del Proyecto de fin de carrera (10 – Set – 2008)
- ✓ Cálculos/Rutinas (avance 50%) (19 – Nov – 2008)
- ✓ Borrador del informe final de PFC (17 – Dic – 2008)
- ✓ Cálculos/Rutinas (avance 100%) (18 – Feb – 2009)
- ✓ Informe final de PFC. (04 – Mar – 2009)

- ✓ Presentación de la defensa de PFC (17 – Mar – 2009)
- ✓ Documentación de seguimiento

La última tarea se extendió durante todo el proyecto y estuvo enfocada en dejar registro de todas aquellas tareas que lo requerían, así como también incluyó la generación de actas de reuniones.

En los anexos 1, 3, 4 y 5 se adjuntan los entregables “Plan de proyecto”, “Informe de justificación de avance de 50%”, “Informe de justificación de avance de 100%” y actas de reuniones respectivamente. Todos ellos fundamentales para hacer un seguimiento del proyecto y detectar desvíos.

3.2 Metodología de estudio

La primera tarea del proyecto fue la elección del tema del mismo. Para esto se evaluaron las distintas posibilidades existentes teniendo en cuenta los intereses de los ejecutores del proyecto, los plazos y recursos. Finalmente el proyecto escogido lleva como título “Implementación de una aplicación interactiva para Televisión Digital Terrestre”. Este tema resultó atractivo debido a su reciente aparición en el mercado, la poca experiencia existente en el Uruguay y la gran aplicabilidad del mismo.

Vale la pena destacar que aquellos proyectos que cuentan con interesados en su desarrollo, o cuyo producto final puede ser adquirido por un cliente resultan sumamente motivantes ya que permiten comprobar su utilidad. Sin embargo esto no sucede con aquellos proyectos que simplemente aportan conocimiento y cuya introducción en el plano comercial no es clara. Este aspecto influye en la elección del tema del proyecto.

Por lo tanto, el proyecto en cuestión aborda una muy innovadora tecnología no conocida hasta el momento por los ejecutores, y de alguna manera recientemente adquirida a nivel mundial. Por esta razón, el proyecto se podría dividir en dos etapas bien diferenciadas pero no independientes. Una primera etapa de investigación, y una segunda etapa de implementación.

La primera etapa consistió en el estudio de normas, manuales, tutoriales, foros, artículos, etc., que permitieron comprender los aspectos medulares de la TDTi y generar así el

conocimiento requerido para la etapa 2. Vale la pena aclarar que no hubo instancias de consulta a docentes o profesionales en cuanto a los aspectos técnicos dado que no se tienen contactos formados en la materia en cuestión.

Durante la segunda etapa se implementó una aplicación de TDTi en base al conocimiento adquirido durante la etapa anterior. La aplicación a desarrollar debía ser de carácter comunitario, por lo que se realizó una encuesta en la que se sondeó los intereses de la sociedad para así generar una aplicación de amplia preferencia. Por otro lado, en paralelo a la etapa inicial, se ejecutaron algunas tareas relacionadas a la implementación, como el diseño de interfaces. La etapa de implementación comprende el desarrollo de código en una herramienta Java genérica y el testeo del mismo en un emulador MHP.

Capítulo 4: Desarrollo

Luego de seleccionado el tema del proyecto a desarrollar y de elaborar el Plan de Proyecto, comenzaron las tareas de ejecución, acompañadas de las de planificación (seguimiento) y documentación. A continuación se detallan aquellas tareas fundamentales del proyecto que ameritan una justificación.

4.1 Tarea 1.1

Búsqueda y estudio de tutoriales, manuales, publicaciones, etc.

La tarea en cuestión consistió en investigar aquellos estándares y tecnologías relacionadas a la TDTI. Dado que el producto final consistía en una aplicación interactiva para TDT de carácter comunitario, y en Uruguay el estándar adoptado para el desarrollo de la TDT es DVB, la investigación se basó principalmente en el estándar MHP, estándar abierto, “socio” de DVB para la generación de aplicaciones interactivas. Esto no sólo habilitará a las emisoras de TV abierta en el Uruguay a emitir la aplicación creada, sino que producirá el conocimiento básico necesario para la creación de nuevas aplicaciones.

Esta tarea dejó como resultado un documento al que se denominó “Guía MHP” y cuya principal función es brindar las instrucciones indispensables a tener en cuenta para crear una aplicación interactiva para TDT bajo el estándar MHP y transmitirla bajo el estándar DVB.

4.2 Tarea 2.1

Selección de posibles aplicaciones

Televisión interactiva aplicada a la TDT

La TDT es la que usualmente conocemos como televisión abierta, aunque también existen sistemas TDT privados. Ésta llega a todos aquellos consumidores que cuentan con un equipo de televisión sin necesidad de suscribirse a ningún servicio en particular. La principal característica de este medio de comunicación es que la información que llega al televidente no es personalizada, lo que significa que todos reciben la misma información y en el mismo instante.

Debido a estas características, la TDT se presta para incorporar aplicaciones interactivas pero de carácter comunitario, brindando información de interés general, no personalizada. Por estas razones el nivel de interactividad no llega a ser bidireccional, es fundamentalmente broadcast, o en algunos casos podría llegar a ser unidireccional.

En este contexto es que se estableció uno de los objetivos del proyecto en cuestión el cual consiste en implementar una aplicación representativa del poder del estándar MHP dentro del sector de servicios ciudadanos, sin la necesidad de la utilización de un canal de retorno que envíe información desde el televidente a la emisora correspondiente.

Selección de posibles aplicaciones de desarrollar

Dado que la aplicación a desarrollar debía ser de índole comunitario, el objetivo principal de ésta debe ser simplemente brindar información al televidente. La siguiente lista muestra posibles contenidos de información de la aplicación a desarrollar:

- EPG (Electronic Programme Guide)
- Pronóstico del tiempo
- Juegos
- Servicios de transporte capitalino
- Información para la realización de trámites del estado
- Deliverys de comida
- Farmacias de turno
- Teléfonos de emergencia
- Cotización de las monedas
- Resultados de deportes
- Resultados de los juegos de azar (Kini, 5 de Oro, Quiniela, etc.)
- Horóscopo

- Cartelera de espectáculos (cines y teatros)
- Ofertas de trabajo
- Cotización de la bolsa de valores de Montevideo.

4.3 Tarea 2.2

Formulación de una encuesta que permitirá determinar los intereses de la sociedad, con el fin de seleccionar la aplicación.

1. Elaboración del brief

Este punto tiene como objetivo declarar de manera breve el objetivo de la encuesta.

Objetivo de la encuesta:

La encuesta está destinada a conocer los intereses de la sociedad en cuanto a qué información le gustaría tener siempre disponible y de forma práctica.

2. Definición del universo o target

Este ítem está enfocado en definir el público objetivo de la encuesta y el medio para la realización de la misma.

Público objetivo:

El público objetivo es la sociedad en general, sin distinción de sexo, edad ni clase social.

Medio:

La encuesta será realizada mediante correo electrónico y de modo personal.

3. Toma de la muestra

La muestra debe ser significativa, aleatoria y representativa. Se considera que en este contexto los intereses de la persona están influenciados fundamentalmente por la edad de la misma.

Por esta razón se entrevistará a 20 personas de cada uno de los siguientes rangos de edad:

- 12 a 18 años
- 19 a 25 años
- 26 a 40 años
- 41 a 65 años
- Más de 66 años

4. Formulación de preguntas

La encuesta fue redactada de manera simple y concreta, intentando no dar lugar a confusiones ni diferentes interpretaciones.

Esquema de la encuesta:

Edad:	<input type="checkbox"/> 12 a 18 años	<input type="checkbox"/> 19 a 25 años	<input type="checkbox"/> 26 a 40 años
	<input type="checkbox"/> 41 a 65 años	<input type="checkbox"/> Más de 66 años	

¿Qué información desearía tener siempre al alcance de la mano, cuando usted lo disponga, gratis y al instante?

De las siguientes opciones elija aquellas 3 que le resulten más útiles:

- Programación de los canales de TV durante los próximos 7 días
 - Pronóstico del tiempo
 - Recorrido de ómnibus
 - Teléfonos de restaurantes y bares con envíos a domicilio
 - Teléfonos de emergencia (policía, bomberos, emergencias médicas, etc.)
 - Teléfono y dirección de farmacias de turno
 - Cotización de la moneda
 - Titulares de diarios
 - Resultados de los juegos de azar (Kini, 5 de Oro, Quiniela, etc.)
 - Cartelera de teatro y cine
- Gracias por su tiempo.

Tabla 4.1: Esquema de la encuesta

4.4 Tarea 2.3

Realización de la encuesta

La encuesta fue realizada de manera tal de cumplir con la planificación de la misma, respetando el público objetivo declarado.

La misma fue realizada personalmente o vía email.

4.5 Tarea 2.4

Procesamiento de los resultados de la encuesta

Los datos de la encuesta fueron procesados elaborando la planilla que se ve a continuación.

		12 a 18 años	19 a 25 años	26 a 40 años	41 a 65 años	Más de 66 años	Total
1	Programación de los canales de TV durante los próximos 7 días	9	2	5	2	4	22
2	Pronóstico del tiempo	8	4	7	12	8	39
3	Recorrido de ómnibus	10	13	7	4	4	38
4	Teléfonos de restaurantes y bares con envíos a domicilio	12	7	4	4	4	31
5	Teléfonos de emergencia (policía, bomberos, emergencias médicas, etc.)	5	4	4	13	9	35
6	Teléfono y dirección de farmacias de turno	1	0	5	3	3	12
7	Cotización de la moneda	0	3	4	4	8	19
8	Titulares de diarios	5	14	12	9	7	47
9	Resultados de los juegos de azar (Kini, 5 de Oro, Quiniela, etc.)	1	0	0	0	7	8
10	Cartelera de teatro y cine	9	13	12	9	6	49

Tabla 4.2: Resultados de la encuesta

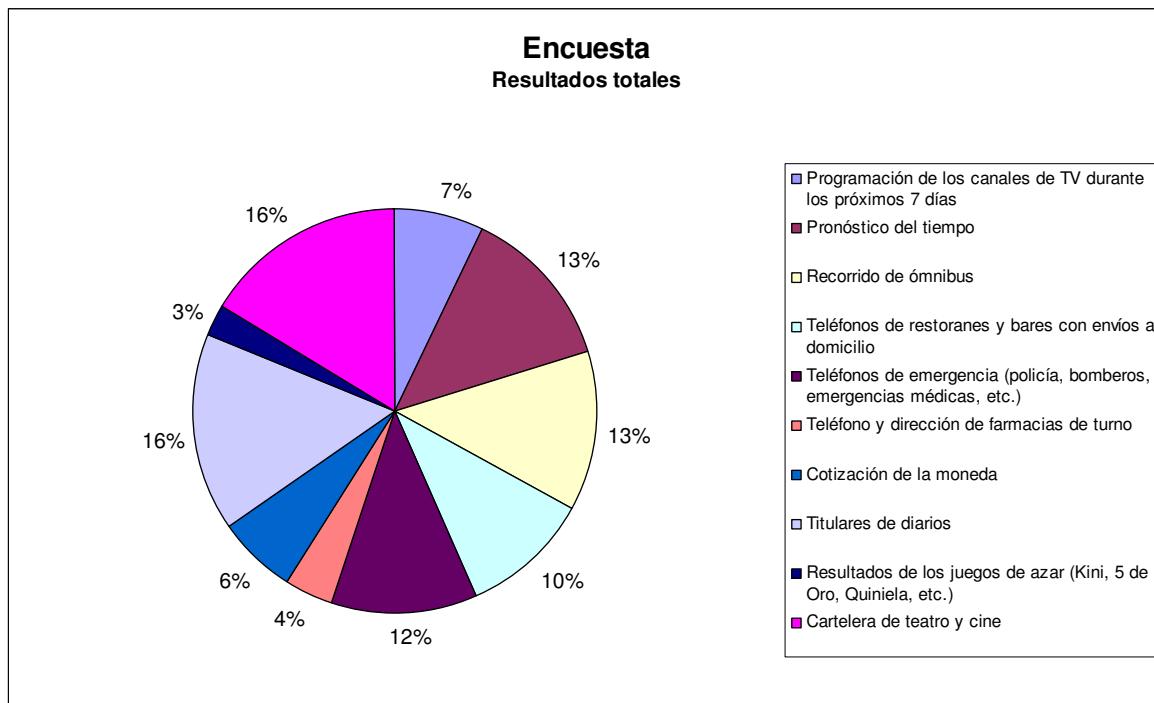


Figura 4.1: Gráfica correspondiente a los resultados de la encuesta

4.6 Tarea 2.5

Definición de la aplicación a desarrollar

Luego de realizado el procesamiento de los datos obtenidos en la encuesta, la opción que tuvo más aceptación fue “Cartelera de Cines y Teatros”. Como se puede apreciar en los resultados, si se analiza la encuesta por franja etárea se observa que la “Cartelera de Espectáculos” en ninguna de éstas obtuvo la mayoría de los votos, sin embargo fue la que más votos obtuvo en total. Si bien es claro que se podría haber realizado otra interpretación de los datos, como por ejemplo ponderado según la importancia de las diferentes edades, se consideró que la TV es un medio masivo y por lo tanto todas las edades deben ser contempladas con el mismo peso. Es por esto que se sumaron todos los votos y se optó por implementar la aplicación “Cartelera de Cines y Teatros”.

4.7 Tarea 3.1

Diseño de Interfases

Esta tarea está enfocada en diseñar las interfaces de la aplicación a nivel de navegabilidad y layout. La navegabilidad refiere a la manera en la que el usuario deberá transitar por la aplicación para obtener la información deseada. Por otro lado el layout describe cómo se ordenará la información en pantalla. Dado que en esta etapa no se conoce con precisión el alcance de la tecnología TDTI no se especifican los colores a utilizar, la inclusión de imágenes ni los botones que permitirán la interacción con la aplicación. Se estudiarán estos aspectos durante la implementación de la aplicación, y se analizará su influencia sobre la ejecución de la misma (tiempo, estética, etc.)

Al diseñar las interfaces se tuvieron en cuenta los siguientes aspectos:

- “Lo necesario, pero lo menos posible”.
- Pantallas con diseños idénticos salvo por el contenido, esto facilita al usuario la navegación.
- Navegación intuitiva.
- Dar idea al usuario de donde está “parado”, cómo llegó a ahí y a donde puede dirigirse.
- Proveer un escape rápido.
- En casos donde se deba usar scroll up o down se hará mediante jumping, lo que significa que se navegará a través del contenido mediante pantallas completas.

Pantallas

Las pantallas se despliegan a continuación:

Pantalla 1



Figura 4.2: “Cartelera de Espectáculos”, Pantalla 1

Pantalla 1.1

Cartelera de Cines



Ver salas y horarios



Ver resumen de películas



Ok



Atrás



Salir

Figura 4.3: Cartelera de Espectáculos”, Pantalla 1.1

Pantalla 1.1.1

Cines – Salas y horarios

Sala x

Teléfono/Dirección

Película 1 Horario1/Horario2/Horario3

Película 2 Horario1/Horario2



Sala y

Teléfono/Dirección

Película 1 Horario1



Ver resúmenes



Atrás



Salir

Figura 4.4: “Cartelera de Espectáculos”, Pantalla 1.1.1

Pantalla 1.1.2

Cines – Resumen de películas

Película 1

Resumen



Película 2

Resumen



Ver salas y horarios



Atrás



Salir

Figura 4.5: “Cartelera de Espectáculos”, Pantalla 1.1.2

Pantalla 1.2

Cartelera de Teatros

Obra 1

Resumen

Sala / Teléfono / Dirección

Horarios



Obra 2

Resumen

Sala / Teléfono / Dirección

Horarios



 Atrás

 Salir

Figura 4.6: “Cartelera de Espectáculos”, Pantalla 1.2

Estructura de navegación

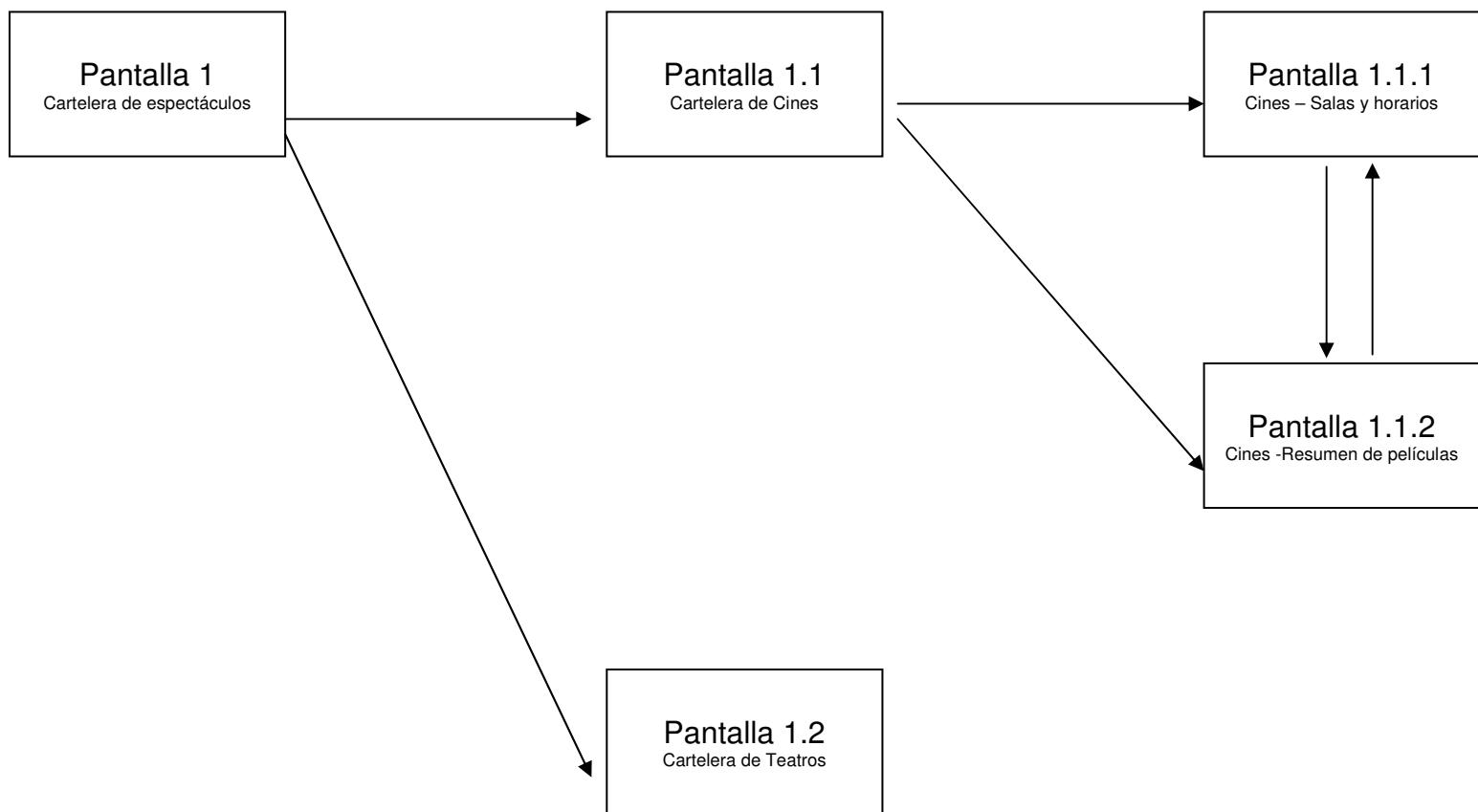


Figura 4.7: “Cartelera de Espectáculos”, Estructura de navegación

4.8 Tarea 3.2

Diagramas de secuencia

En lo que refiere a aplicaciones MHP, es difícil esbozar un diagrama de secuencia, pero sí es importante reflejar el modo en que se ejecutará una aplicación y se accederá a los archivos asociados a la misma. Para esto parece más adecuado realizar un diagrama de flujo.

Antes de describir el diagrama de flujo, vale la pena aclarar algunos aspectos acerca de la aplicación “Cartelera de Espectáculos”:

- La misma será emitida en el estado PRESENT, por lo que al llegar al receptor éste no la ejecutará, pero sí hará que el nombre de la aplicación aparezca en la lista de aplicaciones disponibles.
- La aplicación no hace uso de memoria persistente, por lo que los archivos asociados a la misma podrán ser obtenidos del caché o del carrusel de objetos.
- El modo de caching de la aplicación no es conocido ya que depende del middleware de cada receptor.

Al emitir una aplicación, la emisora no sólo envía sus archivos java y archivos de contenido, sino que también emite señalización que permite al terminal entender qué está recibiendo desde el canal de broadcast y cómo debe actuar. Los parámetros de señalización más importantes son:

- El “DVB-J application location descriptor”. Éste indica al terminal MHP la clase main, el classpath y el directorio base de la aplicación.
- El “Application descriptor”. Este descriptor indica el estado de la aplicación, en este caso PRESENT.

Al llegar la nueva aplicación al terminal, éste la desplegará en la lista de aplicaciones disponibles, y cacheará parte, o todos los archivos asociados a la misma. Al momento que el usuario solicite su ejecución, se llamará al método initXlet() de la clase principal cacheada, y se comenzará a acceder a los archivos necesarios. Para esto, primero se verificará si los archivos se encuentran en el caché, lo que sin lugar a dudas hará de la ejecución una carga rápida. En caso de que los archivos no se encuentren en el caché, se deberá esperar a que el carrusel de objetos los emita. A continuación figura el diagrama de flujo asociado:

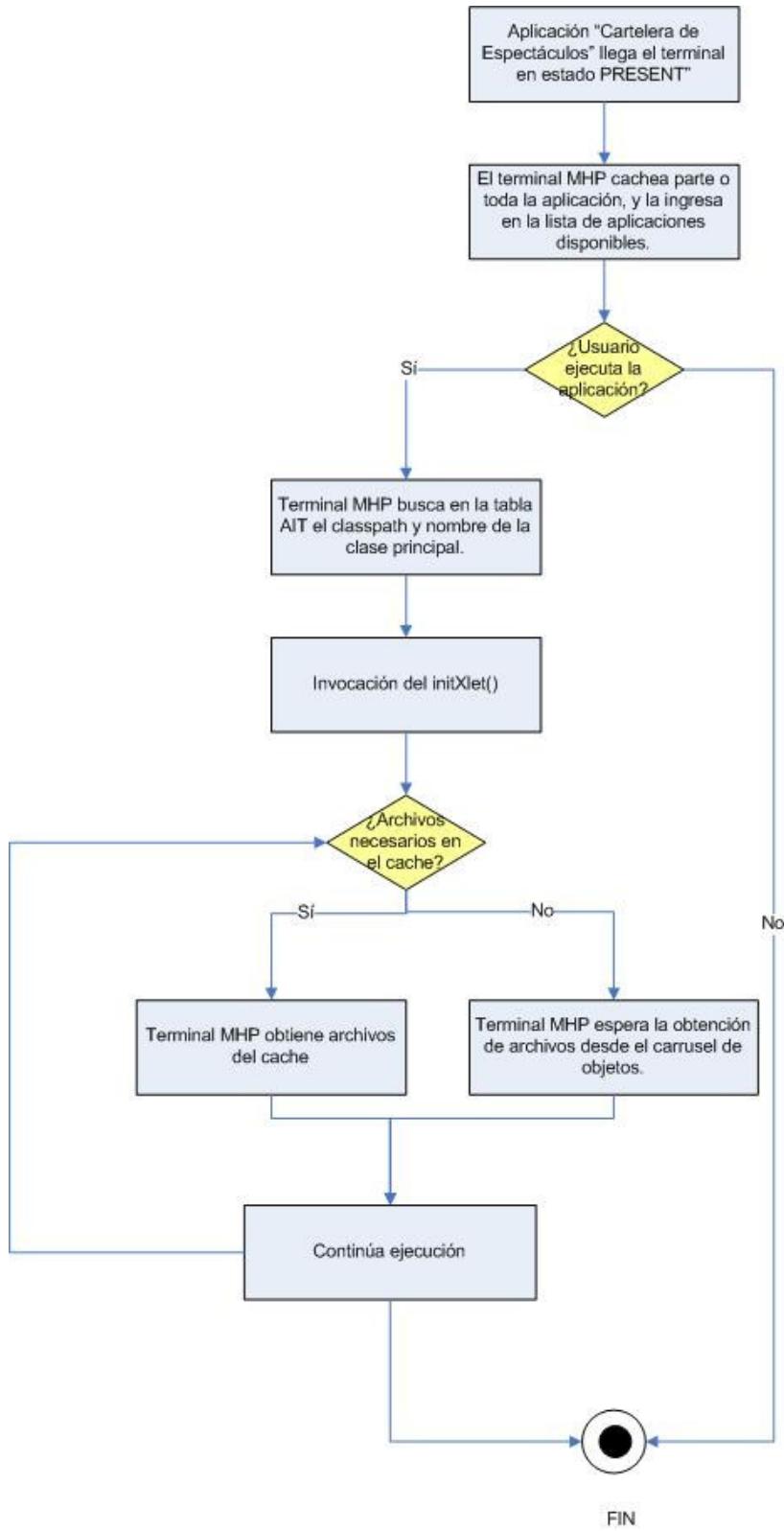


Figura 4.8: Diagrama de flujo de ejecución de la aplicación Cartelera

El tiempo que demora el usuario en percibir la ejecución de la aplicación se llama tiempo de arranque, y el mismo puede ser descrito de la siguiente manera:

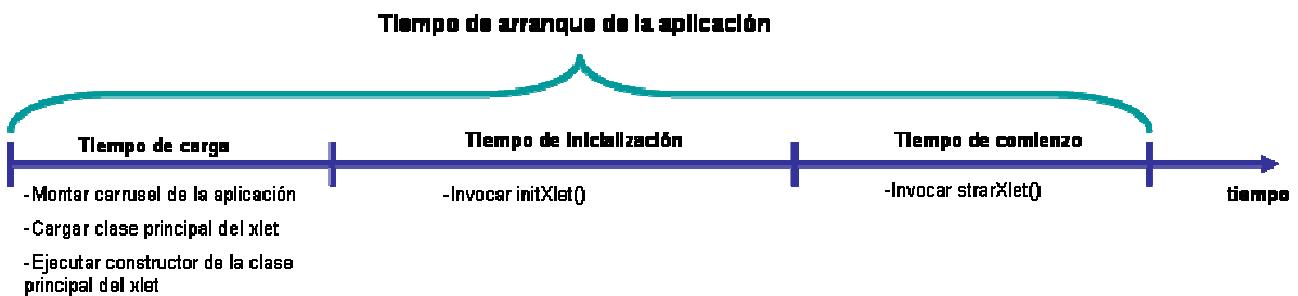


Figura 4.9: Tiempo de arranque de una aplicación

- **Tiempo de carga:** En primer lugar el carrusel asociado a la aplicación debe ser montado. Luego el archivo principal .class del Xlet será cargado en el terminal MHP, para luego ejecutar el constructor del mismo. Si al instanciar la clase principal del Xlet se necesitan otros archivos, éstos también deberán ser cargados.
- **Tiempo de inicialización:** Se ejecuta el método `initXlet()`. Es posible que este método requiera de la carga de otros archivos desde el carrusel. Este tiempo también está directamente relacionado al procesamiento de CPU llevado a cabo.
- **Tiempo de comienzo:** Se invoca el método `startXlet()`.

4.9 Tarea 4.1

Elección de la herramienta de desarrollo Java

La mayoría de las herramientas de desarrollo que facilitan la generación de aplicaciones interactivas según MHP son licenciadas. Sin embargo algunas de ellas cuentan con trials que pueden ser descargados. Las siguientes herramientas fueron descargadas:

- JAME Author
- Cardinal Studio Authoring tool
- Icareus iTV Suite Author

Se evaluaron las tres herramientas en paralelo, éstas facilitan el desarrollo de aplicaciones MHP pero no son flexibles ni transparentes. Esto se debe a que no proporcionan una vista del código a medida que se van diseñando las pantallas por interfaz gráfica. En consecuencia, se utilizó como herramienta de desarrollo Eclipse, que simplemente ofició como compilador luego de que se importaron las librerías MHP. De esta manera la implementación permitía adquirir un conocimiento más profundo de la tecnología.

Vale la pena aclarar que las herramientas anteriormente mencionadas pueden ser utilizadas con fines prácticos y resultan muy fáciles de manipular, permitiendo dar a la aplicación un atractivo “look and feel”.

4.10 Tarea 4.4

Diagrama de clases de la aplicación

El diagrama de clases de la aplicación se presenta a continuación. En el mismo se detallan los principales atributos y métodos de las clases implementadas:

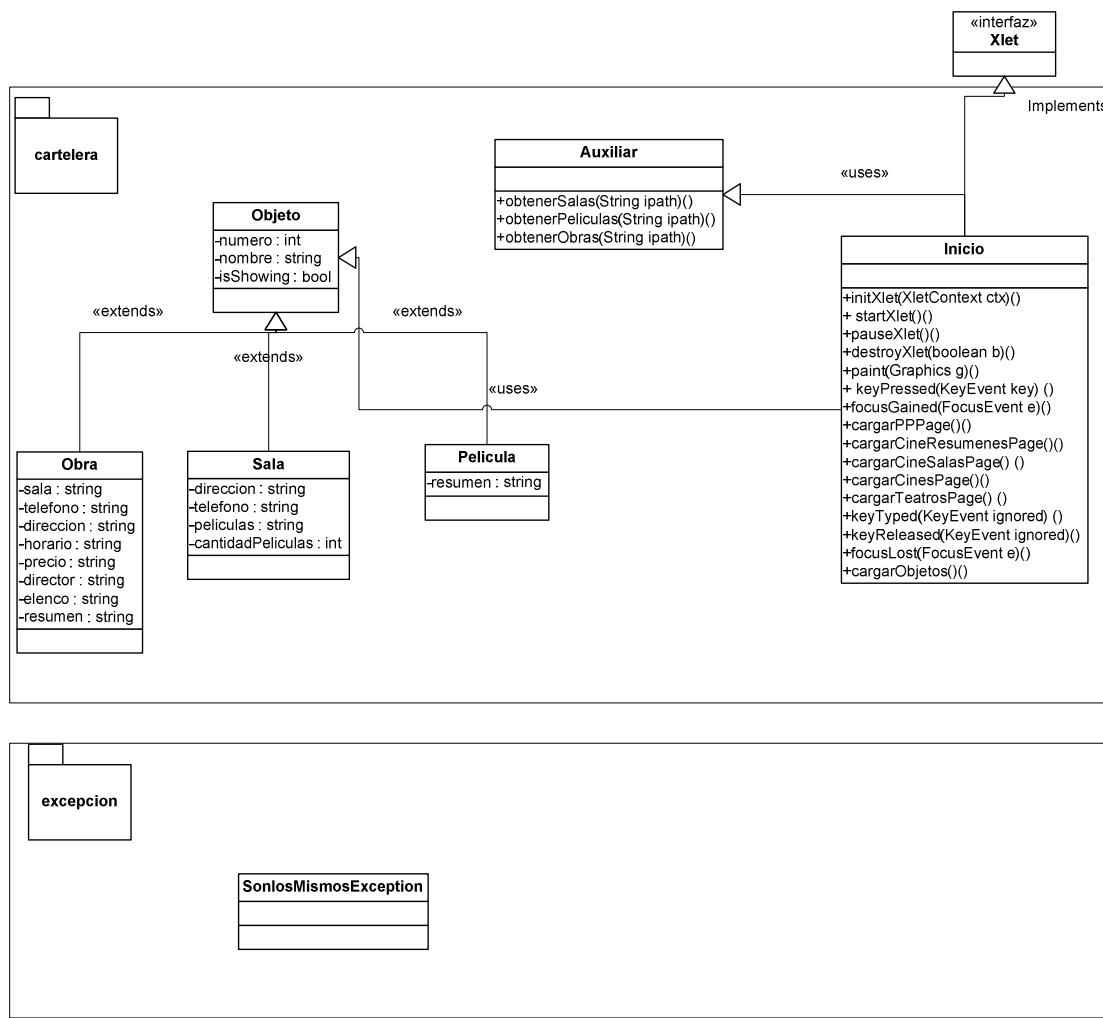


Figura 4.10: Diagrama de clases de la aplicación Cartelera

Como se puede apreciar, la aplicación cuenta con dos paquetes: “cartelera” y “excepcion”. La clase principal, llamada Inicio, es la encargada de desplegar los diferentes contenidos en pantalla. Desde ella se invocan métodos de la clase Auxiliar para manipular el contenido de texto a desplegar en pantalla, entre otras cosas. En la sección 4.11 es posible encontrar una descripción detallada de cada clase.

4.11 Tarea 4.6

Desarrollo de código

Esta tarea consiste en la codificación de la aplicación “Cartelera de Espectáculos” teniendo en cuenta lo investigado en la primera etapa del proyecto, y lo diseñado en la tarea “Diseño de interfaces”. A continuación se presentan los detalles sobre la implementación de la aplicación.

Información general

La aplicación desarrollada tiene como objetivo informar a los televidentes sobre las diferentes opciones que ofrece la cartelera de espectáculos de la ciudad de Montevideo. Los espectáculos contemplados fueron cines y teatros.

Interfase gráfica

La misma está compuesta por 5 pantallas en donde el usuario puede consultar información acerca de las películas exhibidas en el cine, sus horarios y salas, y también información sobre las obras de teatro existentes.

Las pantallas existentes se presentan a continuación. Las mismas fueron capturadas desde el emulador XleTView.



Figura 4.11: Pantalla “Cartelera de Espectáculos”



Figura 4.12: Pantalla “Cartelera de cines”



Cines-Salas y Horarios

Sala: Casablanca
Dirección: 21 de Setiembre 2838
Teléfono: 712 3795
Películas:
El invitado 16:45 - 18:35 - 20:25 - 22:15
El círculo 18:30
La profesión de Irina Palm 16:15 - 18:20 - 20:25 - 22:30
La desconocida 16:15 - 20:15 - 22:30

Sala: Grupocine Arocena
Dirección: Av. Arocena 1660
Teléfono: 901 4242
Películas:
Red de mentiras 20:00 - 22:40
Control total 15:10 - 17:40 - 20:05 - 22:30
High School Musical 3: la graduación 15:05 - 17:30 - Doblada al español
Arráncame la vida 15:20 - 17:40 - 20:10 - 22:30 - Sábado y domingo no se

[SALIR](#) [Altas](#) [Resúmenes](#)

Figura 4.13: Pantalla “Cines – Salas y Horarios”

Cine - Resúmenes de Películas

Nombre: Ceguera

Resumen: Del director brasileño Fernando Meirelles, y basada en la novela "Ensayo sobre la ceguera", de José Saramago. La película narra el drama de una ciudad quejada por una ceguera colectiva.



Nombre: El círculo

Resumen: Film documental de José Luis Charlo y Aldo Garay, acerca de la vida del científico y ex tupamaro Henry Engler.



 [Salir](#)  [Atrás](#)  [Horarios](#)

Figura 4.14: Pantalla “Cine – Resúmenes de Películas”

Cartelera de Teatros

Obra: Barro negro
Sala: Teatro Sobre Ruedas
Teléfono: 915 8618
Dirección: Explanada del Teatro Solís - Buenos Aires s/n
Precio: \$ 180. Tarjeta Joven, jubilados, Socio Espectacular \$120
Horario: Sábado 21:00 - Domingo 20:30 - Último fin de semana
Director: Marcelino Duffau
Elenco: Julio Icasuriaga, Sandra Bartolomeo, Lucy Arregui
Resumen: El espectáculo que marcó un hito en la historia del teatro vuelve a escena en su decimoséptima temporada. El ómnibus de Barro Negro sale de la puerta del Teatro Solís

Obra: Cabaret
Sala: Alianza Sala 1 - Sala China Zorrilla
Teléfono: 908 1953
Dirección: Paraguay 1217
Precio: viernes \$ 250. Sa/Do. \$ 300
Horario: Vi/Sa. 21:30 - Domingo 19:30 - Último fin de semana
Director: Omar Varela
Elenco: Nacho Cardozo, Sara Sabah, Álvaro Pozzolo
Resumen: Nacho Cardozo, Sara Sabah, Álvaro Pozzolo, Ana Rosa, Sergio Pereira y Bettina Mondino encabezan un elenco de 20 artistas incluyendo actores y bailarines.

↓

Salir **Otras**

Figura 4.15: Pantalla “Cartelera de Teatros”

Para la navegabilidad entre las distintas pantallas se utilizan los botones de color que se encuentran estandarizados por la norma MHP y deben ser implementados por todo control remoto certificado bajo este estándar:

- Botón verde: se utiliza para “entrar” a la opción que se encuentre seleccionada. Por defecto la primera opción de la lista es la que queda marcada.
- Botón Rojo: es el único que se encuentra disponibles en las 5 pantallas y tiene la funcionalidad de abortar la ejecución de la aplicación.
- Botón Amarillo: Se utiliza para ir a la pantalla anterior.

- Botón Azul: Permite ir de la pantalla “Resúmenes de películas” a las de “Horarios y salas de películas” y viceversa.

En aquellas pantallas donde aparecen las flechas azules hacia arriba y abajo es posible navegar a través del contenido de las mismas bajo la modalidad “jump scroll”.

Vale la pena aclarar que los botones habilitados en cada pantalla se encuentran dibujados en la parte inferior de las mismas para brindar una guía al usuario. El resto de los botones provistos por el control remoto carecen de funcionalidades en la aplicación.

Guía para el proveedor de contenidos - Información sobre archivos de contenido

La aplicación en cuestión necesita ser alimentada por ciertos contenidos:

- Películas exhibidas, salas y horarios.
- Obras de teatro exhibidas, salas y horarios.

Esta información será desplegada en pantalla, pero para su correcta visualización los archivos de contenido deben cumplir con cierta estructura.

Se cuenta con 3 archivos de contenido:

- Teatros.txt
- Salas.txt
- Películas.txt

Cada uno tiene diferentes datos y es necesario que se respete el orden y el espacio estipulado para cada parámetro de información, para ello a continuación se muestra el contenido y el formato de cada uno de estos archivos.

Teatros.txt

Este archivo contiene toda la información asociada a la cartelera de teatro. Cada obra contiene nueve atributos, éstos son:

- Nombre de la Obra
- Sala en la que se exhibe la obra
- Teléfono de la Sala
- Dirección de la Sala
- Día y hora en la que se exhibe la obra
- Precio
- Director/es de la obra
- Elenco
- Resumen y descripción de la obra

Separador: Cada uno de éstos atributos debe estar separado por el símbolo “::” (doble dos puntos), y cuando se trate de diferentes obras, el último atributo de una y el primero de la otra estarán separados también por el separador “::”.

Cantidad de caracteres: El contenido de todos los atributos, salvo el resumen, debe tener como máximo 67 caracteres, si se excede este número el contenido aparecerá cortado y sólo se presentarán los primeros 67 caracteres. En el caso del resumen, el mismo puede contener un máximo de 201 caracteres que serán desplegados en tres líneas.

El siguiente es un ejemplo de cómo debe ser armado el archivo Teatros.txt.

```
::Nombre de la obra1 :: Sala :: Teléfono :: Dirección :: Horario :: Precio :: Director/es ::  
Elenco :: Resumen :: Nombre de la obra2 :: Sala ...
```

```
::Barro negro::Teatro Sobre Ruedas::915 8618:: Explanada del Teatro Solís
- Buenos Aires s/n::Sábado 21:00 - Domingo 20:30 - Ultimo fin de
semana::$ 180. Tarjeta Joven, jubilados, Socio Espectacular $120::
Marcelino Duffau::Julio Icasuriaga, Sandra Bartolomeo, Lucy Arregui::El
espectáculo que marcó un hito en la historia del teatro vuelve a escena
en su decimoséptima temporada. El ómnibus de Barro Negro sale de la
puerta del Teatro Solís::Cabaret::Alianza Sala 1 - Sala China
Zorrilla::908 1953::Paraguay 1217::Vi/Sa. 21:30 - Domingo 19:30 - Ultimo
fin de semana::viernes $ 250. Sa/Do. $ 300::Omar Varela:: Nacho Cardozo,
Sara Sabah, Álvaro Pozzolo::Nacho Cardozo, Sara Sabah, Álvaro Pozzolo,
Ana Rosa, Sergio Pereira y Bettina Mondino encabezan un elenco de 20
artistas incluyendo actores y bailarines.
```

Cabe destacar que el texto debe comenzar también con el separador “::” para que la visualización funcione correctamente.

Salas.txt

El objetivo de este archivo es proporcionar la información asociada a las diferentes salas de cines. El mismo está compuesto por cuatro atributos, estos son:

- Nombre de la Sala
- Dirección de la Sala
- Teléfono de la Sala
- Películas exhibidas

En pantalla se desplegará la programación de dos salas a la vez, pudiendo navegar entre las diferentes salas a través de las flechas que se indican en pantalla, tanto hacia arriba o hacia abajo.

Separador: Para el armado del archivo se utilizan 2 tipos de separadores. Para separar las salas el símbolo es “;,” (cuatro punto y coma) y para separar las películas dentro de una misma sala y el resto de los atributos mencionados se utiliza el símbolo “::” (doble dos puntos).

Cantidad de caracteres y atributos: Al igual que en el caso del archivo Teatros.txt el contenido de cada atributo no podrá exceder los 67 caracteres. Por otro lado, la cantidad de películas exhibidas en cada sala no podrá exceder el monto de 14 películas.

El siguiente es un ejemplo del archivo Salas.txt

```
;;;; Sala1 :: Dirección :: Teléfono :: Película1 y horarios de la misma :: Película2 y horarios  
de la misma ;;; Sala2 :: Dirección ...
```

```
;;;;Casablanca::21 de Setiembre 2838::712 3795:: El invitado 16:45 -  
18:35 - 20:25 - 22:15::El círculo 18:30::La profesión de Irina Palm  
16:15 - 18:20 - 20:25 - 22:30::La desconocida 16:15 - 20:15 -  
22:30;;;;Grupocine Arocena::Av. Arocena 1660::901 4242:: Red de mentiras  
20:00 - 22:40::Control total 15:10 - 17:40 - 20:05 - 22:30::High School  
Musical 3: la graduación 15:05 - 17:30 - Doblada al  
español::Arráncame la vida 15:20 - 17:40 - 20:10 - 22:30 - Sábado y  
domingo no se exhibe;;;;Grupocine Punta Carretas::José Ellauri 350 nivel  
3::901 4242:: Muerte en un funeral 22:15 - Viernes 0:20 - Sáb y  
Dom no se exhibe::High School Musical 3: la graduación 15:10 - 17:30  
- 19:50 - Doblada al español::Un novio para mi mujer 15:40 - 18:00 -  
20:10 - 22:20 - Vi/Sa. 0:40 - Sábado y domingo se exhibe solo 22:20 y  
0:40 (sábado)::Una chihuahua de Beverly Hills 15:05 - Doblada al  
español::Control total 17:00 - 19:30 - 22:00 - Vi/Sa. 0:30;;;;Opera  
1::18 de Julio 1710::403 1415:: Mamma mia!: la película 16:00 - 18:10  
- 20:20::Una guerra de película 22:30 - Vi/Sa. 0:30
```

El archivo debe comenzar con el separador “;;;;”.

Películas.txt

Este archivo provee la información correspondiente al resumen de cada película, por lo que contiene simplemente dos atributos:

- Nombre de la película
- Resumen de la película

En pantalla se visualizarán dos películas a la vez con su correspondiente resumen.

Separador: Para el armado del archivo se utiliza el separador “::”, como se muestra en el ejemplo a continuación.

Cantidad de caracteres: El nombre de la película no podrá exceder los 67 caracteres y el resumen de la misma podrá tener un máximo de 335 caracteres, lo que corresponde a 5 líneas.

::Película1::Resumen::Película2::Resumen ...

::Control total::Dos personas que no se conocen entre sí, son convocadas por una misteriosa llamada telefónica, y acabarán por involucrarse en el mundo del terrorismo. Para escapara de su peligrosa situación, enfrentarán una mente criminal que dispone de la más sofisticada tecnología.::Arráncame la vida::Basada en la novela homónima de la escritora mexicana Ángeles Mastreta, el film narra un triangulo amoroso signado por luchas políticas en la década de 1930.::Ceguera::Del director brasileño Fernando Meirelles, y basada en la novela "Ensayo sobre la ceguera", de José Saramago. La película narra el drama de una ciudad quejada por una ceguera colectiva.::El círculo::Film documental de José Luis Charlo y Aldo Garay, acerca de la vida del científico y ex tupamaro Henry Engler.

Generación dinámica de los archivos de contenido

Si bien es posible generar los archivos de contenido anteriormente mencionados a mano, y así alimentar a la aplicación MHP “Cartelera de Espectáculos”, se programaron formularios HTML con código php embebido que permitirán hacerlo de manera más amigable. En el Anexo 7, “Generador de contenidos – Cartelera de Espectáculos”, se adjunta información del mismo, instrucciones para su uso, y el código asociado.

Guía para el desarrollador

Estructura de Directorio

La estructura de directorio del proyecto “Cartelera” es la siguiente:

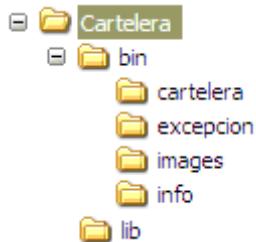


Figura 4.16: Estructura de directorios de la aplicación

Tanto la carpeta “cartelera” como la carpeta “expcion” son parte de la estructura de paquetes y contienen las siguientes clases:

- cartelera.Inicio
- cartelera.Auxiliar
- cartelera.Objeto
- cartelera.Pelicula
- cartelera.Obra
- cartelera.Sala
- excepcion.SonlosMismosException

Por otro lado, la carpeta “images” contiene los siguientes archivos:

- bluespot.png
- cine.png
- downarrow.png

- uparrow.png
- horas.jpg
- resumen.png
- cine.jpg
- greenspot.png
- horas.png
- redspot.png
- teatro.png
- yellowspot.png

Los contenidos se encuentran en la carpeta “info”, que contiene los siguientes archivos:

- Peliculas.txt
- Salas.txt
- Teatros.txt

Finalmente la carpeta “lib” contiene el .jar necesario para la compilación de la aplicación, y contiene el siguiente archivo:

- mhpstubs.jar

Descripción de clases

cartelera.Inicio.java

Esta es la clase principal de la aplicación, y por lo tanto contiene los métodos principales de un xlet: initXlet(), startXlet(), pauseXlet() y destroyXlet(). Tal como recomienda MHP, el constructor de esta clase no realiza ninguna acción, todas las acciones de inicialización se hacen dentro del método initXlet(). Éste último método se encarga principalmente de crear un HScene que actuará como “pizarra” para ir dibujando las diferentes pantallas. Esta es la única clase que formalmente interactúa con el usuario y procesa los eventos relacionados a la presión de botones del control remoto.

Principales campos

```
private HScene scene;
```

El HScene es la “pizarra” sobre la que se dibujarán las diferentes pantallas.

```
private HContainer contX;
```

Existe un atributo de estos por cada pantalla. El HContainer es sobre el cual se ubican los diferentes componentes gráficos. Luego es el HContainer el que se ubica sobre el HScene mediante el comando HScene.add(HContainer). Para saber si un container está visible se utiliza el método HContainer.isShowing().

```
private Image x;
```

Este atributo es utilizado para cargar las imágenes necesarias desde el carrusel de objetos.

```
private HTextButton x;
```

El HTextButton es uno de los tantos componentes gráficos que pueden ser agregados a un HContainer. Este componente es un botón al que se le puede parametrizar el texto que presenta.

```
private HGraphicButton x;
```

El HGraphicButton es como el botón anteriormente detallado, pero sólo presenta una imagen. En caso de querer hacer un botón con texto e imagen es posible utilizar el siguiente código:

```
public class MyButton extends HGraphicButton {  
    private String text;  
    public MyButton(String text) {  
        this.text=text;  
    }  
    public void paint(Graphics g) {
```

```
//despliega la imagen...
super.paint(g);
// despliega el texto
g.drawString(text,hereX,hereY);
}
}

private HStaticText x;
```

El atributo en cuestión representa un cuadro de texto.

En el caso de esta aplicación algunos cuadros de texto fueron utilizados para introducir la información provista por los archivos de contenido. El control del texto para que éste no sobrepase los límites del cuadro se hace a través de métodos implementados en la clase Auxiliar. Sin embargo, si los archivos de contenido exceden la cantidad de caracteres especificada, el texto no aparecerá por completo en el cuadro. También se puede utilizar el DVBTTextLayoutManager, pero igualmente se debería contar con los métodos de control, y la única diferencia radicaría en que al usar el manager se introduciría “(...)” si no es posible desplegar el texto completo.

```
private DSMCCObject x;
```

En estos atributos se cargan los objetos obtenidos del carrusel.

Principales Métodos

```
public void initXlet(XletContext ctx) throws XletStateChangeException;
```

El método initXlet() pone a la aplicación de forma no visible para el usuario mientras solicita los recursos necesarios para la ejecución de la misma. En particular se crea el HScene de la aplicación a través de la clase org.havi.ui.HSceneFactory y se establecen todos los parámetros de la misma. Es importante habilitar el KeyListener del HScene para que puedan capturarse los eventos generados por el control remoto.

```
public void startXlet() throws XletStateChangeException;
```

El método startXlet() se encarga de hacer visible la primer pantalla de la aplicación, el menú inicial, “Cartelera de espectáculos”. Para esto debe invocar el método validate(), poner el HScene visible al usuario, solicitar el “focus” e invocar el método cargarPPPage() que desplegará las opciones de la página principal.

```
public void pauseXlet();
```

El método pauseXlet() será invocado cuando el receptor necesite pausar la aplicación sin liberar por completo los recursos que la misma utiliza. Para esto simplemente hace que el HScene no se visualice.

```
public void destroyXlet(boolean b);
```

destroyXlet() termina la aplicación y por lo tanto elimina el HScene liberando los recursos que la misma consume.

```
public void paint(Graphics g);
```

El método paint() se encarga de actualizar los gráficos de la aplicación, y es llamado de manera automática cuando el HScene necesita ser redibujado. Debe considerarse que aquellas acciones realizadas dentro del paint serán comunes a todas las pantallas. Por esta razón, en el caso de la aplicación “Cartelera de Espectáculos”, el método paint() se encarga de dibujar el botón rojo para terminar con la aplicación.

```
public void focusGained(FocusEvent e);
```

Este método se encarga de colorear los componentes cuando se encuentran seleccionados.

```
public void keyPressed(KeyEvent key);
```

Para manejar la captura de eventos generados por acción de los botones del control remoto se utiliza el método keyPressed(KeyEvent key). Este método a partir del evento capturado y la situación actual de la aplicación decide cómo proceder. Por ejemplo, cada vez que se accione el botón rojo se debe invocar el método destroyXlet(), sin importar en qué pantalla se encuentre la aplicación. Para el resto de los botones se debe tener en cuenta desde qué pantalla fueron invocados. Para ello, dado que cada pantalla está representada por un contenedor (org.havi.ui.HContainer), simplemente se debe consultar si dicho contenedor se encuentra visible mediante el método isShowing (ej : contCines.isShowing()). El método keyPressed captura KeyEvents que pueden ser referenciados por su nombre o por su número; existen algunos cuyos respectivos números se encuentran estandarizados como por ejemplo los botones de color del control remoto, en ese caso conviene usar su número, en caso contrario se los referencia mediante su nombre ej: KeyEvent.VK_DOWN.

```
public void cargarPPPage();  
  
private void cargarCineResumenesPage();  
  
private void cargarCineSalasPage();  
  
private void cargarCinesPage();  
  
private void cargarTeatrosPage();
```

Los métodos en cuestión cargan información en pantalla, botones, imágenes, etc. Cada pantalla se encuentra implementada a través de un HContainer al que se le agregan componentes (HComponent) según lo que se desea visualizar en pantalla. Si bien se podría haber realizado una clase por cada pantalla en este caso se optó por

realizar una versión más unificada de la aplicación concentrando gran parte del desarrollo en una sola clase, la clase Inicio. Esta preferencia responde al hecho de que cuanto más unificada esté la aplicación, más eficiente será la ejecución de la misma, ya que menos archivos deberán ser cargados desde el carrusel de objetos. Para la carga de las cinco pantallas se definió un método que inicializa el contenedor adecuado y agrega los componentes necesarios (HStaticText, HTextButton, HGraphicsButton, etc.).

Para aquellas pantallas que no tienen opción de navegabilidad, lo que significa que no tienen botones, igualmente es necesario colocar un botón Invisible para hacer un requestFocus y así poder seguir capturando eventos. En el caso de esta aplicación, para que las pantallas no pierdan el accionado de eventos, se agregó un HTextButton que sea invisible para el usuario pero que haga un requestFocus.

```
private void actualizarPaginaTeatroArriba();  
  
private void actualizarPaginaTeatroAbajo();  
  
private void actualizarPaginaResumenArriba();  
  
private void actualizarPaginaResumenAbajo();  
  
private void actualizarPaginaSalasArriba();  
  
private void actualizarPaginaSalasAbajo();
```

Estos métodos están asociados a la actualización de aquellas pantallas que permiten ser navegadas mediante un jump Scroll.

```
public void cargarObjetos();
```

El método cargarObjetos es el encargado de cargar todos los objetos que son necesarios por la aplicación a través del acceso al carrusel de objetos. Este método es invocado desde el startXlet() al comienzo de la aplicación. La carga de dichos objetos se realiza de manera asíncrona para que la aplicación no se bloquee ante demoras en la carga de los mismos.

Para el acceso al carrusel de objetos se utilizó como base el código que se encuentra en marco teórico en la sección “2.3.12.4 Ventajas de la carga asíncrona”. Cabe aclarar que se realizaron algunas modificaciones porque dicho código no era compilable tal cual como aparece en dicha sección. Para ser más específicos cuando se instancia el locator a utilizar por la aplicación en vez de utilizar el org.davic.net.Locator se utilizó el org.davic.net.dvb.DvbLocator. El locator es obtenido del Service context; también podría ser ingresado por la emisora en el código fuente.

Es importante aclarar el siguiente aspecto: para acceder a los archivos .txt o imágenes enviados por la emisora es posible hacer uso de ambos códigos presentados a continuación.

OPCIÓN 1: API de acceso standard a archivos

```
BufferedReader in = new BufferedReader ( new InputStreamReader ( new FileInputStream ("name.txt") ) );
```

OPCIÓN 2: DSM-CC API

```
DSMCCObject carouselfile = new DSMCCObject("name.txt");  
carouselfile.asynchronousLoad(this);
```

```
FileInputStream inputStream;
inputStream = new FileInputStream(carouselFile);
```

La diferencia reside en el tiempo de respuesta de la aplicación. Mientras la primera opción necesita esperar a acceder al archivo para continuar, la siguiente es mas eficiente, ya que es un método asíncrono de acceso a la información. Por otro lado debe tenerse en cuenta que el DSM-CC, a diferencia del modo estándar de acceso a archivos, permite no sólo acceder a archivos, sino también a información normal play time o información de stream events.

Es posible utilizar la primera opción para acceder al carrusel de objetos, pero se debe considerar que bajo este método no hay manera de pre-cargar un archivo desde el carrusel de objetos o monitorear las versiones del mismo. De esta manera el carrusel de objetos se convertiría en un filesystem con gran latencia, lo que significa que la aplicación se bloquearía hasta que se hayan cargado los archivos especificados.

cartelera.Auxiliar.java

Dentro de esta clase se encuentran una serie de métodos asociados a la visualización de los contenidos de texto en la pantalla y son específicos de la aplicación. Se encarga de que los contenidos cuenten con los largos y formatos adecuados para ser desplegados en pantalla.

cartelera.Objeto.java

Es la clase padre de las clases “Película”, “Sala” y “Obra”. La creación de este tipo de clase ayuda a manipular la información que se obtiene de los archivos .txt del carrusel de objetos.

Principales campos

```
private int numero;
```

Es el número de identificación del objeto.

```
private String nombre;
```

Es el nombre del objeto.

```
private boolean isShowing;
```

Indica si el objeto se está visualizando en pantalla o no.

Principales métodos

Cada atributo tiene definido su getter y su setter para poder acceder a los datos y setearlos.

cartelera.Pelicula.java

Esta clase hereda de “Objeto” y agrega el atributo de resumen asociado a las películas.

Principales campos

```
private String resumen;
```

Principales métodos

Se implementa el getter y el setter del atributo anterior.

cartelera.Sala.java

Toda la información que puede tener una Sala se manipula mediante esta clase. Al igual que la clase anterior hereda de “Objeto” y agrega algunos atributos.

Principales campos

```
private String direccion;
```

```
private String telefono;
```

```
private String peliculas;
```

Nombre de las películas exhibidas en dicha sala.

```
private int cantidadPeliculas;
```

Cantidad de películas exhibidas en la sala.

Principales métodos

Se implementa el getter y el setter de cada atributo anteriormente mencionado.

cartelera.Obra.java

Al igual que las anteriores hereda de la clase “Objeto” y agrega algunos atributos.

Principales campos

```
private String sala;  
  
private String telefono;  
  
private String direccion;  
  
private String horario;  
  
private String precio;  
  
private String director;  
  
private String elenco;  
  
private String resumen;
```

Principales métodos

Todos estos atributos tiene su getter y setter para manipular la información.

excepcion.SonlosMismosException

Por practicidad es conveniente que cada desarrollo implemente al menos una excepción propia para poder manejar situaciones de error. Esto implica que los casos límites serán contemplados y manejados con este tipo de recurso. Para esto se desarrolla la excepción “SonlosMismosException” que se encuentra bajo el paquete “excepcion”. Esta excepción es lanzada en los métodos de actualización de páginas cuando el usuario acciona eventos que carecen de funcionalidad y por lo tanto no es necesario hacer ninguna actualización de la página que se está visualizando.

Compilación

Si bien es posible utilizar herramientas de desarrollo específicas MHP que ya contengan las clases contra las que se debe compilar la aplicación, también es posible utilizar una herramienta de desarrollo Java genérica. En este último caso para compilar las aplicaciones MHP se necesita un set de clases que lo permitan. Existen implementaciones comerciales de MHP que se obtienen en un JAR, e implementaciones libres y gratuitas MHP, aunque éstas últimas en la mayoría de los casos no son implementaciones completas.

En el caso del proyecto “Cartelera”, para compilar la aplicación, se utilizó la herramienta Java Eclipse, se instalo el J2SE Software Development Kit y se agregaron al proyecto los jars necesarios para implementar MHP. Los archivos .jar son implementaciones de las clases que se definen en MHP y por lo tanto es imprecindible que el proyecto cuente con las mismas. Se probó el funcionamiento de la aplicación con archivos .jar extraídos de los siguientes sitios web.

1. TV without Borders: Se descargaron las stub classes provistas por esta página web (http://www.interactivetvweb.org/resources/code_samples) y luego se agregaron al proyecto. Cabe destacar que este jar incluye los paquetes javax.* y org.* debido a que no es una implementación completa del estándar MHP.
2. Code 4TV: Se obtuvieron los jars publicados en la página en cuestión (<http://www.code4tv.com/c/downloads>). En este caso la implementación de las clases se encuentran en archivos distintos por lo que se agregaron al proyecto los archivos javatv10.jar y havi_dvb_davic112-1.1.jar que se encuentran dentro del rar code4tv-MHP-1.1.2-stubs-v1.1.esp provisto por la página.

Los resultados obtenidos con ambas implementaciones fueron óptimos.

En el Anexo 6 se adjunta el código correspondiente a la aplicación “Cartelera de Espectáculos”.

2. Tarea 5.1

Búsqueda de un emulador MHP

Una de las tareas más complicadas a la hora de desarrollar aplicaciones interactivas es establecer el ambiente de desarrollo MHP para la creación y testeo de Xlets. Existen

sistemas de desarrollo comerciales (versiones de desarrollo de STBs) que rondan los 1000 euros (<http://www.cesnavarra.net>, “TDT: herramientas para desarrollo MHP (Parte 2)”, junio de 2008), o existe la posibilidad de hacer uso de un emulador Xlet y utilizarlo en una PC realizando un testeo local.

En la web existen algunos emuladores MHP cuyo principal objetivo es simular un STB, y por lo tanto permiten evaluar el desempeño gráfico de la aplicación. Durante el proyecto en cuestión se utilizó el emulador XleTView. Éste es uno de los emuladores de Xlets más populares de uso libre, aunque existen otros como OpenMHP.

Vale la pena aclarar que el testeo de la aplicación con este tipo de software deja de lado algunos aspectos importantes de las aplicaciones MHP, como por ejemplo señalización, frecuencia de emisión, acceso al carrusel de objetos, entre otros; y aún no tiene implementado por completo el estándar MHP, por lo que es posible encontrar que un código compila, pero al ejecutarlo en el emulador no funciona. Este es el caso del DVBTTextLayoutManager, DSMCCObject, HtoggleButton, HpermissionDeniedException, entre otros. Para ver el estado actual de implementación del XleTView es posible acceder al siguiente link: <http://www.xletview.org/status/status-0.3.6.html>.

Por estas razones, XleTView es apropiado para principiantes en esta área de conocimiento, pero debe tenerse en cuenta que no sustituye el ambiente real de testeo de las aplicaciones interactivas. Si se quiere un sistema de desarrollo serio se puede recurrir a ADB, Alticast, IRT u Osmosys. Por otro lado XleTView no es adecuado para el desarrollo de aplicaciones, sólo para su testeo.

4.12 Tarea 5.2

Descarga e instalación del emulador MHP

Prerrequisitos

- Plataforma J2SE versión 1.4 o posteriores
- JRE

Descarga

El emulador XleTView está disponible en Internet y no es necesario contar con ningún registro para descargarlo:

<http://sourceforge.net/projects/XleTView/>

La versión descargada es la 0.3.6.

Instalación

Descomprimir el archivo zip dentro del directorio desde donde se ejecutará.

La estructura de directorios es como sigue:

<DIR> config	Contiene los archivos de configuración de XleTView y las aplicaciones.
<DIR> jars	Contiene los jars utilizados por XleTView, no por el usuario.
<DIR> JMF2.1.1	Licencias para usar el JMF.
<DIR> license	Licencias para usar XleTView y los jars.
icon.ico	Ícono de XleTView.
readme.txt	Instrucciones simples para comenzar a utilizar XleTView.
XleTView.jar	La aplicación en sí misma (API MHP, etc.).

4.13 Tarea 5.3

Configuración del emulador MHP

Configurar los canales disponibles

Toda TV tiene una lista de canales disponibles para su uso, por lo que XleTView también. Sin embargo en el XleTView hay que indicar qué canales están disponibles debido a que éste no los detecta de manera automática.

Para configurar los canales se debe editar el archivo config/channels.xml cuya estructura por defecto es como sigue:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<CHANNELS>
  <CHANNEL>
    <NAME>0</NAME>
    <MEDIA>config/defaultbg.jpg</MEDIA>
  </CHANNEL>
</CHANNELS>
```

Cada canal está definido por:

- nombre o número
- media: indica al XleTView qué se debe desplegar en la pantalla al seleccionar este canal, una imagen JPEG (720*576) o un video AVI.

Área segura de pantalla

No todos los receptores de TV tienen pantallas del mismo tamaño, por lo que hasta un 10% de las imágenes pueden perderse por hacer la adaptación, y otro 10% puede distorsionarse. Para evitar estos problemas, se debe trabajar siempre en el área segura de la pantalla, el 80% de la parte central de la misma.

XleTView permite a los desarrolladores definir qué parte de la pantalla se encuentra dentro del área segura, y ésta aparecerá señalada por un borde amarillo.

El archivo config/settings.txt contiene los siguientes campos:

Parámetro	Descripción	Valor por defecto
safearea.show	Habilita/Deshabilita el despliegue del área segura.	true
safearea.color	Color utilizado para bordear el área segura.	#ffee00
safearea.height	Alto del área segura.	556
safearea.width	Ancho del área segura.	700
safearea.x	Offset horizontal del área segura (desde el límite izquierdo de la imagen de TV).	10
safearea.y	Offset vertical del área segura (desde el límite superior de la imagen de TV).	10

Tabla 4.3: archivo /settings.txt del emulador XleTView

Otras configuraciones

La interfase de usuario del XleTView es flexible y es posible modificarla a través del seteo de varios parámetros. Estos parámetros se encuentran en el archivo config/settings.txt. Los valores por defecto de los parámetros son los siguientes:

```

console.font=Helvetica
console fontsize=10
console height=200
console show=true
console width=500
console x=0
console y=0
extra.classpath=
file.applications=config/applications.xml
file.defaultbg=config/defaultbg.jpg
file.settings=config/settings.txt
font.sizeoffset=-2
path.config=config
remote.show=true

```

```
safearea.color=#ffee00
safearea.height=556
safearea.show=true
safearea.width=700
safearea.x=10
safearea.y=10
tv.center=true
tv.screenheight=576
tv.screenwidth=720
tv.x=0
tv.y=0
```

El significado de alguno de ellos se explica a continuación:

Parámetro	Significado
extra.classpath	Classpath adicional en donde se encontrarán las aplicaciones a ejecutar.
file.applications	Ubicación del archivo que define las aplicaciones existentes y como se agrupan las mismas.
file.defaultbg	Imagen utilizada por defecto para la pantalla de TV, puede ser una imagen o un video AVI.
font.sizeoffset	Setea el offset utilizado para el tamaño de fuente. Esto permite modificar el tamaño relativo de la fuente de manera que se visualicen igual que en el STB. Por ejemplo si se setea este valor en 2, lo que se visualizará en el XleTView como 16 se verá en el STB como 18.
remote.show	Mostrar/Ocultar el control remoto.
tv.screenheight	Alto de la pantalla simulada de TV.
tv.screenwidth	Ancho de la pantalla simulada de TV.

Tabla 4.4: Parámetros de configuración la interfase del XleTView

Configuración del control remoto (RCU)

El RCU disponible en el XleTView es bastante simple, y no resultará suficiente para aquellas aplicaciones más complejas que requieran de más teclas. En este caso el RCU

debe ser configurado para que cumpla con los requerimientos del desarrollador. El archivo de configuración del control remoto es el config/remote_control.xml.

El archivo de configuración por defecto es el siguiente:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<remotecontrol width="130" height="100"
    imgroot="config/remote_images/"
    backgroundimage="remote_bg.png"
    backgroundcolor="">

<buttons x="25" y="20" width="100" height="100">
    <button img="button_red.png"
        x="0" y="0"
        width="20" height="14"
        keycode="HRcEvent.VK_COLORED_KEY_0"/>
    <button img="button_green.png"
        x="20" y="0"
        width="20" height="14"
        keycode="HRcEvent.VK_COLORED_KEY_1"/>
    <button img="button_yellow.png"
        x="40" y="0"
        width="20" height="14"
        keycode="HRcEvent.VK_COLORED_KEY_2"/>
    <button img="button_blue.png"
        x="60" y="0"
        width="20" height="14"
        keycode="HRcEvent.VK_COLORED_KEY_3"/>
</buttons>

<buttons x="35" y="45" width="100" height="100">
    <button img="button_arrow_up.png"
        x="20" y="0"
        width="20" height="20"
        keycode="HRcEvent.VK_UP"/>
    <button img="button_arrow_left.png"
```

```
x="0" y="20"
width="20" height="20"
keycode="HRcEvent.VK_LEFT"/>
<button img="button_ok.png"
x="20" y="20"
width="20" height="20"
keycode="HRcEvent.VK_ENTER"/>
<button img="button_arrow_right.png"
x="40" y="20"
width="20" height="20"
keycode="HRcEvent.VK_RIGHT"/>
<button img="button_arrow_down.png"
x="20" y="40"
width="20" height="20"
keycode="HRcEvent.VK_DOWN"/>
</buttons>

<buttons x="35" y="110" width="100" height="100">
<button img="button_1.png"
x="0" y="0"
width="20" height="20"
keycode="HRcEvent.VK_1"/>
<button img="button_2.png"
x="20" y="0"
width="20" height="20"
keycode="HRcEvent.VK_2"/>
<button img="button_3.png"
x="40" y="0"
width="20" height="20"
keycode="HRcEvent.VK_3"/>
<button img="button_4.png"
x="0" y="20"
width="20" height="20"
keycode="HRcEvent.VK_4"/>
<button img="button_5.png"
x="20" y="20"
```

```

        width="20" height="20"
        keycode="HRcEvent.VK_5"/>
<button img="button_6.png"
        x="40" y="20"
        width="20" height="20"
        keycode="HRcEvent.VK_6"/>
<button img="button_7.png"
        x="0" y="40"
        width="20" height="20"
        keycode="HRcEvent.VK_7"/>
<button img="button_8.png"
        x="20" y="40"
        width="20" height="20"
        keycode="HRcEvent.VK_8"/>
<button img="button_9.png"
        x="40" y="40"
        width="20" height="20"
        keycode="HRcEvent.VK_9"/>
<button img="button_ast.png"
        x="0" y="60"
        width="20" height="20"
        keycode="HRcEvent.VK_ASTERISK"/>
<button img="button_0.png"
        x="20" y="60"
        width="20" height="20"
        keycode="HRcEvent.VK_0"/>
<button img="button_square.png"
        x="40" y="60"
        width="20" height="20"
        keycode="HRcEvent.VK_NUMBER_SIGN"/>
</buttons>

<buttons x="25" y="195" width="100" height="100">
<button img="button_exit.png"
        x="25" y="0"
        width="30" height="20"

```

```

        keycode="HRcEvent.VK_ESCAPE"/>
    </buttons>

    <buttons x="45" y="220" width="100" height="100">
        <button img="button_channel_down.png"
            x="0" y="0"
            width="20" height="20"
            keycode="HRcEvent.VK_CHANNEL_DOWN"/>
        <button img="button_channel_up.png"
            x="20" y="0"
            width="20" height="20"
            keycode="HRcEvent.VK_CHANNEL_UP"/>
    </buttons>

</remotecontrol>

```

Como se puede apreciar, este archivo define el elemento `<remotecontrol>`, su alto, ancho, imagen de fondo, color, botones, posición de los mismos. Lo más importante de este archivo reside en cómo se define la función de los botones. Existen códigos que las definen, y deben pertenecer a la lista de clases de `org.havi.ui.HRcEvent`. Por lo tanto, a través de este archivo es posible determinar qué eventos se dispararán al presionar cada botón del control remoto, eventos que serán capturados por la aplicación.

Interacción

A través del control remoto del XleTView es posible interactuar con las aplicaciones. Cada botón del control remoto se encuentra mapeado a una tecla del keyboard de la PC:

Control remoto	PC Keyboard
0-9	0-9
Rojo	F1
Verde	F2
Amarillo	F3
Azul	F4
Flechas	Flechas

Tabla 4.5: Control remoto y teclado de la PC

El resto de los botones del control remoto como Channel UP y Channel DOWN no se encuentran mapeados y deben cliquearse con el mouse.

Ejecución de aplicaciones en el XleTView

Para ejecutar una aplicación se debe ir al menú “Applications” y elegir la aplicación que se desea ejecutar. Es necesario para esto que el XleTView conozca las aplicaciones disponibles.

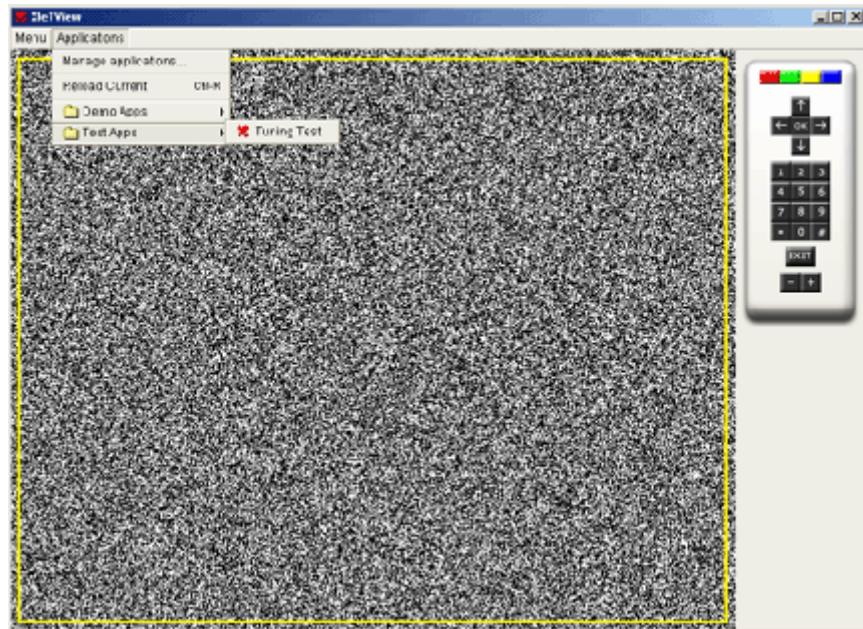


Figura 4.17: Ejecución de aplicaciones en el XleTView

Al seleccionar una aplicación, ésta comenzará automáticamente. Desplegará los gráficos en la pantalla de la TV simulada, y todo mensaje dirigido al system.out o system.err aparecerá en consola si es que se inició XleTView por línea de comandos.

Si se selecciona la opción "Reload Current" del menú "Applications" entonces la aplicación será reiniciada.

¿Cómo notificar al XleTView acerca de las aplicaciones disponibles?

En los receptores MHP es la señalización AIT quien notifica acerca de la existencia de nuevas aplicaciones. En el caso del XleTView se deben “registrar” las aplicaciones de la siguiente manera:

Ir al menú "Applications", elegir la opción "Manage Applications...". En ella se definen las aplicaciones y grupos existentes. Cada aplicación se identificará a través de tres parámetros:

- **Name:** Nombre que se le desea dar al Xlet.
- **Path:** Directorio de ubicación del Xlet.
- **Xlet:** Nombre de la clase main del Xlet.

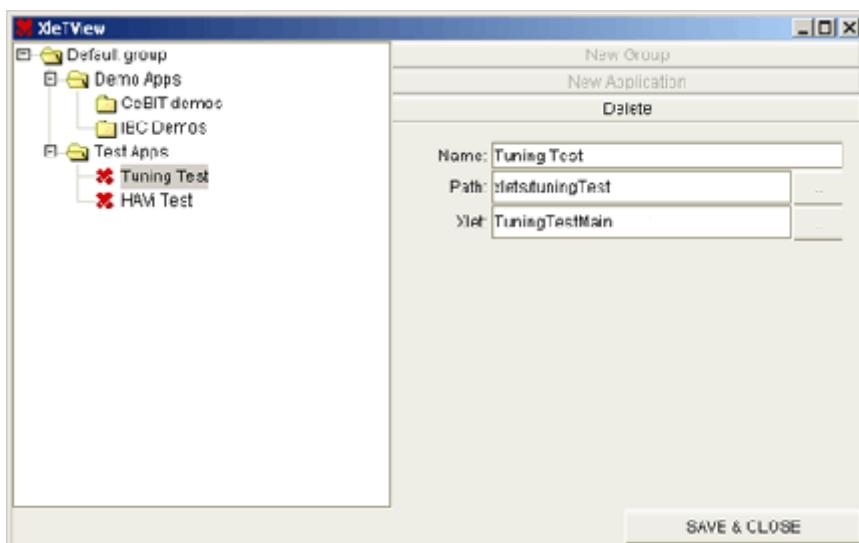


Figura 4.18: Registro de aplicaciones en el XleTView

Toda esta información será almacenada en el archivo applications.xml de la siguiente manera:

```
<!-- top element in this xml file -->
<APPLICATIONS>

    <!-- top element for one Xlet,
        resides in the default group -->
```

```
<APPLICATION>
<!-- name of the Xlet -->
<NAME>Foo Xlet</NAME>

<!-- path to the Xlet -->
<PATH>H:\MHP\foo\classes</PATH>

<!-- the main class of the Xlet -->
<XLET>foo.FooXlet</XLET>
</APPLICATION>

<!-- A second Xlet -->
<APPLICATION>
<NAME>Bar Xlet</NAME>
<PATH>H:\MHP\bar\classes</PATH>
<XLET>bar.barXlet</XLET>
</APPLICATION>

<!-- this makes a group called demos -->
<GROUP NAME="demos">
<APPLICATION>
<NAME>Demo Xlet</NAME>
<PATH>H:\MHP\demo\demo_1\classes</PATH>
<XLET>demo.DemoXlet</XLET>
</APPLICATION>
</GROUP>
</APPLICATIONS>
```

Es importante reiniciar el XleTView en el caso de que se realicen modificaciones directamente en los archivos de configuración, ya que éstos son leídos únicamente al iniciar el programa.

Importante:

No almacenar las aplicaciones bajo el mismo directorio en el que se encuentra el XleTView.jar, ni donde se encuentran las stub classes.

4.14 Tarea 5.4

Testeo de la aplicación y corrección de errores

Como se mencionó anteriormente los emuladores MHP para PC permiten testear la aplicación de manera gráfica, en cuanto a su navegabilidad e interactividad, pero deja de lado varios aspectos tales como:

- acceso al carrusel
- uso de señalización
- canal de retorno

Para comenzar es útil el uso de estos emuladores, aunque luego para un testeo más eficiente se recomienda hacer uso del ambiente real de aplicaciones interactivas (generador de carrusel de objetos, cabecera digital para transmisión de aplicaciones y terminales MHP).

En esta instancia el testeo realizado con el emulador XleTView más que nada verificó el correcto despliegue gráfico de los componentes, y la navegabilidad entre las diferentes pantallas.

Se testeó la aplicación según la siguiente tabla, que contiene el resultado esperado y el resultado obtenido ante cada secuencia de eventos. Dado que durante el desarrollo de la aplicación la misma iba siendo testeada de manera informal con el emulador, al momento de llegar al testeo exhaustivo no se encontraron errores.

Inicio: Pantalla "Cartelera de espectáculos"

Selección				Resultado esperado	Resultado obtenido	Evaluación
1	2	3	4			
"Cartelera de cines"				Pantalla "Cartelera de cines"	Pantalla "Cartelera de cines"	✓
"Cartelera de cines"	"Atrás"			Pantalla "Cartelera de espectáculos"	Pantalla "Cartelera de espectáculos"	✓
"Cartelera de cines"	"Salir"			Desaparece aplicación	Desaparece aplicación	✓
"Cartelera de cines"	"Ver Salas y Horarios"			Pantalla "Cines - Salas y Horarios"	Pantalla "Cines - Salas y Horarios"	✓
"Cartelera de cines"	"Ver Salas y Horarios"	"Atrás"		Pantalla "Cartelera de cines"	Pantalla "Cartelera de cines"	✓
"Cartelera de cines"	"Ver Salas y Horarios"	"Salir"		Desaparece aplicación	Desaparece aplicación	✓
"Cartelera de cines"	"Ver Salas y Horarios"	"Resúmenes"		Pantalla "Cines - Resumen de Películas"	Pantalla "Cines - Resumen de Películas"	✓
"Cartelera de cines"	"Ver Salas y Horarios"	"Resúmenes"	"Atrás"	Pantalla "Cartelera de cines"	Pantalla "Cartelera de cines"	✓
"Cartelera de cines"	"Ver Salas y Horarios"	"Resúmenes"	"Salir"	Desaparece aplicación	Desaparece aplicación	✓
"Cartelera de cines"	"Ver Salas y Horarios"	"Resúmenes"	"Horarios"	Pantalla "Cines - Salas y Horarios"	Pantalla "Cines - Salas y Horarios"	✓

"Cartelera de cines"	"Ver Resúmenes de películas"			Pantalla "Cines - Resumen de Películas"	Pantalla "Cines - Resumen de Películas"	✓
"Cartelera de cines"	"Ver Resúmenes de películas"	"Atrás"		Pantalla "Cartelera de cines"	Pantalla "Cartelera de cines"	✓
"Cartelera de cines"	"Ver Resúmenes de películas"	"Salir"		Desaparece aplicación	Desaparece aplicación	✓
"Cartelera de cines"	"Ver Resúmenes de películas"	"Horarios"		Pantalla "Cines - Salas y Horarios"	Pantalla "Cines - Salas y Horarios"	✓
"Cartelera de cines"	"Ver Resúmenes de películas"	"Horarios"	"Atrás"	Pantalla "Cartelera de cines"	Pantalla "Cartelera de cines"	✓
"Cartelera de cines"	"Ver Resúmenes de películas"	"Horarios"	"Salir"	Desaparece aplicación	Desaparece aplicación	✓
"Cartelera de cines"	"Ver Resúmenes de películas"	"Horarios"	"Resúmenes"	Pantalla "Cines - Resumen de Películas"	Pantalla "Cines - Resumen de Películas"	✓
"Cartelera de teatros"				Pantalla "Cartelera de teatros"	Pantalla "Cartelera de teatros"	✓
"Cartelera de teatros"	"Atrás"			Pantalla "Cartelera de espectáculos"	Pantalla "Cartelera de espectáculos"	✓
"Cartelera de teatros"	"Salir"			Desaparece aplicación	Desaparece aplicación	✓

Tabla 4.6: Casos de testeo de la aplicación Cartelera

Por otro lado también se testearon los siguientes aspectos:

- Correcto funcionamiento del jump scroll no circular.

Este aspecto refiere a cómo se realiza la navegabilidad en aquellas pantallas que habilitan el uso de las flechas “UP” y “DOWN”.

- Adecuado despliegue del texto en el área establecida.

Para aquellas pantallas en las que se debe desplegar texto en más de un renglón, la aplicación controla la cantidad de caracteres por renglón para que así no se pierda ningún carácter ni se corten las palabras. Vale la pena aclarar que esto se cumplirá siempre y cuando el proveedor de contenidos respete las características establecidas en cuanto al largo del texto introducido.

- Contraste de colores

Si bien la pantalla del PC no es igual a la del televisor, los colores utilizados son estándar, y las combinaciones adecuadas.

4.15 Tarea 6.1

Búsqueda de información acerca del canal de retorno TDT

La incorporación de un canal de retorno introduce mejoras en la experiencia de usuario, ya que permite lograr aplicaciones realmente interactivas, permitiendo al usuario enviar información al proveedor de la aplicación.

Los perfiles MHP “Interactive broadcast” e “Internet access” soportan el uso de un canal de retorno para así llevar a cabo la interactividad. Los receptores MHP fabricados bajo estos perfiles deben soportar algunos de los siguientes elementos:

- Módem PSTN
- Cable MODEM
- ADSL
- GPRS

MHP no establece de qué manera implementar el canal de retorno, sólo exige el soporte de TCP/IP y HTTP en alguna de sus versiones. Por esta misma razón es que los

servidores a los que se conecta el canal de retorno no están estandarizados, sólo deben usar TCP/IP.

Tipos de canal de retorno

Actualmente la mayoría de los terminales MHP tienen integrado un módem PSTN, aunque podrían contar con una interfase Ethernet para conexión a Internet o con un módem GSM o raramente con un canal satelital.

Los canales de retorno pueden ser de dos tipos:

- “Always-on”

Este tipo de canal hace uso de una conexión permanente, ya establecida. Ejemplos son cable-modem, xDSL o Ethernet, que finalmente derivará en una conexión IP, ya que se conectará el terminal a un switch o hub y luego a la red del ISP. Para este tipo de canales de retorno es necesario que el abonado esté subscripto a un servicio del ISP.

- “Connection based”

Estos canales de retorno son establecidos sólo cuando es requerido por una aplicación, la información es transferida y la conexión finaliza. Un ejemplo típico es la conexión dial-up. Bajo este panorama se debe tener en cuenta que para hacer uso del canal de retorno las aplicaciones deben contar con los permisos necesarios para establecer la conexión.

Internet vía MHP

MHP 1.0 no exige el soporte de ningún servicio de Internet por parte de los terminales MHP, aunque es posible desarrollar aplicaciones que utilicen el canal de retorno para enviar/recibir información del mundo IP.

MHP 1.1 determina que deben existir aplicaciones residentes para permitir el acceso a Internet, y para esto deben implementar un cliente de email y un navegador.

Seguridad

Se debe tener especial cuidado, ya que código malicioso o bugs en aplicaciones pueden transmitir información confidencial del usuario a través del canal de retorno, no sólo difundiendo información privada sino también pudiendo incurrir en grandes gastos económicos por el uso de la conexión.

Las aplicaciones podrán acceder al canal de retorno si es que un archivo de permisos se lo admite, y en este mismo archivo se especifica el número al que debe discar la aplicación para establecer el enlace. Lo más recomendado es que se le pregunte al abonado si da permisos para establecer la conexión en el momento que la aplicación lo solicita.

Las aplicaciones MHP pueden establecer sesiones TLS y verificar certificados, brindando así mayor seguridad, invisible para el usuario. Los certificados root pueden ser enviados al abonado junto con la aplicación o pueden residir en el terminal.

Implementar un canal de retorno

Para el desarrollador de aplicaciones el uso de un canal de retorno resulta sencillo mediante el uso del paquete java.net, aunque existen algunas diferencias en cuanto al manejo de sesiones.

Java.net asume que las conexiones son permanentes, del tipo “always on”, o sea que la conexión ya se encuentra establecida. Dado que esto no tiene porque ser así en el contexto MHP, éste último debió introducir una API para el manejo de estas cuestiones. Por otro lado, el manejo de sesiones también evita que las aplicaciones usen el canal de retorno de manera excesiva.

API org.dvb.net.rc

Esta API define un administrador de sesiones, y cuenta con las siguientes clases:

- RCInterface: interfase del canal de retorno
- ConnectionRCInterface: extiende la clase anterior, y es la que se utiliza en aplicaciones MHP en las que el canal de retorno no es del tipo “always-on”.

```
public class ConnectionRCInterface extends RCInterface implements
ResourceProxy {

    public boolean isConnected();

    public float getSetupTimeEstimate();

    public void reserve(ResourceClient c, Object requestData) throws
PermissionDeniedException;

    public void release();
}
```

```
public void connect()throws IOException, PermissionDeniedException;

public void disconnect()throws PermissionDeniedException;

public ConnectionParameters getCurrentTarget()throws
IncompleteTargetException;

public void setTarget(ConnectionParameters target) throws
IncompleteTargetException, PermissionDeniedException;

public void setTargetToDefault()throws PermissionDeniedException;

public int getConnectedTime();

public ResourceClient getClient();

public void addConnectionListener( ConnectionListener l);
public void removeConnectionListener( ConnectionListener l);

}
```

ConnectionParameters

El método `setTarget()` de `ConnectionRCInterface` establece los parámetros para la conexión a través de una instancia `ConnectionParameters`, en la que se declara a qué interfase se deberá establecer la conexión.

Toda interfase tiene un target por defecto al que es posible conectarse con los permisos adecuados. Por ejemplo, es posible establecer como interfase por defecto la del ISP asociado al operador de TV.

La clase `ConnectionParameters` define:

- número de teléfono a discar
- usuario
- contraseña

Claramente esta clase está enfocada en establecer conexiones mediante un módem PSTN, donde se hace necesario un manejo de sesiones. Los demás canales de retorno

no necesitan un session manager debido a que la conexión ya está en funcionamiento o el módem ADSL es quien se encargará automáticamente de establecer la conexión.

```
public class ConnectionParameters {  
  
    public ConnectionParameters(String number, String user name,  
String password);  
  
    public ConnectionParameters(String number, String user name, String  
password, InetAddress[] dns);  
  
    public String getTarget();  
    public String getUsername();  
    public String getPassword();  
    public InetAddress[] getDNSServer();  
}
```

Recurso escaso

Debido a que el canal de retorno es un recurso escaso y el acceso a éste debe ser controlado, la clase ConnectionRCInterface implementa a la ResourceProxy. Para esto se usa el RCInterfaceManager, una clase Singleton, cuya referencia se obtiene a través de RCInterfaceManager.getInstance(). Luego de obtenida la referencia se puede usar el siguiente método:

- `getInterfaces()`: devuelve todas las interfaces que la aplicación puede utilizar en un array. Cuando se utilizan conexiones a través de un módem PSTN, se le pasa como parámetro un `java.net.InetAddress`.

Diagrama de flujo – Establecimiento del canal de retorno

La figura a continuación muestra qué debe ejecutar el Xlet para establecer una conexión “Connection based”:

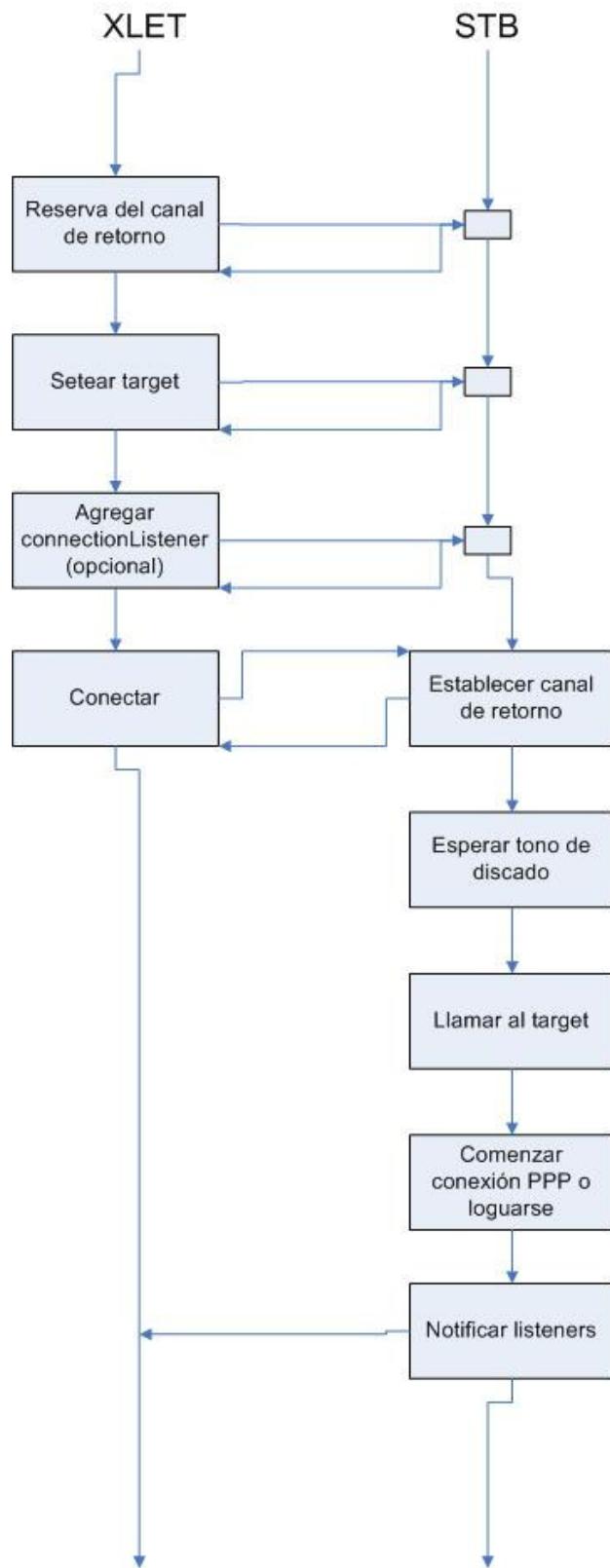


Figura 4.19: Diagrama de flujo_ establecimiento de un canal de retorno

Código de acceso a un servidor remoto

Se debe tener en cuenta que el uso de un módem demora el establecimiento de la conexión entre 30 - 45 segundos. Para evitar el colapso del Xlet debido a este tiempo es que existen 2 threads diferentes, uno del Xlet y otro del terminal MHP.

A continuación figura el código para establecer una conexión del tipo “connection based”:

```
// Se obtiene la referencia al RCInterfaceManager
RCInterfaceManager rcm = RCInterfaceManager.getInstance();

// Se obtiene la lista de interfaces disponibles para la aplicación
RCInterface[] interfaces = rcm.getInterfaces();

/* Se selecciona una interface, pero antes se verifica que es del tipo
deseado "ConnectionRCInterface" */

if (interfaces[0] instanceof ConnectionRCInterface) {
    // Si se entra aca debo conectar la interfase

    ConnectionRCInterface myInterface;
    myInterface = (ConnectionRCInterface) interfaces[0];

    // usamos la interfase
    try {
        // Reservamos la conexión
        myInterface.reserve();

        // Seteo de parámetros parameters
        ConnectionParameters myConnectionParameters;
        myConnectionParameters = new ConnectionParameters
            ("059827074461", "um", "pfc");

        // Seteamos el target de la interface y nos conectamos
        myInterface.setTarget(myConnectionParameters);
        myInterface.connect();

        // SE ENVÍA LO QUE SE DESEA A TRAVÉS DEL CANAL DE RETORNO

        // desconexión, liberacion del recurso
        myInterface.disconnect();
        myInterface.release();
    }
}
```

```
}

    catch (permissionDeniedException e) {
        // no se puede reservar la interfase
        return;
    }

}

else {
//LA CONEXIÓN ES PERMANENTE, USAR
}
```

Tips para el desarrollador

PREFIJOS

Cuando se establece un canal de retorno mediante la PSTN generalmente se necesita digitar un prefijo para así obtener tono y luego conectarse al target. El terminal MHP no detecta si se encuentra conectado al sistema telefónico ni el prefijo que debe utilizar. Para solucionar esto existen 3 estrategias, no todas ellas siempre aplicables:

- Configurar el módem del terminal MHP para que use un prefijo dado. (no siempre posible).
- El Xlet puede discar el prefijo junto con el número de teléfono. Para esto debe existir una manera de ingresar el prefijo de manera manual, y a veces soportar un prefijo nulo cuando se trata de una conexión ya establecida.

USO ASÍNCRONO

El método connect() es asíncrono, esto significa que la conexión debe ser rápida para no bloquear al Xlet. Generalmente éste demora entre 30 y 45 segundos, por lo que se debe tener cuidado de no llamarlo dentro del thread de procesamiento de operaciones del usuario.

SEGURIDAD

Algunos receptores MHP cuentan con lo que se denomina security manager, y muchas veces éstos restringen el establecimiento del canal de retorno.

Existen 2 opciones para solucionar este tipo de problemas:

- Lo más adecuado es enviar en el carrusel un archivo de permisos en el que se habilite a la aplicación a acceder a un canal de retorno, pero esto muchas veces resulta complejo.

Por ejemplo:

Incluir en el carrusel el siguiente archivo con el nombre /org/mhpkdb/returnchannel/manual/dvb.org.mhpkdb.returnchannel.manual.Example06_03ManualConnection.perm.

```
<?xml version="1.0"?>
<!DOCTYPE permissionrequestfile
PUBLIC "-//DVB//DTD Permission Request File 1.0//EN"
"http://www.dvb.org/mhp/dtd/permissionrequestfile-1-0.dtd">
<permissionrequestfile orgid="0x00000000 COMPLETAR CON DATOS DE AIT"
appid="0x0000 COMPLETAR CON DATOS DE AIT">
<returnchannel>
<phonenumber></phonenumber>
</returnchannel>
</permissionrequestfile>
```

- Por otro lado es posible deshabilitar el security manager. Esto resulta necesario cuando el terminal MHP no responde a los archivos de permisos. Ejecutar System.setSecurityManager(null).

Uso de un canal de retorno con el fin de recabar datos a través de una encuesta

Este tipo de aplicaciones son generalmente de interactividad media, por lo que basta con terminales fabricados bajo el perfil MHP “Interactive Broadcast”, envían la información y no esperan respuesta. Igualmente podría ser necesario contar con el “Internet access profile” si por ejemplo se deseara registrar el voto mediante una petición http en una página Web.

Para ir registrando los votos generalmente se hace uso del método “drop call”. Éste consiste en discar un determinado número telefónico dependiendo de la opción votada, el servidor detectará la llamada, registrará un voto, y él mismo terminará la conexión.

Para este tipo de aplicaciones MHP 1.1 define subclases como:

- ConnectionDroppedEvent: Indica a la aplicación que el destino ha recibido la llamada y ha finalizado la misma. Esto es muy útil cuando se implementan aplicaciones del tipo encuesta “drop call”, ya que permite comprobar que el voto ha sido registrado.

Para versiones anteriores de MHP todos los eventos se engloban en ConnectionFailedEvent, lo que significa que la conexión ha sido terminada, pero no indica exactamente la causa del fallo. En el caso de aplicaciones que hacen uso de “drop call” este evento será generado tanto cuando el voto es registrado como cuando el destino se encuentra ocupado, o por alguna otra razón no fue posible registrar el voto.

4.16 Tarea 6.2:

Análisis de la posibilidad de introducir un canal de retorno en aplicaciones interactivas para TDT con el objetivo de realizar encuestas.

La tarea en cuestión tiene como objetivo generar una aplicación en la que se haga uso de un canal de retorno dentro del contexto de la TDT. En estas condiciones el contenido que llega a cada usuario no es personalizable, y por lo tanto no es adecuado desplegar información al usuario en base a la información enviada a través del canal de retorno. A continuación se presentan los detalles sobre la implementación de la aplicación.

Información general

La aplicación en sí misma consiste en una encuesta que permitirá conocer la opinión de los televidentes en el área deseada a través de la parametrización de una pregunta clave y diferentes opciones de respuesta. La respuesta será enviada vía canal de retorno a un servidor final encargado de procesar los datos. Dado que se está trabajando en el contexto de la TDT, y que la mayoría de los terminales MHP cuentan con un módem PSTN, el mecanismo para registrar el voto del usuario es el denominado “drop call”. Esto significa que dependiendo de la respuesta del usuario, el canal de retorno establecerá una llamada con un número correspondiente a través de la red de telefonía fija.

Interfase Gráfica

La interfase gráfica está compuesta simplemente por tres pantallas, una donde se presenta la pregunta con sus diferentes opciones y las otras dos se utilizan para

desplegar el resultado del voto. En caso de que se haya podido registrar el voto se desplegará una pantalla que informe al usuario la acción realizada y en caso de que no se haya podido establecer una conexión, imposibilitando el registro del voto, se desplegará una pantalla de error.

Dichas pantallas se muestran a continuación. Las mismas fueron capturadas a través del emulador XleTView.



Figura 4.20: Pantalla “Encuesta”



Figura 4.21: Pantalla “Encuesta Completada”



Figura 4.22: Pantalla “Error de conexión”

Para navegar entre las pantallas se utilizan los siguientes botones:

- Flechas “UP” y “DOWN”: permiten navegar entre las respuestas de la encuesta.
- Botón Verde: Aparece únicamente en la primer pantalla y tiene como función enviar el voto seleccionado.
- Botón Rojo: Tiene como función abortar la aplicación.

El resto de los botones brindados por el control remoto carecen de funcionalidad y la acción de cualquiera de ellos pasará desapercibida.

Guía para el proveedor de contenidos - Información sobre archivos de contenido

La aplicación deberá ser parametrizada por los siguientes atributos:

- Pregunta de la encuesta
- Opciones de respuesta
- Imagen o logo a desplegar
- Números telefónicos a los que se llamará dependiendo de la respuesta escogida por el usuario.

La información anteriormente mencionada será registrada en el archivo de contenido denominado Encuesta.txt.

Encuesta.txt

Este archivo contiene toda la información asociada a la encuesta. A continuación se visualiza un ejemplo del archivo:

```
Cuál es su postre favorito?;;;Fruta::Helados::Tortas ;;;/images/mammut.png;;;;nro1::nro2::nro3
```

Separador: Cada sección de la encuesta se encuentra separada por el símbolo “; ; ;”. Cuando se trata de varias respuestas o número telefónicos, los mismos se separan por “; ; ;”.

Tamaño del contenido: Para que las diferentes respuestas sean desplegadas por completo, éstas no deberán contener más de 70 caracteres. Por otro lado es posible desplegar hasta 8 opciones de respuesta. Vale la pena aclarar que se debe introducir un número telefónico por cada opción de respuesta existente.

El path corresponde a la ruta (relativa a la aplicación) de la imagen del ícono que se quiere desplegar.

Guía para el desarrollador

Estructura de Directorio

La estructura de directorios del proyecto “Encuesta” es la siguiente:

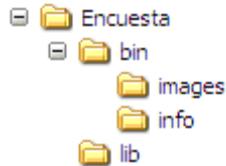


Figura 4.23: Estructura de directorios de la aplicación Encuesta

Dentro del directorio “bin” se encuentran las siguientes clases:

- Encuesta
- DatosEncuesta

La carpeta “images” contiene los siguientes archivos:

- greenspot.png
- mammut.png
- redspot.png

Cabe destacar que el archivo mammut.png se corresponde con el ícono a desplegar en la aplicación, por lo que la ruta que se coloca en el archivo Encuesta.txt debe ser coherente con el nombre del archivo que se encuentra bajo este directorio.

El archivo de contenido se encuentra en la carpeta “info”:

- Encuesta.txt

Finalmente la carpeta “lib” contiene el .jar necesario para la compilación, cuyo nombre es:

- mhbstubs.jar

Descripción de las clases

Encuesta.java

La clase “Encuesta” es la clase principal de la aplicación y es en donde está implementado el Xlet en sí mismo. Por esta razón se incluyen los métodos : initXlet(), startXlet(), pauseXlet() y destroyXlet().

Todas las acciones de inicialización se hacen dentro del método initXlet(). Este último método se encarga principalmente de crear un HScene que actuará como “pizarra” para ir dibujando las diferentes pantallas.

Principales campos

```
private HScene scene;
```

El HScene es la “pizarra” sobre la que se dibujarán las diferentes pantallas.

```
private HContainer contX;
```

Existe un atributo de estos por cada pantalla. El HContainer es sobre el cual se ubican los diferentes componentes gráficos. Luego, es el mismo HContainer el que se ubica sobre el HScene mediante el comando HScene.add(HContainer). Para saber si un HContainer está visible se utiliza el método HContainer.isShowing().

```
private Image x;
```

Este atributo es utilizado para cargar las imágenes necesarias desde el carrusel de objetos.

```
private HTextButton x;
```

El HTextButton es uno de los tantos componentes gráficos que pueden ser agregados a un HContainer. Este componente es un botón al que se le puede parametrizar el texto que presenta.

```
private HGraphicButton x;
```

El HGraphicButton es como el botón anteriormente detallado pero sólo presenta una imagen.

```
private HStaticText x;
```

El atributo en cuestión representa un cuadro de texto.

En el caso de esta aplicación algunos cuadros de texto fueron utilizados para introducir las opciones de respuesta a la encuesta provistas por el archivo de contenido. El control del texto, para que éste no sobrepase los límites del cuadro, se hace a través de métodos implementados en la clase Auxiliar. Sin embargo, si los archivos de contenido exceden la

cantidad de caracteres especificada, el texto no aparecerá por completo en el cuadro. También se puede utilizar el DVBTTextLayoutManager, pero igualmente se debería contar con los métodos de control, y la única diferencia radicaría en que al usar el manager se introduciría “(...)” si no es posible desplegar el texto completo.

```
private DSMCCObject x;
```

En estos atributos se cargan los objetos obtenidos del carrusel.

Principales métodos

```
public void initXlet(XletContext ctx) throws XletStateException;
```

El método initXlet() pone a la aplicación de forma no visible para el usuario mientras solicita los recursos necesarios para la ejecución de la misma. En particular se crea el HScene de la aplicación a través de la clase org.havi.ui.HSceneFactory y se establecen todos los parámetros de la misma. Es importante habilitar el KeyListener del HScene para que puedan capturarse los eventos generados por el control remoto. Además se crea el contenedor principal en donde se agregan posteriormente el resto de los componentes contenidos en la pantalla principal.

```
public void startXlet() throws XletStateException;
```

El método startXlet() se encarga de hacer visible la primer pantalla de la aplicación, el menú inicial. Para esto debe invocar el método validate(), poner el HScene visible al usuario y solicitar el “focus” del componente que se quiera que aparezca seleccionado por defecto.

```
public void pauseXlet();
```

El método pauseXlet() será invocado cuando el receptor necesite pausar la aplicación sin liberar por completo los recursos que la misma utiliza. Para esto simplemente hace que el HScene no se visualice, por lo que todos los componentes de dicho HScene tampoco se visualizarán.

```
public void destroyXlet(boolean b);
```

destroyXlet() termina la aplicación y por lo tanto elimina el HScene liberando los recursos que la misma consume.

```
public void paint(Graphics g);
```

El método paint() se encarga de actualizar los gráficos de la aplicación, y es llamado de manera automática cuando el HScene necesita ser redibujado. Debe considerarse que aquellas acciones realizadas dentro del paint serán comunes a todas las pantallas.

```
private void cargarBotones(Graphics g);
```

Este método es invocado por paint(Graphics g) y se encarga de cargar en pantalla los

botones que son comunes a todas ellas. En este caso el botón rojo, cuya acción es terminar con la aplicación.

```
public void focusGained(FocusEvent e);
```

Este método se encarga de colorear los componentes cuando se encuentran seleccionados. En este caso se colorean las posibles respuestas, en color gris en caso de estar seleccionadas y de color blanco cuando no se encuentran seleccionadas.

```
public void conectar(int nroOpcion);
```

Este método se encarga de establecer el canal de retorno mediante un módem PSTN que dependiendo de la respuesta a la encuesta generará una llamada a través de la red telefónica. Para ello se crea una instancia de la clase RCInterfaceManager y a partir de dicha instancia se obtiene el array de interfaces que se encuentran disponibles para la aplicación. Luego se setean los parámetros característicos de la conexión: el número de teléfono, usuario y contraseña. A continuación se setea el target y se realiza la conexión mediante el método connect(). Finalmente se invoca al método disconnect() para liberar los recursos que están siendo utilizados.

Vale la pena aclarar que en este caso no se ejecuta el método connect() en un thread diferente debido a que mientras se realiza la conexión la aplicación sólo debe esperar al establecimiento del canal de retorno.

```
public void keyPressed(KeyEvent key);
```

Para manejar la captura de eventos generados por acción de los botones del control remoto se utiliza el método keyPressed(KeyEvent key). Este método a partir del evento capturado y la situación actual de la aplicación decide qué hacer en cada caso. Por ejemplo, cada vez que se accione el botón rojo se debe invocar el método destroyXlet(), sin importar en qué pantalla se encuentre la aplicación. Para el resto de los botones se debe tener en cuenta desde qué pantalla fueron invocados. Para ello, dado que cada pantalla está representada por un contenedor (org.havi.ui.HContainer), simplemente se debe consultar si dicho contenedor se encuentra visible mediante el método isShowing (ej : contPpal.isShowing()). El método keyPressed captura KeyEvents que pueden ser referenciados por su nombre o por su número; existen algunos cuyos respectivos números se encuentran estandarizados como por ejemplo los botones de colores del control remoto, en ese caso conviene usar su número, en caso contrario se los referencia mediante su nombre ej: KeyEvent.VK_DOWN.

```
public void handleEvent(Event arg0);
```

El método en cuestión permite capturar diferentes tipos de eventos. En este caso se contemplan únicamente los eventos “ConnectionFailedEvent” (voto no registrado) y “ConnectionDroppedEvent” (voto registrado con éxito), para así distinguir si el voto fue registrado correctamente o no.

```
public void cargarObjetos();
```

Este método se encarga de cargar los objetos que se extraen del carrusel de objetos. En el caso de la implementación del acceso al carrusel de objetos se utilizó como base el

código que se encuentra en marco teórico en la sección “2.3.12.4 Ventajas de la carga asíncrona”. Cabe aclarar que se realizaron algunas modificaciones porque dicho código no era compilable tal cual como aparece en dicha sección. Para ser más específicos cuando se instancia el locator a utilizar por la aplicación en vez de utilizar el org.davic.net.Locator se utilizó el org.davic.net.dvb.DvbLocator. El locator es obtenido del Service context; también podría ser ingresado por la emisora en el código fuente.

DatosEncuesta.java

Es una clase que tiene como objetivo cargar los datos de la encuesta como su nombre indica. Para esto hace uso del archivo obtenido desde el carrusel de objetos “Encuesta.txt”.

Principales campos

`private String i;`

Representa el path donde se encuentra el ícono.

`private String p;`

Representa la pregunta asociada a la encuesta.

`private String op [];`

Array donde se almacenarán las diferentes opciones de respuesta a la pregunta.

`private String tels[];`

Array donde se guardan los números de teléfono a los que se enviará la respuesta.

`private int co;`

Representa la cantidad de respuestas existentes a la pregunta.

Principales métodos

Cada atributo tiene definido su getter y setter para poder acceder a los datos y setearlos.

Compilación

Si bien es posible utilizar herramientas de desarrollo específicas MHP que ya contengan las clases contra las que se debe compilar la aplicación, también es posible utilizar una herramienta de desarrollo Java genérica. En este último caso para compilar las

aplicaciones MHP se necesita un set de clases que lo permitan. Existen implementaciones comerciales de MHP que se obtienen en un JAR, e implementaciones libres y gratuitas MHP, aunque éstas últimas en la mayoría de los casos no son implementaciones completas.

En este caso, para compilar la aplicación, se utilizó la herramienta Java Eclipse, se instaló el J2SE Software Development Kit y se agregaron al proyecto los jars necesarios para implementar MHP. Los archivos .jar son implementaciones de las clases que se definen en MHP y por lo tanto es imprescindible que el proyecto cuente con las mismas. Se probó el funcionamiento de la aplicación con archivos .jar extraídos de los siguientes sitios web.

1. TV without Borders: Se descargaron las stub classes provistas por esta página web (http://www.interactivetvweb.org/resources/code_samples) y luego se agregaron al proyecto. Cabe destacar que este jar incluye los paquetes javax.* y org.* debido a que no es una implementación completa del estándar MHP.
2. Code 4TV: Se obtuvieron los jars publicados por la página en cuestión (<http://www.code4tv.com/c/downloads>). En este caso la implementación de las clases se encuentran en archivos distintos por lo que se agregaron al proyecto los archivos javatv10.jar y havi_dvb_davic112-1.1.jar que se encuentran dentro del rar code4tv-MHP-1.1.2-stubs-v1.1.esp provisto por la página.

Los resultados obtenidos con ambas implementaciones fueron óptimos.

En el Anexo 6 se adjunta el código correspondiente a la aplicación “Encuesta”.

Testeo de la aplicación

Como se mencionó anteriormente los emuladores MHP para PC permiten testear la aplicación de manera gráfica, y en este caso se utilizó principalmente para verificar la correcta navegabilidad de la aplicación.

La aplicación en cuestión consiste únicamente en 3 pantallas, por lo que su navegabilidad es fácilmente verificable. Igualmente, durante el desarrollo de la aplicación la misma iba siendo testeada de manera informal con el emulador.

Cabe destacar que aquellas aplicaciones que hacen uso de un canal de retorno deben ser testeadas sin lugar a dudas en el contexto real, ya que es la única forma de verificar que el canal de retorno ha sido correctamente establecido.

4.17 Tarea 7.1 y 7.2

Estudio del funcionamiento de un “carrusel de objetos”

Establecimiento de los parámetros a tener en cuenta por parte de la emisora para un correcto desempeño de la aplicación ya desarrollada

Las tareas en cuestión tienen como objetivo brindar los lineamientos básicos para la emisión de las aplicaciones MHP “Cartelera de Espectáculos” y “Encuesta”, considerando conceptualmente el funcionamiento del carrusel de objetos, sus parámetros, y el software para generarlos.

Tener un conocimiento básico del carrusel de objetos DSM-CC es importante para crear aplicaciones que funcionen adecuadamente y se carguen de manera rápida. Una de las especificaciones más fáciles de abordar es la ETSI TR 101 202 DVB - Implementation Guidelines for Data Broadcasting. A continuación se detallará el funcionamiento del DSM-CC desde el punto de vista DVB.

DSM-CC y el carrusel MHP

El Digital Storage Media – Command and Control (DSM-CC) soporta la transmisión de datos, marcas de tiempo y archivos sobre MPEG-2. DSM-CC originalmente fue generado para trabajar en redes comunes, y no de broadcasting. Por lo que basa su funcionamiento en el hecho de que los objetos a transmitir están en un nodo de la red, y cualquiera que desee manipularlos debe hacer una petición a ese nodo. Hoy en día es aplicable a los sistemas de TV digital en donde la información se envía desde un transmisor a un receptor imposibilitado de hacer peticiones, sólo recibe información y utiliza lo que desea. Bajo este contexto, la emisora transmite periódicamente los mismos archivos, y el receptor debe esperar a que llegue el deseado. Por este modo de funcionamiento es que el DSM-CC para TV digital se denomina carrusel.

En la figura 4.24, se representa al carrusel de objetos como un círculo, formado por archivos, que gira en el sentido indicado por la flecha azul. El giro se asocia directamente con la emisión de archivos por parte de la emisora, que es cíclico y continuo. El ojo que observa el círculo es el terminal MHP, que debe esperar a que el archivo deseado “aparezca ante sus ojos”.

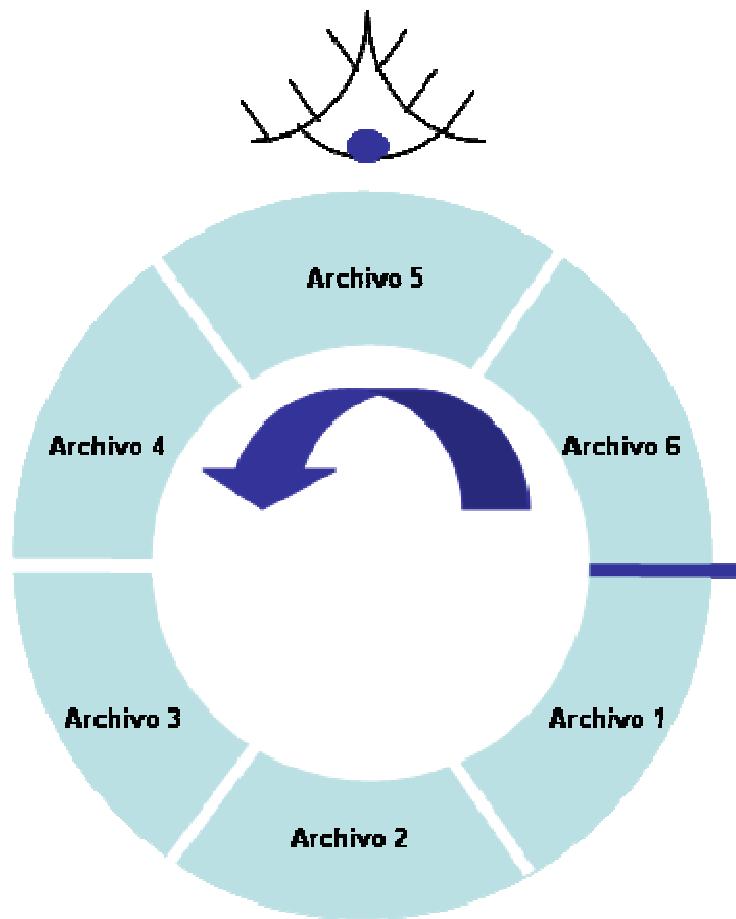


Figura 4.24: Representación lógica del carrusel de objetos

La optimización del carrusel de objetos MHP implica una reducción en el tiempo de espera del usuario, estrechamente asociada al tiempo de carga de los archivos de la aplicación.

Tipos de carrusel

De datos:

Provee a la emisora una manera de enviar información al receptor, sin especificar qué información es, y por lo tanto es el receptor quien debe analizar qué tipo de información está recibiendo para así obtener lo que desea.

De objetos:

Esta es una mejor solución. Se basa en el carrusel de datos para generar una solución estándar más parecida a un filesystem. DVB la utiliza en modo carrusel (circular) y los objetos más comúnmente utilizados en el contexto MHP son:

- Directorios: listado de nombres de archivos, directorios y punteros.
- Service gateway: directorio especial que contiene al directorio raíz.
- Archivos: Son los archivos en sí mismos con la información real que contienen.
- Stream events: Utilizados para stream events, que se explican más adelante.

Carrusel de objetos: Estructura

El carrusel de objetos consiste en un árbol de directorios que se divide en módulos, cada uno contenido uno o más archivos o sub-directorios. El tamaño máximo de un módulo es de 64 KBytes, y un archivo no puede ser repartido en varios módulos. Esto significa que si se tiene un módulo en el que restan “x” KBytes libres, un archivo de “y” KBytes ,con “y”>“x”, tendrá que ser introducido en otro módulo. Por otro lado, si un archivo en sí mismo ocupa más de 64 KBytes entonces deberá tener asociado un módulo exclusivamente para él. Vale la pena aclarar que no existe relación directa entre directorios y módulos, y entonces los archivos de un mismo módulo no tienen por qué pertenecer al mismo directorio.

Ejemplo:

Supóngase la siguiente estructura de directorios:

index.html	1,2 Kbytes
image1.jpg	3,9 Kbytes
image2.jpg	117,5 Kbytes

audio	<directory>
audio/clip1.aiff	25,8 Kbytes
classes	<directory>
classes/Main.class	22,5 Kbytes
classes/Big.class	58,6 Kbytes
classes/Other.class	26,3 bytes

A partir de ésta se componen los siguientes módulos:

Módulo 1: index.html + image1.jpg. (En este módulo ya no entra la image2.jpg) + audio

Módulo 2: image2.jpg (Este archivo ocupa en sí mismo más de 64 KBytes, por lo que un módulo entero debe serle asignado)

Módulo 3: classes + Main.class + Other.class (La clase Big no entra en este módulo)

Módulo 4: Big.class

La información es transmitida por la emisora en módulos, y los módulos son transmitidos uno a continuación de otro. Seguramente existen diversas formas de ordenar los archivos en módulos, pero una sola minimiza el tiempo de espera en el receptor. Para lograr este tiempo es necesario conocer la aplicación en profundidad y saber en qué orden se cargarán los archivos.

Ahora se deben ubicar los módulos en el carrusel. Los mismos podrán ser repetidos si se desea poder acceder a ellos más frecuentemente, pero se debe tener en cuenta que esto agrandaría el tamaño del carrusel y entonces el tiempo de espera aumentaría para el resto de los archivos. Se debe lograr un equilibrio.

Normal play time

Adicionalmente, el DSM-CC permite también transportar marcas de tiempo (timecode o Normal Play Time) que guían al receptor en cuanto a qué información se está desplegando. Si bien MPEG lleva un PCR (Programme Clock Reference), éste actúa como timecode para el flujo MPEG y no resulta útil para TVi. Igualmente, es posible conocer la relación entre el NPT y el clock MPEG-2 a través de los denominados “ticks”. Éstos expresan cuantos tiempos del reloj MPEG-2 equivalen a un NPT.

La peculiaridad del NPT es que puede comenzar en cualquier número y no necesariamente es una secuencia continua. Por lo tanto, si una trama es editada, ésta mantendrá su NPT, entonces sin importar donde aparezca nuevamente, ésta podrá ser identificada por su NPT. Adicionalmente, el NPT permite repetir el mismo número en una misma secuencia y aún así representar diferentes versiones de una misma trama. Para identificar esto el NPT tiene 2 partes:

- NPT (en sí mismo)
- Content ID: permite separar bloques de contenido que comparten un mismo NPT. Un cambio en este identificador permite saber que la trama ha sido editada

Stream Events

El DSM-CC también permite la sincronización entre diferentes flujos de datos. Para esto hace uso de descriptors, y permite al receptor identificar ciertos puntos del flujo sin tener que monitorear el NPT. La emisora en lugar de indicar que cierto programa de TV comienza en los NPT “X” e “Y”, envía un “stream event”, y éste es almacenado en el carrusel tal como cualquier objeto.

Es importante saber que existen dos elementos claves para el manejo de stream events:

- Stream event object: es el objeto en sí mismo, que de manera genérica define diferentes tipos de stream events. Se identifica por un número “event ID” y un nombre explicativo.
- Stream event descriptor: es similar a una instancia de un Stream event object, y describe el stream event específicamente. Éste se identifica a través de un ID, y el NPT en el cual el evento debe ser disparado. Es recomendable enviar el stream event más de una vez para asegurarse de que el receptor lo detecte, por lo menos 5 veces a una frecuencia de 1vez/seg antes de que el evento tenga que ser disparado.

El receptor entonces al recibir un descriptor:

1. chequea que el stream object con el ID correspondiente esté en el carrusel.
2. verifica si es un evento “do it now”, en ese caso lo ejecuta inmediatamente.
3. Si no es un evento “do it now”, se obtiene el NPT del evento. Si existe un evento registrado con el mismo ID y mismo NPT, el actual se descarta. También se descarta si el NPT ya pasó.
4. Al alcanzar el NPT se dispara el evento.

Optimización del carrusel de objetos

Si se le denomina tiempo de arranque al tiempo que demora el usuario en percibir que la aplicación se está ejecutando, estudios realizados por tComLabs (expuestos en el documento “MHP Object Carousel Optimisation” publicado en la página web MHP Knowledge DataBase) demuestran que:

- Generalmente, el tiempo de arranque de la aplicación es directamente proporcional al **ciclo del carrusel**. Esto ocurre con la mayoría de los STBs existentes debido a que los mismos cachean todo el carrusel, por lo que la aplicación no demorará más de un ciclo en cargarse, tiempo en el que el carrusel será completamente cacheado.

Se debe tener en cuenta que si el carrusel es muy grande, éste quizás no pueda ser cacheado por completo en el terminal MHP.

- El tiempo de arranque, sin importar el ciclo del carrusel, siempre demorará al menos “x” segundos dependiendo del terminal MHP. Los “x” segundos se corresponden con el tiempo que demora el STB en procesar archivos, lo que se relaciona directamente con el **poder de la CPU** y no con el carrusel.
- El tiempo de arranque de la aplicación puede ser reducido si el **ancho de banda** (bit rate) del carrusel es aumentado.
- El tiempo de arranque puede ser disminuido si se agrupan los archivos de manera óptima en módulos. Una forma efectiva de **agrupación en módulos** consiste en ubicar los archivos necesarios para comenzar la aplicación en los primeros módulos del carrusel.

Por ejemplo, observando la figura 4.25, si los archivos principales de la aplicación ocupan sólo el 40% del carrusel (desde los 0º a los 144º), existen tres situaciones posibles:

Sea:

“i” el instante de ejecución de la aplicación, momento en que se empiezan a buscar los archivos en el carrusel.

“c” el tiempo de carga de los archivos

$$\text{Si } i = 0^\circ \rightarrow c = 40\%^* \text{ (ciclo del carrusel)}$$

$$\text{Si } 0^\circ < i \leq 144^\circ \rightarrow c = (\text{ciclo del carrusel})$$

$$\text{Si } 144^\circ < i < 360^\circ \rightarrow c \leq 60\%^* \text{ (ciclo del carrusel)}$$

Por lo tanto, agrupando los archivos, el terminal MHP logrará cachear los módulos necesarios para comenzar la aplicación en un tiempo promedio menor a un ciclo del carrusel.

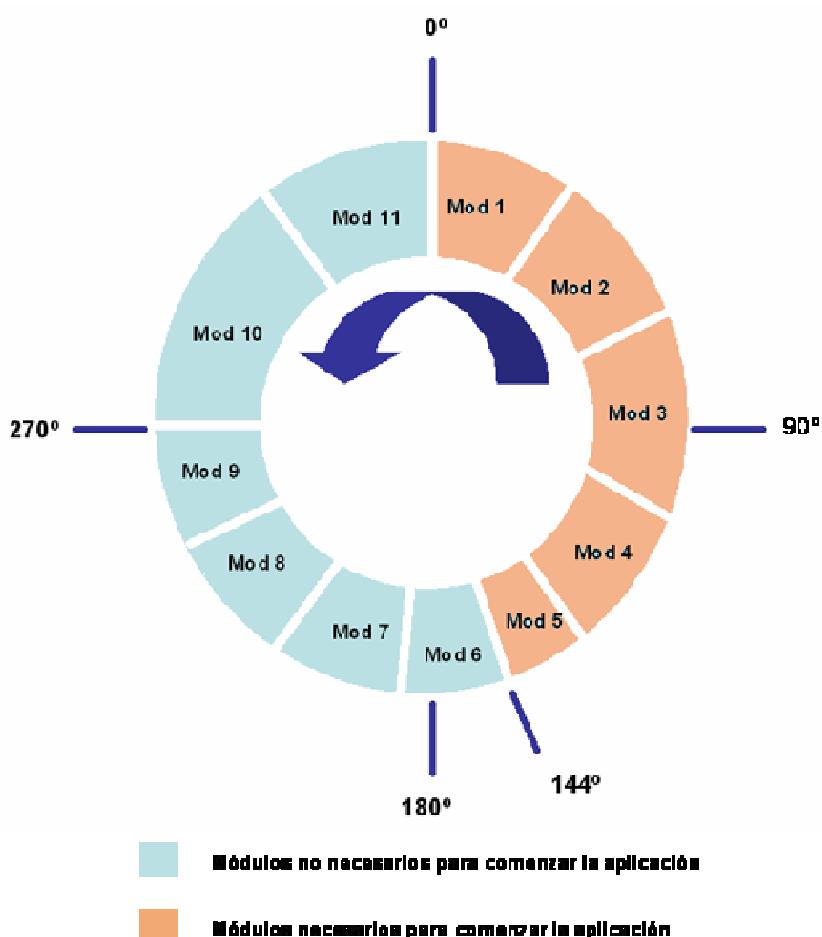


Figura 4.25: Optimización del carrusel por agrupación en módulos

- El tiempo de arranque puede ser reducido si los módulos claves para la aplicación son emitidos de manera **repetida** en un ciclo de carrusel.

DSM-CC respecto a la aplicación desarrollada “Cartelera de Espectáculos”

Las aplicaciones desarrolladas no depende de ningún evento del sistema (programa, horario, etc.), y debido a que brindan información de interés general deben estar disponibles la mayor cantidad de tiempo posible.

Dado que no es posible conocer de manera certera si el terminal cacheará o almacenará toda la aplicación o parte de la misma, la emisora deberá enviar la aplicación y sus archivos de manera periódica. Si se considera que el tiempo en que demora en enviar el carrusel un mismo archivo equivale al tiempo de descarga de la aplicación, entonces se puede definir la frecuencia de envío del archivo. A los ojos de un usuario que desea ver la “Cartelera de Espectáculos”, la aplicación no puede demorar más de 5 segundos en cargarse, por lo tanto si la emisora tiene un carrusel de ancho de banda “A” (KBytes/seg), entonces la aplicación deberá ser ubicada cada “T” (KBytes) a lo largo del carrusel, siendo $T \leq 5 * A$. Se puede realizar un razonamiento análogo para la aplicación “Encuesta”.

Por otro lado dado que la aplicación consiste en pocas pantallas, y el usuario puede solicitar rápidamente ir de una a otra, es necesario cargarlas “todas juntas”, por lo tanto sería adecuado introducir todos los archivos de contenido y clases de la aplicación en el mismo módulo, y si es posible repetir el módulo a lo largo del ciclo de vida del carrusel.

En conclusión, para optimizar el uso del carrusel de objetos se deben tener en cuenta los siguientes parámetros:

- ancho de banda del carrusel
- tamaño del carrusel
- Localización de los módulos y agrupación de archivos en los mismos
- Repetición de módulos claves en el ciclo del carrusel

Respecto a la agrupación de archivos en módulos, dado el tamaño de los archivos del proyecto “Cartelera” y del proyecto “Encuesta”, se recomienda la siguiente distribución:

 cartelera	39,9 KBytes	MÓDULO 1
 excepcion	0,5 bytes	MÓDULO 1
 images	32,8 KBytes	MÓDULO 2
 info	3,29 KBytes	MÓDULO 1

Figura 4.26: Agrupación en módulos del proyecto “Cartelera”

Estando los archivos y directorios anteriormente especificados dentro de la carpeta “ProyectoCartelera”.

 Encuesta	12,5 KBytes	MÓDULO 2
 DatosEncuesta	1,34 KBytes	MÓDULO 2
 info	0,09 KBytes	MÓDULO 2
 images	16,9 KBytes	MÓDULO 2

Figura 4.27: Agrupación en módulos del proyecto “Encuesta”

Los archivos y directorios que figuran arriba se encuentran dentro de la carpeta “ProyectoEncuesta”.

Por otro lado, dado que la emisora generará el carrusel con un software, y el mismo también emitirá la tabla de señalización AIT, vale la pena aclarar el seteo de los siguientes parámetros de la AIT para la aplicación “Cartelera”:

```
#Nombre de la aplicación y el lenguaje
name esp "Cartelera de Espectáculos"

#Flag de control: 1=autostart 2=present 3=destroy 4=kill
control 2

#Directorio base de la aplicación (a modo de ejemplo)
#Debe ingresarse el path bajo el que se coloca la carpeta “ProyectoCartelera”
basedir "C:/ProyectoCartelera"

#Classpath extension
```

```
#Debe ingresarse la ruta desde el directorio base hasta el
# paquete de la clase principal
classpath "/"

#Clase inicial
class "cartelera.Inicio"
```

Para la aplicación “Encuesta” los parámetros serían:

```
#Nombre de la aplicación y el lenguaje
name esp "Encuesta"

#Flag de control: 1=autostart 2=present 3=destroy 4=kill
control 2

#Directorio base de la aplicación (a modo de ejemplo)
#Debe ingresarse el path bajo el que se coloca la carpeta “ProyectoEncuesta”
basedir "C:/ProyectoEncuesta/"

#Classpath extension
classpath "/"

#Clase inicial
class "Encuesta"
```

El directorio base y el classpath seteados anteriormente deben permitir localizar los archivos de la aplicación. Estos parámetros no pudieron ser verificados, pero conceptualmente son los apropiados.

En el Anexo 10 se exponen algunos productos adecuados para la generación del carrusel de objetos y señalización AIT.

4.18 Tarea 8.1

Contactar autoridades pertinentes para negociar una posible prueba de la aplicación cuando ésta es transmitida conjuntamente con el flujo de video

La presente tarea experimentó algunas dificultades debido a que requería de una gran capacidad de negociación con las autoridades pertinentes para lograr el testeo de la aplicación en el contexto real de la TV digital. Se evaluó la posibilidad de realizar dicho testeo en alguna de las emisoras uruguayas de TV abierta.

En el contexto MHP existen tareas que son claramente responsabilidad de la emisora, como por ejemplo la señalización de la aplicación, multiplexación de la misma en la trama de audio y video ya existentes, manejo del carrusel de objetos, entre otras. Sin embargo, las emisoras en el Uruguay todavía no tienen un conocimiento firme desarrollado acerca de estos puntos. Dado que el proyecto en cuestión tiene como principal objetivo el desarrollo de una aplicación MHP, y no profundiza en los aspectos a tener en cuenta por las emisoras se estudió qué roles tomaría cada una de las partes durante el testeo de la aplicación. Las operadoras de TV, Canal 10 y Canal 5, expresaron que actualmente cuenta con una cabecera de transmisión digital pero sin soporte MHP, por lo que el proyecto TvDTI debería acondicionarla. Dado que el cronograma del proyecto no permitía abordar estas tareas, y no se contaba con los recursos de SW necesarios, el testeo no fue realizado.

El 18 de diciembre de 2008, se realizó la primera reunión de trabajo en el marco del proyecto Sala+ enfocado en la “TV DIGITAL: Visión, Perspectivas Y Oportunidades”. La misma tuvo como principal objetivo presentar el proyecto Sala+, y discutir cuáles son los principales actores vinculados a la TvD. (En el Anexo 9 se adjunta el acta de la reunión).

Sala+ se define como “*Una nueva iniciativa respaldada por la Comisión Europea, que apunta a apoyar el desarrollo de cooperación I+D entre Europa y Latinoamérica en el sector de las Tecnologías Audiovisuales en Red (Networked Electronic Media, NEM)*”. El estudio de los actores fundamentales para el desarrollo de la TvD realizado en la reunión tuvo como principal objetivo entender cuál es la posición de Uruguay frente a esta nueva tecnología, los avances existentes, y cómo se debe trabajar de manera conjunta para desarrollar la TvD mediante la aplicación a proyectos respaldados por SALA+.

En la reunión en cuestión se definió el mapa de actores necesarios para el desarrollo de la TvD, y todos ellos concordaron en que antes de aplicar a algún proyecto en el marco de Sala+ o tomar alguna decisión, es necesario establecer las condiciones necesarias para instalar adecuadamente la tecnología digital en nuestro país. Con esto se hace referencia principalmente a que es necesaria la toma de decisiones políticas y regulatorias para saber en qué terreno se está maniobrando y luego recién comenzar a ejecutar. Por ejemplo, uno de los aspectos criticados refiere a que no existe declaración del gobierno respecto a la cantidad de operadores que podrán existir para la TvD, ni cómo se asignarán las frecuencias. Los participantes de la reunión hicieron principal incapié en que introducir la TvD no debe consistir simplemente en trasladar “lo viejo” a digital, sino que debe aprovecharse para introducir nuevos contenidos que sean atractivos para el usuario y que le permitan a él mismo disfrutar del cambio tecnológico.

Bajo ninguna circunstancia debe entrarse al ámbito digital de manera obligada, “empujados” por el apagón analógico.

Para interés del proyecto TvDTi se pudo recabar información acerca de los posibles ambientes de testeo existentes. A continuación se detalla la institución o empresa en cuestión, la persona de contacto y el estado de las instalaciones de Televisión digital:

- Televisión Nacional Uruguay (Canal 5)

Contacto: Pablo Barbaletta. Director técnico

TVNU actualmente cuenta con una cabecera de televisión digital que está siendo utilizada para pruebas en un canal asignado de manera temporal por la URSEC. Sin embargo, esta cabecera aún no cuenta con el soporte de MHP para interactividad. Próximamente, una empresa española se encargará de introducir interactividad en el equipamiento existente, a través de la instalación de un software generador de carrusel de objetos que también puede ser conocido como servidor de aplicaciones.

- Multicanal

Contacto: Fabio Baudo. Gerente técnico.

Multicanal brinda a sus clientes servicios digitales, sin embargo no cuenta con middleware MHP para el soporte de interactividad. La introducción de interactividad en su sistema implicaría una gran inversión no sólo del lado de la emisora sino también del cliente que deberá cambiar sus decodificadores digitales.

- Agencia Nacional de Investigación e Innovación (ANII)

Contacto: Gustavo Romay. Consultor

Se manejó la posibilidad de conseguir mediante esta organización un STB MHP de desarrollo para el testeo de aplicaciones. Sin embargo no parece viable la obtención de equipamiento de este tipo en el corto plazo.

- Entretenimiento Hispano.

Contacto: Manuel Cicarello

Entretenimiento Hispano es una empresa dedicada al desarrollo de software que próximamente comenzará a desarrollar aplicaciones MHP. Bajo financiación de la ANII contará con equipamiento MHP (STBs, STBs de desarrollo, emuladores, herramientas de desarrollo), y también recibirá capacitación en cuanto al estándar y el desarrollo en Java de aplicaciones interactivas. Este nuevo emprendimiento tendrá lugar alrededor de mayo de 2009.

Luego de visualizado este panorama se puede llegar a la conclusión de que aún no se cuenta con un entorno de pruebas adecuado para aplicaciones interactivas MHP en el Uruguay. Sin embargo, la ANII ha negociado con el Ministerio de Industria Turismo y Comercio de España el uso gratuito del iLAB de la UPM para las instituciones académicas de Uruguay, y para empresas el costo será mínimo. Las personas contactadas anteriormente coincidieron en que, por el momento, esta es la opción más accesible para el testeo de aplicaciones MHP.

4.19 Tarea 8.2

Testeo

Dado que en el Uruguay actualmente no existe una cabecera de transmisión digital en condiciones de soportar la emisión de aplicaciones MHP, se decidió realizar el testeo en un laboratorio instalado por la UPM denominado iLAB. La ANII (Agencia Nacional de Investigación e Innovación, Uruguay) ha pactado un convenio con el MITYC (Ministerio de Industria Turismo y Comercio, España) para habilitar a centros académicos uruguayos a acceder de forma gratuita al iLAB. Sin embargo, no fue posible obtener el acceso a través de la ANII. Por otro lado, se intentó contactar a la UPM directamente desde la Universidad de Montevideo, pero tampoco se obtuvo respuesta.

Como segunda opción se decidió utilizar el testeo Online de aplicaciones interactivas disponible en <http://www.mhp-knowledgebase.org/>. Dicha página expone diferentes centros online para la prueba de aplicaciones desarrolladas bajo el estándar MHP. Actualmente se ofrecen tres centros, éstos son:

- University of Duisburg-Essen (UDE), Essen, Germany

- Institut für Rundfunktechnik (IRT), Munich, Germany
- Instituto Tecnológico de Aragón (ITA), Zaragoza, Spain

Para acceder a dichos centros se requiere de autorización previa. Por esta razón se realizó la reserva de los mismos vía web, seleccionando fecha, hora y duración del servicio. Lamentablemente los tres sitios se encontraban bajo reforma imposibilitando el testeo de la aplicación. En el caso del centro UDE surgieron problemas al momento de cargar los archivos de la aplicación. Se informó de esta falla al contacto publicado en el link, pero no se obtuvo respuesta. En el resto de los centros el acceso no se pudo concretar ya que se encontraban datos de baja.

A pesar de que ninguno de los testeos pudo ser realizado, se llevó a cabo un estudio del funcionamiento del iLAB. A continuación se detalla la información asociada al mismo.

iLAB: “Plataforma interconectada para la Definición, implementación y pruebas de servicios, aplicaciones y productos basados en el estándar MHP”.

Definición

El i-LAB es un laboratorio que brinda un entorno de pruebas para las aplicaciones desarrolladas bajo el estándar MHP. Éste puede ser accedido de manera remota o local, permitiendo la prueba de aplicaciones en diferentes receptores MHP.

Necesidad

Definición e implementación de un entorno remoto de pruebas de aplicaciones desde el que se pueda interactuar con un número significativo de receptores que soportan MHP.

La existencia del iLAB permitirá realizar pruebas a un precio reducido sin tener que incurrir en el sobrecosto de una red privada de difusión.

Justificación de Impacto

La existencia de un iLab agilizará la implantación del estándar MHP permitiendo testear las aplicaciones desarrolladas en un entorno confiable, de manera fácil y rápida. Para instituciones académicas el acceso al iLab puede resultar gratis, mientras que para las empresas el costo es bajo.

Descripción

El sistema consiste en una plataforma de prueba y validación de aplicaciones MHP que se podrá acceder a través de la web, y dará como resultado una serie de reportes finales acerca del desempeño de la aplicación. Por otro lado permite validar los receptores MHP, que a pesar de ya contar con la certificación MHP a través del Test Suite, pueden no ser del todo eficientes, ya que el testeo anteriormente mencionado no tiene en cuenta aspectos como:

- ejecución simultánea de aplicaciones
- facilidad de uso del terminal

Arquitectura del sistema UPM

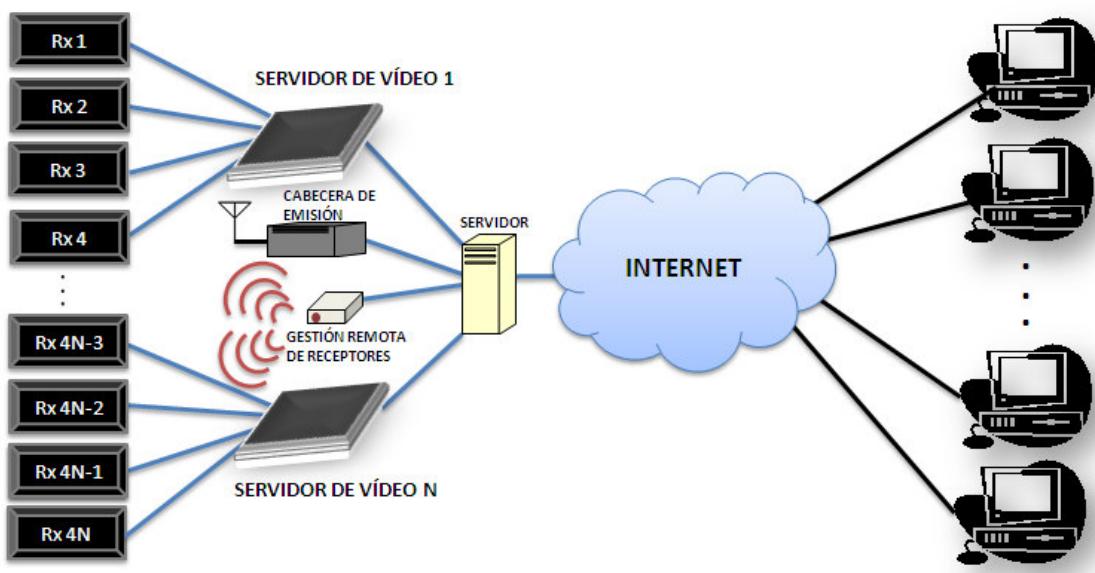


Figura 4.28: Arquitectura del iLAB

A través de una conexión a Internet, el usuario puede ubicar su aplicación en los servidores de video, para que ésta luego sea emitida por una cabecera de televisión digital, y finalmente obtenida vía aire por los receptores MHP.

Acceso al iLAB

Existen varios laboratorios bajo el proyecto iLAB. Uno de ellos es el de la UPM. En este caso se profundizará en el iLAB de la Universidad Politécnica de Madrid (UPM). Esta universidad no sólo ha generado convenios con la ANII para permitir su uso desde el Uruguay, sino que también posee acuerdos con la Universidad de Montevideo (UM).

Es posible acceder al laboratorio vía web concretando previamente una reserva. El usuario puede probar su aplicación de manera deseada en diferentes receptores DVB-T visualizando la salida en su propio monitor.

Requisitos de acceso

- Conexión a Internet. Velocidad recomendada: 3 Mbps o superior
- Navegador web recomendado: Firefox 2.0.0.15
- Habilitar puertos: 4500, 4502, 4503.

Validación de una aplicación

Para que una aplicación sea validada, ésta debe:

- Adecuarse al estándar MHP: Se analizará el código fuente detectando código redundante o no optimizado.
- Desempeñarse adecuadamente de manera individual.
- Coexistir con otras aplicaciones (compartir memoria, recursos, etc.)
- Ser de fácil navegabilidad y usabilidad.

El iLAB no sólo permitirá visualizar la aplicación en el monitor del usuario, sino que también entregará un reporte final sobre el desempeño de la aplicación. En las figuras presentadas a continuación se muestra de manera conceptual los testeos a realizar sobre las aplicaciones o receptores MHP para luego generar el informe final.

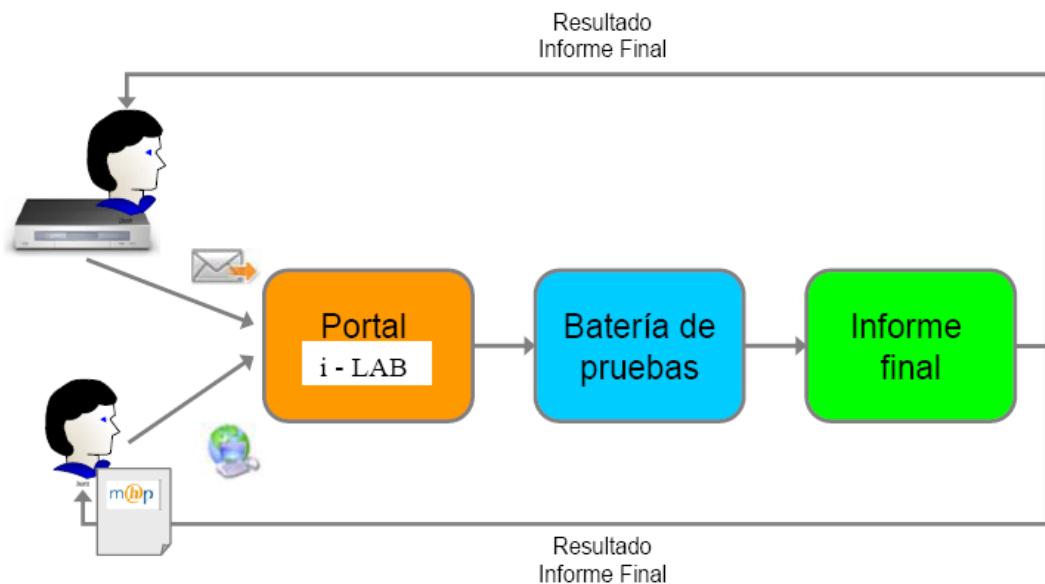


Figura 4.29: Reporte entregado por el iLAB

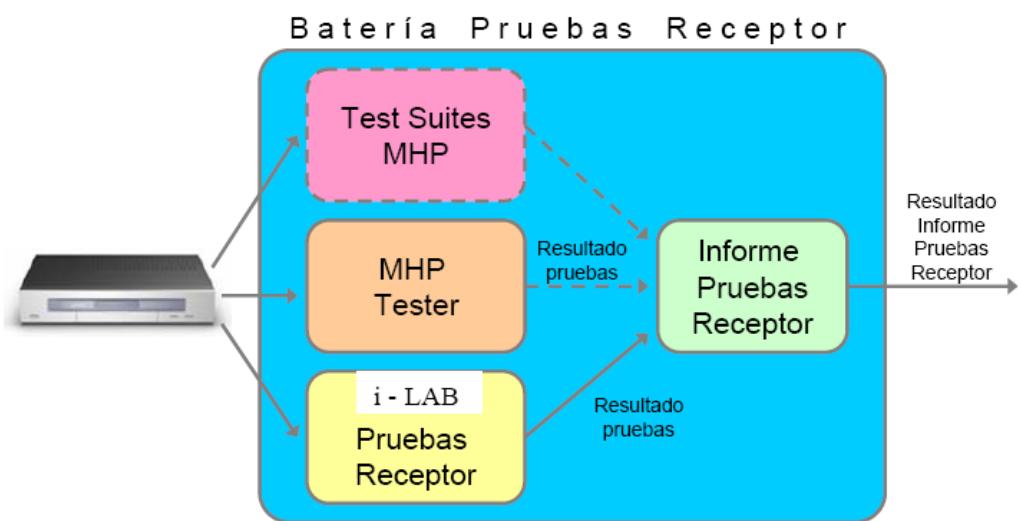


Figura 4.30: Batería de pruebas del iLAB para los receptores MHP

En el caso del proyecto TvDTi la batería de pruebas más útil se corresponde con la siguiente figura:

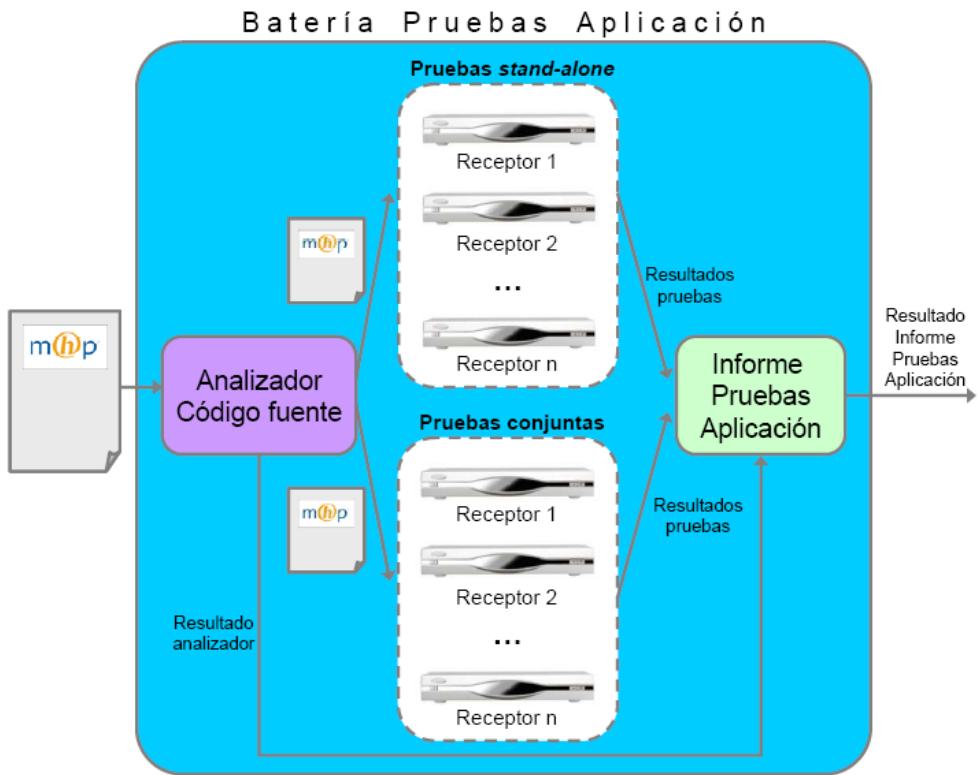


Figura 4.31: Batería de pruebas del iLAB para las aplicaciones MHP

Tipos de test

Existen dos tipos de testeos a realizar sobre las aplicaciones:

- Test de aplicación definido por el usuario: Dirigido mediante comando a distancia virtual en pantalla.
- Test de aplicación predefinido: Comprueba las funcionalidades de la aplicación enviada por el usuario.

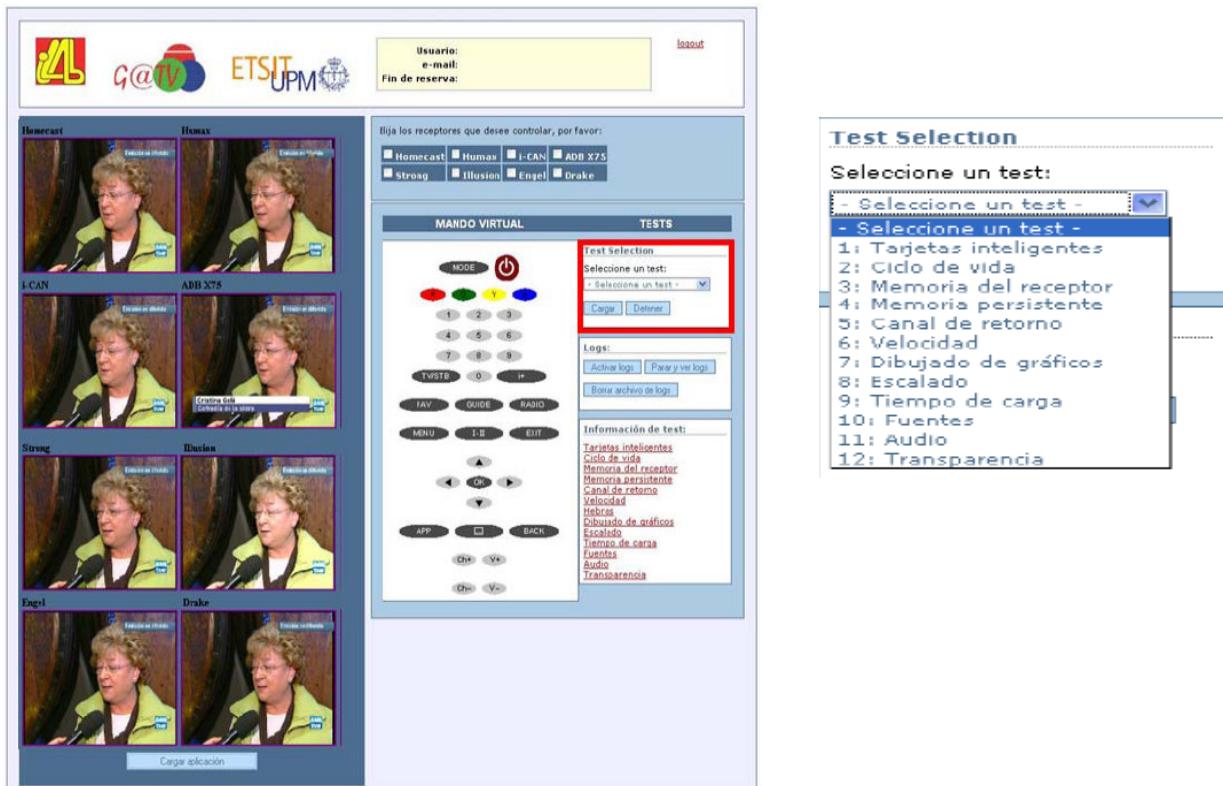


Figura 4.32: Interfase gráfica del iLAB.

Algunos de los tests predefinidos son:

- Ciclo de vida

Muestra los diferentes estados que va atravesando la aplicación enviada por el usuario.

- Memoria del receptor

Determina la cantidad de memoria utilizada y disponible en el receptor.

- Memoria persistente

Analiza la capacidad de lectura y escritura de datos en la memoria del receptor.

- Canal de retorno

Comprueba la correcta conexión del receptor de televisión digital al canal de retorno.

- Velocidad

Determina la velocidad de lectura y escritura de datos en la memoria persistente.

- Dibujado de gráficos

Analiza las capacidades de los receptores en cuanto al dibujado de líneas, polígonos y texto.

- Escalado

Determina qué factores de escalado son soportados por los distintos receptores.

- Tiempo de carga

Aplicación que determina el tiempo que tarda en cargar la aplicación del usuario en los diferentes receptores.

- Fuentes

Comprueba qué tipos de letra son soportados por los receptores.

En el Anexo 8, “Manual de uso del iLAB”, se adjunta la documentación para el uso del iLAB. La misma fue brindada por la ANII (Agencia Nacional de Investigación e Innovación, Uruguay.)

Capítulo 5: Resultados

Resultados obtenidos

Con el objetivo de contribuir con el desarrollo de aplicaciones interactivas para TDT destinadas a proporcionar un servicio ciudadano, el proyecto TvDTi logró los siguientes resultados:

Generación de un documento explicativo sobre las tecnologías involucradas en la TDT

Se generó un documento al que se denominó “Guía MHP”. El mismo se corresponde con el marco teórico del proyecto TvDTi, y presenta una visión general de la televisión digital interactiva, permitiendo entender los conceptos básicos de la misma, y brindando los conocimientos fundamentales para el desarrollo y testeo de aplicaciones MHP. Dicha guía tiene como objetivo trascender el presente proyecto aportando a la comunidad de desarrollo de aplicaciones interactivas.

Desarrollo de aplicaciones interactivas para TDT de carácter comunitario

Se generaron dos aplicaciones MHP en el contexto de la TDT abierta en el Uruguay. Si bien las mismas no tienen en cuenta todos los aspectos MHP, como por ejemplo seguridad, éstas son claros ejemplos de aplicaciones MHP, y cuentan con comentarios y un documento explicativo que permiten entender fácilmente las bases del desarrollo.

Se pretende generar la transferencia de conocimiento adecuada que permita extender el ciclo de vida de las aplicaciones desarrolladas más allá del período de tiempo del Proyecto Final de Carrera; utilizándose como base para seguir avanzando en conocimiento, generando nuevas versiones más sofisticadas.

Alcance de objetivos

Al inicio del proyecto se establecieron mediante el documento “Plan de proyecto” los objetivos principales del mismo. A continuación se enumeran los objetivos específicos propuestos y los resultados obtenidos al respecto:

1. Establecimiento de un ambiente de desarrollo para aplicaciones interactivas de televisión digital.

Se estudiaron las diferentes opciones existentes para el desarrollo y testeo de aplicaciones interactivas. En el caso del proyecto TvDTI se escogió hacer uso de herramientas de desarrollo java genéricas, un jar con la implementación MHP, y un emulador MHP para el testeo de aplicaciones.

En la sección 2.3.1 se establece de manera teórica el equipamiento necesario para la instalación de un laboratorio MHP. En el Anexo 10, “Productos para la instalación de un ambiente de desarrollo MHP”, se mapea dicho equipamiento con productos existentes en el mercado.

2. Implementar una aplicación representativa del poder del estándar MHP dentro del sector de servicios ciudadanos, sin la necesidad de la utilización de un canal de retorno que envíe información desde el televidente a la emisora correspondiente.

Se implementó una aplicación denominada “Cartelera de Espectáculos”. La misma fue elegida según una encuesta realizada sobre las aplicaciones de carácter comunitario preferidas por los televidentes. En la sección 4.11 se describe la aplicación en cuestión y se anexa el código correspondiente.

3. Determinar la configuración requerida del lado de la emisora para un correcto funcionamiento de la aplicación.

Durante la tarea “Búsqueda y estudio de tutoriales, manuales, publicaciones, etc.” se pudieron identificar aquellos aspectos a tener en cuenta por parte de la emisora a la hora de transmitir aplicaciones MHP. En base a esto se llegó a la conclusión de que la pieza clave para la emisión de aplicaciones es el generador de carrusel, por lo que en la sección 4.17 del documento en cuestión se profundiza en el uso y funcionamiento del carrusel de objetos.

Adicionalmente, en el Anexo 10 se mencionan algunos productos ofrecidos actualmente en el mercado para la implementación del generador de carrusel de objetos.

4. *Se estudiarán las diferentes posibilidades que existen de implementar un canal de retorno en el contexto de la TDT.*

En la sección 4.15 se aborda este tema. Finalmente se seleccionó el mecanismo más apropiado para la realización de encuestas haciendo uso del canal de retorno. Adicionalmente se desarrolló la aplicación correspondiente, aunque este punto no estaba incluido en los objetivos iniciales. La misma es descrita en la sección 4.16 y se anexa su código.

En cuanto a este objetivo vale la pena aclarar que la aplicación fue desarrollada y luego testeada en un emulador MHP, por lo que el correcto funcionamiento del canal de retorno no pudo ser verificado.

5. *Se analizará la posibilidad de testear la aplicación en el contexto real de funcionamiento.*

El testeo en cuestión iba a realizarse en el iLAB de la UPM, laboratorio accesible mediante la Web. Sin embargo, a pesar de haber estudiado el funcionamiento del mismo, y las funcionalidades ofrecidas, el acceso al laboratorio no fue obtenido debido a cuestiones de negociación entre la ANII y el gobierno español. Igualmente se expone una descripción del iLAB y el manual de usuario del mismo.

El alcance de este objetivo habría enriquecido mucho los resultados alcanzados, ya que reafirmaría la aplicabilidad del proyecto en cuestión. Permitiendo comprobar el funcionamiento del canal de retorno y el acceso al carrusel de objetos, entre otras cosas.

Capítulo 6: Conclusiones

En este apartado se exponen las conclusiones obtenidas a partir de la realización del proyecto TvDTi. Se detallan también las limitaciones observadas en el sistema desarrollado, así como las líneas de trabajo futuras.

Limitaciones

La mayoría de los objetivos establecidos por el proyecto TvDTi fueron alcanzados de manera satisfactoria, aunque se encontraron algunas limitaciones:

- Fuentes de información insuficientes acerca de MHP.

Las aplicaciones desarrolladas se encuadran en el marco de la TV digital y la interactividad. Dado a que la TV digital interactiva es un proceso bastante actual, las tecnologías implicadas no tienen el grado de madurez deseable para un desarrollo cómodo. Si bien este proyecto puede ser tomado como punto de partida para comenzar a desarrollar aplicaciones MHP, el mismo no puede considerarse una capacitación íntegra de MHP.

- Falta de recursos de software y hardware para el desarrollo y testeo de aplicaciones MHP de modo eficiente.

Dado que el Uruguay aún no ha profundizado en la TDTi, no existen instituciones con los recursos necesarios para desarrollar y testear aplicaciones MHP, por lo que resultó difícil finalizar el proceso de entendimiento de esta nueva tecnología.

Debido a la existencia de esta limitación es que se optó por utilizar la infraestructura ofrecida por el iLAB de la UPM para el testeo de aplicaciones en el ambiente real. Si bien existe un convenio entre la ANII y el gobierno español para la utilización de dicho laboratorio de manera gratuita, al momento de solicitar el acceso al mismo, éste no fue otorgado. Esto se debe a la burocracia existente entre los entes involucrados.

Si se desea explotar la industria de desarrollo de aplicaciones MHP en el Uruguay, tanto para consumo interno como para exportación, se debe contar con una plataforma de testeo local. La utilización de un laboratorio en las afueras del país no resulta del todo serio. Creemos firmemente que se debe apostar al desarrollo de las aplicaciones

interactivas MHP debido al éxito que ha tenido la exportación de software en los últimos años.

Perspectivas

MHP proveerá nuevas oportunidades a los operadores de TV, al permitir la convergencia de las tecnologías más populares hoy en día: la computadora y la televisión. Sin embargo, la migración a TDTi requerirá de modificaciones del lado de la emisora, más específicamente en las áreas de Transmisión (la pieza clave será el carrusel de objetos) y la producción de contenidos (el mayor desafío será en el ámbito creativo, en donde las emisoras deben ser capaces de crear nuevos servicios), y del lado del cliente, quien debe adaptar su equipamiento a la tecnología digital.

A continuación se detallan aquellos factores que se consideran necesarios para una exitosa implementación de la TDTi en el Uruguay:

- Introducción de políticas públicas

Al incorporar la nueva tecnología digital, ésta impactará a nivel social, económico, político y cultural, y renovará no solamente en el aspecto tecnológico sino también la misión de los medios de comunicación. Por esta razón Uruguay debe tomar conciencia de la importancia de contar con medios de comunicación electrónica actualizados, que permitan mantener a la población conectada con el mundo desarrollado. Dada la experiencia europea durante la implementación de la TDT, es posible decir que ésta debe ser llevada a cabo bajo estrictas políticas públicas. Los operadores de TV abierta también deben participar de manera activa en el proceso de migración, acompañando las políticas regulatorias, los calendarios impuestos, las campañas de comunicación, etc. Uno de los principales desafíos a afrontar por parte de las autoridades públicas es el de garantizar y promover la correcta inversión en infraestructura y generar los planes de ayuda necesarios para lograr un acceso universal al nuevo servicio.

- Desarrollo de una “Killer application”

La introducción de la TDT en el Uruguay no debe consistir únicamente en una renovación tecnológica, sino que debe aprovecharse al máximo para generar nuevos servicios a través de la interactividad. Los usuarios finales deben percibir el cambio de tecnología como un beneficio, por lo tanto es importante implementar aplicaciones interactivas bajo

el rol “Killer application”, que implicará una asimilación definitiva de los usuarios, y los impulsará a adquirir decodificadores digitales con soporte MHP. Por estas razones, creemos que la popularización de la interactividad en el Uruguay depende directamente de la aparición de una “Killer application”.

- Utilización de un estándar abierto de interactividad: MHP

Por otro lado parece importante destacar que la introducción de interactividad de acuerdo a un estándar abierto, como MHP, resulta beneficiosa. Un estándar abierto se encuentra disponible públicamente y cualquiera puede utilizarlo sin necesidad de pagar regalías, por lo que habilita la compatibilidad e interoperabilidad entre distintos componentes tanto de software como de hardware. Adicionalmente este tipo de estándares tienden a generar un mercado libre y dinámico, permitiendo la sofisticación y la mejora continua.

- Formación de recursos humanos en materia de la TDT

Para lograr una implantación de la TDT en la región es necesario contar con la formación rápida de recursos humanos que puedan cubrir las necesidades de “Know-how” para realizar el despliegue de la nueva red, y desarrollar aplicaciones interactivas de óptima calidad.

Tiempo de vida de la TDT abierta en el Uruguay

La siguiente reflexión es acerca de la “vida útil” de la TDT abierta. El 30 de diciembre de 2008, el gobierno anunció oficialmente la ejecución del Proyecto de Convergencias para el Acceso a la Recreación y al Desarrollo de Alternativas Laborales y Emprendimientos Sustentables (Plan Cardales), a través del cual todos los ciudadanos podrán acceder a información, entretenimiento, Internet, televisión para abonados y servicios de telefonía. La idea es llegar con un plan de acceso diferencial y con distintos paquetes tecnológicos de telecomunicaciones al conjunto de los ciudadanos, buscando un servicio universal. Frente a esta situación, podría pensarse que la TDT abierta debería ser completamente reemplazada por la TV privada, sin embargo esto no es totalmente cierto. Si bien la nueva TDT tendrá grandes objetivos enfocados en la industria y la economía, ésta seguirá siendo el único medio de comunicación pluralista, con fines sociales y culturales.

Capítulo 7: Recomendaciones

El mundo de la TV digital es muy amplio como para abarcarlo por completo en un único proyecto. Durante el proyecto TvDTi se profundizó en temas relacionados a la implementación de aplicaciones MHP, pero existen varios otros puntos a atacar en futuros proyectos que también contribuirán a desarrollar el conocimiento técnico en el Uruguay acerca de esta nueva tecnología:

- Montaje y configuración de un ambiente de desarrollo real a “pequeña escala” para el testeo de aplicaciones interactivas MHP, en el Uruguay.
- Emisión (broadcasting) de aplicaciones interactivas.
- Diseño de receptores MHP.
- Seguridad en el marco de la TVi.
- Aplicaciones interactivas para la TV por cable.
- Implementación de aplicaciones altamente interactivas (transacciones comerciales, quizzes, etc.) que requieran de una canal de retorno, seguridad robusta y emisión unicast.

Cualquiera de estos proyectos no sólo tiene valor académico, sino también comercial. Podrían llevarse a cabo en el plano académico, pero sería conveniente llevarlos al plano laboral al menos como una pasantía. Ésta podría ser de gran utilidad en el contexto de una emisora, y sin lugar a dudas permitiría contar con más recursos de hardware, software, y adicionalmente le daría al proyecto un vuelco comercial de manera inmediata.

Finalmente en base al proyecto desarrollado podría prepararse una práctica de Laboratorio para los alumnos de la carrera Ingeniería telemática en la que se diera una

breve introducción a la TvDTi, la interactividad y el estándar MHP, y se solicitará desarrollar un código sencillo, simplemente como demostración de esta innovadora tecnología y despertar el interés de los alumnos.

Glosario y abreviaciones

.class	Al utilizar el lenguaje de programación Java se generan archivos ".java" que contienen el código fuente y que serán posteriormente compilados en archivos ".class".
AIT	Application Information Table. Tabla definida por MHP para proveer información básica sobre las aplicaciones.
API	Application Programming Interface. Define cómo debe invocarse una determinada función de un programa desde una aplicación. Cuando se estandariza una plataforma se estipulan una serie de APIs a las que deben adaptarse todos los desarrolladores.
ARIB	Association of Radio Industries and Businesses. Organización que tiene como objetivo conducir y fomentar la investigación y el desarrollo en todo lo involucrado al uso del espectro electromagnético.
ATSC	Advanced Television Systems Committee. Asociado a los estándares Norteamericanos de televisión.
AWT	Abstract Window Toolkit. En español Kit de Herramientas de Ventana Abstracta destinado a la presentación de gráficos, interfaces de usuarios y sistemas de ventanas en una plataforma Java.
Broadcast	Radiodifusión. Refiere a la distribución de audio y/o señales de vídeo que transmiten los programas a una audiencia masiva o no.
CA	Conditional Access. Refiere a la protección de contenido mediante el cumplimiento de ciertos criterios antes de que el acceso sea garantizado.
Classpath	Indica dónde se deben buscar los archivos a compilar o ejecutar, sin tener que escribir en cada ejecución la ruta completa.
Compilar	Proceso de traducción de un código fuente (escrito en un lenguaje de programación de alto nivel, por ejemplo Java) a lenguaje máquina para que pueda ser ejecutado por la computadora.

DAVIC	Digital Audio Video Council. Asociación de compañías para promover la interoperabilidad de equipamiento audiovisual a través de la estandarización.
DTV	Digital TV. Televisión Digital.
DSM-CC	Digital Storage Media Command and Control. Formato para transmisión de contenidos e información de control en una trama MPEG-2.
DVB	Digital Video Broadcasting. Organización que promueve estándares aceptados internacionalmente para el despliegue de televisión digital. Es el estándar europeo para la televisión digital.
DVB-J	Las aplicaciones DVB-J están escritas en Java, haciendo uso del API MHP, y consiste en un conjunto de clases que son emitidas junto con el flujo de audio y video de TV.
DVB-HTML	Las aplicaciones DVB-HTML son un conjunto de páginas HTML que son transmitidas como parte de un servicio.
ES	Elementary Stream.
GEM	Global Executable Multimedia Home Platform.
GUI	Graphical User Interface.
HAVI	Home Audio Video Interoperability.
HTML	HyperText Markup Language (Lenguaje de Marcas de Hipertexto). Lenguaje sumamente difundido para la construcción de páginas web. Permite describir la estructura y el contenido en forma de texto.
HTTP	Hyper Text Transfer Protocol. Protocolo de transferencia de hipertexto usado en cada transacción de la Web (WWW).
HW	Hardware.
IP	Internet Protocol. Protocolo no orientado a conexión usado para la comunicación de datos a través de una red de paquetes comutados.

iTV	Interactive Television.
Java	Lenguaje de programación orientado a objetos. Fue desarrollado por Sun Microsystems al principio de la década de los 90. La programación en Java es compilada en bytecode y luego ejecutada por una máquina virtual Java.
JMF	Java Media Framework.
JVM	Java Virtual Machine.
MHP	Multimedia Home Platform.
MIME	Multipurpose Internet Mail Extensions. Serie de especificaciones dirigidas al intercambio transparente de todo tipo de archivos a través de Internet.
MPEG	Moving Pictures Expert Group.
MPEG-2	Técnica de compresión de audio y video definida por MPEG.
PC	Personal Computer (computadora personal).
Pixel	Picture Element. Es un único punto en una imagen gráfica. Los monitores gráficos muestran imágenes dividiendo la pantalla en miles (o millones) de pixeles, dispuestos en filas y columnas.
PSI/SI	Program Specific Information/Service Information.
PSTN	Public Switched Telephone Network. Se le denomina así al sistema de telefonía fija sobre cobre para transportar la voz.
Relación de aspecto	Razón de proporcionalidad de una imagen. Es la proporción entre el ancho y alto de la misma. Se calcula dividiendo el ancho por el alto de la imagen visible en pantalla, y se expresa normalmente como «X:Y».
SMS	Short Messaging Service (conocido en el ámbito de la telefonía móvil).
STB	Set-top box. Receptor MHP conectado al televisor.
SW	Software.

TDT	Televisión Digital Terrestre.
TS	Transport stream.
TV	Televisión / Televisor.
TvDTi / TDTI	Televisión Digital Terrestre Interactiva.
UHF	Ultra High Frequency. Banda del espectro electromagnético que ocupa el rango de frecuencias de 300 MHz a 3 GHz. Uno de los servicios UHF más conocidos son los canales de televisión abierta.
URL	Uniform Resource Locator. Secuencia de caracteres, de acuerdo a un formato estándar, que se usa para nombrar recursos, como documentos e imágenes en Internet, por su localización.
XML	Extensible Markup Language. Generalmente utilizado para darle formato al contenido MHP.

Bibliografía

<http://es.wikipedia.org/wiki/MHP>

30 de enero de 2009

<http://www.mhp.org/>

30 de enero de 2009

<http://www.televisiondigital.electronicafacil.net/Sections-article21-p1.html>

30 de enero de 2009

<http://www.mhp-knowledgebase.org/>

30 de enero de 2009

<http://www.mhproject.org/>

30 de enero de 2009

<http://www.interactivetvweb.org/>

30 de enero de 2009

<http://www.xletview.org/>

22 de enero de 2009

<http://code4tv.com>

30 de enero de 2009

<http://forums.sun.com/index.jspa>

30 de enero de 2009

<http://www.cesnavarra.net/cesdigital/Lists/Noticias%20CESDigital/AllItems.aspx>

30 de enero de 2009

http://kristal-priv.ita.es/ilab/index.php?option=com_content&task=view&id=46&Itemid=69

30 de enero de 2009

<http://www.televisiondigital.electronicafacil.net/Sections-article25-p1.html>

30 de enero de 2009

<http://elearningco.wordpress.com/2008/03/22/posibilidades-interactivas-de-la-iptv-versus-la-tdt/>

30 de enero de 2009

<http://www.migaceta.com/2006/12/competencia-entre-tdt-y-iptv.html>

30 de enero de 2009

<http://sociedaddelainformacion.telefonica.es/jsp/articulos/detalle.jsp?elem=4642>

30 de enero de 2009

<http://videobits.tv/2007/06/22/%C2%BFque-pasara-tras-el-apagon-de-2010/>

30 de enero de 2009

http://convergence.blogs.ie.edu/archives/2006/03/hacia_la_nueva_1.php

30 de enero de 2009

<http://www.ipodizados.com/un-escenario-del-futuro-tdt-conclusiones-de-alc-zar-por-gonzalo-martin>

30 de enero de 2009

http://www.vnunet.es/es/vnunet/opinion/2008/03/24/tv_digital_en_latinoamerica_el_futuro

08 de febrero de 2009

<http://www.elmundo.es/elmundo/2008/08/29/comunicacion/1220001244.html>

08 de febrero de 2009

<http://www.larepublica.com.uy/politica/347054-plan-cardales-inclusion-e-igualdad-de-oportunidades>

08 de febrero de 2009

Morris, Steven. Smith-Chaigneau, Anthony. Interactive TV Standards: A Guide to MHP, OCAP, and JavaTV. Focal Press, 2005. 608p.

Formato electrónico: http://books.google.com.uy/books?id=Tv9_WcYPXtwC

30 de enero de 2009

Schwalb, Edward M. ITV Handbook: Technologies and Standards. Prentice Hall PTR, 2003. 724p.

Formato electrónico: http://books.google.com.uy/books?id=DM_EV4yJKYUC

30 de enero de 2009

Anexos

Contenido

- Anexo 1. Plan de proyecto
- Anexo 2. Análisis FODA
- Anexo 3. Informe de justificación de avance de 50%
- Anexo 4. Informe de justificación de avance de 100%
- Anexo 5. Actas de reunión
- Anexo 6. Código desarrollado
- Anexo 7. Generador de contenidos – Cartelera de Espectáculos
- Anexo 8. Manual de uso del iLAB
- Anexo 9. Proyecto Sala+. Acta de reunión nº 1, 18 de diciembre de 2008
- Anexo 10. Productos para la instalación de un ambiente de desarrollo MHP

Anexo 1: Plan de proyecto

Anexo 2: Análisis FODA

Anexo 3: Informe de justificación de avance de 50%

Anexo 4: Informe de justificación de avance de 100%

Anexo 5: Actas de reunión

Anexo 6: Código desarrollado

Anexo 7: Generador de contenidos – Cartelera de Espectáculos

Anexo 8: Manual de uso del iLAB

Anexo 9: Proyecto Sala+. Acta de reunión nº 1, 18 de diciembre de 2008

Anexo 10: Productos para la instalación de un ambiente de desarrollo MHP
