Universidad Zaragoza

Trabajo Fin de Grado

Desarrollo de una aplicación de subtitulado mediante pictogramas para una smart tv y navegadores web

Autor:

Miguel Hernández Boza

Director:

Eduardo Lleida Solano

Escuela de Ingeniería y Arquitectura

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

Julio 2015

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

D./Dª. Miguel Hernández Boza,

con nº de DNI 17767068M en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) Grado, (Título del Trabajo) Desarrollo de una aplicación de subtitulado mediante pictogramas para una smart tv y navegadores web,

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 9 de junio de 2015

Fdo: Miguel Hernández Boza

Desarrollo de una aplicación de subtitulado mediante pictogramas para una smart tv y navegadores web

Resumen

El TFG realizado trata de solventar los impedimentos que tienen las personas con discapacidad en el ámbito de la comunicación, ayudando a que comprendan mejor lo que están viendo mediante el desarrollo de una aplicación que utilice un subtitulado hecho con pictogramas. Se ha creado una aplicación para televisiones Samsung Smart TV así como una aplicación web donde poder ver todo el contenido. Ambas aplicaciones han sido realizadas a partir de código HTML5, CSS3 y JavaScript, aunque también se han necesitado algunas herramientas como XAMPP o SDK Eclipse para poder programar o exportar las aplicaciones.

En la realización del TFG se contó con la colaboración de José Manuel Marcos, logopeda y especialista en pedagogía terapéutica del colegio público de educación especial "Alborada" de Zaragoza. José Manuel indicó las pautas a seguir en la correcta elección de los pictogramas y su disposición en la pantalla.

La utilidad de este TFG podría ser la de crear material audiovisual para el aprendizaje del lenguaje pictográfico, con diferentes niveles de dificultad y con una serie de cuestionarios de evaluación.

Índice general

Índice Figuras

Índice de tablas

Capítulo 1

Introducción

Las personas discapacitadas se encuentran en la vida real con situaciones de efectiva desigualdad e incluso llegan a aislarse y alejarse de la sociedad por el simple hecho de no poder expresar sus sentimientos o no saber interpretar al receptor de la conversación. Este hecho pone de manifiesto la falta de medios existente con respecto a estas personas, ya que los pocos sistemas desarrollados hasta ahora provienen del extranjero y tienen unos costes muy elevados.

Con este TFG se intenta mejorar su calidad de vida y en consecuencia su integración dentro de la sociedad paliando las desigualdades y discriminaciones. Para ello se ha adoptado una solución basada en pictogramas, explicados de forma más exhaustiva en el capítulo 2, facilitados por *ARASAAC* [13] (Portal Aragonés de la Comunicación Aumentativa y Alternativa) que expresan palabras a través de representaciones gráficas.

Figura 1 Pictograma de ejemplo "HOLA"

Estos pictogramas suponen una esquematización de lo que se está viendo a través de subtítulos, ayudando a la comprensión sin dejar de prestar atención al vídeo, un ejemplo es el pictograma de "hola" de la figura 1.

Teniendo claro el objetivo del producto final, ha sido implementado el subtitulado tanto en una aplicación para televisiones, cuyas compañías como es el caso de Samsung ponen a disposición del usuario herramientas de desarrollo para la creación de aplicaciones, como en una

página web facilitando así su difusión a un mayor número de personas de manera sencilla y gratuita.

1.1 Motivación y objetivo final

La motivación de este TFG está basada en la ayuda e integración de las personas con problemas de comunicación ya que son las más vulnerables a la marginación social puesto que algo tan normal para todos como es ver la tv se convierte en algo incomprensible para personas con estas barreras.

La solución planteada para la mejora de estas situaciones ha sido llevada a cabo a través de pictogramas, debiendo tener en cuenta, que actualmente no se ha encontrado nada parecido puesto que al tratarse de un sector tan específico las grandes empresas no apuestan por ello lo que se convierte, además de en una ventaja como es la de ser pioneros en un ámbito tan importante socialmente, en la posibilidad de crear unas bases utilizables en un futuro para la emisión desde las cadenas de televisión.

Por lo tanto el objetivo final es lograr transformar lo que existe actualmente a un subtitulado mediante pictogramas que sea totalmente compatible, automatizando todo el proceso. El TFG logra que cualquier persona con acceso a internet pueda visualizar los vídeos con una subtitulación hecha con pictogramas y en un futuro se pueda seleccionar en los televisores de los hogares.

1.2 Herramientas y lenguajes utilizados

En el desarrollo de las aplicaciones, han sido utilizados los lenguajes más compatibles con las plataformas que íbamos a utilizar, como son HTML5[1], CSS3[2] y Javascript[3]. HTML5 es la quinta versión del lenguaje básico de la World Wide Web, esta nueva versión incorpora nuevas etiquetas para mostrar contenidos multimedia. CSS3 es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML. Javascript es un lenguaje utilizado para crear páginas web dinámicas. Con estos lenguajes se ha diseñado la web inicial, utilizando como editor la herramienta Brackets[9], que permite organizar y editar todos los ficheros de manera muy visual.

Para la traducción de los subtítulos se ha utilizado una aplicación creada en lenguaje Java[4] y el entorno de desarrollo SDK Eclipse[8].

Con respecto a la aplicación en la smart tv, se ha utilizado el SDK Samsung smart tv[5] con el que se ha programado todas las funcionalidades a través de JavaScript, y además se ha cambiado la disposición de la aplicación a través de HTML5 y CSS3. Dentro de la aplicación también se usa XML para el fichero RSS (Really Simple Syndication, un formato XML para compartir contenido en una web) del cuál la aplicación sincroniza el contenido.

Para el servidor Apache se ha usado la herramienta XAMPP[7] que proporciona directamente el servidor montado y que será necesario para cargar las aplicaciones que exportamos desde el SDK Samsung a la televisión.

Finalmente, en la conversión automática del fichero SRT (Formato de subtítulos creado por el software SubRip) a un lenguaje más apropiado para la representación de pictogramas, se ha usado un programa en Java en el entorno de Eclipse[8] y la herramienta freeling[21]. Esta herramienta pone a disposición los resultados de la investigación del lenguaje natural UPC, la herramienta es una biblioteca de la prestación de servicios de análisis del lenguaje, permite al usuario analizar ficheros de texto y ofrece el análisis morfosintáctico de cada palabra del texto además de otras muchas opciones.

1.3 Estructura de la memoria

El contenido de la memoria viene estructurado de la siguiente manera

- **Capítulo 1:** Se ha realizado una introducción del TFG y la motivación que ha llevado a la realización de este.

- **Capítulo 2:** Análisis de la situación actual de la comunicación aumentativa, los pictogramas y el subtitulado.

- **Capítulo 3:** Bloque principal de la memoria donde se desarrolla todo el proceso de creación de las aplicaciones, comentando sus diferentes versiones, tanto web como smart tv. Además de la generación automática del fichero de subtitulado.

- **Capítulo 4:** Se realiza una conclusión final y un posible desarrollo en líneas futuras.

En la parte final se encuentran los diferentes anexos donde vienen explicados con más detenimiento las partes de programación de las herramientas y aplicaciones, así como los manuales de uso.

- **Anexo A:** Explicación de la plataforma elegida y de las especificaciones requeridas.

- **Anexo B:** Explicación detallada del proceso de subtitulado en las aplicaciones, generación del fichero mediante las herramientas creadas en java y manual de uso.

- **Anexo C:** Explicación del SDK de Samsung y ficheros de la aplicación para Smart Tv.

- **Anexo D:** Manual de uso de la instalación del servidor Apache y de la exportación de la aplicación a una televisión Samsung.

Capítulo 2

Análisis previo y estudio del arte

2.1 Comunicación Aumentativa

El término comunicación tal y como establece la R.A.E. (Real Academia Española) se refiere a la "transmisión de señales mediante un código común a emisor y receptor". A este respecto debemos plantearnos ¿qué ocurre cuando el receptor no entiende el código? Para solucionar este problema de incomprensión del código por parte del receptor debemos utilizar un nuevo código de comunicación, que en el caso que nos ocupa serán los pictogramas, los cuales pueden ser utilizados en sistemas aumentativos o alternativos

- Sistemas aumentativos.

Complementan al lenguaje oral que por sí solo no es suficiente es decir, conllevan el uso de herramientas de ayuda para transferir el mensaje que pretendemos comunicar.
Las herramientas de ayuda de que se sirven este tipo de sistemas deben ser universales, como pueden ser los carteles o señales de tráfico, que ayuden a todos por igual facilitando la comprensión.

- Sistemas alternativos.

Sustituyen al lenguaje oral, como señalar imágenes en un tablero o agendas visuales.
Ambos sistemas, aunque de diferentes modos, permiten que personas con problemas de comunicación puedan relacionarse e interactuar con los demás de manera igualitaria, teniendo así las mismas oportunidades y derechos en la sociedad.

Dentro de la comunicación aumentativa no hay que olvidar la suma importancia de utilizar un vocabulario lo más preciso posible, para ello

en el subtitulado llevado a cabo en el producto de este TFG han sido utilizados verbos en infinitivo, pictogramas impersonales así como una lectura de apoyo a la hora de la realización de las frases del libro "Lectura Fácil Métodos de Redacción y Evaluación"[22] de Oscar García Muñoz Igualmente se han tenido en cuenta reglas de comunicación, educación y tiempos para no intervenir entre los interlocutores.

Este sistema de subtitulado mediante pictogramas resulta efectivo y refleja mejoras a través de un uso repetido y continuado reforzado a su vez con una rutina de ejercicios de comprensión. Para evaluar la eficacia del sistema de subtitulado se usarían cuestionarios relacionados con lo visto constando éste de una serie de preguntas cuyas respuestas estarían basadas en pictogramas.

Como barreras a este método nos encontramos con la velocidad de aparición del subtitulado, se necesitan unos 3 segundos por pictograma para comprender lo que representa; y la carga de pictogramas por frase, no deben aparecer en la imagen demasiados pictogramas. Ambas barreras se pueden solucionar expandiendo al máximo los tiempos de aparición y resumiendo el subtitulado inicial.

2.2 Pictogramas

El creador del primer sistema universal de comunicación mediante símbolos fue Charles Bliss[15], quien creo el sistema Bliss, aún utilizado en todo el mundo y que es usado en personas con parálisis cerebral sin habla. Un ejemplo de este sistema se puede ver en la figura 2.1.

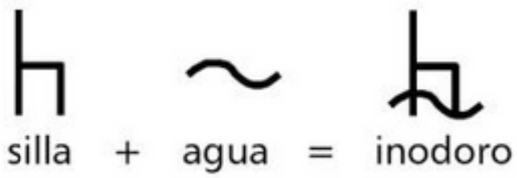

Figura 2.1 Ejemplo sistema Bliss

Este sistema simbólico gráfico-visual representa conceptos y permite comunicarse a personas sin habla, usando dibujos para representar estados y colores diferenciando así la interpretación. A partir de aquí se crean estrategias para aumentar el vocabulario, en este caso se usa los

pictográficos, imágenes que tienen un gran parecido con lo que quieren representar. Bliss abogaba la idea de no construir frases largas y omitir preposiciones, artículos, etc.

En 1981 Roxana Mayer[15] diseño unos criterios, simbolizar palabras y conceptos de uso frecuente en la comunicación cotidiana, que fueran para todas las edades, fotocopiables y fáciles de distinguir, con un vocabulario similar al del sistema Bliss, este sistema es SPC (Símbolos Pictográficos para la Comunicación). Un ejemplo del sistema SPC se puede ver en la figura 2.2.

Figura 2.2 Ejemplo sistema SPC

En el TFG se usan los pictogramas de ARASAAC[14], los cuales pueden ser libremente utilizados teniendo como única condición la imposibilidad de lucrarse a través de su uso en aplicaciones. Todos ellos se pueden descargar desde la página oficial de ARASAAC y existen un total de 14.000 pictogramas.

Para subtitular son elegidos pictogramas impersonales para evitar problemas al identificarse con el emisor, no se usan plurales y buscando siempre el que se adapta al contexto que se está viendo, por ejemplo no es lo mismo querer algo que a alguien, se usan diferentes pictogramas para cada cosa. Un ejemplo de una frase preparada con pictogramas es la figura 2.3

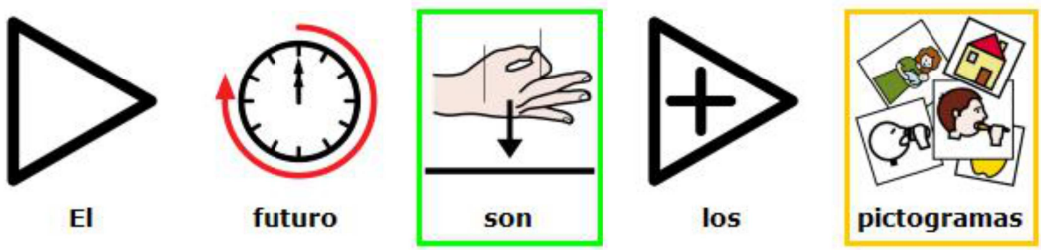

Figura 2.3 Ejemplo de una frase hecha en pictogramas

2.3 Subtitulado

El doblaje junto con la subtitulación es una de las modalidades más extendidas de traducción. Los subtítulos son la transcripción en texto que aparece debajo del vídeo que se está reproduciendo, aportando una ayuda para las personas sordas para comprender lo que están viendo.
Los últimos estudios han demostrado los beneficios del subtitulado en el ámbito de la didáctica de lenguas en general. Un material sin subtitular tiende a crear un alto nivel de ansiedad e inseguridad, diversos experimentos han podido demostrar que la incorporación de subtítulos a este material proporciona una respuesta instantánea y, por tanto, un refuerzo positivo que contribuye a crear una sensación de seguridad y ayuda a sentirse preparados para enfrentarse a este tipo de material audiovisual sin necesidad de apoyo textual.
Existen multitud de formatos para la subtitulación, estos son los más utilizados:

- SRT (Sub Rip)
- SUB (MicroDVD)
- VTT (El más moderno, incluido en la etiqueta <track> de html5)
- XML (Lenguaje de marcas extensible)
- STL (STereo Lithography)

En la actualidad no existe ningún estándar de subtitulación de forma gráfica por lo que en este TFG cogeremos el fichero original .srt y lo traduciremos a un subtitulado pictográfico, realizando cambios en las palabras usadas, eliminando la información que no sea útil y siendo lo más precisos posible. El fichero srt modificado pasará por un programa hecho en java que creará un fichero javascript donde sincronizará cada aparición de los pictogramas mediante la inyección de código html. Es algo que no existe actualmente y por ello se ha llegado a esta solución para poder implementar el subtitulado pictográfico.

Capítulo 3

Desarrollo de las aplicaciones

3.1 Planteamiento inicial

Para comenzar, se propone hacer un reproductor inicial donde se reproduzca un vídeo y se puedan ver los subtítulos clásicos, de texto. Para hacer esto se van a utilizar distintas etiquetas HTML5, el vídeo HTML[12] es un nuevo estándar y no está vinculado a ningún otro formato de vídeo. La adición de un reproductor de vídeo se realiza con una sola linea HTML, en la figura 3.1 se puede ver un ejemplo.

HTML

```
<video src="demo.mp4" controls autoplay >HTML5 Video is required for this example</video>
```

Figura 3.1 Línea de código HTML5 para incluir reproductor

El atributo *src* señala el archivo que se va a reproducir. El atributo *controls* indica al explorador que muestre los controles de reproducción integrados, la función y el aspecto dependerán del explorador utilizado. El atributo autoplay es booleano e indica que el vídeo se reproduzca cuando se cargue. En la tabla 1 se muestran todos los atributos que se pueden usar con vídeo.

Atributo	Descripción
Src	Señala el archivo que se va a reproducir.
controls	Indica al explorador que muestre los controles de reproducción integrados
Poster	La imagen que aparece en el reproductor cuando el vídeo no estaádisponible o no ha comenzado su reproducción

Loop	Atributo booleano que reproduce el vídeo ininterrumpidamente
Muted	Atributo booleano que silencia el vídeo
autoplay	Atributo booleano que reproduce automáticamente el vídeo cuando este se cargue
preload	Atributo booleano que define una clave sobre cuanto almacenar en el buffer
Height	Establece el alto del reproductor de vídeo en pixels
Width	Establece el ancho del reproductor de vídeo en pixels

Tabla 1 Valor de los atributos de la etiqueta vídeo

Los formatos de vídeo soportado por HTML5 son:

- WebM
- Ogg
- MP4 o H.264 (AAC o MP3)
- WAVE PCM

En función del explorador el elemento vídeo selecciona el que puede reproducir. Para añadir subtitulos al reproductor, hay que incluir el elemento *track,* este permite la reproducción de vídeos con subtítulos. El código de ejemplo se puede ver en la figura 3.2.

HTML

```
<video src="video.mp4" controls autoplay loop>
  <track src="en_track.vtt" srclang="en" label="English" kind="subtitles" default>
</video>
```

Figura 3.2 Código HTML de un reproductor con subtítulos

Los diferentes atributos del elemento track se pueden ver en las tablas 2 y 3. En la tabla 2 tenemos las diferentes opciones del elemento *kind.*

Opciones de Kind	Descripción
captions	Define la traducción del diálogo y efectos sonoros
chapters	Define los titulos de los capítulos
descriptions	Define la descripción textual del contenido del

	vídeo
metadata	Define el contenido usado mediante scripts, no visible
subtitles	Define el subtitulado, para mostrarlos en el vídeo

Tabla 2 Valor de las opciones de kind

Para el resto de atributos , se pueden ver en la tabla 3.

Atributo	Descripción
Src	URL del archivo de texto temporizado, acepta vtt o ttml
Srclang	Idioma del archivo de texto temporizado, solo informativo
Label	Etiqueta para identificar el texto temporizado seleccionado
Default	Especifica el elemento de pista predeterminado

Tabla 3: Valor de los atributos de la etiqueta track

3.2 Página web

3.2.1 Primeras soluciones

Una primera idea fue incluir las imágenes en el fichero de subtitulado, ya que la etiqueta <track> puedes incluir diferentes tipos, mediante el atributo *kind*. Ninguno de estos permite meter las imágenes directamente sobre el fichero, ya que este lee el texto sin interpretarlo.

La siguiente idea fue la de crear un slider, usando jQuery (biblioteca de JavaScript), pero no se podía controlar con el vídeo la aparición de las distintas imágenes. Además era un método manual muy costoso.

Otra idea fue la de introducir dos vídeos al mismo tiempo, donde uno reprodujera el vídeo y el segundo fuera representando las imágenes, de modo que si se preparan con anterioridad podían estar perfectamente sincronizados, pero esta solución también presentaba varios problemas, uno de ellos es que tenías que presionar el play al mismo tiempo para que funcionará correctamente, y otro que dejaba inviable esta opción era que a la hora de exportarla a una smart tv no sería posible, ya que no permite

el multriprocesado, no se pueden crear dos instancias del reproductor al mismo tiempo.

Por lo tanto la idea que se utiliza es la de utilizar código javascript para sincronizar el vídeo y el subtitulado, utilizando el tiempo actual del vídeo para saber qué imagen mostrar e inyectar código html para mostrar las imágenes que queramos.

3.2.2 Creación y sincronización del subtítulo

Lo primero es saber en qué tiempo se encuentra el vídeo reproducido actualmente, con este valor podemos crear una serie de condiciones para después saber cuándo va cada una de las imágenes del subtitulado.
El fichero JavaScript comprueba si se han cargado todos los elementos de la página para después añadir el evento "timeupdate" a nuestro reproductor con id="video"y que realice la función de sincronización. El valor del tiempo actual es video.currentTime y hay que multiplicarlo por 1000 para tenerlo en milisegundos. Este evento se activa 4 veces por segundo. Con este valor, se sigue la estructura de código if-then-esle en la cual se van concatenando las condiciones de aparición de imágenes.
En la figura 3.3 tenemos un ejemplo de un fichero de subtitulado .srt.

```
1
00:00:02,020 --> 00:00:04,019
yo soy PEPPA

2
00:00:04,021 --> 00:00:07,019
yo soy hermano GEORGE
```

Figura 3.3 Extracto fichero srt

Es necesario tener el valor inicial y final de la aparición de cada frase del subtítulo, esto se consigue utilizando el método split() el cual divide cada frase en palabras.

Una vez se tienen todos los valores, se pasan a milisegundos y se escriben en el fichero de salida javascript. Finalmente se divide la frase palabras y se colocan dependiendo del número que aparezcan en el contenedor adecuado.

La inserción de código html permite colocar la imagen y el texto en el contenedor apropiado para su correcta visualización. Donde no se quiere que se vea nada se dejan vacíos, para borrar los pictogramas que hayan con anterioridad.
Para no tener que generar este fichero manualmente, se crea un programa en Java en el cual se coge el fichero srt preparado con las palabras que queremos que sean traducidas a imágenes previamente seleccionadas.

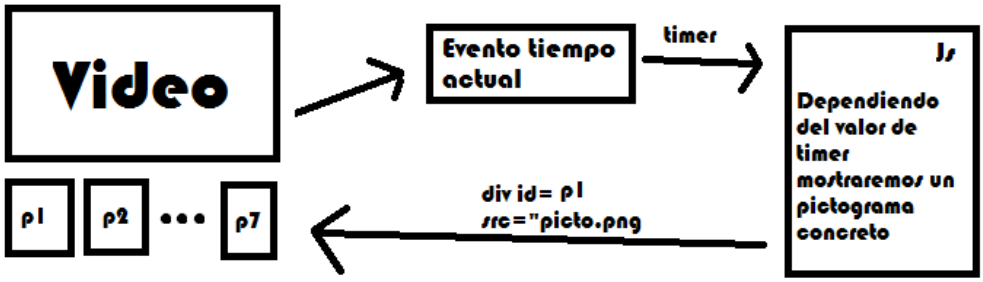

Figura 3.4 Esquema de sincronización

En la figura 3.4 podemos apreciar todo lo comentado anteriormente, donde el evento es el que nos proporciona el valor actual del tiempo de reproducción con el cual comparar.
La herramienta en Java que automatizará el proceso de creación del fichero javascript lo podemos ver el esquema en la figura 3.5. A partir de un fichero srt preparado, se logra crear el fichero javascript de sincronización. Este es el caso del navegador, en el capítulo 3.3 veremos que hay varias diferencias pero la esencia es la misma. Todo el código referente a este apartado se podrá visualizar en el Anexo B**.**

Figura 3.5 Esquema de creación del fichero de sincronización

3.2.3 Posicionamiento del vídeo y de los pictogramas

Las primeras soluciones fueron la de colocar el reproductor de vídeo y debajo un contenedor donde inyectar toda la frase en una única imagen, podemos ver lo que ocurre en la figura 3.6.

Figura 3.6 Ejemplo de frase completa con subtítulo

Esto tiene una serie de inconvenientes, ya que primero tenemos que preparar cada una de esas imágenes con anterioridad, si queremos que sea un proceso automático, tendríamos que diseñar una arquitectura en la cual dependiendo de la frase sacara una imagen distinta que fuera la composición de los pictogramas de cada palabra de la frase. Por otro lado, siempre intentaría llenar todo el contenedor por lo que cuando hubiera pocos pictogramas se distorsionarían. Por lo tanto, esta solución no era válida.

La solución que se llevó a cabo fue realizar una serie de contenedores, uno para cada uno de los pictogramas hasta un máximo de siete. Todos ellos tienen un pequeño contenedor debajo donde aparece el texto que

representa la imagen. Esta solución era la que más se adaptaba a nuestras necesidades y fue la que finalmente se usa. Todo el código html y css se puede consultar en el Anexo B pero en la figura 3.7 tenemos un extracto.

```
<!-- Contenedor Pictogramas -->
<div id="contenedorPictos">
    <div id="Picto11" class = "Pic1"></div>
    <div id="Txt11" class = "Text1"></div>
    <div id="Picto22" class = "Pic2"></div>
    <div id="Txt22" class = "Text2"></div>
    <div id="Picto33" class = "Pic3"></div>
    <div id="Txt33" class = "Text3"></div>
    <div id="Picto44" class = "Pic4"></div>
    <div id="Txt44" class = "Text4"></div>
    <div id="Picto55" class = "Pic5"></div>
    <div id="Txt55" class = "Text5"></div>
    <div id="Picto66" class = "Pic6"></div>
    <div id="Txt66" class = "Text6"></div>
    <div id="Picto77" class = "Pic7"></div>
    <div id="Txt77" class = "Text7"></div>
</div>
```

Figura 3.7 Código ejemplo de los contenedores de los pictogramas

3.2.4 Aspecto final

Una vez la funcionalidad de nuestro reproductor es la correcta, se añade una página de estilos css3 para que la visualización sea lo más cómoda posible. Los tamaños se reajustan dependiendo del soporte electrónico que se utilice.

Se puede acceder a este reproductor desde la dirección:

http://dihana.cps.unizar.es/~eduardo/pictosub/

En la figura 3.8 se puede observar el aspecto de la página web final, en la parte superior aparece el índice de vídeos que contiene la página, en este caso dos. La ideo posterior sería incluir una pestaña de evaluación donde apareciera un cuestionario sobre lo visto en los vídeos. Debajo aparece el reproductor con los controles y debajo los pictogramas, no ocultando en ningún momento el vídeo.

Figura 3.8 Aspecto final aplicación web

3.3 Aplicación para Samsung Smart Tv (PictoAPP)

3.3.1 Creación del proyecto

Para crear la aplicación se necesita un entorno de trabajo en el que poder programar y exportar el proyecto que se va a realizar, por lo que nos descargamos la última versión SDK que nos facilita Samsung para desarrollar aplicaciones : Eclipse SDK TV Samsung 5.1. Este framework se utiliza para crear aplicaciones basadas en web.

Una vez descargado y preparado para su uso, se empieza creando el proyecto, para ello usaremos una base ya creada por el equipo de Samsung donde se puede ver un ejemplo básico de un reproductor con una lista de reproducción, será este el que se usa inicialmente para crear la aplicación ya que implementa el reproductor.

En la figura 3.9 se puede ver el esquema que sigue el framework de Samsung.

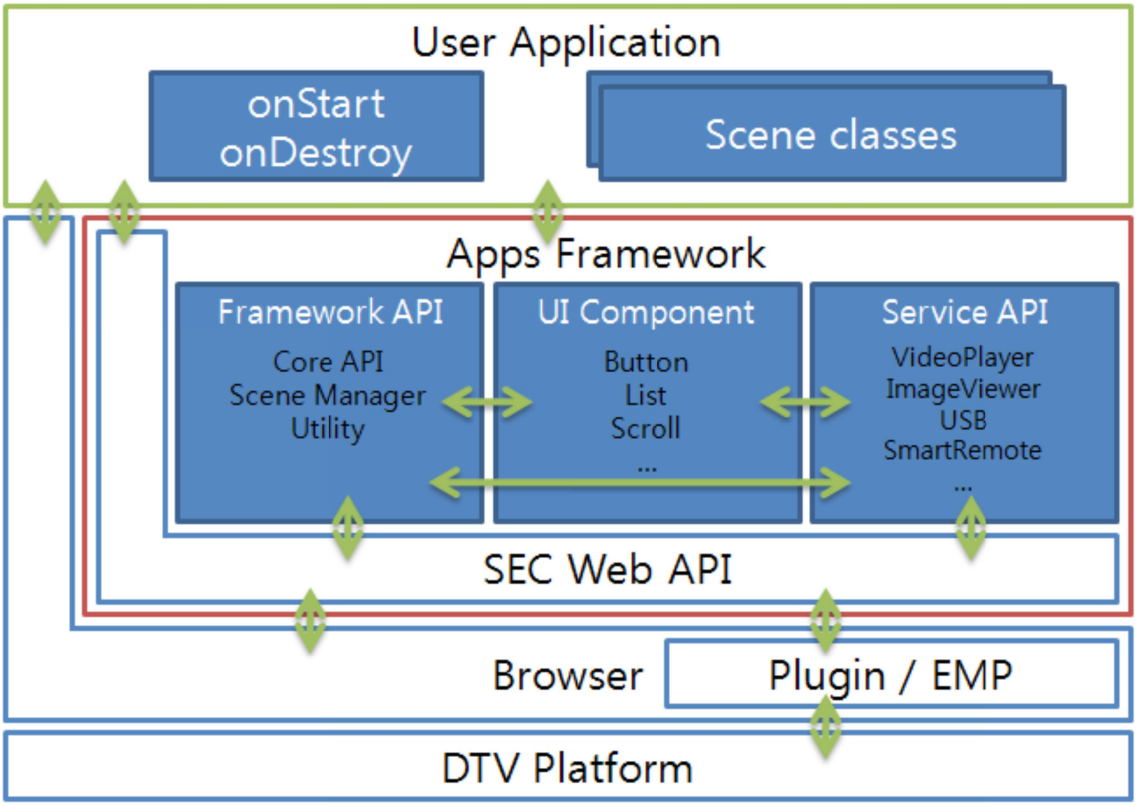

Figura 3.9 Framework Samsung Smart tv

3.3.2 Estructura de la aplicación

La estructura de la aplicación aparece en la figura 3.10 y esta subdivida en varias partes que se detallan en las tablas 4 y 5.

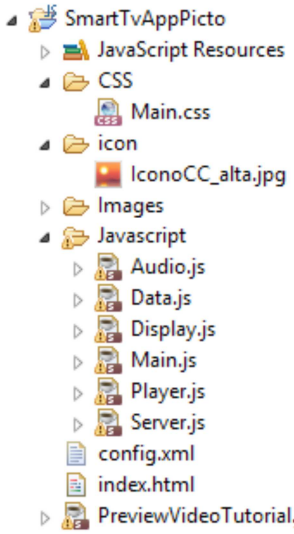

Figura 3.10 Estructura de la aplicación smart tv

Archivos/Carpetas	Descripción
Common/Api	Contiene los módulos comunes
CSS	Contiene los estilos de la aplicación
Images	Contiene las imágenes de la aplicación
Javascript	Contiene los Javascripts de la aplicación
Config.xml	Contiene la información para ejecutar la aplicación
Index.html	Archivo HTML que ejecuta la aplicación

Tabla 4. Estructura de Archivos/Carpetas de la aplicación smart tv

Clase	Descripción
Main	Manejo y coordinación de todos los componentes de la aplicación
Player	Controla la reproducción del audio y vídeo de la aplicación
Audio	Controla el volumen de audio utilizando un plugin
Server	Maneja la recuperación de un feed RSS desde el servidor de datos usando AJAX
Data	Almacenamiento de datos de vídeo dentro de la aplicación
Display	Encargado de presentar la información de la aplicación

Tabla 5. Estructura de clases en la aplicación smart tv

En este apartado se explican los aspectos más importantes de la aplicación y las modificaciones realizadas. En el Anexo C se podrá parte de todo el código utilizado.

En el fichero config.xml tenemos la información de la aplicación y las características que queramos que tenga, por ejemplo resolución, información del autor.

En el fichero html, en la cabecera indexamos todos los ficheros que vamos a usar como webApis, plugins, páginas de estilos... En el body hay insertado el código de nuestra aplicación web, salvo por la etiqueta <video> ya que para la reproducción del vídeo se usan los ficheros javascript.

Toda la información se encuentra alojada en nuestro servidor, y para sincronizarla a la aplicación se usa el controlador server.js. Dentro del código vemos que maneja la recuperación de un feed RSS desde el

servidor de datos usando AJAX, y la url viene de nuestro servidor donde tenemos un fichero videoList.xml donde se encuentra el contenido de nuestra aplicación (nombre, url vídeo, descripción).
Fichero Audio.js y Data.js no se modifican ya que no es necesario.
Main.js se modifica para poder utilizar 2 teclas del mando y poder ocultar y mostrar el índice de los vídeos o la descripción de este. En este caso fueron tvKey.KEY_GUIDE y tvKey.KEY_INFO.
Player.js es modificado para la resolución del vídeo, podemos ver como queda en la figura 3.11, y la función donde se va actualizando el tiempo actual de reproducción, donde incluimos nuestra función de subtitulado, en la figura 3.12 tenemos el extracto del código.

```
Player.setWindow = function()
{
    //this.plugin.Execute("SetDisplayArea",0, 0, 960, 540);
    Player.AVPlayer.setDisplayRect({
        top: 0,
        left: 0,
        width: 960,
        height: 540
    });
};
```

Figura 3.11 Extracto Player.js función setWindow

```
Player.setTotalTime = function()
{
    console.log('setTotalTime : ' + Player.AVPlayer.getDuration());
    Display.setTotalTime(Player.AVPlayer.getDuration());
}
```

Figura 3.12 Extracto Player.js función setTotalTime

Display.js tenemos la disposición de todos los elementos, la actualización del tiempo, cambio de modo ventana o pantalla completa, etc.

3.3.3 Apariencia de la aplicación

Se modifica varias veces la apariencia hasta dar con la fórmula que mejor se adaptar a nuestras necesidades, a continuación se muestran las diferentes apariencias.
Algunas de ellas tienen de colores diferentes cada una de las partes, para poder diferenciarlas de la mejor manera posible, aunque después al exportarla a la televisión se quiten.

La disposición inicial de la aplicación la podemos ver en la figura 3.13

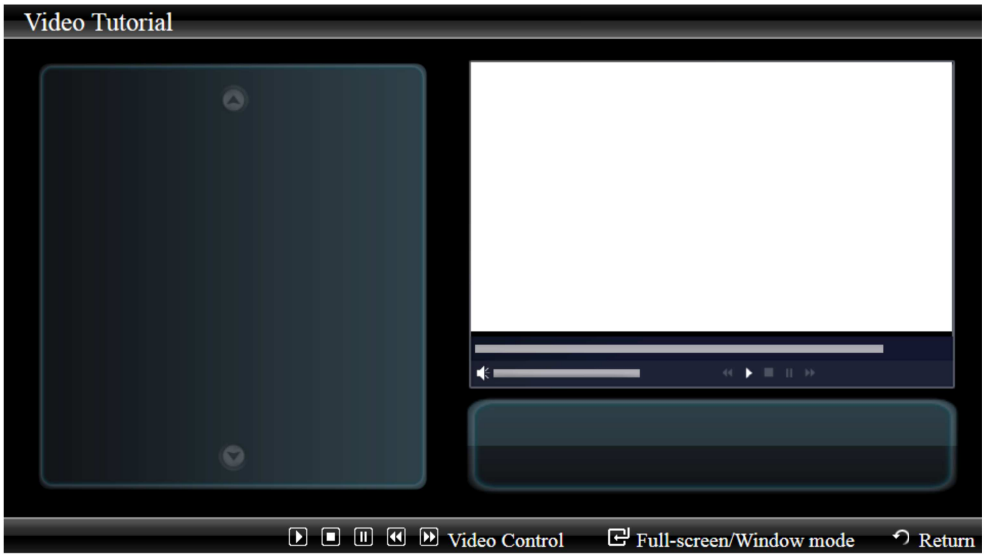

Figura 3.13 Apariencia inicial de la aplicación para smart tv

En la parte derecha aparece el índice de vídeos que se pueden seleccionar, a la izquierda hay dos partes, en una el reproductor y debajo una pequeña descripción. Introducir en esta estructura un subtitulado es casi inviable por problemas de espacio.

La 2ª disposición de la aplicación la podemos ver en la figura 3.14

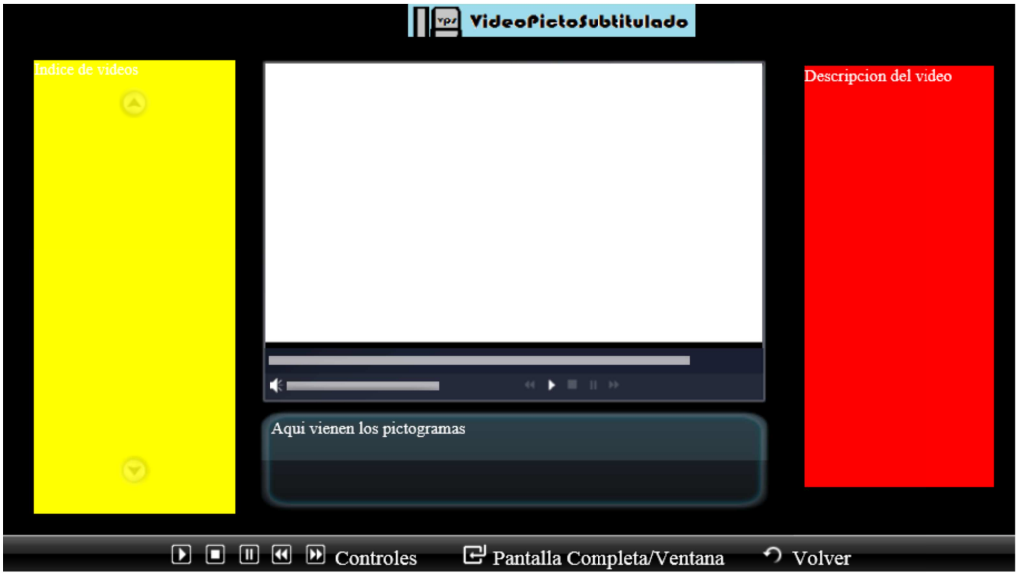

Figura 3.14 2ª disposición de la aplicación smart tv

Aquí centramos el reproductor y colocamos el índice de vídeos a la derecha y la descripción a la derecha, en la parte de abajo iría los pictogramas.

La 3ª disposición de la aplicación la podemos ver en la figura 3.15

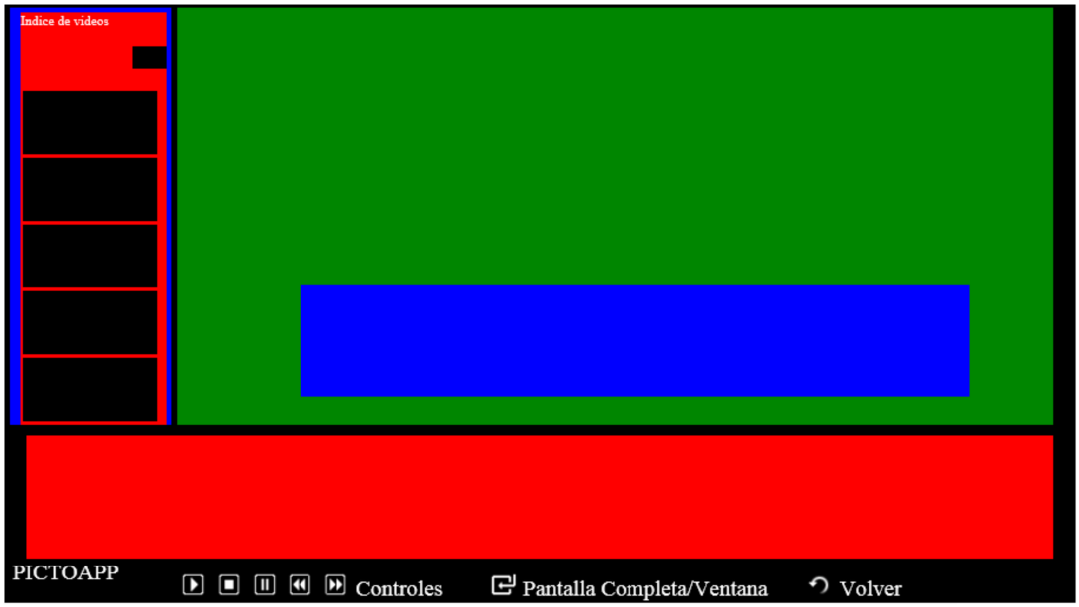

Figura 3.15 3ª Disposición de la aplicación smart tv

En este caso el reproductor ya es de un tamaño considerable y el contenedor azul es el de la descripción, el cual aparece o desaparece si usamos un botón del mando. Toda la parte de abajo es para el subtitulado, pero se puede ver que no está centrado y causa problemas de visualización.

El aspecto final de la aplicación lo podemos ver en la figura 3.16

Figura 3.16 Aspecto final de la aplicación smart tv

En este caso tanto el índice como la descripción del vídeo aparecen superpuestas, pero con el botón de información o de ayuda del mando desaparecen o aparecen. La parte de abajo también desaparece con la del índice para la mejor visualización del vídeo. Los subtítulos aparecen en un tamaño lo suficientemente grande para poder verse sin problemas pero sin llevar a molestar en la visualización de los vídeos.

3.3.4 Servidor Apache y exportando la aplicación

Para poder exportar la aplicación a un televisor Samsung, es necesario crear un servidor desde el cual conectarse desde la televisión. En este caso se ha utilizado un panel de control muy utilizado para entornos de red local XAMPP, donde instalado ya tienes un servidor apache configurado, servidor ftp, etc. Este apartado se verá en profundidad en el Anexo D.1

Una vez está instalado, para poder exportar la aplicación tenemos que configurar en eclipse la ip del servidor para que la televisión sepa donde se tiene que conectar y que tiene que descargar. Una vez exportada correctamente, nos conectamos desde el televisor al servidor y sincronizamos, se instalará nuestra aplicación y estará lista para usarse, todos los pasos están explicados en el anexo D.1.

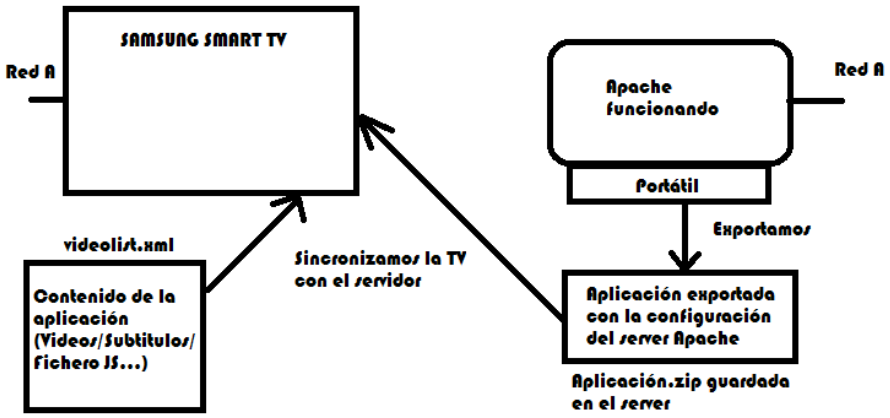

Figura 3.17 Esquema de sincronización con el servidor Apache

En el esquema de la figura 3.17 tenemos todo lo mencionado anteriormente, una vez exportada correctamente la aplicación al servidor Apache, este lo arrancamos y esperamos a que la TV se sincronize con él.

Cápitulo 4

Posible utilidad y líneas futuras

4.1 Conclusiones

Estoy muy satisfecho con la calidad del trabajo realizado, han sido muchísimas las modificaciones y pruebas que se han realizado hasta llegar a los productos finales, probando con todas las combinaciones que nos iban surgiendo, para al final llegar a un producto lo más agradable posible, y a la vez sencillo de entender.
En este TFG, a falta de un estándar para el subtitulado gráfico, se ha creado un nuevo formato que facilita tanto la conversión de formatos estándar de texto como la posterior visualización tanto en SmarTV como en entorno web. El proceso de creación del subtitulado se ha automatizado lo máximo posible para que simplemente ejecutando un par de programas en java tengamos todo lo necesario para poder ver el contenido que queramos subtitulado con pictogramas.

Espero que más adelante pueda seguir trabajando en el tema del subtitulado y ver si se puede ayudar a un gran número de personas con este tipo de herramientas. Algunos ejemplos por los que se puede avanzar es el estudio con eye tracking de la posición optima de los pictogramas en el reproductor, o el estudio de la carga cognitiva para vídeos de estas características.

La mayor utilidad que puede tener nuestras aplicaciones es a la hora de la enseñanza, para que sea un refuerzo para las personas con discapacidades de comunicación y puedan relacionar mucho antes los significados de los pictogramas. El mayor problema que tienen es que aportan más información de la que ya te da el vídeo que estas visualizando, por eso hemos tenido que crear unas reglas para no sobrepasar la información cognitiva que aparece. Por ejemplo el número óptimo de pictogramas que aparezcan por frase serían hasta 3 al mismo

tiempo, y que aparezcan por lo menos 2/3 segundos por imagen para poder comprenderlo. También existe el problema de subtitular dibujos animados, ya que ellos ya están esquematizados y puede llevar a confusión, las pruebas realizadas han sido en distintos tipos de vídeos y para diferentes edades, desde peppa-pig hasta tráilers de películas.

Actualmente las herramientas de creación de ficheros de subtitulado gráfico están siendo utilizadas para la creación de material didáctico, para mejorar el uso de los pictogramas mediante el subtitulado de vídeos y un cuestionario posterior, acelerando el proceso de familiarización con estos elementos de comunicación.

4.2 Mejoras y lineas futuras

Las principales mejoras que pueden tener las aplicaciones creadas son en el apartado del contenido, ya que este subtitulado está pensado para ser usado en televisión o cine, y que sean los gestores de contenido los que proporcionen un subtitulado pictográfico donde cada persona desde su casa pueda activarlos y visualizarlos, como ocurre con el lenguaje de signos. Otro apartado a desarrollar sería la conversión automática de un fichero original srt a uno que utilice lenguaje pictográfico.
En este TFG se ha utilizado una herramienta de lematización del texto y después se trata la salida para tener un fichero compatible, pero esto se podría desarrollar más a fondo, usando mejores reglas y perfeccionando los pictogramas a usar dependiendo del contexto en el que se realiza la comunicación. Finalmente la posición horizontal de los pictogramas puede ser un problema si distrae demasiado la atención y sería una excelente idea investigar una mejor posición para el subtitulado donde podamos tener un refuerzo a lo que estamos viendo pero sin que sea una molestia.
Existe trabajo por hacer pero con este TFG se ha creado una base desde la cual trabajar y poder avanzar.

Bibliografía

[1]Lenguaje HTML5 disponible en http://www.w3schools.com/html/default.asp, fecha de consulta: 24 de noviembre de 2014
[2] Lenguaje CSS3 disponible en http://www.w3schools.com/css/default.asp y http://librosweb.es/, fecha de consulta: 26 de noviembre de 2014
[3] Lenguaje Javascript disponible en http://www.w3schools.com/js/default.asp, fecha de consulta: 27 de noviembre de 2014
[4] Lenguaje java disponible en http://chuwiki.chuidiang.org/index.php?title=Lectura_y_Escritura_de_Ficheros_en_Java, fecha de consulta: 16 de febrero 2015
[5] SDK Eclipse Samsung smart tv descargado en http://www.samsungdforum.com/, fecha de consulta: 15 de diciembre de 2014
[6] Información sobre el fucionamiento de samsung smart tv http://www.samsungdforum.com/SamsungDForum/ForumDashBoard/c305cb96c72a9e5b, fecha de consulta: 12 de enero de 2015
[7]XAMPP descargado en https://www.apachefriends.org/es/index.html, fecha de consulta: 2 de febrero de 2015
[8]Eclipse descargado en https://eclipse.org/, fecha de consulta: 20 de febrero de 2015
[9]Brackets descargado en http://brackets.io/, fecha de consulta: 1 de diciembre de 2014
[10]Documentación aplicación smart tv en https://samsungstad.com/, fecha de consulta: 14 de enero de 2015 (Actualmente no disponible)
[11]Desarrollo de aplicación smart tv e http://developer.samsung.com/, fecha de consulta: 15 de enero de 2015
[12]Información sobre eventos para programación de la web : https://msdn.microsoft.com/es-es/default.aspx, fecha de consulta: 9 de enero de 2015
[13]Pictogramas descargados de www.catedu.es/arasaac/, fecha de consulta: 2 de febrero de 2015
[14]Información y descarga de pictogramas arasuite.proyectotico.es/index.php, fecha de consulta: 3 de febrero de 2015

[15]Comunicación aumentativa y alternativa : guía de referencia / Dolores Abril Abadín, Clara I. Delgado Santos, Ángela Vigara Cerrato, 1ª edición
[16]Comunicación sin habla: comunicación aumentativa y alternativa alrededor del mundo / Anne Warrick 1ª edición
[17]Sistemas alternativos de comunicación / María Sotillo (coord.) ; [prólogo, Angel Rivière ; 3ª edición.
[18] Especificaciones de televisores LG en http://developer.lge.com/resource/tv/RetrieveDevPlatformList.dev, fecha de consulta: 9 de diciembre de 2015
[19] Especificaciones de televisores Samsung en http://www.samsungdforum.com/Devtools/Spec, fecha de consulta: 8 de diciembre de 2015
[20] Estudio de ventas en https://www.strategyanalytics.com/, fecha de consulta: 4 de mayo de 2015
[21] Descarga de Freeling http://nlp.lsi.upc.edu/freeling/, fecha de consulta: 6 de abril de 2015
[22] Lectura Fácil Métodos de Redacción y Evaluación de Oscar García Muñoz, 1ª edición: noviembre 2012. Editado por Real Patronato sobre Discapacidad y Ministerio de Sanidad, Servicios Sociales e Igualdad

ANEXO A

Elección de la plataforma

A.1 Plataformas existentes

En la actualidad, existen varias plataformas para el desarrollo de aplicaciones smart tv, en este apartado se van a ver las caracteristicas de algunas de ellas. Las principales marcas de televisiones por ventas son:

- Samsung
- LG
- Sony

Según un estudio de Strategy Analytics, Samsung lidera el mercado con un 26% del segmento de televisores inteligentes, seguido de LG por 16% y Sony con un 11%.

Las especificaciones funcionales de cada uno vienen en las siguientes tablas:

Type	Feature		TV/AV	SDK 5.1
App Engine	HTML		HTML5	HTML5
	DOM		DOM 3	DOM 3
	CSS		CSS3	CSS3
	Javascript		JSC	JSC
Flash	SWF		Flash 11.1 / ActionScript 3.0	No Soportado
	streaming		RTMP/RTMPe	No Soportado
DRM	PlayReady		Supported	No Soportado
		HTTP	Soportado	Soportado
		HTTPS		
		MMS*2		

VOD	Streaming	RTP/RTSP		
	Adaptative streaming	HAS LS	No Soportado	No Soportado
		MPEG-DASH	Soportado	Soportado
		HLS LS	Soportado	Soportado
Live Streaming	Adaptative Streaming		Soportado (HLS LS y WidevineLS)	Soportado (HLS LS)

Tabla 6 : Especificaciones Samsung Smart tv

Se puede observar que en la tabla de especificaciones de Samsung, se cumplen todos los requisitos necesarios para la realización del proyecto.

Type	Feature		NetCast 4.5 LM14 (2015)
App Engine	HTML		HTML5 Soportado parcialmente
	DOM		Soportado
	CSS		CSS3 Soportado parcialmente
	Javascript		SquirrelFish Extreme
Formatos soportados	JPEG / PNG		Soportado
	GIF		Soportado
DRM	PlayReady		Play Ready v2 Soportado
VOD	Streaming	HTTP	Soportado
		HTTPS	
		MMS*2	
		RTP/RTSP	
		HAS LS	No Soportado
		MPEG-DASH	Soportado

	Adaptative streaming	HLS LS	Soportado
Live Streaming	Adaptative Streaming		Soportado (HLS LS y WidevineLS)

Tabla 7 : Especificaciones LG

Cubre todas las necesidades para la realizació de este proyecto.

Type	Feature		NetCast 4.5 LM14 (2015)
App Engine	HTML		HTML5 Soportado
	DOM		Soportado
	CSS		CSS3 Soportado
	Javascript		SquirrelFish Extreme
Flash	SWF		Soportado
	Streaming		Soportado
DRM	PlayReady		No Soportado
VOD	Streaming	HTTP	Soportado
		HTTPS	
	Adaptative streaming	HAS LS	No Soportado
		MPEG-DASH	Soportado
		HLS LS	Soportado
Live Streaming	Adaptative Streaming		Soportado (HLS LS)

Tabla 8 : Especificaciones Sony internet tv

Con la alianza de Google con Sony ha sido posible la implementación del sistema operativo android en los televisores para satisfacer las exigencias para el desarrollo de aplicaciones.

A.2 Plataforma elegida

La elección final ha sido la de Samsung Smart tv por su madurez en el entorno de desarrollo de aplicaciones para smart tv. Cuenta con un SDK propio integrado en Eclipse el cuál ha ido mejorando en las distintas versiones desde hace 3 años. Actualmente se encuentra en la versión SDK Samsung smart tv 5.1 que es la que se usa en este proyecto. Samsung cubre el 100% de nuestras necesidades para la realización de la aplicación y nos da una mayor flexibilidad a la hora de realizarla y al ser la líder en ventas podría llegar al mayor número de personas.

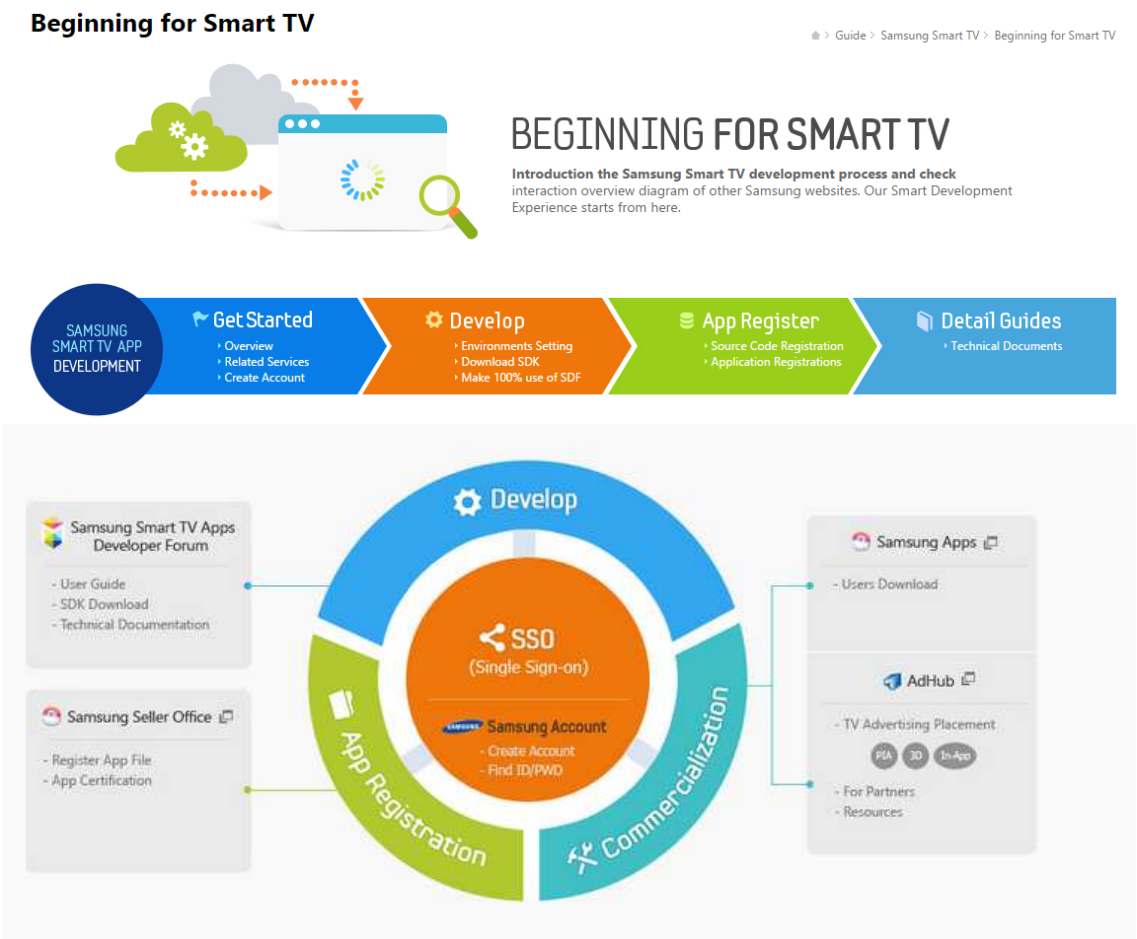

Figura A.1 Samsung smart tv developers

ANEXO B

Sincronización del subtitulado

B.1 Automatización en la generación del fichero srt

En este apartado veremos el código java donde se automatiza el proceso de modificación de un fichero de subtitulado convencional, con formato srt, a uno con el mismo formato pero que podamos usar para poder subtitular mediante pictogramas.
El código inicial pertenece al primer paso, coger el fichero original y seleccionar la parte que nos interesa, y codificar el fin de línea para tener la transformación que aparece en las figuras B.1 y B.2

```
1
00:00:02,020 --> 00:00:04,019
yo soy PEPPA

2
00:00:04,021 --> 00:00:07,019
yo soy hermano GEORGE
```

Figura B.1 : Fichero original .srt

```
yo soy PEPPA fl
soy hermano GEORGE fl
ser MAMAPIG ser PAPAPIG fl
PEPPA fl
```

Figura B.2 Fichero .srt modificado

Con este fichero es más fácil trabajar a la hora de procesarlo, ya que solo es necesario ir leyendo linea por linea y tenemos la codificación de fin de linea para saber cuando acaba una frase y empieza otra. Para lograr esto se ha usado el esquema de la figura B.3

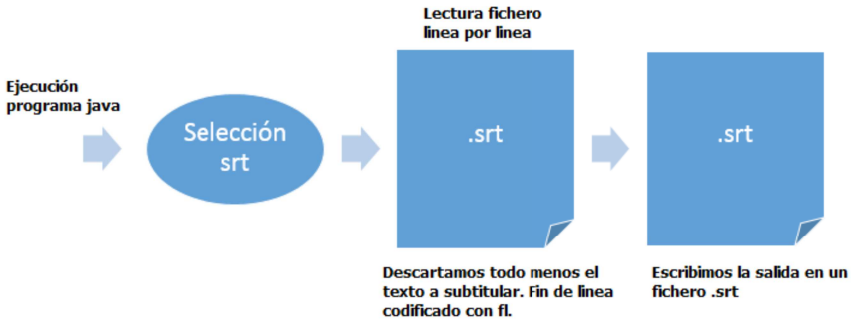

Figura B.3 : Esquema de modificación inicial

Al ejecutarlo nos pedirá que escojamos un fichero, este será el fichero de subtitulado original que queremos modificar para que después sea más sencillo su tratamiento. La salida la guarda en un fichero srt en la ruta del programa freeling que se usará para la lematización del texto. Esta herramienta pone a disposición los resultados de la investigación del lenguaje natural UPC.
Al usar la herramienta freeling, en nuestro caso freeling-3.1, le pasamos el texto a codificar. Usamos el siguiente comando desde la shell:

analyzer.exe -f %FREELINGSHARE%\config\es.cfg < PeppaSub_Mod.srt > subtituladocodif.srt

donde la variable de entorno %FREELINGSHARE% equivale a la ruta donde tengamos instalado el freeling.
La salida que se obtiene despues de usar freeling la podemos ver en la figura B.4.

```
yo yo PP1CSN00 1
soy ser VSIP1S0 1
PEPPA peppa NP00000 1
fl fl AQ0CS0 0.587402
soy ser VSIP1S0 1
hermano hermano NCMS000 0.990196
GEORGE george NP00000 1
fl fl AQ0CS0 0.587402
```

Figura B.4 : Fichero lematizado

Vemos que cada palabra tiene su codificación, estas se usan para saber que palabras queremos en el subtitulado y cuáles no, desde un fichero de

reglas. En nuestro caso hemos usado un fichero de reglas que se puede ver en la figura B.5 y las que coincidan saldrán en el subtítulo.

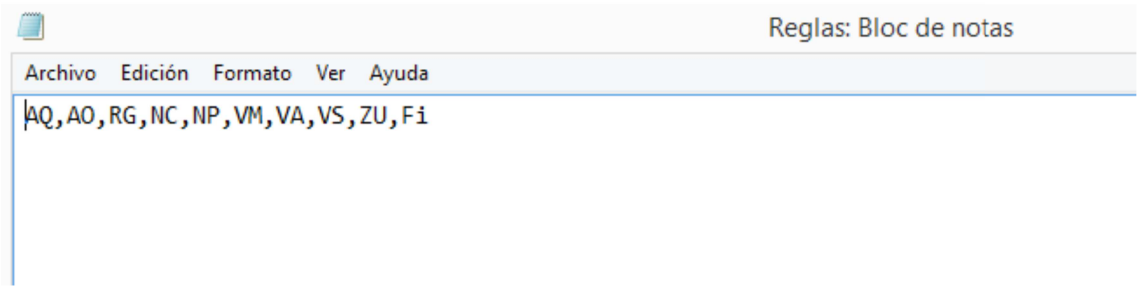

Figura B.5 : Fichero de reglas

Las posibles reglas a usar vienen determinadas en el manual de freeling, en la figura B.6 aparecen resumidas.

```
Las siguientes codificaciones con sus explicaciones:

A para adjetivos                                    C para conjunciones
        Q Calificativo                                      C Coordinada
        O Ordinal                                           S Subordinada
R para adverbios                                    I para interjecciones
        G General                                   S para preposiciones
        N Negativos                                         P Preposiciones
D para determinantes                                F para signos de puntuación
        D Demostrativos                             Z para numerales
        P Posesivo                                          d partitivo
        T Interrogativo                                     m Moneda
        E Exclamativo                                       p porcentaje
        I Indefinido                                        u unidad
        A Articulo                                  W para fecha/hora
N para nombres == lema en singular
        C Común
        P Propio
V para verbos == Siempre el lema sera el infinitivo
        M Principal
        A Auxiliar
        S Semiauxiliar
P para pronombres
        P Personal
        D Demostrativo
        X Posesivo
        I Indefinido
        T Interrogativo
        R Relativo
        E Exclamativo
```

Figura B.6 : Codificación de las reglas

Con el fichero de salida de freeling, la herramienta nos permite crear automáticamente el fichero javascript que nos permite subtitular. El proceso que sigue para crearlo se puede ver en la figura B.7.

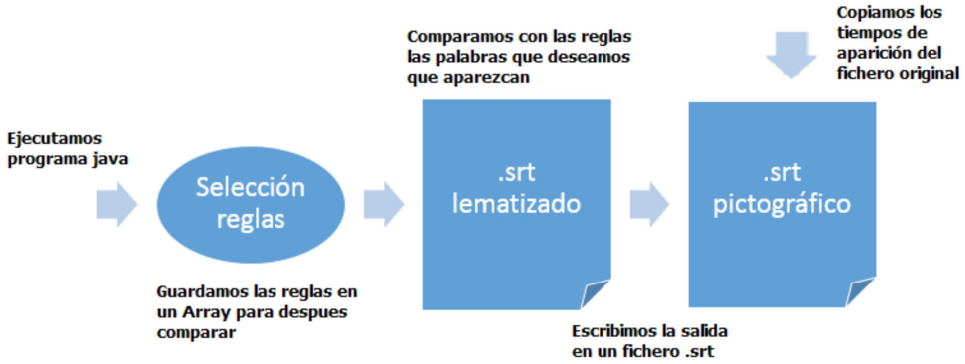

Figura B.7 : Esquema creación subtitulado pictográfico

Vemos en la figura B.7 que lo primero nos pide que le pasemos el fichero de reglas que va a usar el programa, después va creando un nuevo fichero con la misma codificación que la del srt anterior, pero con las palabras ya cambiadas, se tiene una salida intermedia la tenemos en la figura B.8

PeppaSubPreFinal: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
ser peppa
ser hermano george
ser mamapig ser papapig
peppa
charco barro
```

Figura B.8 : Ejemplo fichero intermedio

Con el fichero de la figura B.8, se abre el original y cambia las frases de subtitulado por las del fichero, consiguiendo así la solución deseada, en la figura B.9 podemos ver la salida final.

```
1
00:00:02,020 --> 00:00:04,019
ser peppa

2
00:00:04,021 --> 00:00:07,019
ser hermano george
```

Figura B.9 : Salida final del fichero .srt

B.2 Manual de uso para generar el fichero srt

Una vez se tienen instaladas las herramientas Eclipse y Freeling-3.1. Los pasos a seguir son los siguientes:

- Ejecutar el programa SubtModIni.java
- Realizar el siguiente comando en la consola de nuestro ordenador, hay que realizarlo en la ruta donde este freeling-3.1/bin

analyzer.exe -f %FREELINGSHARE%\config\es.cfg < PeppaSub_Mod.srt > subtituladocodif.srt

- Ejecutar el programa Codiftsrt.java

Con esto se ha convertido el fichero srt original en uno modificado para la subtitulación mediante pictogramas.

B.3 Automatización en la generación del fichero js

Se subtitula el vídeo usando un fichero javascript que sincroniza el tiempo de reproducción con la inyección de código HTML. Para generar este fichero se necesita un proceso de automatización, ya que crear un fichero para cada vídeo resulta inviable. Se ha creado un programa en java que realiza este proceso de procesado de texto para lograr tener el fichero deseado, hay diferencias entre si es para el navegador o para la smart tv.

ENTORNO WEB:

Este es un extracto del código del programa, donde se ve como calcula el tiempo inicial y final en milisegundos a partir del fichero

```
pw.write(" ( fTime > ");
// Leo linea de tiempo
strLinea = buffer.readLine();
// Separo tiempo inicia y final ( Descarto flecha )
tiempos = strLinea.split(" ");
// Separo cada dígito.
t_ini = tiempos[0].split(":");
t_fin = tiempos[2].split(":");
// Separo segundos y milisegundos
s_ini = t_ini[2].split(",");
s_fin = t_fin[2].split(",");
// Opero para tener todo en milisegundos y poder comparar con el tiempo
actual.
```

```
time_ini = Integer.parseInt(s_ini[1]) + (Integer.parseInt(s_ini[0]) * 1000) +
(Integer.parseInt(t_ini[1])   *   60000)   +   ((Integer.parseInt(t_ini[0])   *
1440000));
time_fin = Integer.parseInt(s_fin[1]) + (Integer.parseInt(s_fin[0]) * 1000) +
(Integer.parseInt(t_fin[1])   *   60000)   +   ((Integer.parseInt(t_fin[0])   *
1440000));
pw.write(time_ini + " ) && ( fTime < " + time_fin + " )){\n");
```

Sigue el esquema de la figura B.10.

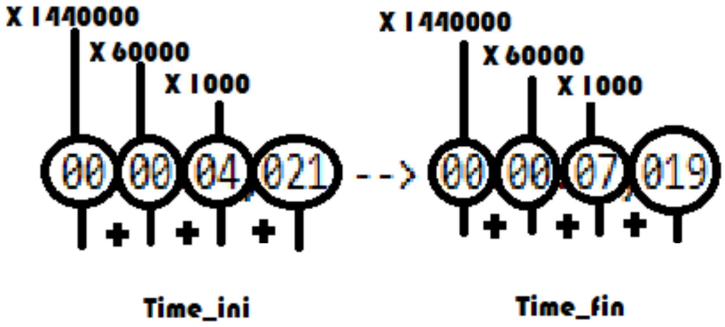

Figura B.10 : Esquema de obtención valores

Por otro lado tenemos la parte de la inyección de código HTML, lo realizamos mediante el metodo .innerHTML que permite, a través del identificador, colocar cualquier tipo de código en ese contenedor, este extracto es para cuando solo tenemos un pictograma a representar.

```
strLinea = buffer.readLine();
// Leo pictos y dependiendo del numero los coloco de una manera.
pictos = strLinea.split(" ");
if (pictos.length == 1){
pw.write("document.getElementById(\"Picto4\").innerHTML='<img
src=\"Pictopng/"+pictos[0]+".png\"height=\"100%\"width=\"100%\"></img>';\n");
pw.write("document.getElementById(\"Txt4\").innerHTML      =      \""      +
pictos[0].toUpperCase() + "\";\n");
pw.write("      document.getElementById(\"Picto1\").innerHTML = \"\";\n");
pw.write("      document.getElementById(\"Picto2\").innerHTML = \"\";\n");
pw.write("      document.getElementById(\"Picto3\").innerHTML = \"\";\n");
pw.write("      document.getElementById(\"Picto5\").innerHTML = \"\";\n");
pw.write("      document.getElementById(\"Picto6\").innerHTML = \"\";\n");
pw.write("      document.getElementById(\"Picto7\").innerHTML = \"\";\n");
pw.write("      document.getElementById(\"Txt1\").innerHTML = \"\";\n");
pw.write("      document.getElementById(\"Txt2\").innerHTML = \"\";\n");
pw.write("      document.getElementById(\"Txt3\").innerHTML = \"\";\n");
pw.write("      document.getElementById(\"Txt5\").innerHTML = \"\";\n");
pw.write("      document.getElementById(\"Txt6\").innerHTML = \"\";\n");
pw.write("      document.getElementById(\"Txt7\").innerHTML = \"\";\n");
pw.write(" }else if (");}
```

Este programa crea un fichero y escribe el código necesario para la sincronización, dependiendo del número de palabras que aparecen

colocará los pictos en unas posiciones determinadas. Un fragmento de la salida es lo siguiente:

```
function init() {
 var video = document.getElementById("video");

//  display the current and remaining times
 video.addEventListener("timeupdate", function () {
//  Current time   var vTime = video.currentTime;
var fTime = 1000 * vTime;


 if( ( fTime > 2020 ) && ( fTime < 4019 )){
      document.getElementById("Picto3").innerHTML = '<img
src="Pictopng/yo.png" height ="100%" width="100%"></img>';
      document.getElementById("Picto4").innerHTML = '<img
src="Pictopng/ser.png" height ="100%" width="100%"></img>';
      document.getElementById("Picto5").innerHTML = '<img
src="Pictopng/PEPPA.png" height ="100%" width="100%"></img>';
      document.getElementById("Txt3").innerHTML = "YO";
      document.getElementById("Txt4").innerHTML = "SER";
      document.getElementById("Txt5").innerHTML = "PEPPA";
      document.getElementById("Picto1").innerHTML = "";
      document.getElementById("Picto2").innerHTML = "";
      document.getElementById("Picto6").innerHTML = "";
      document.getElementById("Picto7").innerHTML = "";
      document.getElementById("Txt1").innerHTML = "";
      document.getElementById("Txt2").innerHTML = "";
      document.getElementById("Txt6").innerHTML = "";
      document.getElementById("Txt7").innerHTML = "";
 }else if ( ( fTime > 4021 ) && ( fTime < 7019 )){
      document.getElementById("Picto3").innerHTML = '<img
src="Pictopng/ser.png" height ="100%" width="100%"></img>';
      document.getElementById("Picto4").innerHTML = '<img
src="Pictopng/hermano.png" height ="100%" width="100%"></img>';
      document.getElementById("Picto5").innerHTML = '<img
src="Pictopng/GEORGE.png" height ="100%" width="100%"></img>';
      document.getElementById("Txt3").innerHTML = "SER";
      document.getElementById("Txt4").innerHTML = "HERMANO";
      document.getElementById("Txt5").innerHTML = "GEORGE";
      document.getElementById("Picto1").innerHTML = "";
      document.getElementById("Picto2").innerHTML = "";
      document.getElementById("Picto6").innerHTML = "";
      document.getElementById("Picto7").innerHTML = "";
```

[…]

Vemos como este fichero recoje mediante el evento timeudpate el tiempo actual del vídeo este se usa para comparar con los tiempos del subtitulado. El programa anterior ha pasado todo a milisegundos para tener los rangos más precisos posibles. El método para inyectar código HTML es innerHTML que permite hacer lo que necesitamos.
Para el caso de la smart tv se genera un fichero javascript que recoje el subtitulado de todos los vídeos que contenga el servidor. Además inyecta

el código HTML usando el evento widgetapi.putInnerHTML que nos permite hacer lo que deseamos. A continuación se muestra un extracto del código del programa en java.

SMART TV:

```
while ((strLinea = buffer.readLine()) != null)   {
	// Escribo capitulo actual
	pw.write(" (cap == \'" + ficherosSubs[i] +"\')");
	strLinea = buffer.readLine();
	// Leo pictos y dependiendo del numero los coloco de una manera.
	pictos = strLinea.split(" ");
	if (pictos.length == 1){;
		pw.write("   img1 = '<img
	src=\"http://155.210.156.59/App_Demo/Pictopng/" + pictos[0] + ".png\"
	height =\"100\" width=\"125\"></img>';\n");
		pw.write("   widgetAPI.putInnerHTML(picto1, negro);\n");
		pw.write("   widgetAPI.putInnerHTML(picto2, negro);\n");
		pw.write("   widgetAPI.putInnerHTML(picto3, negro);\n");
		pw.write("   widgetAPI.putInnerHTML(picto4, img1);\n");
		pw.write("   widgetAPI.putInnerHTML(picto5, negro);\n");
		pw.write("   widgetAPI.putInnerHTML(picto6, negro);\n");
		pw.write("   widgetAPI.putInnerHTML(picto7, negro);\n");
		pw.write("   widgetAPI.putInnerHTML(Text1, negro);\n");
		pw.write("   widgetAPI.putInnerHTML(Text2, negro);\n");
		pw.write("   widgetAPI.putInnerHTML(Text3, negro);\n");
		pw.write("   widgetAPI.putInnerHTML(Text4,\"" +
		pictos[0].toUpperCase() + "\");\n");
		pw.write("   widgetAPI.putInnerHTML(Text5, negro);\n");
		pw.write("   widgetAPI.putInnerHTML(Text6, negro);\n");
		pw.write("   widgetAPI.putInnerHTML(Text7, negro);\n");
		pw.write(" }else if (");
	}
```

Podemos ver la gran similitud con el programa anterior, en el extracto vemos el ejemplo cuando solo tenemos un pictograma a mostrar.
La salida de esta herramienta nos dará el fichero javascript deseado, se puede ver el extracto del código en la figura B.11.

Es solo una muestra ya que la extensión del fichero de la demo llega a las 2978 lineas.

```
 if( (cap == 'ElFrancotirador') && (( time > 8020 ) && ( time < 20019 ))){
      img1  =  '<img  src="http://155.210.156.59/App_Demo/Pictopng/SONIDO.png"
height ="100" width="125"></img>';
      img2  =  '<img  src="http://155.210.156.59/App_Demo/Pictopng/CAMIÓN.png"
height ="100" width="125"></img>';
      widgetAPI.putInnerHTML(picto1, negro);
      widgetAPI.putInnerHTML(picto2, negro);
      widgetAPI.putInnerHTML(picto3, img1);
      widgetAPI.putInnerHTML(picto4, negro);
      widgetAPI.putInnerHTML(picto5, img2);
      widgetAPI.putInnerHTML(picto6, negro);
      widgetAPI.putInnerHTML(picto7, negro);
      widgetAPI.putInnerHTML(Text1, negro);
      widgetAPI.putInnerHTML(Text2, negro);
      widgetAPI.putInnerHTML(Text3, "SONIDO");
      widgetAPI.putInnerHTML(Text4, negro);
      widgetAPI.putInnerHTML(Text5, "CAMIÓN");
      widgetAPI.putInnerHTML(Text6, negro);
      widgetAPI.putInnerHTML(Text7, negro);
 }else if ( (cap == 'ElFrancotirador') && (( time > 20020 ) && ( time < 26019
))){
      img1  =  '<img  src="http://155.210.156.59/App_Demo/Pictopng/SONIDO.png"
height ="100" width="125"></img>';
      img2 = '<img src="http://155.210.156.59/App_Demo/Pictopng/RESPIRAR.png"
height ="100" width="125"></img>';
      widgetAPI.putInnerHTML(picto1, negro);
      widgetAPI.putInnerHTML(picto2, negro);
      widgetAPI.putInnerHTML(picto3, img1);
      widgetAPI.putInnerHTML(picto4, negro);
      widgetAPI.putInnerHTML(picto5, img2);
```

Figura B.11 : Extracto código javascript smart tv

B.4 Automatización en la generación del fichero js

En este caso el manual es bastante simple, para el entorno de la smart tv, los ficheros de vídeo y subtitulos tienen que tener el mismo nombre, guardarse en la ruta corresponiente a la que apunta el programa java (en este caso el servidor) y ejecutarlo.

Para el caso de la web, realiza el subtitulado vídeo por vídeo, generando un javascript para cada uno de ellos, ya que el nombre del vídeo aparece con la ruta y para hacerlo compatible con cualquier equipo habría que comparar esas rutas para cada equipo.
Por lo que en este caso simplemente elegir el fichero srt para generarle su javascript automáticamente.

ANEXO C

SDK Samsung Smart TV

C.1 Especificaciones Hardware

Ordenador/Portatil:

- Processor: Dual Core 1.5GHz / Single Core 3GHz or higher
- RAM: 2GB o superior
- OS(32bit and 64bit supported):
  - **Windows**: (Recomendado),WindowsXP Service Pack 2 o superior
  - **MacOS X**: Intel-based hardware, OS versions 10.6 and above.
  - **Linux**: Versión de Linux que soporte VirtualBox 2.4.2
- Resolución de pantalla : 1280 x 1024 o superior
- HDD: 5GB o superior

Smart TV:

En el caso de este TFG hemos utilizado el modelo de televisión samsung UE55F6400AW con las siguientes especificaciones:

- Televisor 139,7 cm (55'')
- Full HD
- 1920x1080 pixeles de máxima resolución
- Analógico y digital
- 20 W
- Eficiencia energética +A

C.2 Implementación de la aplicación

En este anexo vamos a ver el esquema de los subsistemas que tiene la aplicación para smart tv y la explicación de los métodos usados.
En la figura C.1 podemos ver un esquema de funcionamiento de toda la aplicación.

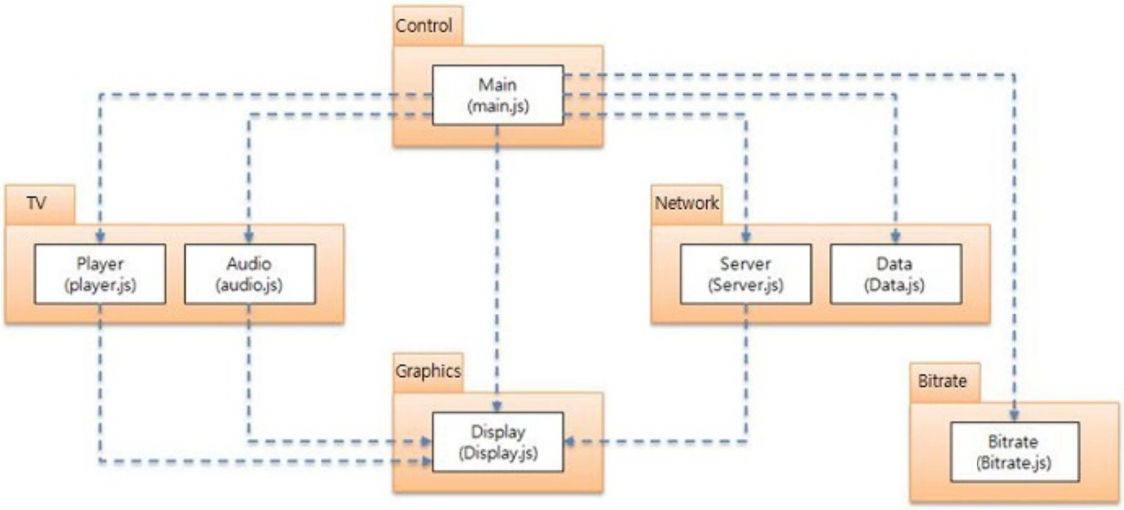

Figura C.1 : Estructura de los ficheros javascript en la aplicación smart tv

La configuración de la aplicación viene en el fichero config.xml, el cual puede verse en la figura C.2.

config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<widget xmlns="http://www.samsung.com/">
    <previewjs>PreviewVideoTutorial</previewjs>
    <type>user</type>
    <cpname/>
    <cplogo/>
    <cpauthjs/>
      <ThumbIcon>icon/IconoCC_alta.jpg</ThumbIcon>
    <BigThumbIcon>icon/IconoCC_alta.jpg</BigThumbIcon>
    <ListIcon>icon/IconoCC_alta.jpg</ListIcon>
    <BigListIcon>icon/IconoCC_alta.jpg</BigListIcon>
    <ver>1.1</ver>
    <mgrver/>
    <fullwidget>y</fullwidget>
    <movie>y</movie>
    <srcctl>y</srcctl>
    <ticker>n</ticker>
    <childlock>n</childlock>
    <audiomute>n</audiomute>
    <videomute>n</videomute>
    <dcont>y</dcont>
    <widgetname>Pictosub</widgetname>
    <description>Aplicacion que subtitula mediante pictogramas.</description>
    <width>960</width>
    <height>540</height>
    <author>
        <name>Miguel Hernández</name>
        <email>miguelhernandez2907@gmail.com</email>
        <link/>
        <organization>Unizar</organization>
    </author>
<autoUpdate/>
<category>Education</category>
</widget>
```

Figura C.2 : Fichero config.xml

Se puede ver que la aplicación es de tipo usuario, versión de la aplicación 1.1, si se quiere ir mejorando se puede ir cambiando la versión y volviendo a sincronizar con la televisión, vemos que acepta el modo pantalla completa y también se ve el nombre de la aplicación, descripción, resolución, en este caso 960x540 por que con las otras dos que acepta (1280x720 y 1920x1080) daba problemas la resolución del reproductor. Las demás opciones son informativas de la aplicación.

Por otro lado el index.html, es donde esta toda la estructura de la aplicación, en la figura C.3 se puede observar la cabecera donde adjuntamos todos los ficheros que va a necesitar la aplicación, como por ejemplo las API del mando a distancia, los ficheros javascript que controlan la aplicación, plugins de audio y vídeo, nuestra función de subtitulado o la página de estilos.

index.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/div/html4/strict.dtd">
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
        <title>App PictoSub</title>

        <!-- Common widget API -->
        <script type='text/javascript' language='javascript' src='$MANAGER_WIDGET/Common/API/Widget.js'></script>
        <script type='text/javascript' language='javascript' src='$MANAGER_WIDGET/Common/API/TVKeyValue.js'></script>
        <script type="text/javascript" language="javascript" src="$MANAGER_WIDGET/Common/webapi/1.0/webapis.js"></script>

        <!-- Widget code -->
        <script language="javascript" type="text/javascript" src="Javascript/Main.js"></script>
        <script language="javascript" type="text/javascript" src="Javascript/Server.js"></script>
        <script language="javascript" type="text/javascript" src="Javascript/Data.js"></script>
        <script language="javascript" type="text/javascript" src="Javascript/Player.js"></script>
        <script language="javascript" type="text/javascript" src="Javascript/Display.js"></script>
        <script language="javascript" type="text/javascript" src="Javascript/Audio.js"></script>

        <!--  Codigo de subtitulado -->
        <script language="javascript" type="text/javascript" src="http://155.210.156.59/App_Demo/js_sub/Fsubtitulado.js"></script>

        <!-- Style sheets -->
        <link rel="stylesheet" href="CSS/Main.css" type="text/css">

        <!-- Plugins -->
        <object id="pluginPlayer" border=0 classid="clsid:SAMSUNG-INFOLINK-SEF"></object>
        <object id="pluginAudio1" border=0 classid="clsid:SAMSUNG-INFOLINK-SEF"></object>
        <object id="pluginTVMW1" border=0 classid="clsid:SAMSUNG-INFOLINK-SEF"></object>

    </head>
```

Figura C.3 : Index.html cabecera

La aplicación se descarga todo el contenido de nuestro servidor, usando el fichero videolist.xml, lo podemos ver en la figura C.4 un extracto. Para descargarse este fichero, se encarga Server.js, tenemos 2 extractos del código en las figuras C.5 y C.6.
Server.js coge el fichero videolist.xml de la ruta que le pasemos, puede ser externa o dentro de la propia aplicación. Mediante AJAX recoge los datos y se los pasa al main.js para que realice las acciones oportunas.

videoList: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
<?xml version="1.0"?>
<rss version="2.0">
    <channel>
        <item>
            <title>Peppa-Pig-Cap01</title>
            <link>http://155.210.156.59/App_Demo/video/Peppa-Pig-Cap01.mp4</link>
            <description>
                Descripcion del capitulo 1:
                Una pequeña familia de cerditos.
                Con ellos, los más pequeños de la casa conocerán el mundo que les rodea.
                En este caso juegan con charcos de barro y acaban llenos de barro!!
            </description>
            <track>1</track>
        </item>
        <item>
            <title>Peppa-Pig-Cap02</title>
            <link>http://155.210.156.59/App_Demo/video/Peppa-Pig-Cap02.mp4</link>
```

Figura C.4 : Fichero videolist.xml

En la figura C.4, cada item es un vídeo a reproducir, cada uno tiene su título, descripción y la ruta de la que descargarse el vídeo en link.

Server.js

```
var Server = {
    /* Callback function to be set by client */
    dataReceivedCallback : null,
    XHRObj : null,
    url : "http:155.210.156.59/App_Demo/xml/videoList.xml"
};

Server.init = function() {
    var success = true;
    if (this.XHRObj) {
        this.XHRObj.destroy();
        // Saves memory
        this.XHRObj = null;
    }
    return success;
}
```

Figura C.5 : Fichero Server.js url

```
// Parse RSS
// Get all "item" elements
items = xmlElement.getElementsByTagName("item");

for (index = 0; index < items.length; index++) {
    titleElement = items[index].getElementsByTagName("title")[0];
    descriptionElement = items[index].getElementsByTagName("description")[0];
    linkElement = items[index].getElementsByTagName("link")[0];
    linkTrack = items[index].getElementsByTagName("track")[0];

    if (titleElement && descriptionElement && linkElement && linkTrack) {
        videoNames[index] = titleElement.firstChild.data;
        videoURLs[index] = linkElement.firstChild.data;
        videoDescriptions[index] = descriptionElement.firstChild.data;
        videoTrack[index] = linkTrack.firstChild.data;
    }
}
Data.setVideoNames(videoNames);
Data.setVideoURLs(videoURLs);
Data.setVideoDescriptions(videoDescriptions);
Data.setVideoTrack(videoTrack);
if (this.dataReceivedCallback) {
    this.dataReceivedCallback();
    /* Notify all data is received and stored */
}
```

Figura C.6 : Fichero Server.js Parse RSS

Vemos que en Server.js utilizamos funciones del fichero Data.js, este se encarga de enlace entre el fichero Server.js y Main.js, en la figura C.7 tenemos un extracto del fichero Data.js donde se ven las funciones utilizadas para la descripción, setVideoDescription y getVideoDescription.

Data.js

```
var Data =
{
    videoNames : [ ],
    videoURLs : [ ],
    videoDescriptions : [ ],
    videoTrack : [ ]
}

Data.setVideoDescriptions = function(list)
{
    this.videoDescriptions = list;
}

Data.getVideoDescription = function(index)
{
    var description = this.videoDescriptions[index];

    if (description)    // Check for undefined entry (outside of valid array)
    {
        return description;
    }
    else
    {
        return "No description";
    }
}
```

Figura C.7 : Data.js

Toda la aplicación esta controlada por el Main.js, en la figura C.8 podemos ver la función de updateCurrentVideo, que carga el vídeo que esté seleccionado, y representa la descripción del vídeo actual y la posición en el índice.

```
Main.updateCurrentVideo = function(move)
{
    Player.setVideoURL( Data.getVideoURL(this.selectedVideo) );

    Display.setVideoListPosition(this.selectedVideo, move);

    Display.setDescription( Data.getVideoDescription(this.selectedVideo));

}
```

Figura C.8 : Main.js updatecurrentVideo

Las acciones a realizar vienen determinadas por la función KeyDown que está dentro del Main.js, en la figura C.9 se puede ver un extracto del código y en la tabla 9 la relación de todos los botones con las descripciones de las acciones que realizan.

```
96 Main.keyDown = function()
97 {
98     var keyCode = event.keyCode;
99     alert("Key pressed: " + keyCode);
100
101     switch(keyCode)
102     {
103         case tvKey.KEY_GUIDE:
104             alert("INDEX");
105             this.displayIndex();
106             break;
107         case tvKey.KEY_INFO:
108             alert("DESCRIPTION");
109             this.descriptionMode();
110             break;
111         case tvKey.KEY_YELLOW:
112             sf.service.AVSetting.show(function asd(){
113                 Main.enableKeys();
114             });
115
116              break;
117         case tvKey.KEY_BLUE:
118             sf.service.AVSetting.hide();
119             break;
120         case tvKey.KEY_RETURN:
121         case tvKey.KEY_PANEL_RETURN:
122             alert("RETURN");
123             Player.stopVideo();
124             widgetAPI.sendReturnEvent();
125             break;
126             break;
```

Figura C.9 : Main.KeyDown

Botón	Descripción de la acción
KEY_GUIDE	Mostrar/ocultar el índice de los vídeos y la ayuda en la navegación
KEY_INFO	Mostrar/ocultar la descripción del vídeo
KEY_YELLOW	Mostrar configuración del vídeo
KEY_BLUE	Ocultar configuración del vídeo
KEY_PANEL_RETURN	Parada del vídeo y mandamos el evento de volver al evento anterior
KEY_PLAY	Inicialización de la reproducción del vídeo
KEY_STOP	Parada de la reproducción del vídeo y vuelve al inicio
KEY_PAUSE	Parada de la reproducción del vídeo pudiendo continuar desde ese punto
KEY_FF	Adelanto 5 seg el tiempo de reproducción
KEY_RW	Atraso 5 seg el tiempo de reproducción

KEY_VOL_UP	Aumentoel volumen del vídeo
KEY_VOL_DOWN	Disminución el volumen del vídeo
KEY_UP	Selección del vídeo superior
KEY_DOWN	Selección del vídeo inferior
KEY_PANEL_ENTER	Mostrar/ocultar modo pantalla completa
KEY_MUTE	Silenciar el vídeo

Tabla 9 : Relación botón acción

La resolución del vídeo se configura desde el Player.js, en la figura 3.11 se puede ver la función utilizada, además este fichero controla la reproducción del vídeo con las diferentes funciones de pausa, stop, resume y play vídeo. Podemos observar un extracto del código en la figura C.10 donde vemos la inicialización el reproductor, el cuál utiliza el plugin de avplay.

```
48 Player.init = function()
49 {
50     var success = true;
51     alert("success vale :  " + success);
52     this.state = this.STOPPED;
53     try{
54         var playerInstance = webapis.avplay;
55         webapis.avplay.getAVPlay(Player.onAVPlayObtained, Player.onGetAVPlayError);
56 
57     }catch(e){
58         alert('######getAVplay Exception :[' +e.code + '] ' + e.message);
59     }
60     return success;
61 };
62 
```

Figura C.10 : Player.js init

La parte donde se controla el modo pantalla completa o ventana se puede ver dentro de Main.js en la figura C.11.

```
251 Main.setFullScreenMode = function()
252 {
253     if (this.mode != this.FULLSCREEN)
254     {
255         Display.hide();
256 
257         Player.setFullscreen();
258 
259         this.mode = this.FULLSCREEN;
260     }
261 }
262 
263 Main.setWindowMode = function()
264 {
265     if (this.mode != this.WINDOW)
266     {
267         Display.show();
268 
269         Player.setWindow();
270 
271         this.mode = this.WINDOW;
272     }
273 }
```

figura C.11 : Main.js pantalla completa / ventana

Con todo el funcionamiento claro de la aplicación, en la figura C.12 se puede ver un extracto del código del index.html donde aparece la descripción y el índice. Por otro lado el código de los pictogramas en el index se puede ver en la figura 3.7 y la página de estilos de este en la figura C.13.

```
    <div id = "description" class = "style_navi3" style='display:none;'></div>
</div>
<!-- Parte Izquierda con el índice de los videos a reproducir -->
<div id="leftHalf">
    <div id="videoList" class="style_videoList"> Indice de videos
        <div id="video0"></div>
        <div id="video1"></div>
        <div id="video2"></div>
        <div id="video3"></div>
        <div id="video4"></div>
        <div id="videoCount"></div>
    </div>
</div>
```

Figura C.12 : Descripción y videolist en index.html

```
#Picto1
{
    position:absolute;
    width:125px; height:90px;
    text-align:middle;

}
#Text1
{
    position:absolute;
    top:95px;
    width:125px; height:30px;
    text-align:center;
    color:black;

}
#Picto2
{
    position:absolute;
    left:125px;
    width:125px; height:90px;
    text-align:middle;

}
#Text2
{
    position:absolute;
    left:125px; top:95px;
    width:125px; height:30px;
    text-align:center;
    color:black;

}
```

Figura C.13 : Main.css pictogramas

Todo el código de la aplicación para smart tv vendrá dada en soporte de CD adjunto a la memoria.

ANEXO D

D.1 Instrucciones de la exportación de este TFG

Una vez tenemos la aplicación creada, necesitamos exportarla al servidor apache para poder despues sincronizarla con la smart tv. Lo primero de todo es descargarnos el servidor Apache, en el caso de este TFG se ha optado por utilizar el panel de control XAMPP[7], una vez descargado e instalado, te monta el servidor automáticamente y preparado con una configuración básica, si se quiere modificar habría que modificar el fichero httpd.conf pero en nuestro caso no es necesario.
Una vez tenemos nuestro servidor montado, activamos el servidor apache, en la figura D.1 podemos ver el panel de control.

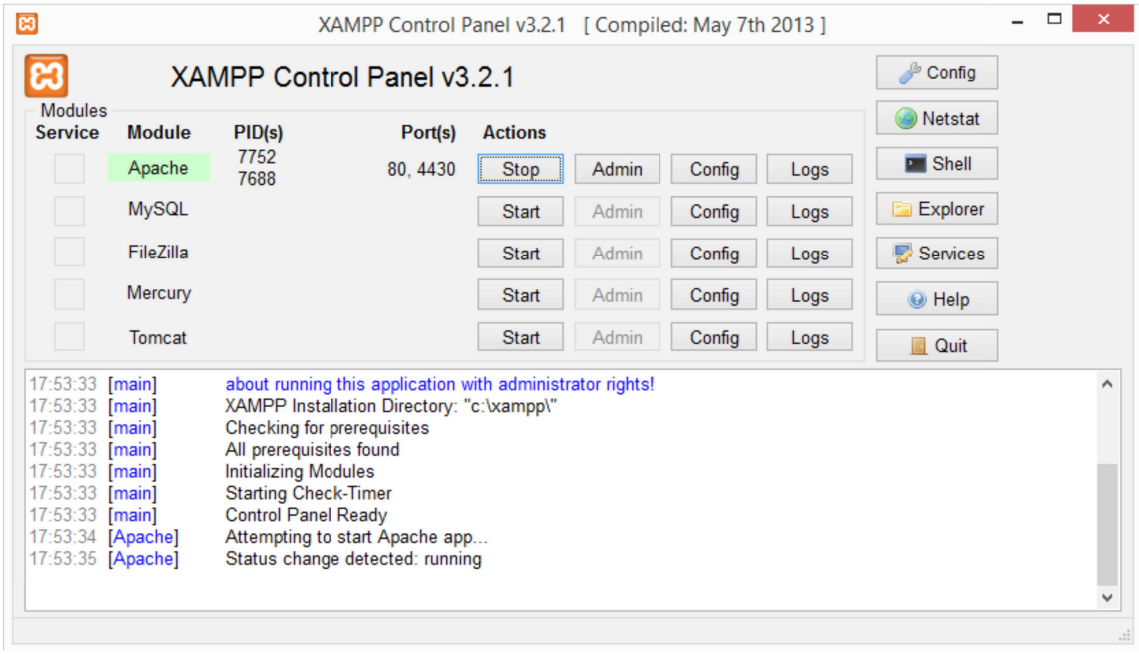

Figura D.1 : Panel de control del servidor Apache

¿Cómo exportamos la aplicación? Desde el propio sdk. Lo primero de todo es configurar en eclipse para que configure la aplicación exportada correctamente. En la figura D.2 tenemos la configuración de este TFG

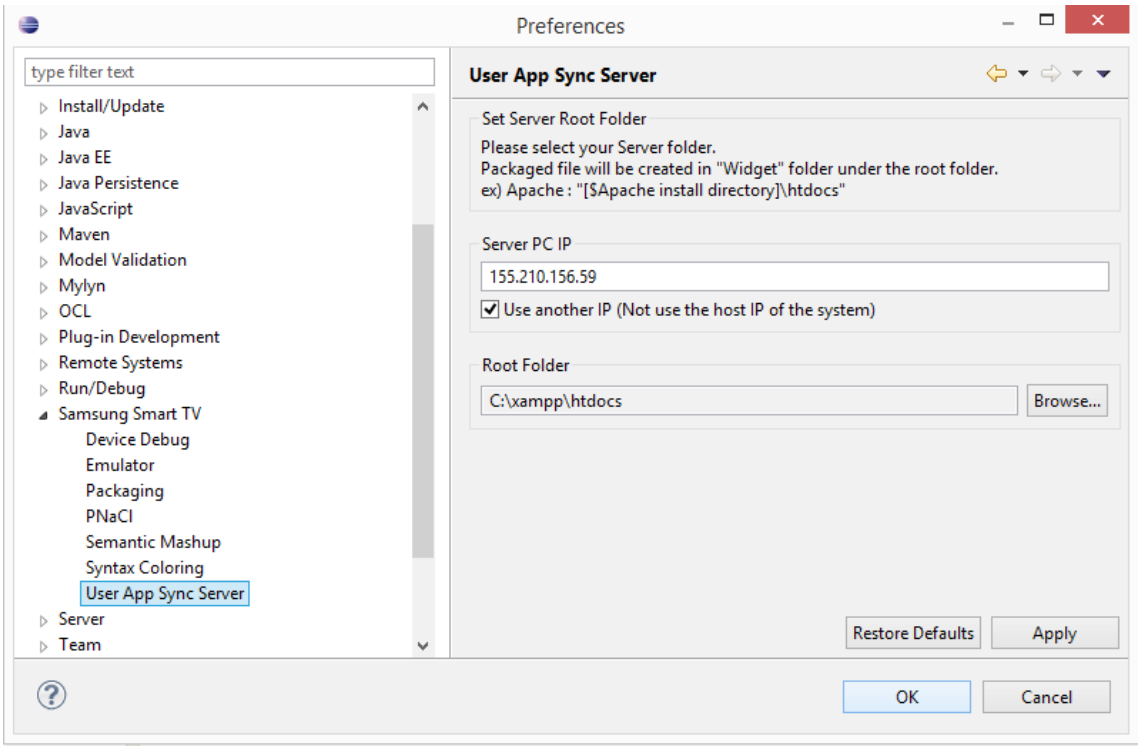

Figura D.2 : Configuración del SDK

Una vez configurado nuestro SDK a las necesidades que tengamos, procedemos a exportar la aplicación. El primer paso es seleccionar el proyecto y botón derecho para seleccionar "Export". En la figura D.3 lo podemos ver.

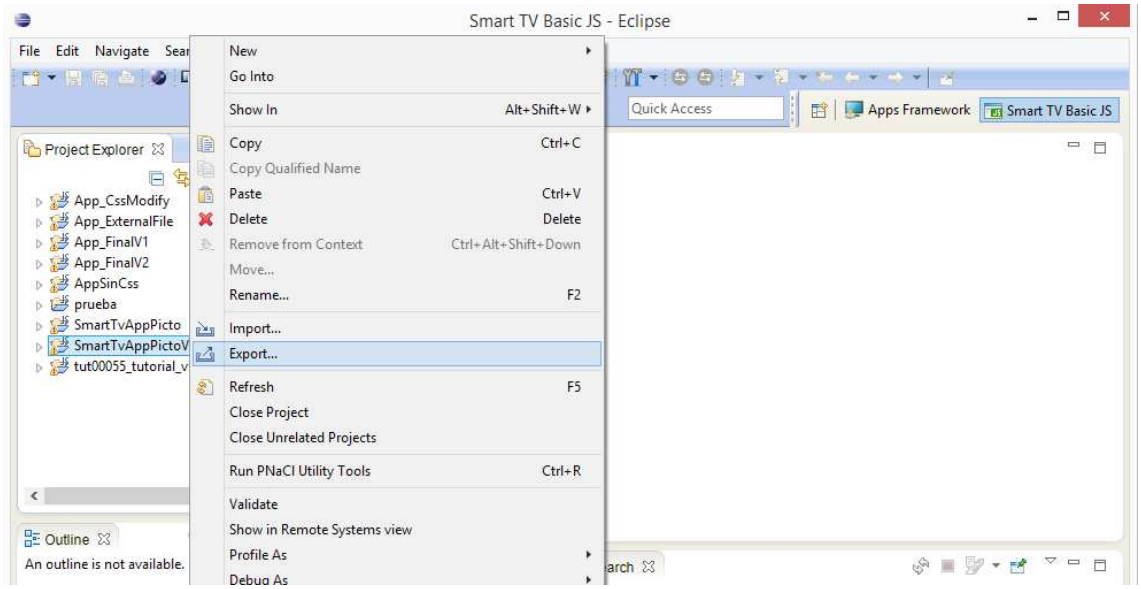

Figura D.3 : Exportamos la aplicación

Una vez le damos a "Export" seguimos los pasos que nos indican las figuras D.4 y D.5.

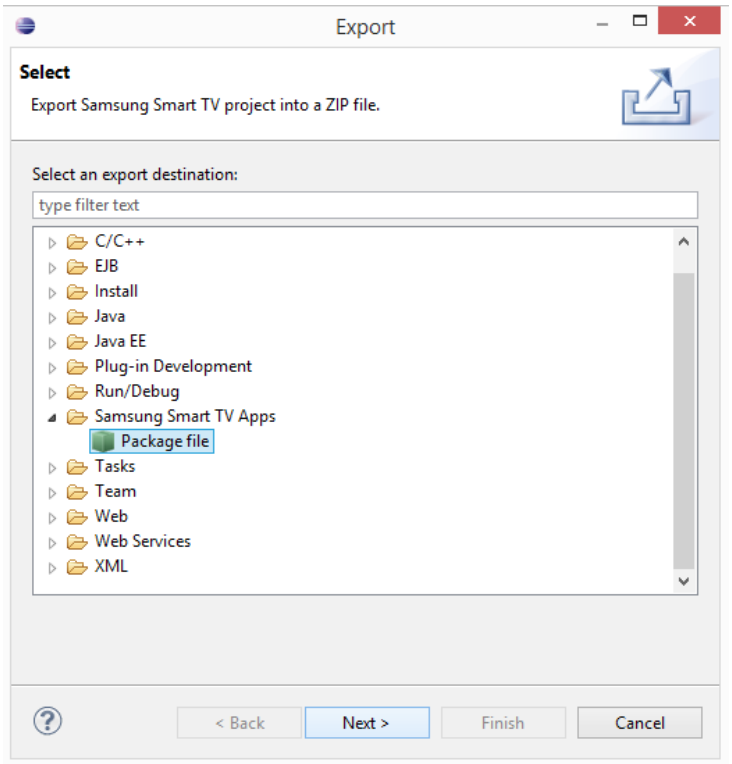

Figura D.4 : Opciones 1 exportar aplicación

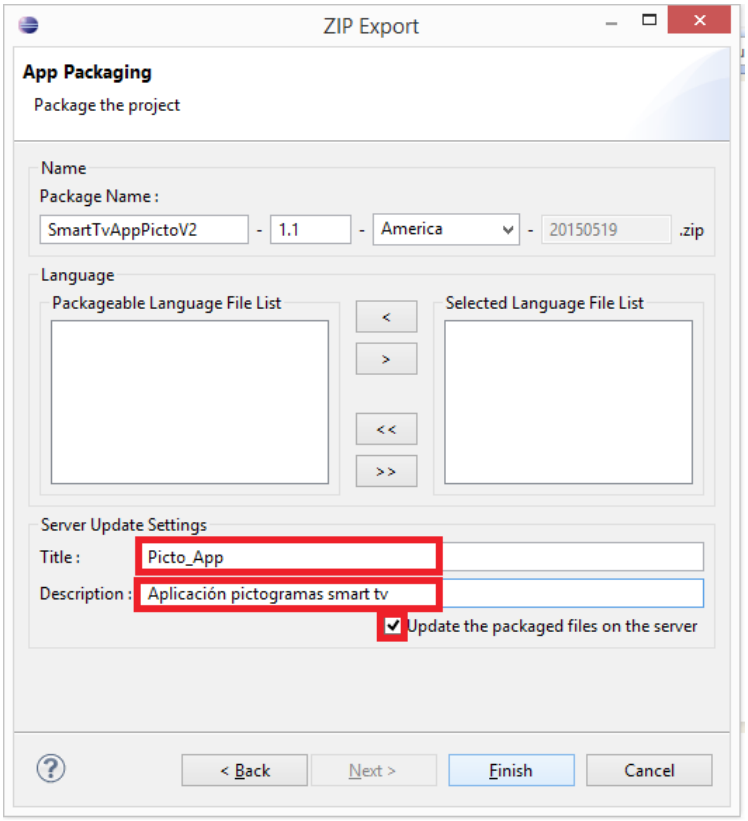

Figura D.5 : Opciones 2 exportar aplicación

Es necesario seleccionar el boton de subir los archivos en el servidor para que funcione correctamente.

Con esto ya tendríamos la aplicación exportada en el servidor en formato .zip. Lo guarda en la ruta xampp/htdocs/Widget y modifica el fichero widgetlist, que es el que utiliza el televisor a la hora de sincronizarse para saber que es lo que se tiene que descargar. En la figura D.6 tenemos el fichero widgetlist.xml

Figura D.6 : Fichero widgetlist.xml

Ahora nos falta configurar nuestra IP a la para que funcione correctamente y seamos nosotros el servidor al que se va a conectar la televisión. En la figura D.7 podemos ver la configuración usada en este TFG pero puede cambiar dependiedo del caso.

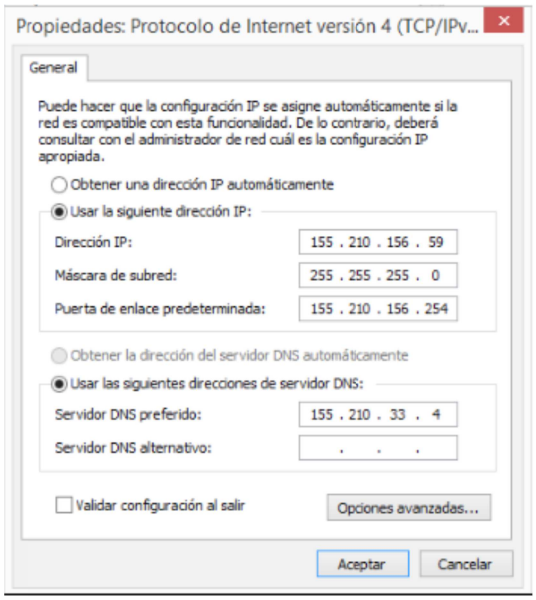

Figura D.7 : Configuración Ip en este TFG

Si se ha realizado correctamente, los valores del fichero widgetlist deberían ser los mismos que los de la aplicación al exporarla.
En este momento, podemos conectar el portátil a la misma red en la que se encuentra la smart tv y comenzar con la sincronización. Lo primero es

entrar en el modo desarrollador, entramos desde el menu smarthub y seleccionamos cuenta samsung y despues en login. En las figuras D.8, D.9 y D.10 se puede ver el proceso.

Figura D.8 : Smart Features

Figura D.9 : Cuenta samsung menu smarthub

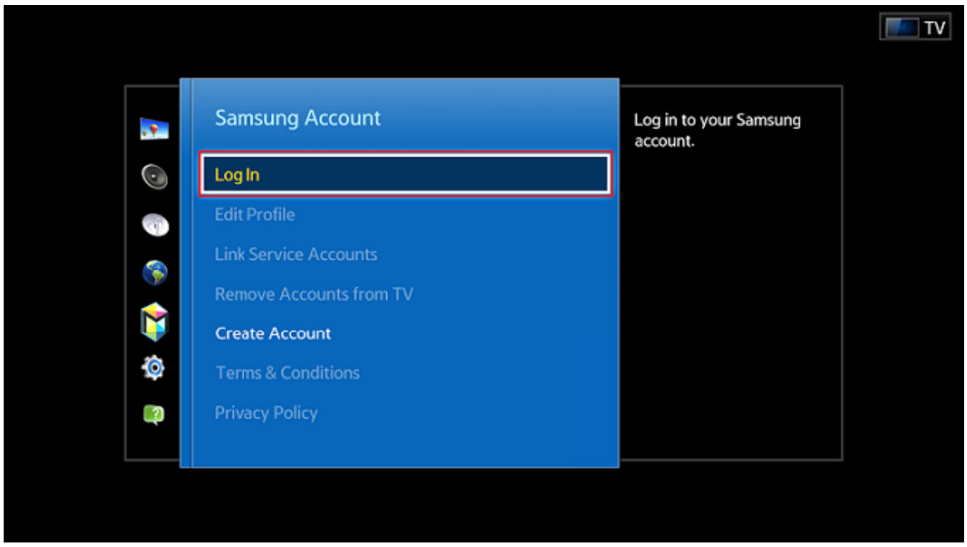

Figura D.10 : Log in menu cuenta samsung

Al seleccionar "Log in" tendremos que insertar el usuario y contraseña del modo desarrollador. Para ello ponemos como usuario: develop y dejamos la contraseña vacia. En la figura D.11 se puede ver como se ingresan los datos. Con esto ya tenemos la televisión lista para poder sincronizar aplicaciones, así que nos metemos en el menu de SmartHub desde el menu principal o utilizando el botón que incluye el mando a distancia. En la figura D.12 podemos ver el SmartHub.

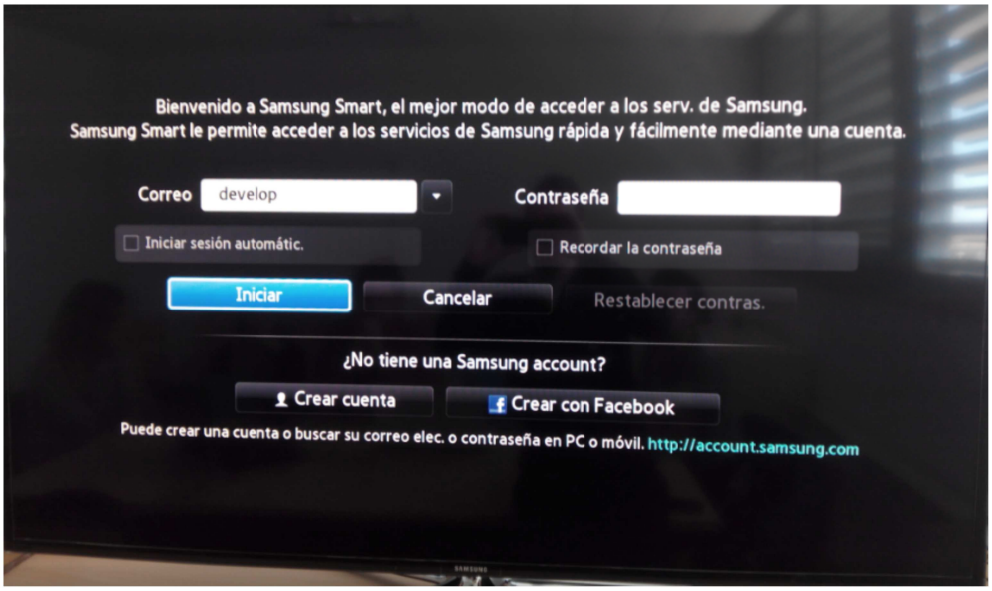

Figura D.11 : Datos como develop SmartHub

En la figura D.12 presionamos en Mas Apps para pasar al menu donde podemos sincronizar nuestras propias Apps.

Figura D.12 : Menu SmartHub

Una vez dentro nos vamos a la parte superior de la figura D.13 para presionar opciones, este nos desplegará un menu en el cual seleccionaremos IP settings para configurar la ip de nuestro servidor. Se puede ver este proceso en las figuras D.14 y D.15.

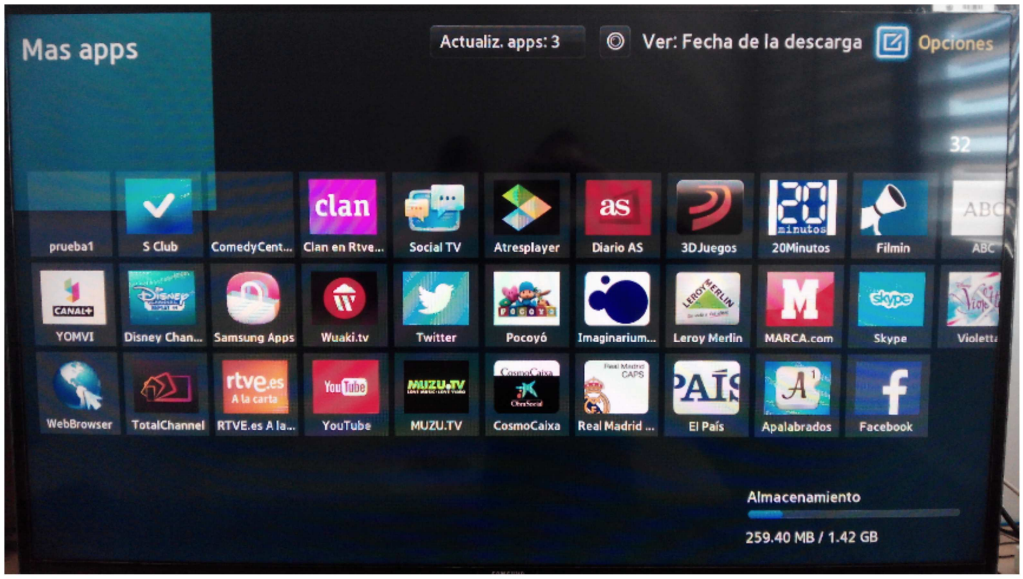

Figura D.13 : SmartHub opciones

En el caso de este TFG, en la figura D.15 colocamos la siguente ip:

155.210.156.59

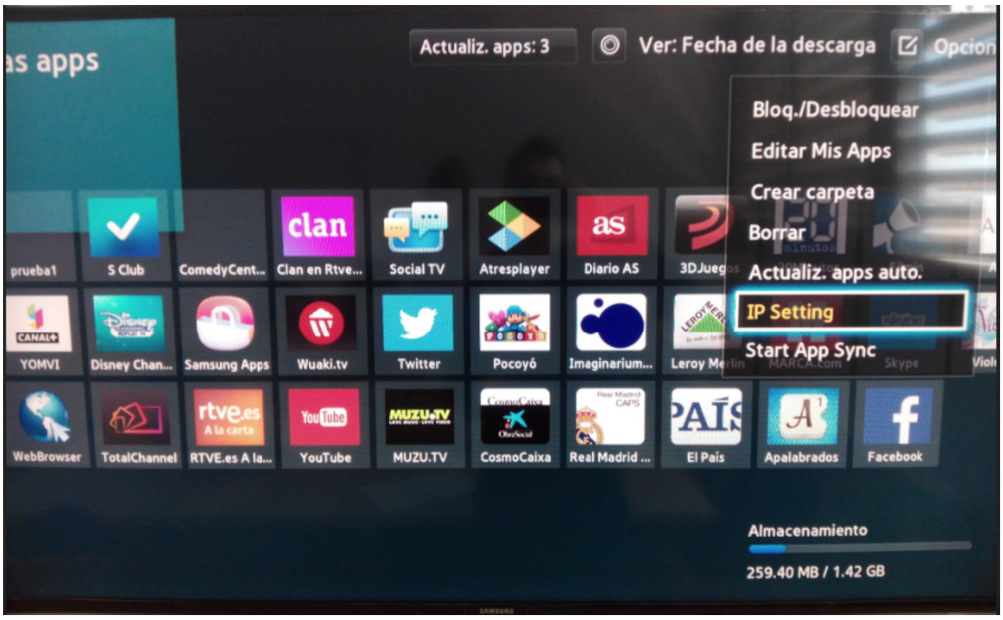

Figura D.14 : IP Setting

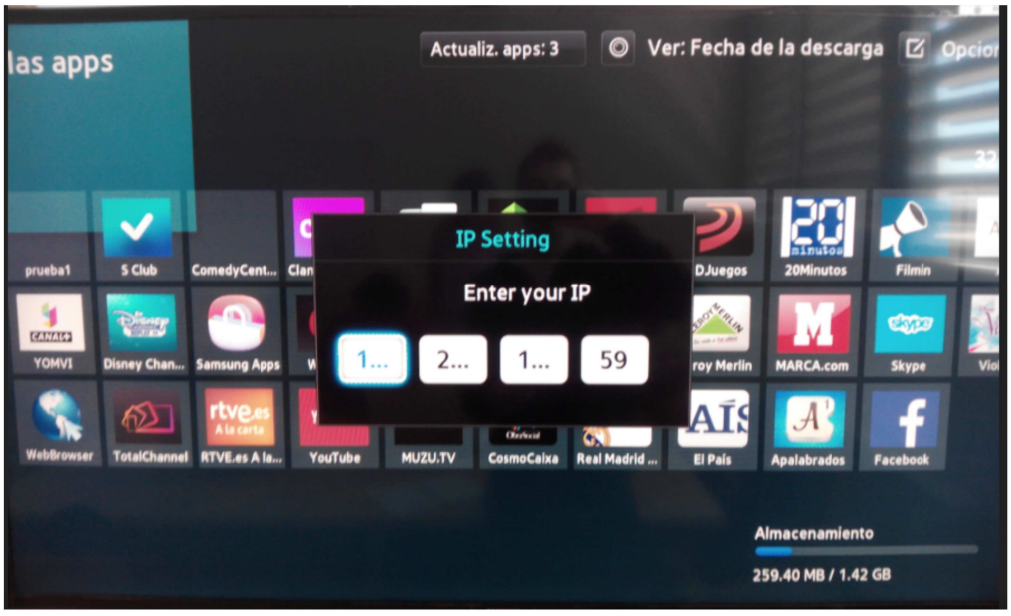

Figura D.15 : Ip del server 155.210.156.59

Finalmente le damos a sincronizar nuestra aplicación, en la figura D.16 podemos ver el procedimiento y en la D.17 vemos que ocurre si hemos seguido correctamente los pasos y todo funciona bien.

Figura D.16 : Start App Sync

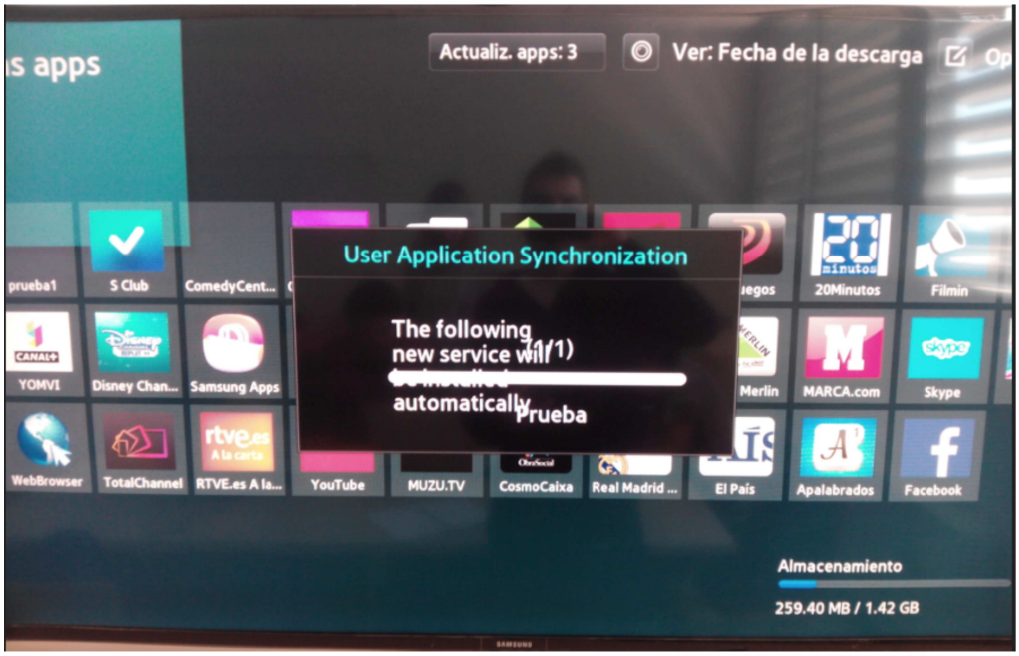

Figura D.17 : Correcta sincronización

Pueden surgir diversos problemas, vamos a intentar analizar cada uno de ellos para saber que hacer en caso de que se produzcan.

Error de conectividad

Puede ser que a la hora de exportar no tengamos correctamente conectado el cable rj45 a nuestro portátil y eso provocará que la smart tv busque constatemente con quién conectarse. Este error también puede

suceder si hemos configurado mal el SDK y al exportarla tiene otra ip, con lo que la smart tv no sabrá de donde se tiene que descargar los ficheros. Puede darse el caso que el firewall nos de problemas pero al deshabilitarlo deberían cesar. En la figura D.18 nos aparece la pantalla que saldría en estos casos.

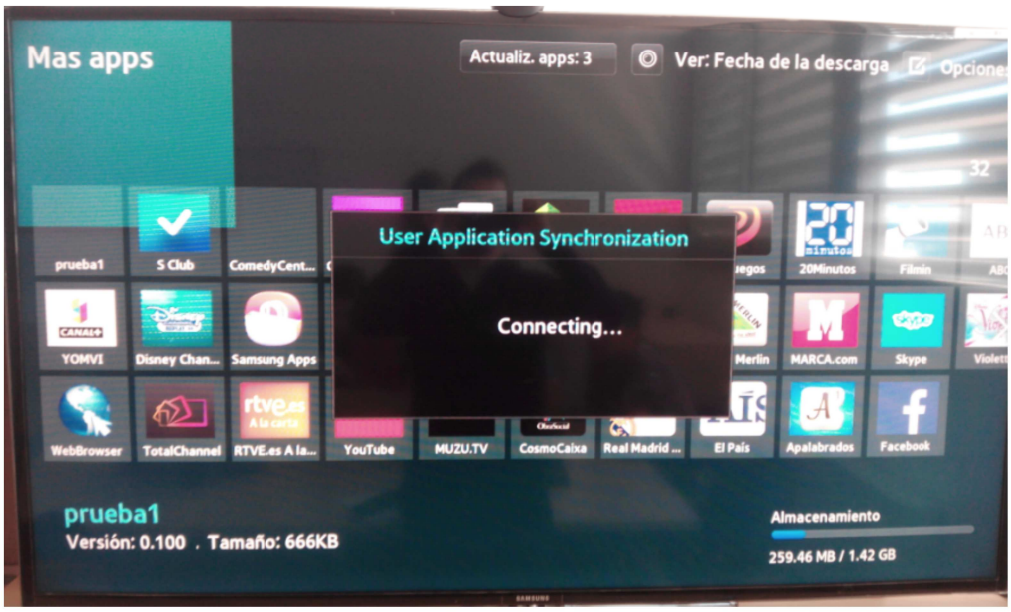

Figura D.18 : Error de conectividad

Error de desarrollador

Este error es probable por una mala configuración en la conexión entre el portátil y la televisión, hay que mirar bien como hemos configurado el IP Setting de la smart tv. Por otro lado Si este error persiste, vuelve a mirar todo el tutorial de la exportación de la aplicación ya que algún paso no has seguido correctamente y por eso aparece este fallo. En la figura D.19 podemos ver lo que nos saldría en la pantalla.

Estos son los errores que nos pueden aparecer en la exportación de una aplicación, si la aplicación no la soporta la smart tv, dejará exportarla pero a la hora de iniciarla será cuando nos bloquee la televisión o ni tan siquiera nos abra nada.

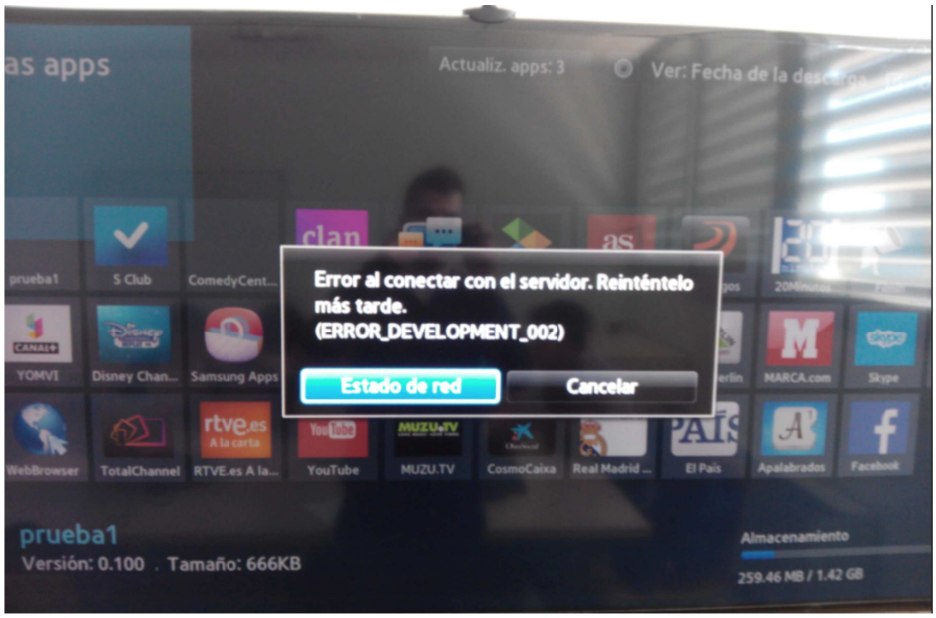

Figura D.19 : Error Development 002