Control Automatizado de Audiencias Televisivas

Proyecto de Final de Carrera

Rubén López Navarro

Tutor: Gustavo Medina del Rosario

Tutor: Agustín Trujillo Pino

Las Palmas de Gran Canaria

Enero de 2014

Índice general

Capítulo 1

Introducción

1.1. Objetivos del proyecto

En este proyecto, propuesto por la empresa Singular Factory, se pretende llevar a cabo un prototipo de un medidor de audiencias televisivas automatizado, debido a que los audímetros actuales presentan una serie de inconvenientes que se presentarán más adelante. Como iremos viendo a lo largo de este documento, se trata de una tarea dividida en unos bloques principales y que, aunque a priori parezcan estar claras, resultan en una serie de disyuntivas y desaciertos que obligan al repetido replanteamiento del sistema o de sus partes desde un punto de vista tecnológico.

Para llevar a cabo la automatización, es necesario realizar el recuento de espectadores de forma autónoma. La forma en la que se ha decidido abordarse este problema es mediante técnicas de reconocimento facial a través de imagen, que a su vez ofrece las ventajas de poder proporcionar información adicional al número de personas, en función de las características del reconocedor utilizado.

Es necesario que el prototipo sea completo y funcional, de forma que sea posible hacerse una idea de la escalabilidad del mismo y la posibilidad de su aplicación comercial en el caso de que se prosiga su desarrollo. Por lo tanto, para garantizar la completitud del mismo, debe abarcar la recopilación de información de audiencias en los hogares de los televidentes, el almacenamiento y análisis de dichos datos, y la muestra de los mismos a los interesados de una forma fácilmente entendible.

Por otra parte, la introducción en el mercado de las SmartTV (televisiones inteligentes) da la opción al estudio de la incorporación del sistema de audiencias en las mismas. Más adelante veremos las conclusiones tomadas al respecto y las decisiones de diseño y desarrollo en base a ellas.

1.2. La televisión como medio de comunicación

La prehistoria de la televisión se extiende, aproximadamente, desde finales del siglo XIX hasta 1935. Durante este período un grupo de investigadores en los países tecnológicamente más avanzados (EEUU, Gran Bretaña, Francia, Alemania) buscan transmitir imágenes a distancia. Se trataba de captar imágenes utilizando una cámara, transmitir esas imágenes a través del aire y recibirlas en un aparato receptor a cierta distancia de donde originariamente se habían captado: la televisión.

Tras una serie de inventos, marchas y contramarchas, en los años 20 surgen los dos primeros modelos de televisión: por un lado, la televisión **mecánica**, por otro, la **electrónica**. Ambas se desarrollaron de forma paralela en un período caracterizado por la lucha entre distintas compañías e inventores por la adopción de un único sistema.

Aunque la calidad de las imágenes del sistema mecánico empleado por Baird mejoró notablemente con el transcurso del tiempo, siempre quedó muy por debajo de la obtenida con su competidora. No podía resistir la comparación.

Por su parte, la televisión electrónica, fue creada por el científico ruso-norteamericano Vladimir Zworykin, que trabajaba en la estadounidense RCA hacia finales de los años 20. Así, en 1931 la RCA colocó una antena emisora en la terraza del Empire State Building, el edificio más alto de Nueva York, y comenzó con sus emisiones de pruebas.

Al otro lado del Atlántico, la inglesa EMI se lanzó a trabajar en la televisión electrónica. Los ingenieros de EMI realizaron una demostración a la BBC sobre su sistema televisivo. Los especialistas no tuvieron dudas: la calidad de este sistema era muy superior. En enero de 1935, una comisión investigadora creada por el gobierno británico para definir la posición del Estado en materia televisiva optó por la televisión electrónica debido a la superior calidad respecto al otro sistema y el 2 de noviembre de 1936, la

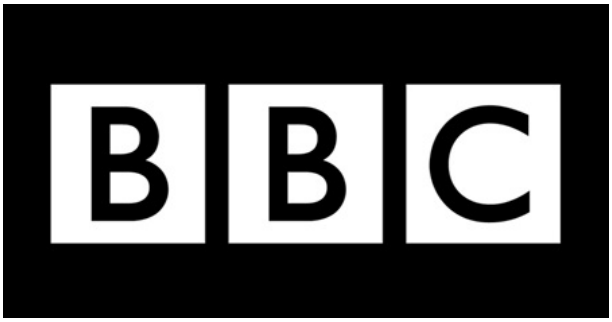

Figura 1.1: La BBC, la cadena británica más relevante en el desarrollo de la televisión

BBC comenzó sus transmisiones desde los estudios londinenses de Alexandra Palace.

Hacia mediados de la década de 1930 las transmisiones tienden a regularizarse y a crecer en las principales urbes (Londres, Berlín, París, Nueva York).

El primer gran evento retransmitido por televisión fueron los Juegos Olímpicos de Berlín de 1936. El éxito fue total. La recepción de las emisiones tuvo lugar en lugares públicos: "teatros" con capacidad para 50 personas y pantallas de cerca de dos metros de diagonal.

Figura 1.2: Imagen televisada de los JJOO de Berlín de 1936, recuperada por Televisión Española

Este rápido desarrollo tropezó con la escasa fabricación de receptores, hecho que se corrigió a finales de la década. Fue en Gran Bretaña en 1937 y en

los EEUU en 1939 cuando comenzó la fabricación de aparatos receptores en serie para uso doméstico-familiar. Todos estos avances se vieron congelados con el estallido de la Segunda Guerra Mundial, que obligó a los gobiernos a concentrar su esfuerzo económico en los materiales de guerra.

Concluida la II Guerra Mundial, los gobiernos y los sectores industriales ligados a la televisión volvieron su mirada a la pequeña pantalla en un contexto de recuperación social, económica y tecnológica. Muchos países europeos fueron haciendo sus primeras pruebas públicas y en España, las empresas Philips y RCA organizaron en 1948 sendas exhibiciones de televisión en Barcelona y Madrid respectivamente.

A medida que los servicios de televisión se regularizaron fueron ganando fervorosos adeptos allí donde comenzaban las emisiones y estableciendo un novedoso equilibrio en relación a los otros medios existentes (prensa, cine y radio). Asimismo aparecieron dos modos diferentes de entender la televisión en Occidente: mientras que en los EEUU, y luego en Iberoamérica, la industria televisiva se asentó en redes de empresas privadas y comerciales en competencia, en la Europa del Oeste se desarrollaron fuertes sistemas públicos y nacionales de radio y televisión.

Un país se destaca por la conformación de una poderosa industria televisiva: EEUU. Hacia 1952 se calcula que unas 108 emisoras estadounidenses daban servicio a unos 21 millones de televisores. De forma paralela creció la publicidad. Las cifras son reveladoras: de una inversión publicitaria de un poco más de 10 millones de dólares, en 1950, se pasó a 1.500 millones, en 1960. A comienzos de los años 50 la diferencia entre los EEUU y el resto de los países desarrollados era notoria. Durante los primeros meses de 1952, en Gran Bretaña sólo se habían vendido 1,2 millones de televisores, en Francia cerca de 10.600, y en la entonces Alemania Federal apenas se contabilizaban 300 aparatos.

La televisión también empezó a causar furor en Japón. La NHK, comenzó a operar en 1953 y al año siguiente hizo lo propio la primera estación comercial.

La incorporación del color supuso otra revolución. EEUU se convirtió, en 1953, en el primer país en contar con televisión en color con su propio sistema de emisión: el NTSC, que actualmente sigue en funcionamiento en los EEUU, Canadá y Japón, entre otros países.

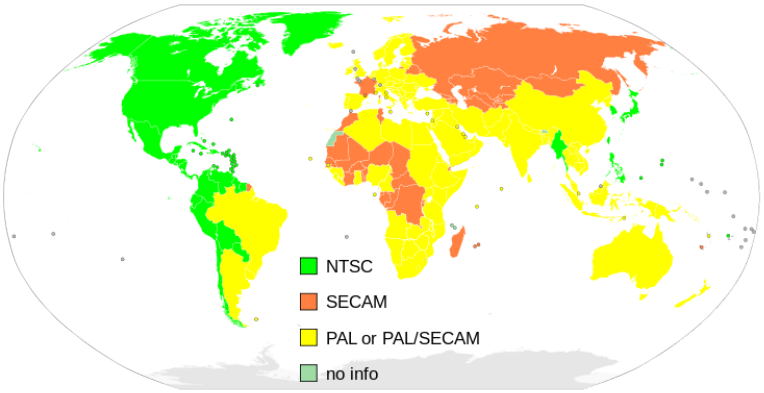

Figura 1.3: Sistemas de televisión utilizados en el mundo

Posteriormente en Europa se pusieron en marcha una serie de investigaciones para perfeccionar el sistema estadounidense. Los resultados de éstas dieron lugar a dos sistemas de televisión en color. En 1959, el Gobierno galo puso en marcha el sistema SECAM; y cuatro años más tarde, de la mano de Telefunken, apareció el sistema alemán PAL. Hoy en día estos dos sistemas son todavía utilziados en Europa, siendo PAL el utilizado en España.

El siguiente paso fue lógico: la internacionalización de las emisiones. Hoy día los intercambios entre países y entre organismos internacionales se basan en la difusión de programas a través de los satélites de comunicaciones. Así, desde el lanzamiento del primer satélite Sputnik, en 1957, y de satélites cada vez más perfeccionados, los intercambios de informaciones y de programas se han multiplicado. El primer acontecimeinto retransmitido de forma internacional, marcando un nuevo hito en la televisión, fueron los Juegos Olímpicos de Tokyo de 1964.

Hacia 1989, comenzó la segunda generación de operadores televisivos vía satélite. En poco tiempo la creación de diversas plataformas multicanales vía satélite en todos los países fue un hecho.

Las antenas parabólicas empezaron a proliferar. Un paso más se dará cuando las plataformas se digitalicen totalmente. En EEUU, a mediados de 1994, se pone en marcha la primera plataforma digital. Poco más tarde se hizo lo propio en Europa. Hoy en día, todas las televisiones están dando el salto a la era digital, produciéndose el "apagón" de la televisión analógica a partir del año 2006, cuando los Países bajos fueron los primeros en abandonar

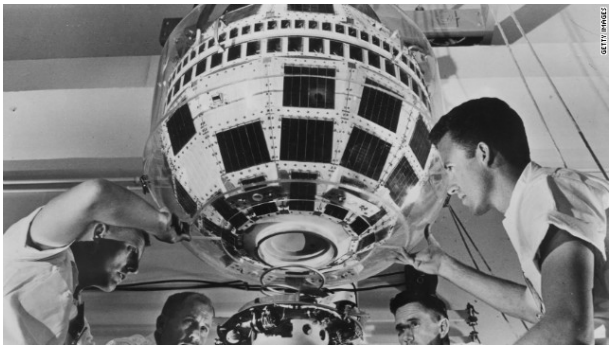

Figura 1.4: Telstar 1, el primer satélite usado para emitir señal de televisión

completamente el contenido analógico. En España este hecho se produjo en 2010, y cada año multitud de países se suman a esta tendencia. Estados Unidos tiene planeada esta acción para el año 2015, aunque la mayor parte de sus emisiones ya son de tipo digital.

Se estima que para 2018, todos los países habrán adoptado la TDT (Televisión Digital Terrestre) para la retransmisión de contenido televisivo, fecha para la cual el apagón analógico habrá concluido a nivel mundial.

1.3. Consumo de televisión

El consumo televisivo de 2012 en España es de 246 minutos de media diaria por individuo, lo que supone un importante incremento en relación al año pasado, y un récord histórico en el consumo anual de la televisión en nuestro país. El mes de noviembre es el de mayor consumo con 269 minutos, siendo éste también un récord histórico en el consumo mensual.

Los 246 minutos consumidos por persona y día suponen un nuevo récord de consumo televisivo en España. Es el cuarto año consecutivo de incrementos y la cifra supone 7 minutos más que el año anterior. Las nuevas tecnologías, dispositivos y pantallas, así como el conjunto de las redes sociales (1 de cada 3 tuits es sobre televisión) no parecen afectar al medio televisivo, sino más bien al contrario; retroalimentan al producto audiovisual.

Por ámbitos, las regiones más consumidoras de televisión ha sido Andalucía, Aragón y Valencia, mientas que Galicia y Madrid son las comunidades donde el consumo televisivo es menor.

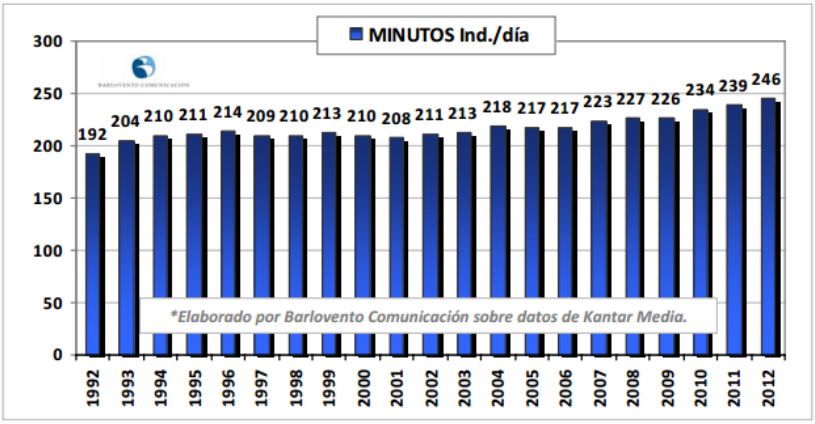

Figura 1.5: Niveles de consumo televisivo entre 1992 y 2012

Sin embargo, a pesar de este incremento en la audiencia, 2012 (y lo que va de 2013) es un tiempo de dificultades para el sector. Las televisiones públicas se encuentran inmersas en un clima de incertidumbres políticas, financienras y laborales. TVE está a la espera de conocer la resolución de Bruselas sobre sus fuentes de financiación (están en el aire las aportaciones de los operadores telefónicos y el resto de cadenas privadas), una decisión crucial para todo el sector por las importantes consecuencias que conllevaría en términos económicos.

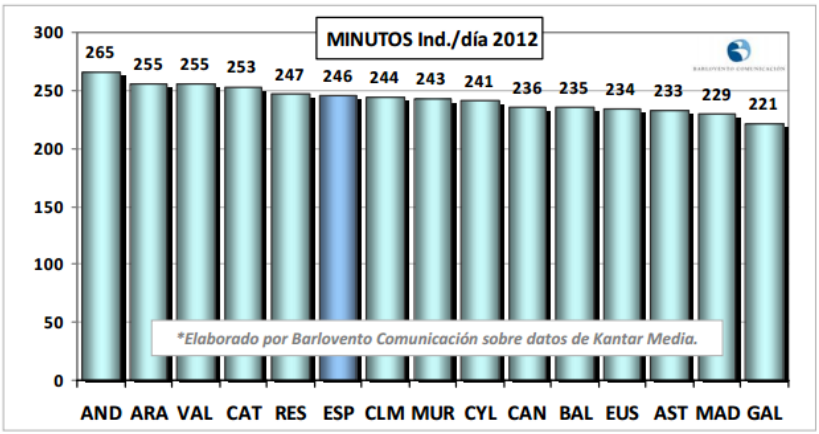

Figura 1.6: Niveles de consumo televisivo por comunidades autónomas en 2012

Los entes autonómicos, por su parte, se encuentran en una situación extremadamente delicada. Las reducciones presupuestarias de las que han sido objeto en los últimos tiempos, están rovocando en la mayoría de los

casos reajustes de personal y servicios básicos asociados a su funcionamiento (EREs en plantillas, redimensionamientos empresariales, cierre de canales, externacionalizaciones, ahorro extremo...)[1].

[1]Todo el apartado toma como fuente el informe oficial del análisis televisivo de 2012 de Barlovento Comunicación

Capítulo 2

Estado del Arte

Cuando se tomó interés en conocer los niveles de audiencia en los distintos canales televisivos, la primera maniobra fue heredar los sistemas que ya se utilizaban para conocer las audiencias de las emisoras de radio. El **sondeo** es el primero y más elemental de estos métodos, consistente en la entrevista con una muestra de la población de espectadores acerca de sus gustos televisivos. Estas entrevistas fueron primero en persona, aunque debido a su enorme coste económico y temporal se abandonó por las entrevistas teléfónicas. En algunos países aún se mantiene este formato, a pesar de estar plagado de inconveniencias, tanto de tipo humano (incapacidad de recordar si un programa se vió un determinado día, vergüenza de admitir la visualización de ciertas emisiones) como de tipo técnico (mala representación de la población por parte de la muestra tomada). A día de hoy este método aún se utiliza en ocasiones, aunque la evolución tecnológica lo ha hecho cambiar, siendo la encuesta realizada normalmente por correo electrónico o mediante formularios en una página web.

Debido a todos los problemas mencionados, surgió un nuevo método, el **panel** con diario de escucha, conocido comúmenmente como el panel. Esta técnica también fue heredada de la radio y requiere que cada uno de los integrantes de la muestra anote cada día en un impreso su consumo de televisión. El formulario consiste en una parrilla con las 24 horas del día divididas en fracciones de 15 minutos.

Se creó esta técnica para evitar algunos problemas que ocasionaba la entrevista. El individuo ya no tiene que recurrir a recordar lo que vio en un

determinado momento, y el entrevistador se hace innecesario, lo que hace el estudio más barato y se reduce el riesgo de que el entrevistado mienta sobre su consumo, ya que la presión psicológica frente a un formulario es mucho menor que al responder a un entrevistador humano.

Por supuesto, este método también conlleva algunas desventajas. Los formularios se han de imprimir y entregar, y cuando hayan sido rellenados, se han de recoger, lo que supone un gasto elevado. Otro de los problemas radica en en la falta de respuestas de los usuarios, ya que es difícil que una persona recuerde rellenar el formulario cada 15 minutos. A pesar de las desventajas, sus ventajas hacen que se considere un sistema superior al sondeo.

El primer método basado en tecnología y no heredado de los utilizados en la medición de audiencias radiofónicas, es el utilizado actualmente en la mayoría de países. El **audímetro** (o audiómetro) es un dispositivo que se coloca en el salón del hogar de los espectadores que se toman como muestra, y que se encarga de recoger los datos de consumo de televisión de los mismos. Este aparato se conecta a un televisor compatible con el mismo y dispone de un mando a distancia con el que los espectadores deben identificarse.

Originalmente, estos dispositivos únicamente podían recoger el canal visualizado en cada momento en el hogar, ofreciendo por lo tanto información sobre el consumo televisivo de la familia, pero no a nivel individual. Actualmente, cada miembro del hogar está registrado y tiene asignado un botón, que deberá pulsar para identificarse cada vez que encienda la televisión. Eso le permite saber al aparato no sólo quien está viendo la televisión, sino su sexo y edad, datos claves en la muestra. Cada vez que haya invitados, estos también deben identificarse con unos botones especiales. Si los miembros del hogar no se identifican, el aparato va emitiendo una señal acústica que se va

Figura 2.1: Audímetro contabilizando 3 usuarios registrados y 2 invitados

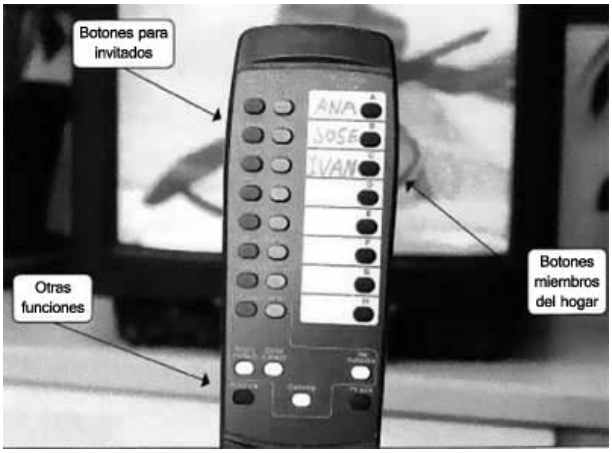

Figura 2.2: Control remoto de un audímetro tradicional

haciendo cada vez mayor, hasta que se identifiquen. Esta medida de presión, aunque útil para mantener los datos fiables, es muy invasiva e irritante para el usuario. La información recogida se va a almacenando a lo largo del día en la memoria del aparato, y es enviado a los servidores centrales al finalizar la jornada, donde son procesados.

Como puede apreciarse, todos los métodos mencionados hasta ahora son altamente invasivos para el espectador y además requieren de su cooperación y sinceridad para que los datos sean fiables. Precisamente estos dos aspectos son los que este proyecto intenta solucionar, para así conseguir unos datos de audiencia más cercanos a la realidad y, por lo tanto, más valiosos que los utilizados actualmente.

2.1. Estado actual de los audímetros en España

En España, los audímetros actuales son de uso manual. Una empresa (TNS) instala estos dispositivos en los domicilios de una serie de seleccionados (4500 por todo el país) de los que se extrapolan los datos para toda la población (de 45 millones de habitantes, 10.000 veces más grande que la muestra).

La muestra total está dividida en 9 submuestras, 8 para cada una de las comunidades autónomas con televisión propia que son Andalucía, Galicia, Castilla la Mancha, Cataluña, Comunidad Autónoma vasca, Canarias,

Madrid y Valencia, y una novena submuestra para el resto de la península y Baleares.

Es TNS quien decide en qué hogar poner un audímetro y estas personas pueden aceptar o rechazar la oferta. Como aliciente, TNS les gratifica mediante un programa de fidelización mediante la cual la familia puede seleccionar un regalo de un catálogo según los puntos que haya conseguido. En muchas ocasiones, estas recompensas son vales de descuento de compras en cadenas con las que TNS está asociada.

2.2. Repercusión económica de los datos de audiencia

Toda esta insistencia y mejoría en la obtención de los datos de audiencia televisiva tienen una motivación, y como en la mayoría de casos, es de tipo económico. Esta motivación viene dada debido a que el precio de la publicidad en un canal está estrechamente relacionada con el número de espectadores para ese canal en ese momento del día, y por lo tanto, con el programa que se está emitiendo. Debido esto, las cadenas usan la información de audiencia para determinar que programas emiten o dejan de emitir, y para determinar el precio (o el precio de salida, en caso de subasta de espacio publicitario, como sucede en la SuperBowl americana) de los anuncios, y las compañías que quieren publicitarse para conocer los momentos en los que más le interesa publicitar su producto o servicio en función del número de espectadores y su segmentación. Es por esto por lo que la información de audiencias televisivas es muy valiosa, y las grandes compañías pagan sumas voluminosas por ellas.

Para que nos hagamos una idea de la magnitud de las cifras que se mueven en el mundo de la publicidad, en el año 2011 se invirtieron **12.061 millones de euros** en publicidad en España, lo que representa un 1,12 % del PIB del país para dicho año. De este total, 2.237,2 millones fueron destinados a publicidad televisiva[1].

Explicado esto, es evidente la posible monetización del proyecto: La venta de la información obtenida por el mismo, o bien la venta del sistema completo

[1] fuente: `http://www.infoadex.es/Resumen_Estudio_Inversiones_InfoAdex_2012.pdf`

para que lo explote un tercero.

Capítulo 3

Plan de trabajo

El plan de trabajo de un proyecto de envergadura debe llevarse a cabo con cuidado, pero no perjudicar posteriores etapas de desarrollo. La primera tarea, como acostumbra a ser en este tipo de actividades, será realizar un análisis del sistema que se quiere materializar. Este análisis deberá incluir los aspectos tradicionales como la identificación de actores, los casos de uso y la identificación de requisitos, y deberá ser acompañado de un estudio sobre las Smart-TVs (televisiones inteligentes) y sus alternativas, de forma que una de las primeras decisiones pueda ser tomada y se elija una ruta de trabajo.

Una vez decidida la rama en la que se va a trabajar, se deberá comenzar un nuevo estudio sobre las herramientas de reconocimiento de rostros, tanto en sus capacidades y limitaciones como en las alternativas existentes. Además, gracias al conocimiento de los requisitos de usuario y software, analizados anteriormente, se podrá llevar a cabo una fase de investigación y experimentación con las distintas herramientas software que cubran las necesidades que se nos han planteado.

Una vez se conozcan las herramientas a utilizar, se entrará en un ciclo de diseño y desarrollo iterativo, y veremos como durante el mismo surgen problemas por un lado, y se realizan descubrimientos por otro, que abrirán camino a herramientas alternativas a las estudiadas previamente.

Este ciclo de desarrolo iterativo, basado en prototipos, concluirá cuando el sistema alcance un estado en el que se cumplan los requisitos establecidos previamente, momento en el que comenzarán las etapas de optimización y

pruebas.

Por último, se llevará a cabo una recopilación y adaptación de la información desarrollada, y su inclusión en este documento que se presenta, así como la redacción de los apartados faltantes.

3.1. Planificación temporal

Todas las tareas mencionadas en el apartado anterior deben ser planificadas, así como estimar la duración temporal de cada una de ellas. A priori, la planificación temporal del proyecto es la que sigue, aunque finalmente se produjeron una serie de eventos y descubrimientos que alargaron unas actividades y acortaron otras.

- Estudio del estado del arte de los medidores de audiencias televisivas (10 horas)
- Estudio sobre la usabilidad de las Smart-TV y sus respectivos SDK (100 horas)
  - Sistema operativo
  - Hardware
  - Puertos de entrada/salida
- Estudio de las alternativas en PC a las Smart-TV (50 horas)
  - Receptores de señal TDT funcionales en Linux
  - Reproductores de señal de televisión en Linux
- Elección final entre Smart-TV o miniPC.
- Análisis y estudio de librerías para tratamiento de imágenes (100 horas)
- Integración y pruebas de detección de rostros (50 horas)
  - Sólo número de caras
  - Intentar identificar otros parámetros (número, sexo, grupo de edad)

- Estudio y experimentación con las tecnologías a usar en el servidor y selección de las mismas (120 horas)
    - Gestión y almacenamiento de datos
    - Servidores web
    - Herramientas de diseño y programación web
- Prototipo de medidor de audiencia simple (50 horas)
- Pruebas de envío de datos al servidor (50 horas)
- Desarrollo de la aplicación web que gestione la información y permita su visualización en un navegador (150 horas)
- Prototipo final integrado (100 horas)
- Optimización (50 horas)
- Pruebas y test (50 horas)
- Memoria del proyecto (100 horas)

Hay que tener en cuenta que el número de horas para cada tarea no será realizada en una sola vez, sino que se repartirán entre las distintas iteraciones del proyecto, según se vayan mejorando y ampliando los prototipos. Tampoco el orden de las tareas aquí mencionadas tiene por qué ser relevante a la hora de desarrollar las mismas, ya que en un proyecto que incluye tanta búsqueda e interacción entre sistemas es fácil que el plan de acción se modifique durante el desarrollo.

Capítulo 4

Metodología

El proyecto se organizará siguiendo un paradigma de desarrollo iterativo e incremental, teniendo los hitos de desarrollo marcados en forma de prototipos.

El desarrollo iterativo es un método de contrucción de productos cuyo ciclo de vida está compuesto por un conjunto de iteraciones, las cuales tienen como objetivo entregar versiones del software. Cada iteración se considera un subproyecto que genera un subproducto del software y documentación, permitiendo al usuario tener puntos de vista y control más rápido.

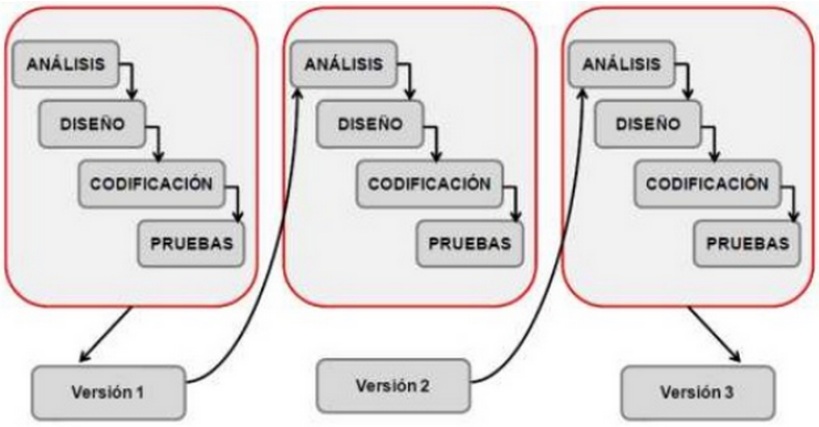

Figura 4.1: Esquema del modelo iterativo de desarrollo

Este paradigma es especialmente útil al tener un doble tutorizado. Por una parte, cada iteración genera un nuevo prototipo que es fácilmente enseñable a los tutores en reuniones con ellos, tanto en la Universidad como

en la empresa Singular Factory, y que en estas reuniones surjan nuevas ideas para el próximo prototipo.

Además, al tratarse de un tipo de desarrollo evolutivo es sencillo que los requisitos cambien a medida que se van sucediendo las iteraciones, aunque también gracias a su naturaleza, es más sencillo adaptarse a los mismos, que como veremos más adelante, se produjeron en más de una ocasión.

Capítulo 5

Arquitectura del Sistema

Normalmente, la selección de la arquitectura a utilizar dependería del resultado y las conclusiones de un proceso de análisis. Sin embargo, debido a que la arquitectura del sistema viene práctimente impuesta desde fuera (formaba parte de la propuesta del proyecto por parte de la empresa Singular Factory), además ser bastante evidente, se trata en un apartado separado antes del análisis.

Además, se muestra al lector antes del proceso de análisis, ya que la arquitectura elegida de antemano facilita la exposición de dicho proceso de forma más organizada y clara.

Debido a la distinta localización de cada una de las partes del sistema, es evidente que necesita ser dividido en módulos. En este caso, parece que la división ideal es ternaria, teniendo por una parte el subsistema que se encontrará en el hogar de los espectadores (de ahora en adelante subsistema cliente) y que es el responsable de recoger los datos de audiencia del hogar, un subsistema que reciba estos datos y los analice (a partir de ahora subsistema servidor de datos) y un subsistema encargado de mostrar estos datos de forma comprensible. Este último subsistema, en base a decisiones que se expicarán más adelante, muestra estos datos en un entorno web, por lo que lo llamaremos subsistema servidor web.

Cabe destacar que la ubicación física de los subsistemas servidor de datos y servidor web puede ser en máquinas distintas o en la misma, aunque no por ello dejan de ser subsistemas separados entre sí.

A continuación, entraremos a estudiar ligeramente más en profundidad

las funcionalidades, necesidades y expectativas de cada uno de los subsistemas, aunque entraremos a trabajar estos apartados en profundidad en los capítulos siguientes.

5.1. Subsistema cliente

Este subsistema es el encargado de mostrar el canal de televisión seleccionado por el usuario, realizar un registro del mismo canal, junto con el momento temporal, los espectadores y el resto de datos que se consideren oportunos. También es el responsable de enviar dichos datos al servidor utilizando el protocolo elegido.

Los datos de audiencia son obtenidos a través de una aplicación basada en visión por computador y que obtiene las imágenes a través de una pequeña cámara instalada en la misma habitación donde la familia ve la televisión (o la televisión principal en el caso de disponer de más de una).

5.2. Subsistema servidor de datos

El servidor de datos trata la información recibida de los clientes en el hogar realizando tareas de *data mining*, haciéndolos fácilmente manejables a la hora de representarlos. Además, es el encargado de almacenar todos estos datos a largo plazo en una base de datos y ponerlos a disposición del servidor web cuando los necesite. Para realizar sus tareas deberá disponer, como mínimo, de un gestor de bases de datos y alguna forma diseñada para mantener accesibles y ordenados los datos que recibe.

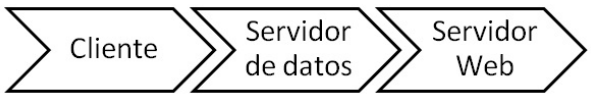

Figura 5.1: Flujo de datos entre los módulos del sistema

5.3. Subsistema servidor web

El servidor web es el encargado de generar una aplicación web a través de la cual los usuarios puedan consultar los datos disponibles en el subsistema servidor de datos, que deben ser representados de forma comprensible y útil para los mismos.

Capítulo 6

Análisis

El objetivo del proceso de análisis es la obtención de una especificación detallada del sistema que satisfaga las necesidades de los usuarios o clientes y sirva de base para el posterior diseño del sistema. Comenzaremos por llevar a cabo una descripción inicial del sistema, más en profundidad que en la introducción del documento, entrando en características adicionales como los factores legales a tener en cuenta o el alcance del sistema o los casos de uso del sistema.

Posteriormente se entra en la identificación y estudio de los casos de uso según los actores que interactúan con las distintas partes del proyecto, clasificándolos según el subsistema con el que están relacionados.

6.1. Definición del sistema

Actualmente en España y la mayoría de países se dispone de una o más empresas especialmente dedicadas al control y medición de audiencias televisivas. En la mayoría de los casos, esta medición es realizada mediante audímetros manuales instalados en el hogar de algunos telespectadores cuidadosamente elegidos, formando una muestra extrapolable al resto de la población. Con este proyecto pretende darse un paso más en la medición de audiencias, acercándonos a la automatización del proceso.

Los principales motivos para perseguir la eliminación de la interacción del espectador con el audímetro son los siguientes:

- Aumento en la comodidad del espectador.

  Al no estar el espectador forzado a registrar constantemente su actividad televisiva, la invasión en su actividad es notablemente menor. Debido a esto, se espera que un mayor número de espectadores estén dispuestos a participar en los sistemas de medición de audiencias, aumentando así el tamaño de la muestra y por lo tanto reduciendo la posibilidad de obtener una información sesgada.

- Reducción en la sensibilidad del sistema debido a la indiferencia o vergüenza de los espectadores.

  Es perfectamente normal, e incluso característico del ser humano, la pérdida de interés e incluso el hastío al realizar una tarea determinada a lo largo del tiempo. Esto en el sistema de audiometría actual se traduce en que a medida que el tiempo pasa, los datos de audiencia de un espectador concreto se van volviendo menos fiables, ya que cada vez le resulta más pesado registrar sus idas y venidas. Sin embargo, si se le fuerza de alguna forma (como los pitidos mencionados en el apartado del estado del arte) se produce una gran invasión e irritación en el usuario, llegando incluso al punto en el que puede desear mentir en los datos debido a las molestias que se le están cansando.

  Además, hay que tener en cuenta la posibilidad de que ciertos espectadores, por motivos sociales como la vergüenza o el complejo, no quieran compartir que acostumbran a ver ciertos contenidos, como casos de hombres que ven programas generalmente orientados al público femenino o adultos que disfrutan de ciertas emisiones para niños, como series de dibujos animados. Todos estos casos introducen ruido e imprecisiones en las mediciones de audiencia.

Cuanto más precisos y garantizables sean los datos de audiencia, más útiles son y por lo tanto mayor es su valor económico.

Para llevar a cabo esta automatización, se confía la tarea de detección de personas a la tecnología de detección de rostros. Esta tecnología es ampliamente utilizada en mútiples circunstancias como en las cámaras fotográficas digitales, permitiendo centrar el enfoque de la fotografía alrededor de las personas presentes en la misma. Existe la posiblidad de utilizar otro tipo de sensores instalados en la habitación, como sensores de presión en un

sillón para detectar el número de espectadores, aunque de las alternativas disponibles, es la detección de rostros a través de imagen la que puede aportar una mayor cantidad de información.

Es necesario encontrar para este subsistema un detector de rostros, a ser posible gratuito, que no sólo reconozca el número de espectadores, sino que además ofrezca algún tipo de información adicional de forma que los datos de audiencia estén enriquecidos y puedan compararse a los obtenidos hoy en día. En caso contrario, a pesar de tratarse de datos con mayor precisión, al ofrecer menos tipos de datos esto les restaría mucho valor real. A priori, el reconocimiento de sexos parece ser una tecnología conocida y ya utilizada en ciertas ocasiones, por lo que es deseable y factible incorporarlo al proyecto, así como estudiar qué otros atributos pueden obtener los distintos reconocedores.

Junto con el sistema de reconocimiento, es evidentemente necesario mostrar los canales de televisión al usuario. Por ello, es necesario un dispositivo en forma de televisión o pc con el software y hardware adecuado para esta tarea.

Una vez obtenida la información es necesario enviarla y realizar con ella las tareas necesarias para su organización y posterior representación, de forma que sean útiles para los posibles clientes. Para llevar a cabo el almacenamiento y análisis de grandes volúmenes de información existen múltiples herramientas, tanto gratuitas como de pago, y por ello es necesario realizar un estudio exhaustivo sobre las mismas.

Una vez la información esté lista hay que ponerla a disposición de los usuarios. El servidor web debe ofrecer la información de alguna manera, que será diseñada más adelante, aunque a priori se manejan las alternativas de crear informes que cubran un cierto intervalo temporal (mensuales, trimestrales...), o la generación dinámica de gráficas o tablas en función de la información solicitada.

6.2. Identificación del entorno tecnológico

El entorno tecnológico necesario se basa en las necesidades de infraestructura tecnológica, divididas en función de los distintos elementos que componen el sistema.

Reproductor

El sistema reproductor donde el espectador podrá visualizar el contenido televisivo debe ser capaz de ejecutar aplicaciones con unas características determinadas, incluido el uso de una cámara para la detección de rostros. Se barajan como opciones las ya mencionadas SmartTVs o un pequeño PC con Linux. En caso de utilizar un PC, el mismo debe ser silencioso y poco invasivo en el salón del espectador.

Servidor

La máquina que se ejecute como servidor debe disponer de un ancho de banda considerable para poder interactuar con los aplicaciones cliente sin saturarse, y a su vez capacidad de cómputo para llevar a cabo la minería de datos. Además, será el responsable de atender las peticiones por parte de los usuarios finales que desean conocer los datos de audiencia.

Cámara

La cámara usada en el sistema cliente debe ser de buena resolución y con adaptabilidad a los cambios de luz, así como con conectores compatibles con los del televisor o PC. Ante el amplio mercado de cámaras disponibles, se llevará a cabo un estudio posterior.

6.3. Especificación de Éstandares y Normas

Según la LOPD (Ley Orgánica de Protección de Datos) un dato de carácter personal es cualquier información numérica, alfabética, fotográfica, acústica o de cualquier otro tipo concerniente a personas físicas identificadas o identificables, tanto la relativa a su identidad (como nombre y apellidos, domicilio, filiación, una fotografía o video, etc...) como la relativa a su existencia y ocupaciones (estudios, trabajo, enfermedades, etc.)

Es evidente que nuestro sistema se ve afectado por la Ley de Protección de Datos ya que toma imágenes de los espectadores en sus hogares, que constituyen un dato de carácter personal. Debido a esto, es necesario, por una parte que la autorización pertinente figure en el contrato a la hora de instalar el sistema de audiometría del espectador, y por otra, que estas imágenes sean seguras y no accesibles por terceros.

Esta restricción legal añade un criterio de seguridad que deberá ser tenido en cuenta a la hora de diseñar la aplicación.

6.4. Identificación de usuarios finales

En un servicio en el que se obtienen datos de un colectivo siempre disponemos de dos conjuntos de usuarios finales: el formado por la muestra de la población estudiada y aquellos interesados en adquirir y utilizar los datos obtenidos.

- Espectadores que participan el sistema de medición de audiencia.
    - Son todos aquellos espectadores que tienen instalado un audímetro automatizado instalado en casa. Por lo tanto, forman la muestra estudiada.
- Clientes interesados en adquirir los datos de audiencia.
    - Generalmente, estos datos son principalmente solicitados o adquiridos por las cadenas televisivas, las empresas publicistas, y todas aquellas empresas que deseen publicitarse en telivisión y tengan poder adquisitivo suficiente para comprarlos.

6.5. Establecimiento de requisitos

Los requisitos del sistema son aquellas características importantes (o imprescindibles, en algunos casos) que deben cumplirse para considerar que el producto final cumplirá con sus objetivos de forma satisfactoria. A continuación se presentan, como de costumbre, separados en función del subsistema al que pertenecen. Además, dentro de cada sección los requisitos están ordenados por prioridad.

- Requisitos del subsistema cliente en el hogar
    - La aplicación debe mostrar al espectador el canal solicitado.
    - El sistema debe ser capaz de recoger los datos de audiencia que se deseen.

- La cámara debe tener una resolución y calidad de imagen suficiente para llevar a cabo una buena tarea de detección de rostros.
- El subsistema de ser ser capaz de enviar los datos de audiencia al subsistema servidor.
- La aplicación deberá ser estable, y capaz de recuperarse en caso de error o cierre inesperado.
- El espectador debe ser capaz de interactuar con el sistema como si se tratara de un televisor corriente.
- Los datos manipulados por la aplicación deberán seguir el tratamiento exigido por la Ley de Protección de Datos.

- Requisitos del subsistema servidor de datos
  - El servidor debe ser capaz de recibir los datos de múltiples subsistemas cliente.
  - El subsistema debe almecenar la información recibida de forma útil para su uso.
  - La aplicación listener (la encargada de recibir la información) deberá ser estable, y capaz de recuperarse en caso de error o cierre inesperado.
  - El subsistema debe disponer de la capacidad de almacenamiento, ancho de banda y la potencia de cálculo suficientes para realizar sus tareas.
  - El servidor debe ser seguro, tanto para evitar robos de información como contra la inyección de información falsa.

- Requisitos del servidor web
  - La aplicación web debe mostrar la información solicitada de forma clara y *human readable*
  - La aplicación debe ser lo más segura posible, evitando los ataques más populares como la inyección de JavaScript en los formularios.
  - El tiempo desde que la información se solicita hasta que se muestra debe ser aceptable. Teniendo en cuenta los volúmenes de infomación manejados, ciertas demoras son esperables.

- Debe ofrecer cierta flexibilidad al usuario a la hora de elegir qué datos quiere consultar.
- Es recomendable que sea accesible desde todo tipo de dispositivos con navegador y usable a cualquier resolución.

6.6. Especificación de casos de uso

Un caso de uso es una secuencia de transacciones que son desarrolladas por un sistema en respuesta a un evento que inicia un actor sobre el propio sistema. Un actor es una entidad externa al sistema que se modela y que puede interactuar con él, o partes del sistema que deben llevar a cabo acciones determinadas en respuesta a determinados eventos. El objetivo del diagrama de casos de uso es representar los distintos requisitos funcionales que se esperan de una aplicación o sistema y cómo se relaciona con su entorno (usuarios u otras aplicaciones o sistemas).

En el caso de nuestro sistema, disponemos de dos tipos de actores distintos: los usuarios finales, ya sean tanto los espectadores cuyo consumo televisivo queremos analizar, como clientes que desean adquirir y utilizar la información obtenida, y las distintas partes de los subsistemas que deben realizar sus respectivas tareas y comunicarse los resultados entre sí.

A continuación, se presenta el diagrama de casos de uso del sistema en formato UML. Sin embargo, debido a motivos de espacio sobre el papel, se dividarán en tres imágenes representando en cada una de ellas los casos de uso relacionados con cada subsistema. Las relaciones entre los distintos subsistemas deberían ser bastante intuitivas debido a sus nombres, aunque se detallarán posteriormente en la descripciones individuales de cada caso de uso.

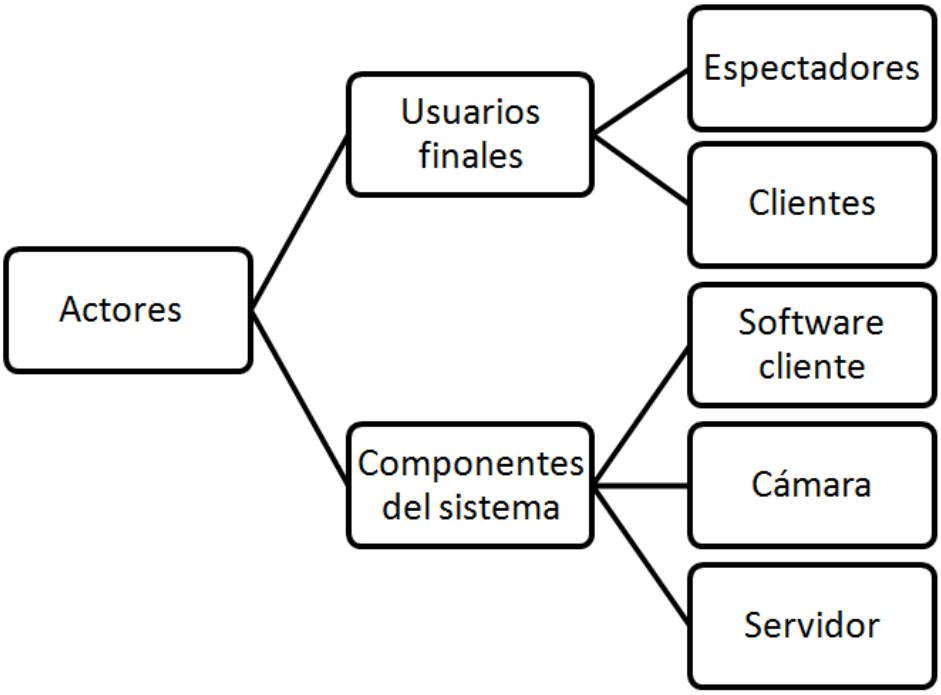

Figura 6.1: Esquema de los actores que interactúan en el sistema

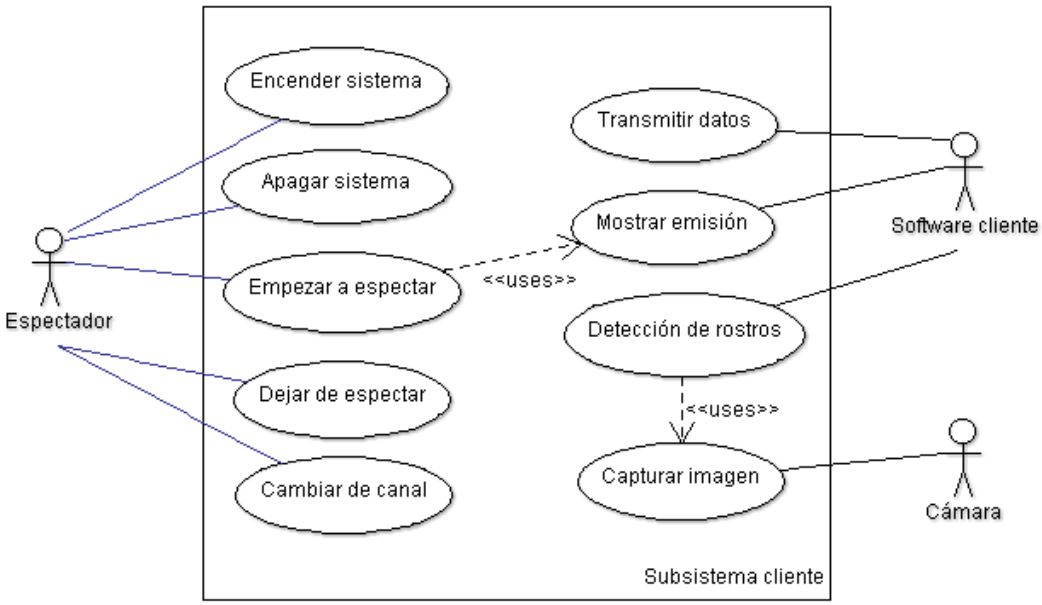

Figura 6.2: Casos de uso del subsistema cliente

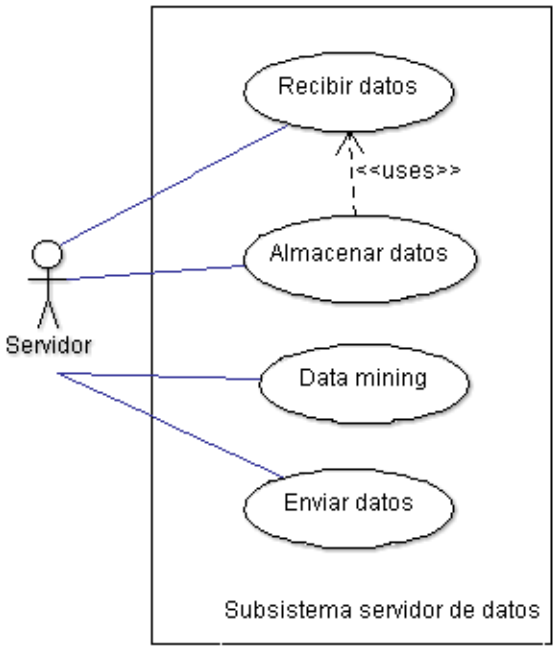

Figura 6.3: Casos de uso del subsistema servidor de datos

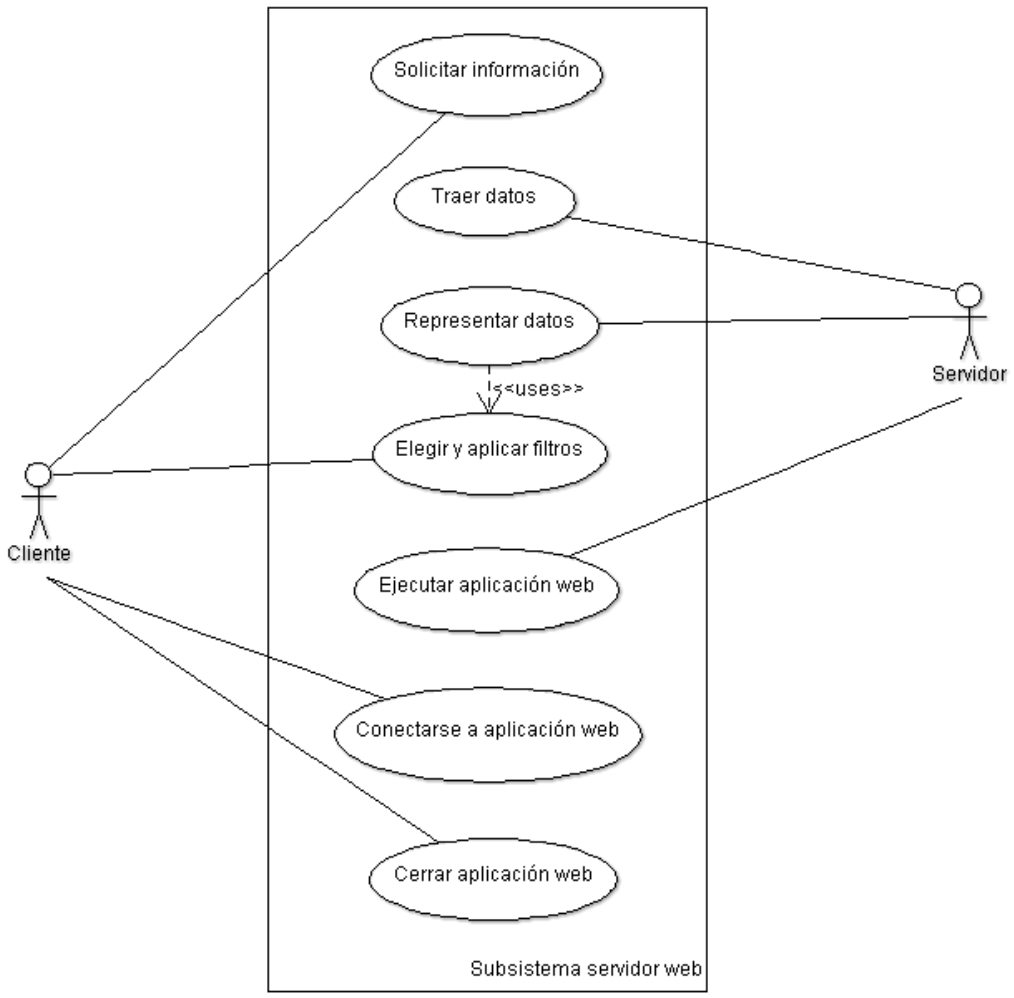

Figura 6.4: Casos de uso del subsistema servidor web

6.7. Descripción de casos de uso

En este apartado se incluye una ficha con la información relevante sobre cada uno de los casos de uso mostrados anteriormente. Además, en estas fichas se podrán distinguir más claramente los puntos de intercambio de información entre los subsistemas.

Nombre	Encender sistema.
Subsistema	Cliente.
Actores	Espectador.
Descripción	El espectador enciende el sistema cliente en el hogar, dispuesto a ver la televisión. Comienza a su vez la medición de audiencia.
Precondiciones	El sistema ha de estar apagado para poder encenderlo.
Postcondiciones	El sistema queda encendido, pudiendo realizarse los demás casos de uso relacionados con el mismo.
Excepciones	Ninguna.

Nombre	Apagar sistema.
Subsistema	Cliente.
Actores	Espectador.
Descripción	El espectador apaga el sistema cliente en el hogar, puesto que no va a ver más la televisión por el momento. Se interrumpe la medición de audiencia para ese hogar.
Precondiciones	El sistema ha de estar encendido para poder apagarlo.
Postcondiciones	El sistema queda apagado, impidiendo la realización de los demás casos de uso relacionados con el mismo.
Excepciones	Ninguna.

Nombre	Empezar a espectar.
Subsistema	Cliente.
Actores	Espectador.
Descripción	El espectador comienza a ver la televisión, ya sea el primero o uniéndose a alguien que ya esté haciéndolo en el hogar.
Precondiciones	El sistema ha de estar encendido, y mostrando la programación.
Postcondiciones	El espectador queda listo para ser detectado.
Excepciones	Ninguna.

Nombre	Dejar de espectar.
Subsistema	Cliente.
Actores	Espectador.
Descripción	El espectador deja de ver la televisión.
Precondiciones	El espectador debe haber comenzado a ver la televisión previamente.
Postcondiciones	El espectador no puede ser detectado.
Excepciones	Ninguna.

Nombre	Cambiar de canal.
Subsistema	Cliente.
Actores	Espectador.
Descripción	El espectador cambia el canal de televisión. Es una acción que en caso de que haya más de un espectador afecta a todos los presentes, ya que todos pasan a ver el nuevo canal.
Precondiciones	El sistema ha de estar encendido.
Postcondiciones	En los datos de audiencia se comienza a registrar el nuevo canal.
Excepciones	Ninguna.

Nombre	Transmitir datos.
Subsistema	Cliente.
Actores	Software cliente.
Descripción	El software envía los datos de audiencia al subsistema servidor de datos a través de la conexión a internet.
Precondiciones	El sistema ha de estar encendido y deben existir datos de audiencia.
Postcondiciones	Los datos de audiencia del hogar pasan a estar disponibles en el servidor de datos.
Excepciones	1- Problemas con la conexión a internet. 2- El servidor no está disponible.

Nombre	Mostrar emisión.
Subsistema	Cliente.
Actores	Software cliente.
Descripción	El software muestra en la pantalla el canal que el o los espectadores han elegido.
Precondiciones	El sistema ha de estar encendido.
Postcondiciones	
Excepciones	1- Problemas con la recepción de la señal TDT.

Nombre	Capturar imagen.
Subsistema	Cliente.
Actores	Cámara.
Descripción	La cámara recoge imágenes que pone a disposición del sotware para sus tareas de reconocimiento.
Precondiciones	El sistema debe estar encendido.
Postcondiciones	El sistema dispone de imágenes sobre los que aplicar la detección de rostros.
Excepciones	1- Problemas de conectividad entre la cámara y el resto del hardware. 2- Defectos en la cámara que producen una captura de imagen errónea.

Nombre	Detección de rostros.
Subsistema	Cliente.
Actores	Software cliente.
Descripción	El software contabiliza el número de expectadores y obtiene información adicional sobre los mismos en base a las imágenes obtenidas por la cámara.
Precondiciones	El sistema debe estar encendido, y la cámara debe estar recogiendo imágenes.
Postcondiciones	El sistema conoce el número de espectadores presente y otros atributos sobre los mismos, por lo que puede generar los datos de audiencia.
Excepciones	1- Las condiciones de luz son insuficientes para realizar la tarea de forma correcta o consistente.

Nombre	Recibir datos.
Subsistema	Servidor de datos.
Actores	Servidor.
Descripción	El servidor recibe los datos de audiencia de los distintos clientes en los hogares de los espectadores.
Precondiciones	El servidor debe estar en marcha y listo para recibir los datos.
Postcondiciones	El servidor dispone de nuevos datos que almacenar junto con los ya presentes, si existen.
Excepciones	1- Problemas en la conexión a internet del servidor.

Nombre	Almacenar datos.
Subsistema	Servidor de datos.
Actores	Servidor.
Descripción	El servidor almacena los datos de audiencia recibidos.
Precondiciones	El servidor debe estar en marcha y haber recibido datos previamente.
Postcondiciones	El servidor dispone de nuevos datos almacenados para el proceso de *data mining*.
Excepciones	

Nombre	Data mining.
Subsistema	Servidor de datos.
Actores	Servidor.
Descripción	El servidor gestiona y organiza los datos de audiencia presentes de forma que seán fácilmente accesibles y utilizables. Este caso se ejecuta de forma pasiva cada cierto tiempo o al recibir datos nuevos.
Precondiciones	El servidor debe estar en marcha y disponer de datos almacenados.
Postcondiciones	El servidor dispone de de datos listos para ser utilizados en la aplicación web.
Excepciones	

Nombre	Enviar datos.
Subsistema	Servidor de datos.
Actores	Servidor.
Descripción	El servidor envía al servidor web la información que éste le ha solicitado.
Precondiciones	El servidor debe estar en marcha y haber recibido una solicitud de datos por parte del servidor web.
Postcondiciones	El servidor web dispone de información para mostrar en la aplicación web.
Excepciones	1- Problemas en la conexión a internet del servidor.

Nombre	Solicitar información.
Subsistema	Servidor web.
Actores	Cliente.
Descripción	El cliente solicita los datos de audiencia deseados en el servidor web.
Precondiciones	El servidor web debe estar en marcha y la aplicación web disponible.
Postcondiciones	La aplicación web dispone de la información necesaria para ser representada.
Excepciones	

Nombre	Traer datos.
Subsistema	Servidor web.
Actores	Servidor.
Descripción	El servidor web solicita y recibe la información que necesita desde el servidor de datos.
Precondiciones	El servidor de datos debe estar disponible.
Postcondiciones	La aplicación web dispone de la información que el cliente ha solicitado.
Excepciones	1- La solicitud realizada es inválida o el cliente no dispone de permisos para verla.

Nombre	Representar datos.
Subsistema	Servidor web.
Actores	Servidor.
Descripción	La aplicación web muestra los datos solicitados por el cliente.
Precondiciones	El cliente ha solicitado visualizar unos datos de audiencia.
Postcondiciones	
Excepciones	1- Fallo en la conexión entre el cliente y el servidor.

Nombre	Elegir y aplicar filtros.
Subsistema	Servidor web.
Actores	Cliente.
Descripción	El cliente elige los datos de audiencias que quiere conocer.
Precondiciones	La aplicación web funciona correctamente.
Postcondiciones	La aplicación web conoce los datos que debe filtrar.
Excepciones	1- Fallo en la conexión entre el cliente y el servidor.

Nombre	Conectarse a la aplicación web.
Subsistema	Servidor web.
Actores	Cliente.
Descripción	El cliente abre la aplicación web en un navegador de su elección.
Precondiciones	La aplicación web funciona correctamente.
Postcondiciones	El cliente se encuentra frente a la aplicación web, listo para consultar los datos que desee conocer.
Excepciones	1- Fallo en la conexión entre el cliente y el servidor.

Nombre	Conectarse a la aplicación web.
Subsistema	Servidor web.
Actores	Cliente.
Descripción	El cliente abre la aplicación web en un navegador de su elección.
Precondiciones	La aplicación web funciona correctamente.
Postcondiciones	El cliente se encuentra frente a la aplicación web, listo para consultar los datos que desee conocer.
Excepciones	1- Fallo en la conexión entre el cliente y el servidor.

Nombre	Cerrar la aplicación web.
Subsistema	Servidor web.
Actores	Cliente.
Descripción	El cliente cierra la aplicación web que tenía abierta.
Precondiciones	La aplicación web funciona correctamente, y el cliente ya la tenía abierta.
Postcondiciones	El cliente abandona la aplicación web, y ya no puede consultar los datos de audiencia.
Excepciones	

Capítulo 7

Herramientas utilizadas y experimentación

Como veremos a continuación, aunque los objetivos del proyecto están claros, se presentan una serie de alternativas a nivel tecnológico a la hora de llevarlo a cabo. Por ello, fue necesario realizar una etapa de investigación y experimentación previa a la toma de decisiones sobre las herramientas a utilizar. Aunque es atípico desarrollar estos apartados de forma conjunta, al estar tan interrelacionados en esta ocasión, se presentan de forma simultánea de forma que el lector pueda conocer una por una las disyuntivas que se presentaron y la solución tomada en cada una de ellas antes de presentarle la siguiente.

A pesar de que es común separar un apartado como este en herramientas software y herramientas hardware, debido a la extensión y a la gran cantidad de alternativas que se presentaron al realizar el estudio para encontrar las herramientas adecuadas, dividiremos en su lugar este capítulo por subsistemas, al igual que los anteriores. Como nota adicional, cabe mencionar que las decisiones tomadas en las herramientas a utilizar fueron variando a lo largo del desarrollo del proyecto, como iremos viendo en el capítulo de desarrollo.

7.1. Herramientas del subsistema cliente

La primera decisión que hubo que tomar fue sobre la herramienta a utilizar para la visualización de los canales de televisión. A priori la intención, en caso de ser viable, era utilizar una SmartTV, de forma que el único dispositivo adicional necesario fuera una pequeña cámara, que en algunos modelos hasta era posible que viniera integrada en el marco del televisor. La alternativa sería utilizar un PC como dispositivo que recibe la señal de TDT (con el receptor adecuado) y que la mostrara en un televisor o monitor. Existen ordenadores de muy pequeñas dimensiones con capacidad de cálculo suficiente para realizar esta tarea, por lo que tampoco es un gran inconveniente.

SmartTV

La televisión inteligente (traducido del ingés SmartTv) es un nuevo concepto que describe la integración de Internet y de las características Web 2.0 a la televisión digital. Estos dispositivos se centran en los medios interactivos en línea, en la televisión por Internet y en otros servicios como el vídeo a la carta. Además, las compañías suelen poner un SDK a disposición de los desarrolladores que quieran crear aplicaciones que se ejecuten en estos entornos, y es habitual ver pequeños juegos, servicios de grabación programada, alquiler de contenidos, y otros muchos conceptos de aplicaciones. Por lo tanto, se llevó a cabo un estudio sobre las diversas marcas y modelos de SmartTV y los SDK que se ofrecen. En particular se estudiaron las SmartTV de Samsung, LG y Sony, ya que parecen ser las más completas y avanzadas del mercado.

Fue necesario descargar los SDKs disponibles para cada una de estas plataformas y realizar un estudio de viabilidad para nuestro subsistema cliente en cada una de ellas. Se presentan las acciones llevadas a cabo y las conclusiones obtenidas al finalizar estos estudios.

El primer SDK sobre el que se decidió investigar, principalmente debido a la propuesta de la empresa SingularFactory, fue el de Samsung. Por lo tanto, fue necesario registrarse en la web de desarrolladores de Samsung (de forma gratuita) y descargar el entorno de desarrollo que allí ofrecen. Tras su instalación se procedió a, junto con una lectura rápida de la documentación

Figura 7.1: Smart TV de Samsung, donde se pueden observar el menú de aplicaciones

del mismo, llevar a cabo un ligero prototipo de aplicación que simplemente mostrara datos en pantalla. Durante el desarrollo se pudo observar que el SDK incorpora un simulador, de forma que el desarrollador pueda comprobar como interactuaría su aplicación con un mando a distancia tradicional de televisión.

Las aplicaciones para este sistema se desarrollan utilizando HTML5, CSS y Javascript, por lo que funcionan de forma análoga a aplicaciones web y no fue necesario el estudio de nuevos lenguajes de programación, y con descubrir unas pocas funciones ofrecidas por la API fue suficiente.

La desagradable sorpresa vino cuando se pudo comprobar que no era posible crear aplicaciones transparentes o que ocuparan únicamente parte de la pantalla, de forma que pudieran verse los canales de televisión a través de la aplicación, ni que hubiera forma de redirigir el contenido de dichos canales a las aplicaciones. Las aplicaciones se ejecutaban en un entorno embotellado, completamente aislado de la reproducción de los canales de televisión. Debido a este gran inconveniente, fue necesario abandonar el SDK de Samsung por el momento.

Sin embargo, debido a que algunas de las aplicaciones predefinadas de Samsung si eran capaces de interactuar con los canales de televisión, se llegó a la conclusión de que el SDK interno de la compañía no presentaba este inconveniente. Debido a ello se realizó una solicitud formal a Samsung para utilizar el sistema de desarrollo interno de la empresa, explicándoles los objetivos del proyecto y esperando que mostraran cierto interés en el mismo y

Figura 7.2: Emulador del SDK de Samsung, mostrando una aplicación de ejemplo

nos dieran los medios que necesitábamos para realizar la tarea. Esta petición fue realizada por la empresa SingularFactory, ya que suponíamos que tendría más peso que la petición individual de un programador.

Obviamente, este tipo de solicitudes tienen un largo tiempo de respuesta, y no íbamos a estar esperando sin hacer nada, especialmente teniendo en cuenta la alta probabilidad de rechazo de la solicitud o incluso la ausencia de respuesta. Sobre la marcha se siguieron estudiando los otros dos SDK que mencionamos anteriormente. Como nota, nombrar que la respuesta de Samsung llegó meses más tarde, denegando la solicitud.

El siguiente SDK con el que se decidió comenzar a realizar pruebas fue el de Sony. Sin embargo, se descubrió que Sony está fuertemente asociado a la plataforma GoogleTV, que utiliza su propio sistema de distribución de contenidos, y las aplicaciones del SDK se ejecutan en ese entorno, y se desarrollan utilizando el SDK tradicional de Android. Nuevamente, se realizaron las mismas pruebas que en el caso del SDK de Samsung, y desgraciadamente las conclusiones fueron las mismas: seguía sin ser posible comunicar las aplicaciones con la reproducción de los canales televisivos o extraer información de los mismos.

Por último, aunque ya con pocas esperanzas, se repitió el proceso por el SDK de los televisores de LG. Aunque el proceso de instalación fue un poco más extraño que los anteriores, debido a la necesidad de utilizar una máquina

virtual (necesidad impuesta por LG), pronto estuvo listo. El formato de las aplicaciones a desarrollar, desgraciadamente, era el mismo que el de Samsung: aplicaciones en HTML5 que se ejecutan en un entorno encapsulado. Nuevamente, un SDK que no nos sirve.

Viendo que ninguno de los SDK convencionales de las SmartTV nos era válido para el proyecto, se planteó una última opción antes de descartar por completo las SmartTV: los sistemas operativos alternativos, que sustituyen al sistema incorporado en el televisor. El resultado de la búsqueda fue SamyGO. SamyGO[1] es un firmware personalizado creado para televisores inteligentes de Samsung, y la inversión temporal en investigación sobre el mismo fue notable. Sin embargo, fue finalmente descartado debido a la heterogeneidad del sistema entre distintos modelos de televisor, y el hecho de que solo funcionaba de forma segura y estable en los modelos mas antiguos de SmartTV, que a día de hoy ya son complicados de conseguir. Esto, junto con el riesgo al instalar un nuevo firmware de volver inservible un aparato de coste económico elevado, hicieron que esta opción fuera descartada antes de intentar ponerla en prueba.

Debido a todos estos escollos encontrados en el camino de las SmartTV, se optó por cambiar de entorno a la segunda opción que barajábamos desde un principio: un PC con un sistema Linux, que será preparado para cubrir las necesidades del proyecto. Al optar por seguir este camino, surge una nueva necesidad en forma de un receptor de señal de televisión y el software adecuado para mostrar las emisiones al espectador, así como la elección de una distribución de Linux acorde a las necesidades del proyecto.

PC

A continuación se realiza un análisis de las distintas alternativas existentes en el mercado para actuar como mini-pc. Una vez presentadas sus características y sus precios, se presentará una conclusión presentando la mejor o mejores alternativas. Sin embargo, para evitar el gasto que supone, durante el desarrollo de la aplicación y las pruebas se utilizaron ordenadores de sobremesa y portátiles convencionales. En caso de llevar el proyecto a la realidad, la aplicación cliente debería instalarse en el dispositivo seleccionado.

[1]Más información sobre SamyGO puede encontrarse en `www.wiki.samygo.tv`

Mac Mini

La primera opción que se presenta, debido a su fama y como punto de referencia, es el Mac Mini. Con su procesador Intel Core i5 de doble núcleo a 2,5 GHz y sus 4GB de memoria RAM, cumple con los requisitos funcionales de forma muy holgada. Se distribuye con sistema operativo OSX Mavaericks, aunque es posible (y necesario para nosotros) formatearlo a través de sus puertos USB para instalar la distribución de Linux que más nos convenga.

Figura 7.3: Mac Mini

En cuanto a la conectividad, presenta salidas de video HDMI y Thunderbolt, siendo la primera la que nos interesa, al ser un conector estándar presente en la mayoría de televisores modernos, y puerto de red Ethernet para la transferencia de datos, así como capacidad wi-fi integrada. Su principal inconveniente es su precio: se comercializa a 650 euros la unidad.

Intel NUC BOXDC3217BY

Ligeramente mayor en tamaño y peso al Mac Mini, esta unidad de Intel presenta una capacidad computacional ligeramente inferior al mismo. Sus cálculos son realizados por un procesador Intel i3 3271U a 1,8 GHz, acompañado por 8 GB de memoria RAM DDR3.

Para gestionar la entrada y salida de datos, este sistema presenta tres puertos USB y una salida de vídeo HDMI. Si esta unidad presenta un inconveniente, es la inexistencia de toma de red por cable y por lo tanto su dependencia de que en el hogar se disponga de una buena señal wi-fi en el punto donde se instale. Su coste es de 350 euros por unidad, algo más de la mitad del Mac Mini.

MSI Wind Box DC110

Este pequeño ordenador, ensamblado por la famosa marca MSI, es una opción algo más económica que la anterior, con cierto recorte en la capacidad

Figura 7.4: Intel NUC BOXDC3217BY

de procesamiento de datos. El procesador encargado del mismo es un Celeron 847 de doble núcleo a 1,1 GHz y cuenta con 2 GBs de memoria RAM. Se distribuye de fábrica con Windows 8 preinstalado, aunque es sencillo formatearlo e instalarle un sistema Linux a través de sus puertos USB.

Figura 7.5: MSI Wind Box DC110

En el apartado de salida de video presenta una novedad: junto con el ya esperable puerto HDMI, encontramos una salida de video VGA, lo que aumenta ligeramente la compatibildiad del dispositivo con monitores. Además, a diferencia del modelo anterior, en esta ocasión sí disponemos de un puerto de red Ethernet, siendo esta una ventaja importante a tener en cuenta, así como sus 6 puertos USB y su salida de audio digital. Su precio es de 320 euros la unidad.

Raspberry Pi model B

Mostrar en este apartado la Raspberry Pi puede resultar algo polémico debido a su baja capacidad computacional. Sin embargo, sabiendo que se utiliza a menudo como Media Center en hogares y otras muchas tareas, junto con su reducido precio y volumen, no me gustaría descartarlo sin poder realizar las pruebas pertinentes. Desgraciadamente, estas pruebas quedan pendientes de realizarse en un futuro. El procesador de la Rasperry Pi es un ARM11 a 700Mhz, lo cual es un recorte substancial con respecto a los modelos mostrados anteriormente. Sin embargo, al tener en cuenta que existen versiones de Linux especialmente diseñadas para ser ultraligeras y utilizadas en este dispostivo, es posible que sea suficiente.

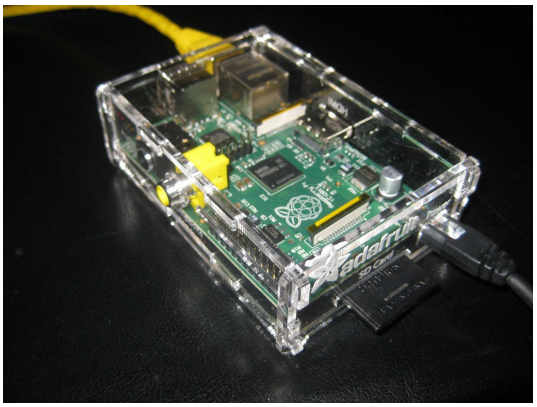

Figura 7.6: Raspberry Pi

El modelo presenta 512 MBs de RAM, 2 puertos USB, salida de video HDMI y utiliza memoria flash como disco duro. Dispone de salida de vídeo HDMI. La conectvidad a internet es a priori exclusivamente mediante cable Ethernet, aunque se suele vender junto a un adaptador wi-fi para USB. Cabe destacar además sus reducidas dimesiones(8.5 x 5.4 centímetros) y su más que bajo consumo eléctrico, de sólo 5 Watios. El paquete con carcasa y receptor wi-fi, que parece el más adecuado para la tarea, ronda los 50 euros.

Como conclusión y basándonos en los criterios ya mostrados, parece que en caso de ser viable la ejecución de la aplicación en la Rasberry Pi, sería esta la adecuada debido a su reducido coste y precio. En caso contrario, se debería optar por utilizar la MSI Wind Box DC110, siguiente en precio y que dispone de la capacidad de cálculo y de toda la conectividad necesaria.

Distribución de Linux

Actualmente el mercado de distribuciones de Linux más populares y prácticas se reducen a elegir un sistema basado en Debian, o bien Fedora (basado en RedHat). Por costumbre y conocimiento previo, así como por el gran número de aplicaciones y repositorios disponibles por defecto, se eligió un sistema Ubuntu, basado en Debian. Durante el desarrollo se han hecho pruebas en distintas versiones de este sistema, y funciona correctamente en todas aquellas que tengan menos de 18 meses. Es posible que funcione correctamente en versiones más antiguas, pero no se garantiza. Aun así, al ser este proyecto un sistema cerrado y no un producto software, el sistema vendrá instalado y preparado de antemano, por lo que no hay riego de conflictos entre distintas versiones.

Receptor TDT

Al decidir basar el subsistema cliente en un PC con Linux, es necesario añadir un dispostivo hardware que se encargue de la recepción de la señal. En el mercado existen múltiples dispositivos que cumplen esta función, aunque únicamente unos pocos de ellos funcionan adecuadamente en Linux [2]. Además, por algún motivo, aquellos que se distribuyen con drivers universales compatibles suelen ser más caros. Afortunadamente, fue posible averiguar la existencia de un dispositivo TDT de bajo coste y completamente funcional en Linux que no requiere la instalación de ningún software adicional, el receptor AverMedia Black HD[3]. Este modelo tiene la ventaja adicional de poder elegir entre la conexión a la toma de antena del hogar donde se instala el dispositivo (el sistema utilizado por la mayoría de televisores) o una pequeña antena desplegable incorporada que, aunque no presente una gran potencia y no sea de ayuda en situciones rodeadas por edificios más altos, puede ser útil en determinadas situaciones donde la antena del edificio no esté disponible.

Reproductor de televisión

Evidentemente, en un sistema para medir las audiencias es necesario mostrar al espectador los canales de televisión que desee ver. Para esta tarea,

[2]Más información acerca de modelos y drivers en `www.linuxtv.org/wiki`

[3]Agradecimientos a D. Jonán Cruz Martín por la mención y el préstamo del aparato para el desarrollo y las pruebas

Figura 7.7: Receptor TDT AverMedia Black HD, mostrando la conexión hembra para antena

existen múltiples aplicaciones de software libre, como Me-TV o Kaffeine. A pesar de que Me-TV dispone de alguna funcionalidad adicional para el espectador, Kaffeine ofrece al desarrollador la posibilidad de consultar la configuración de los canales en un fichero SQLite (una adaptación ligera de MySQL).

Esta característica le otorga una gran utilidad a Kaffeine, ya que es posible indicar al servidor el canal que se está mostrando, requisito indispensable para llevar el proyecto a buen puerto. Por lo tanto, será esta la aplicación a utlizar.

Cámara

Para las labores de desarrollo iniciales no es necesario que la cámara cumpla ningún requisito especial, y cualquier webcam es válida para el objetivo que se persigue. Sin embargo, en una situación real donde las condiciones de iluminación son muy variadas, sería interesante que tuviera un buen balance de blancos (para evitar que situaciones muy luminosas puedan cegar a la cámara) o incluso, para cubrir los casos más extremos, visión por infrarrojos, para aquellas personas que gustan de ver la televisión a oscuras.

Librería de reconocimiento

La librería de reconocimento a utilizar es una decisión muy importante.

De sus características dependerán no sólo los datos que es posible obtener de los espectadores, sino también en gran medida la precisión de la información obtenida.

La librería de tratamiento de imágenes de código libre más popular en el momento es OpenCV. OpenCV no sólo permite crear y modificar imágenes, sino que incluye una multitud de herramientas utilizables en tareas de detección y clasificación de rostros. Sin embargo, el uso de esta librería es de bajo nivel, muy cercano a todas las subtareas que hay que realizar paso a paso para obtener unas detecciones adecuadas, y no incluye las librerías de imágenes necesarias para entrenar el sistema, lo cual complica notablemente su uso. Esto es debido a que el sistema de reconocimento usado por OpenCV está basado en entrenamiento y experiencia, y es necesario una base de datos de rostros fuerte para obtener buenos resultados.

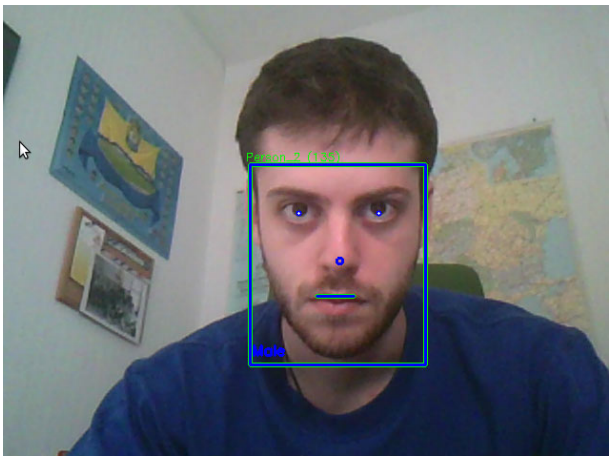

Figura 7.8: Captura de imagen de Encara2, reconociendo al sujeto como hombre

Sin embargo, sería ideal encontrar una librería que actuara como capa de abstracción entre OpenCV y nuestro subsistema, debido a la complejidad de realizar estas tareas directamente con OpenCV. En la propia ULPGC existe una librería de detección de rostros desarrollada por el SIANI llamada Encara2 y, al solicitarle el uso de la misma para el proyecto al Dr. Modesto Castrillón Santana lo permitió de buen grado. Afortunadamente, esta librería incluye, aparte de reconocimiento de rostros, una función adicional de clasificación por sexos, lo cual es útil para segmentar la población. Por lo tanto, los datos de audiencia que se aportarán en esta versión del proyecto serán número de espectadores y segmentación por sexos, aunque cambiando

la librería de reconocimiento podría oferse cualquier información que ésta obtuviera de las imágenes.

Sin embargo, hay que tener en cuenta que existen otras librerías de reconocimiento de rostros con funciones adicionales, y que pueden ser útiles en etapas futuras del proyecto. Una de estas librerías es eMotion, una librería diseñada especialmente para añadir detección de expresiones faciales para detectar el estado emocional de las personas. Incorporando una librería de estas características se podría incorporar un nivel único y novedoso a los datos de audiencia.

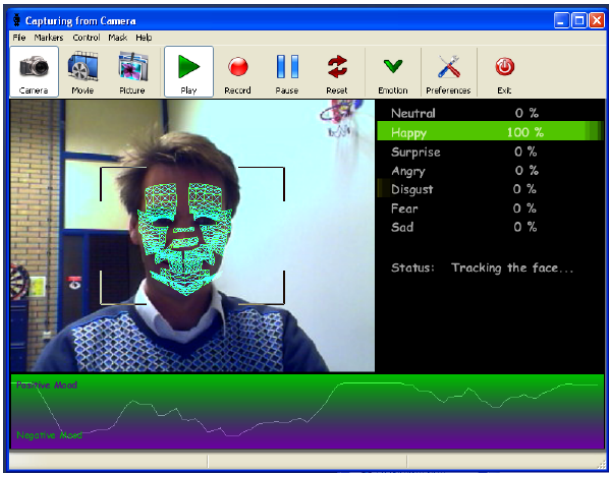

Figura 7.9: eMotion detectando expresiones faciales

7.2. Herramientas del subsistema servidor de datos

Un sistema encargado de gestionar grandes volúmenes de datos requiere de una herramienta de gestión de los mismos y una base de datos en la que almacenarlos. Normalmente, estas herramientas ya vienen diseñadas para trabajar un tipo de base de datos en particular, por lo que la elección realmente se limita a elegir la herramienta a utilizar. Estas herramientas son interesantes de utilizar ya que facilitan enormemente la inserción y extracción de la base de datos, actuando como capa de abstracción intermedia. Se contemplaron las opciones que se muestran a continuación.

Pentaho

Pentaho es una herramienta especialada en lo que ellos llaman *Big Data* o gestión de grandes volúmenes de datos. Esta herramienta pone a disposción del usuario un sistema de visualización de la información muy completo, y con un panel similar al de Google Analytics. Son lideres en el sector y una de las principales ventajas que presentaría para este proyecto son tipos de gráficas adicionales como mapas de calor, de forma que se podría añadir una segmentación geográfica a los datos de audiencia. Funciona sobre MongoDB, una base de datos no relacional de alto rendimiento, capaz de soportar casi diez veces más de peticiones por unidad de tiempo que las bases de datos relacionales como MySQL.

Su único inconveniente es que, al ser una solución profesional pensada para grandes empresas, las licencias son costosas. No se puede incluir un precio exacto debido a que hacen presupuestos individualizados, pero en los casos más baratos son de varios cientos de euros. Debido a ello, y a que realmente en un prototipo sencillo no se van a manejar volúmenes de datos tan grandes, se ha optado por buscar una solución alternativa.

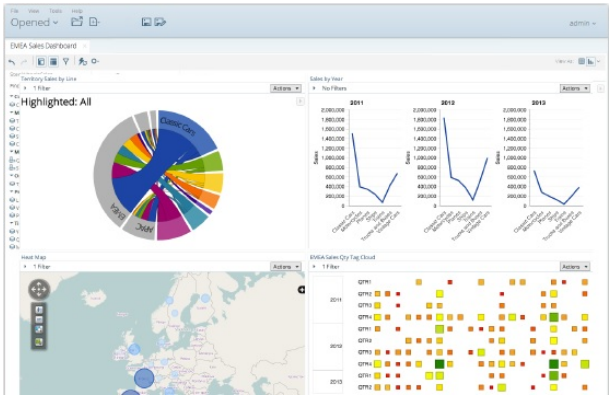

Figura 7.10: Un dashboard de Pentaho mostrando información

FnordMetric

FnordMetric es una herramienta de código libre para la gestión y visualización de datos. Está programado en Ruby y funciona sobre una base de datos Redis, basada en el almacenamiento en tablas de hashes (clave/valor), siendo también de código libre. Aunque no es tan completo como Pentaho, tiene herramientas para almacenar información con órdenes directamente desde aplicaciones, exportar gráficas directamente en html5 de forma que

sean fácilmente embebibles, y la creación de dahboards personalizados en las herramientas administrativas. Sin embargo, se echan en falta opciones adicionales para otros tipos de gráficas y su velocidad de cómputo no es especialmente elevada (sin llegar a ser realmente lento), aunque para la situación que tenemos entre manos es perfectamente válido.

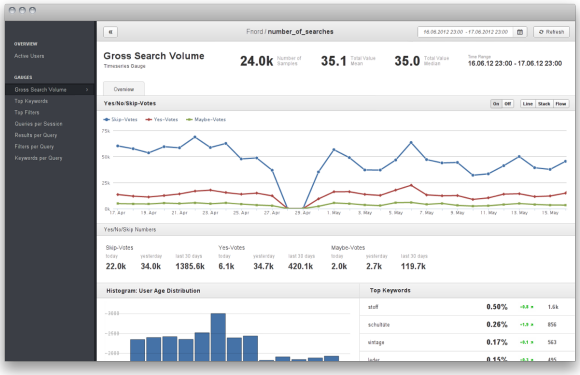

Figura 7.11: Un dashboard de FnordMetric mostrando información

Desarrollo de un sistema propio

Es cierto que el diseño y creación de un software a medida para nuestras necesidades sería una medida excelente, seguramente basándose en sistemas de bases de datos no relacionales, mucho más rápidas que aquellas basadas en tablas. Sin embargo, incluso en el caso de FnordMetric, mucho más sencillo que Pentaho, el volumen de código y su complejidad tanto a la hora de almacenar datos como de representarlos queda muy lejos de la magnitud esperada en un proyecto como este, teniendo en cuenta que ha sido desarrollado por un equipo durante varios años.

Veremos como en las primera etapas del proyecto, antes de llegar a estas conclusiones, se planteaba gestionar los datos con un desarrollo propio, y como posteriormente al tomar conciencia de la complejidad de la tarea y con el oportuno descubrimiento de FnordMetric, se utilizará este último a partir de un punto determinado.

7.3. Herramientas del subsistema servidor web

Servidor web

Cierto es que al desarrollar una aplicación web, la cantidad de opciones que se presentan es enorme. Por un lado, necesitamos un servidor web, como Apache. Aunque actualmente existen alternativas a Apache como NGINX, especialmente diseñado para aplicaciones con un gran volumen de usuarios (utilizado por Facebook, por ejemplo), Apache sigue teniendo el mayor porcentaje de uso con una gran diferencia, y al no tener ninguna necesidad específica que cubrir y conocer ya la herramienta, será que la se utilice.

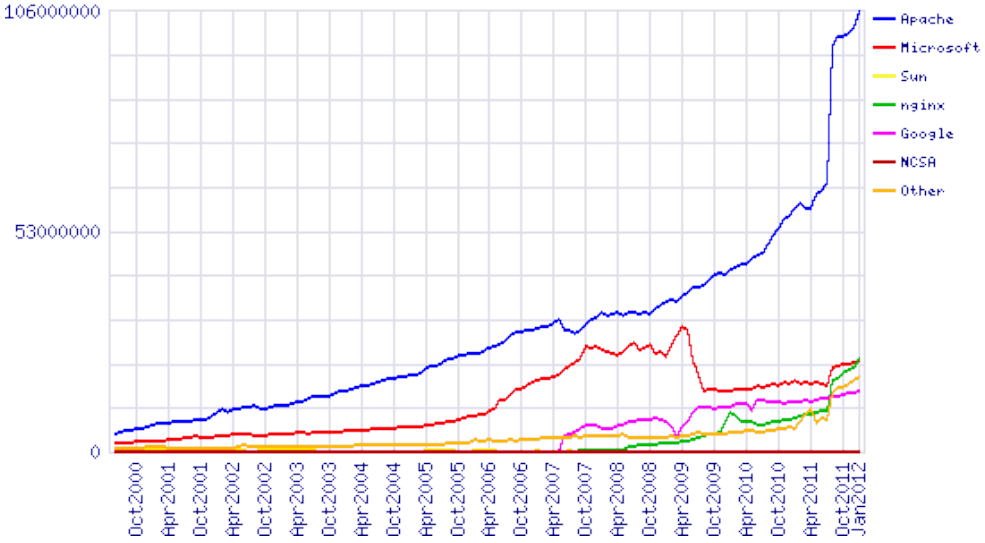

Figura 7.12: Comparativa del uso de servidores web

Lenguaje de programación y CMS/Framework

Resuelto este punto, sobre Apache se pueden utilizar múltiples alternativas en forma de CMS o frameworks, y diversos lenguajes de programación. Actualmente se encuentran en alza Ruby on Rails, Python y el siempre presente PHP. Por su similitud a C y ya que Apache ya viene configurado por defecto para funcionar con este lenguaje, será PHP el utilizado, apoyándose en el uso de JavaScript para algunas funcionalidades. Antes de conocer Fnordmetric, se esperaba que la aplicación web fuera mucho más compleja

de lo que finalmente resultó ser. Por ello, en los primeros prototipos de este subsistema se contaba con el uso de alguna herramienta que facilitara la tarea, en particular algún framework, ya que un gestor de contenido (CMS) está fuera de lugar en una web con este fin. Zend Framework y Symfony2 son dos grandes opciones con características similares, y el único motivo que llevó a la instalación de Symfony2 en lugar de la de Zend fue recomendación por parte de la empresa Singular Factory ya que es ampliamente utilizado allí, de forma que pudiera servirme como experiencia en la herramienta en caso de que en un futuro me uniera a la plantilla de la misma.

Sin embargo, en la versión final dejó de utilizarse este fraework debido a que descubrimientos realizados durante el desarrollo simplificaron el desarrollo web, haciendo la complejidad adicional de un framework innecesaria, y desarrollando la aplicación web directamente en PHP y JavaScript, con la ayuda de unas pocas herramientas adicionales.

JQuery

JQuery es una librería JavaScript ampliamente utilizada, que incluye multitud de características por defecto, desde algunas muy genéricas como creación, modificación y eliminación de elementos del DOM, hasta otras muy específicas como funciones de autocompletado, calendarios, menús, y otras muchas características fácilmente incluibles en cualquier web.

Bootstrap

Bootstrap se define a sí mismo como un "Elegante, intuitivo y potente framewrok de desarrollo para el front-end de aplicaciones web". Y realmente, eso es lo que es. Bootstrap está formado, principalmente, por hojas de estilo CSS que modifican la forma en la que se maqueta el diseño de una web. En lugar de pensar en forma de etiquetas DIV que han de ser dimensionadas a mano, utiliza un sistema de clases basadas en una plantilla (o grid) de 12 columnas, donde nuestros contenidos ocupan un ancho igual al número de columnas que elijamos. Como característica adicional, estas columnas son flotantes y de ancho variable en función de la resolución del usuario, por lo que la aplicación funciona por defecto en pantallas de todos los tamaños, incluidos dispositivos móviles.

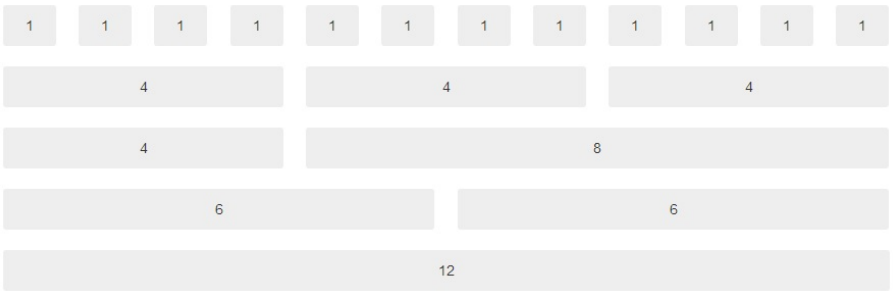

Figura 7.13: Ejemplo del grid de Bootstrap, mostrando el ancho de los distintos elementos en número de columnas

JpGraph

JpGraph es una librería de PHP orientada a la creación de imágenes en un entorno web basándose en una serie de datos. Esta herramienta incluye múltiples tipos de gráficas (aunque algunos de ellos requieren la adquisición de una licencia valorada el 98$), a pesar de que su aspecto no sea demasiado pulido. Se utilizó en las primeras etapas del proyecto, hasta que se cambió el manejo de datos a FnordMetric.

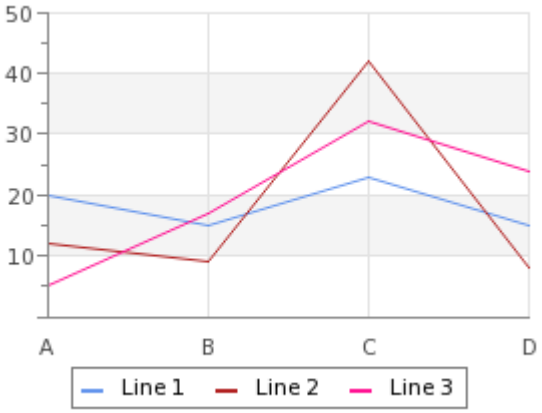

Figura 7.14: Ejemplo de gráfica generada con JpGraph

7.4. Otras herramientas

Aparte de las herramientas específicas usadas en cada subsistema, son necesarias una serie de herramientas adicionales genéricas, más propio del

desarrollo que de el proyecto en sí. Algunas de ellas son genéricas a todos los proyectos (como un editor de textos o un entorno de desarrollo) aunque son incluidas aquí para mostrar algunas opciones preferidas sobre otras, y mis preferencias personales.

Editor de texto o IDE

En una labor de desarrollo, normalmente es necesario o bien un IDE completo que incluya configuración del proyecto, como NetBeans, o bien un editor de texto plano y utilizar las herramientas de compilación en línea de comandos. He optado por esta segunda opción, tanto por adquirir cierta experiencia en la labor de compilación y enlazado, como por predilección de un editor determinado, descubierto durante la realización del proyecto.

Un editor de texto, para ser útil y no poner obstáculos a la productividad durante el desarrollo, debe incluir ciertas características. Dos de estas características son la apertura de carpetas completas en lugar de ficheros, de forma que se pueda disponer del árbol del proyecto en un lateral del editor, y la búsqueda en múltiples ficheros de forma simultánea. Una vez estas características están cubiertas, hay otras muchas que son de agradecer, como las marcas de indentado, minimizado de bloques de código, chequeo de sintaxis y otras tantas. Sin embargo, es complicado encontrar un editor que cuente con todas las opciones que uno desea, y normalmente hay que elegir las características a sacrificar en pos de otras.

Al comenzar el proyecto, el editor de preferencia utilizado era NetBeans. Netbeans dispone de muchas características prácticas, e incluso sus propias herramientas de compilación y enlazado, que nunca llegaron a utilizarse. Su gestión de árboles de ficheros es buena, y la búsqueda en múltiples ficheros, aunque algo lenta si el número de archivos es elevada, es funcional y práctica. Además, dispone de autocompletado y chequeo de sintaxis. Sin embargo, las

opciones de personalización de la interfaz son limitadas y, al estar basado en Java, su velocidad a veces es muy lenta comparada con la de otros editores. Debido a este último problema, cuando algún fichero necesitaba una edición rápida y NetBeans no estaba abierto, utilizaba Gedit, que es a mi parecer un editor completamente insuficiente para la labor de programación, al menos sin instalarle módulos adicionales.

Sin embargo, durante el desarrollo y comentando con compañeros algunas alternativas a estos editores debido a las pequeñas frustraciones que me iban produciendo, encontré lo que hasta a día de hoy es la solución perfecta para mí: SublimeText2. SublimeText es un editor de texto plano especialmente diseñado para labores de programación. Incluye un excelente número de características que todo gran editor puede desear, tanto muchas de las ya comentadas como búsquedas multifichero y apertura de carpetas, así como otras adicionales como la capacidad de configurar todos los aspectos de la interfaz, alta velocidad de respuesta en todas las operaciones, una gran variedad de comandos y macros especialmente diseñados para múltiples lenguajes de programación (y la posibilidad de definir macros nuevas), edición en columna de varios ficheros de forma simultánea, y su característica estrella: una vista previa del fichero en el lateral del fichero, que permite acceder a cualquier parte del código con un solo click. La lista de características es mucho más amplia, pero no serán explicadas aquí[4].

Control de versiones

En una labor de desarrollo siempre es importante disponer de un sistema de control de versiones, que al mismo tiempo cumple como copia de seguridad si el repositorio está ubicado fuera del equipo donde se trabaja, aparte de otras ventajas como el desarrollo por ramas. Por experiencia previa se eligió Git, utilizando como repositorio las herramientas gratuitas de BitBucket.

Editor de imágenes

Durante el proyecto, han sido pocas y contadas las tareas de edición de imagen realizadas (muchas de ellas durante el desarrollo de este documento),

[4]Para más información o descargar el editor, se puede acudir a `www.sublimetext.com`

aunque para ellas ha sido necesario una herramienta adecuada. Por mantenernos utilizando software gratuito se ha utilizado **GIMP**, un software para edición de imágenes digitales libre y de código abierto.

Generador de PDF

Tanto para este documento, como para la mayor parte de la documentación generada durante el proyecto, se ha utilizado **LaTeX** como software para crear los ducumentos PDF. LaTeX es un sistema de creación de textos clasificado como software libre, orientado a la publicación de textos científicos en cualquiera de sus formatos. Cabe destacar del mismo que no es editor de tipo WYSIWYG (What You See Is What You Get, que en castellano vendría a ser algo como "Lo que ves es lo que obtienes") sino que es texto plano con etiquetas que hay que pasar por el generador para obtener el documento final.

Capítulo 8

Desarrollo

Como ya se ha mencionado anteriormente, el paradigma de desarrollo utilizado ha sido el iterativo e incremental, utilizando la generación de prototipos como forma de gestionar los hitos del proyecto. Se van a presentar cuatro prototipos principales en los que se pueden apreciar los cambios más relevantes y la inclusión progresiva de las características principales, aunque realmente sean más los estados intermedios que se han dado. Esta omisión se realiza principalmente para evitar la duplicación de estados muy parecidos y para simplificar la lectura del capítulo.

8.1. Prototipo 1

Al comienzo de este prototipo, solamente se conocía cómo se quería que fuera el proyecto y que herramientas ibamos a utilizar (aunque algunas de ellas estuvieran sujetas a cambios). El principal objetivo inicial era crear un esqueleto del sistema completo, incluyendo cada uno de los tres subsistemas, de forma que en adelante el trabajo se centrara en completar y ampliar este esqueleto.

El primer subsistema que se abordó fue el subsistema cliente en el hogar. Para ello, era necesario conseguir hacer funcionar la librería de reconocimento de rostros, Encara2, al menos hasta el punto de ser capaces de contabilizar el número de rostros detectados. Cierto es que la curva de aprendizaje para aprender a trabajar con la herramienta fue dura, principalmente por un error conceptual: al cabo de un tiempo quedó claro que era más sencillo crear la

aplicación dentro de Encara2, en lugar de intentar incluir Encara2 dentro de un proyecto externo.

Además, se presentó un problema adicional de dependencias: Encara2 depende de OpenCV, y este a su vez de otras muchas librerías. Además, OpenCV se distribuye de forma no compilada, y preparado para ser compilado utilizando la herramienta CCmake. Mi desconocimento sobre esta herramienta era absoluto y llevó un tiempo conseguir una compilación correcta de la librería.

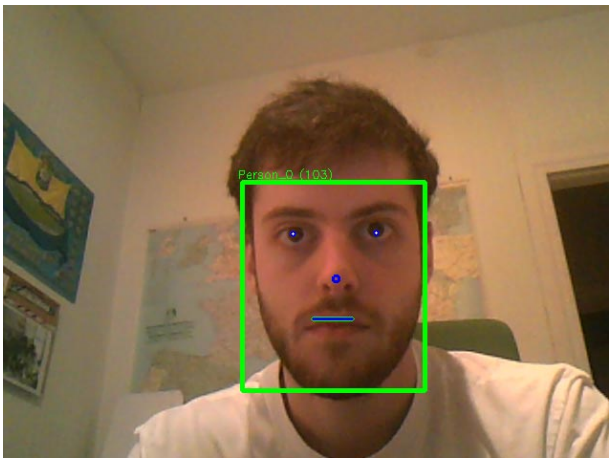

Figura 8.1: Encara2 detectando rostros (sin distinguir sexos)

Por lo tanto, la tarea de conseguir que el proyecto de ejemplo de Encara2 funcionara llevó un tiempo considerable, y que más adelante quedó patente que podría haberse resuelto de una forma mucho más sencilla utilizando un gestor de paquetes como *apt-get*, que se encarga de gestionar automáticamente e instalar todas las dependencias de un paquete al instalarlo, lo que trivializa la instalación de OpenCV.

Una vez se consiguió ejecutar la aplicación de ejemplo de Encara2, era necesario conseguir contabilizar el número de rostros presentes en las imágenes capturadas. Tras cierta investigación en el código de la librería, al no existir una documentación propiamente dicha, se ubicó la forma de acceder a la información necesaria. Al comenzar la ejecución se instancia un objeto de la clase *ENCARAFaceDetector* que representa el detector de rostros, y que dispone de un método llamado *GetFacialData()* que devuelve un puntero a un objeto de la clase *CFacialDataPerImage* que contiene diversa información sobre los rostros detectados en la imagen que se está tratando. Uno de

estos atributos es el número de rostros, en un atributo del objeto llamado *NumFaces*.

```
CFacialDataperImage* FacialData;
FacialData = ENCARAFaceDetector->GetFacialData();
Faces = FacialData->NumFaces;
```

Conociendo ya el número de rostros de rostros detectados en cada momento, era necesario almacenar esta información de algún modo. Se decidió almacenar la información en un log que sería lo que se enviaría de forma rutinaria al servidor cada cierto tiempo. Para que estos datos fueran fácilmente extraíbles del log, debían seguir alguna estructura con la que fuera sencillo trabajar. Para ello, se eligió en este primer prototipo utilizar lenguaje XML, con el que es muy sencillo estructurar la información por bloques. La estructura que seguiría este log es la que se muestra a continuación.

```
<?xml version="1.0" encoding="UTF-8" ?>
<Log>
   <Block>
      <Timestamp> _TIMESTAMP_ </Timestamp>
      <Channel> _CHANNEL_ </Channel>
      <Viewers> _VIEWERS_ </Viewers>
   </Block>
   <Block>
      ...
   </Block>
   ...
</Log>
```

En este log se aprecian una serie de elementos XML cuyo contenido depende de los datos obtenidos en tiempo de ejecución. Los datos a insertar se muestran en mayúsculas y entre guiones bajos. El significado de los distintos elementos, bastante obvio si se conoce la lengua inglesa, se muestra a continuación.

- **Block:** Un bloque de datos de audiencia. Representa un estado de

audiencia (canal y número de espectadores) que se prolonga hasta el comienzo del siguiente bloque.

- **Timestamp:** Representa el instante temporal en el que comienza el bloque en el que se encuentra. Se utiliza el timestap de UNIX.
- **Channel:** El canal que se está visualizando en el intervalo de tiempo que representa el bloque en el que se encuentra.
- **Viewers:** El número de espectadores detectados en el intervalo de tiempo que representa el bloque en el que se encuentra.

Viendo como funciona la estructura de bloques diseñada, se deberá cerrar el bloque actual y crear uno nuevo cada vez que se cambie de canal o varíe el número de espectadores. Sin embargo, la reproducción y detección del canal actual no entra en esta etapa del desarrollo (lo veremos más adelante), y una vez el almacenamiento del log está listo, pasamos a trabajar en el lado del servidor, para seguir construyendo el esqueleto básico del que hablábamos antes.

En el servidor la primera tarea que se realizó fue llevar a cabo las instalaciones mínimas para un servidor web, con las herramientas que habíamos mencionado anteriormente. Por lo tanto, se instaló un servidor Apache2 utilizando *apt-get*. Sin embargo, Apache2 no viene preparado por defecto para manejar PHP5. En caso de que las novedades del lenguaje quisieran ser utilizadas más adelante, se procedió a la instalación de las muchas librerías que se necesitan para disponer de PHP5 al completo. Finalmente, lo comandos ejecutados para la instalación fueron los siguientes.

```
sudo apt-get install apache2
sudo apt-get install php5 libapache2-mod-php5 php5-mysql
   php5-curl php5-gd php5-idn php-pear php5-imagick php5
   -imap php5-mcrypt php5-memcache php5-ming php5-ps
   php5-pspell php5-recode php5-snmp php5-sqlite php5-
   tidy php5-xmlrpc php5-xsl
```

Como vimos antes, el Framework de elección que iba a contener la aplicación web en un principio era Symfony2, y el paso siguiente a tener listo

el servidor web es proceder a una instalación limpia del mismo. En la web del producto recomiendan utilizar **Composer** para la instalación del mismo. Composer es un gestor de paquetes PHP, que se encarga de mantener al día todas las dependencias entre los mismos. Por lo tanto, procedemos a instalar Composer utilizando la instrucción de línea de comandos que los diseñadores muestran en su página web.

```
curl -s http://getcomposer.org/installer | php
```

Una vez disponemos de Composer ejecutamos, nuevamente en línea de comandos, una instrucción que se encuentra en la documentación de Symfony2, que crea un proyecto en blanco de Symfony2 e instala automáticamente todas las dependencias del mismo.

```
php composer.phar create-project symfony/framework-
    standard-edition path/to/install
```

En este caso la ruta de instalación es la carpeta **www** localizada bajo el **home** del usuario utilizado durante el desarrollo, quedando la ruta como **/home/ker/www/symfony**. Con el proyecto ya creado, el siguiente paso era instalar JPGraph (la librería de generación de gráficas en PHP) y realizar un pequeño dummy con ella, para aprender su manejo. Una vez descargada la librería y descomprimida dentro del proyecto, se crearon un controlador y una vista basándose en códigos de ejemplo que se encuentran en las web tanto del framework como de la librería. El código del controlador es algo extenso, pero se muestra como ejemplo la parte donde se guardan los datos de ejemplo y se hacen las llamadas a JPGraph para crear la imagen, así como la llamada a la vista que las representará.

```
$ydata = array(11,3,8,12,5,1,9,13,5,7);
// Create the graph. These two calls are always required
$graph = new \Graph(350,250);
$graph->SetScale('textlin');
// Create the linear plot
```

```
$lineplot=new \LinePlot($ydata);
$lineplot->SetColor('blue');
// Add the plot to the graph
$graph->Add($lineplot);

...

$redirect = $this->render('Bundle:Folder:file.html.twig',
    array(
   'EncodedImage' => $image,
    ));
```

Y del código de la vista resaltamos únicamente la línea HTML donde se incorpora la imagen al contenido de la página.

```
<img src=" {{ EncodedImage }}" />
```

Como última tarea de este primer prototipo, faltaba instalar un sistema gestor de bases de datos. Esto es debido a que, como ya se ha comentado, originalmente la tarea de almacenamiento y minería de datos iba a ser desarrollada a mano. Por ello, se instaló un sistema MySQL utilizando nuevamente la línea de comandos:

```
sudo apt-get install mysql-server mysql-client phpmyadmin
```

Como puede verse se incluye en la instalación el paquete **phpmyadmin**, el cual permite visualizar y manejar los contenidos de la base de datos en un etorno web.

Con esto finalizaría el primer prototipo. Veamos a continuación un resumen de las características de las que se dispone en este punto del desarrollo.

- Instalación completa de Encara2 y sus dependencias.
- Detección y contado de rostros funcional.

- Generación de log en XML (a falta de incorporar el canal).
- El servidor web Apache2 está listo.
- Instalación del framework de desarrollo web Symfony2 e integración de la librería JPGraph en el mismo.
- MySQL ya está preparado para almacenar datos.

8.2. Prototipo 2

Después del volumen de trabajo y los avances realizados en el primer prototipo, los avances realizados en el segundo prototipo pueden paracer poco significativos. En este segundo prototipo principalmente se tomaron como objetivos la conectividad entre el subsistema cliente y el servidor. Además se llevó a cabo un cambio de formato en el log de audiencia generado por motivos que se explican más adelante.

Cuando hablamos de conectvidad del cliente al servidor, realmente lo que queremos decir es que el cliente ha de ser capaz de conectarse a la base de datos e incorporar a la misma el log que se ha generado. Para realizar esta tarea desde una aplicación en C++, existen distintas librerías suministradas por terceras partes. Por seguridad y estabilidad, nos ceñiremos a la librería oficial ofrecida por MySQL, llamada MySQLconnector.

Esta librería puede ser descargada desde la página oficial de MySQL, aunque requiere disponer de una cuenta de usuario de Oracle, que puede ser creada de forma gratuita. Además, su uso requiere de la instalación de dos paquete adicionales en el sistema, disponibles en los repositorios de Ubuntu a través de apt-get.

```
sudo apt-get install libmysqlclient libmysqlclient-dev
```

Una vez instalados los paquetes e incluida la librería descargada en el proyecto, es necesario conectar a la base de datos para poder incorporar el log a la misma. Sin embargo, antes de esto es necesario decidir cuál será el criterio para saber cual es el momento adecuado de enviar el log a la base de datos. Se barajaron dos opciones entre las que elegir:

- Utilizar un criterio temporal constante: Siempre que una hora termine, o una vez al día siempre a la misma hora.
- Al cerrar la aplicación.

Entre las dos opciones se tomó la segunda por ser más sencilla y comenzar a trabajar cuanto antes en la conexión con la base de datos, sabiendo que este criterio podría cambiarse con relativa facilidad en cualquier momento.

Una vez lista las dependencias únicamente faltaba la propia conexión a la base de datos y el envío del log, con lo que la interconexión entre los subsistemas cliente y servidor quedaría completa. A continuación se muestran algunas líneas de código relevantes entre todas las que se utilizaron para esta tarea.

```
#include <mysql.h>
...
MYSQL *conn;
conn = mysql_init(NULL);
...
if (mysql_real_connect(conn, "localhost", "zetcode", "
   passwd", NULL, 0, NULL, 0) == NULL)
   {
   printf("Error %u: %s\n", mysql_errno(conn),
      mysql_error(conn));
   exit(1);
   }
...
mysql_close(conn);
```

No se muestra el proceso de envío del log ya que ocupa un volumen demasiado elevado para la cómoda lectura del documento, y es principalmente tratamiento de ficheros y ristras en C, poco relevante para el desarrollo del proyecto.

Además, se introdujo el cambio de formato del log que se mencionaba anteriormente. La evolución se produce al sustituir el lenguaje XML por **JSON**. La expresión JSON (habitualmente leída como el nombre Jason en

inglés) proviene de las siglas de *JavaScript Object Notation*. A continuación se muestra como queda la estructura del log utilizando el nuevo lenguaje.

```
{
   "blocks":[
   {
      "timestamp": "_TIMESTAMP_",
      "channel": "_CHANNEL_",
      "viewers": "_VIEWERS_"
   },

   {
      ...
   }
   ...
   ]
}
```

Como puede obersevarse, la estructura del log es muy semejante al de formato XML, aunque con unas pocas diferencias. Para empezar, el elementro raíz **<Log>** ya no tiene un equivalente en JSON, mostrándose directamente los bloques. La segunda diferencia es que en lugar de múltiples elementos **<Block>**, se utiliza un único elemento **blocks** cuyo contenido es un vector, conteniendo cada vector todos elementos de los que disponía antiguamente un bloque.

La principal ventaja de este cambio de formato consiste en que los ficheros en formato JSON presentan una mayor densidad de datos, es decir, ficheros del mismo tamaño contienen un mayor volumen de información. Esto en nuestro caso es especialmente útil debido a que se espera un gran número de bloques en cada log así como un gran número de clientes generando dichos logs. Hay que recordar que esta información ha de ser enviada a través de la red hasta el servidor y posteriormente almacenada y procesada, por lo que reducir el espacio que ocupan es muy interesante.

Con esto el segundo prototipo está listo, y procedemos a enumerar los cambios así como las características actuales en este punto del desarrollo.

- Instalación completa de Encara2 y sus dependencias.

- Detección y contado de rostros funcional.
- Generación de log en JSON (a falta de incorporar el canal).
- El servidor web Apache2 está listo.
- Instalación del framework de desarrollo web Symfony2 e integración de la librería JPGraph en el mismo.
- MySQL ya está preparado para almacenar datos.
- Conexión a la base de datos MySQL desde el subsistema cliente y almacenado de los logs en la misma.

8.3. Prototipo 3

Este tercer prototipo es el que trajo el mayor número de reconsideraciones y cambios en el desarrollo, especialmente en el servidor. Sin embargo, se comenzó buscando completitud en el subsistema cliente, ya que una pieza fundamental del mismo sigue faltando: La reproducción de los canales de televisión y la detección de canales. Por ello, los esfuerzos se volcaron en esta tarea. El caso óptimo era poder encontrar un reproductor que fuera controlable directamente desde la aplicación. Aunque no se encontró ningún reproductor con dicho requiisito, se encontró una alternativa similar: Kaffeine (el reproductor multimedia de KDE) resultó ser completamente controlable desde línea de comandos. Esta funcionalidad es explotable mediamente llamadas a la función **system** de la librería **stdlib** desde la aplicación.

El primer paso es instalar Kaffeine, disponible en los repositorios de Ubuntu, utilizando el ya tradicional apt-get.

```
sudo apt-get install kaffeine
```

A continuación se muestran algunos fragmentos de código relevantes en el manejo de Kaffeine desde nuestra aplicación, como la apertura del reproductor al lanzar el programa o el cambio de canal.

```
string startPlayer = "kaffeine --fullscreen --channel 1";
system (startPlayer.c_str());

...

int channelChange(bool Increasing){
   ...
   if (Increasing)
      channel++;
   else
      channel--;
   ...
   changeChannel = "kaffeine --fullscreen --channel ";
   changeChannel.append(channelContainer);
   system(changeChannel.c_str());
   ...
}
```

El código que se muestra al comienzo de la muestra es el lanzamiento del reproductor, y la función channelChange es la encargada de realizar el cambio de canal. Sin embargo, esta aproximación produce un efecto secundario no deseado: Cuando se realiza una llamada por consola a Kaffeine para que cambie de canal, dicha aplicación toma el foco del sistema, por lo que perdemos el control sobre nuestra aplicación. Esto es un gran problema y es necesario solventarlo.

Este inconveniente ciertamente era preocupante, y casi hace abandonar por completo la opción utilizar Kaffeine. Sin embargo, tras cierta investigación se descubrió una aplicación para linux cuya funcionalidad es la gestión de ventanas en el escritorio, contando con funciones como mover o redimensionar una ventana, o nuestro deseado cambiar el foco a cualquier aplicación. El nombre de la aplicación es **wmctrl** y se puede instalar a través de los repositorios de Ubuntu a través del método ya mostrado repetidas veces.

Una vez instalada esta aplicación, fue necesario añadir la siguiente instrucción a continuación todas todas aquellas acciones que hicieran que el foco del sistema pasara a Kaffeine, de forma que retornara inmediamente a nuestro sistema.

```
system("wmctrl -a Capture");
```

El parámetro **-a** significa que la acción a realizar es un cambio de foco, y Capture es el nombre de la ventana a la que se va a realizar el salto. Este nombre viene dado por la ventana creada en por OpenCV al comenzar la detección de rostros.

Siendo posible ya de forma completamente funcional el cambio de canal, ya puede incorporarse a los logs el canal actual. Sin embargo, por el momento solo es posible almacenar el número del canal y no su nombre, ya que esa información está oculta en algún lugar dentro de Kaffeine, que veremos en el próximo prototipo. Con los logs ya completos, pasamos a atacar el servidor y mostrar los grandes cambios que anticipábamos antes.

Es el momento de comenzar a plantearnos la aproximación completa del sistema de minería de datos. Se comenzó a llevar a cabo un diseño, pero nunca parecía satisfactorio, en parte debido a su gran complejidad, en parte porque parecía inabordable llevarlo a cabo dentro de la planificación temporal del proyecto. Afortunadamente, investigando soluciones para estas dificultades, apareció una herramienta que cambió todo el concepto de la estructura del servidor: **FnordMetric**[1].

Fnordmetric es una aplicación que se ejecuta sobre Ruby y que depende de una instalación de una base de datos Redis, por lo que el proceso de instalación aparte de desconocido por ser tecnologías nunca utilizadas, no fue trivial. Aunque a priori parece posible instalar Ruby utilizando los repositorios de Ubuntu, esto no es recomendable debido a que los paquetes állí presentes son antiguos y no contienen una funcionalidad muy útil: **RVM**, o Ruby Version Manager.

Por lo tanto, la aproximación para instalar ruby es algo más compleja, e incluye el uso de cURL para la obtención e instalación de los paquetes necesarios. Destacar que todavía no estamos instalando ruby, sino RVM.

```
sudo apt-get install curl
```

[1]Para más información acerca de otras soluciones que se barajaron, consultar el capítulo de herramientas de este mismo documento

```
curl -L get.rvm.io | bash -s stable --auto
```

Una vez ejecutadas estas órdenes, necesitamos recargar el bash profile para tener acceso a RVM, lo que puede realizarse con este sencillo comando.

```
. ~/.bash_profile
```

Antes de proceder a instalar ruby, debemos instalar todas las dependencias que pueden conocerse mediante la orden **rvm requirements**, tarea que puede llevarse a cabo con apt-get y que no va a mostrarse debido al gran número de dependencias que se producen. Una vez las dependendencias estén listas, podemos instalar ruby.

```
rvm install 2.0.0
rvm use 2.0.0
```

Por último solo queda asignar la versión más actual de ruby como la versión por defecto. Para ello comprobamos la versión con la orden **ruby -v** y la asignamos como la preferida. En el momento de realizar esta tarea, el comando fue el siguiente (sujeto a cambios según ruby publique nuevas versiones).

```
rvm --default use 2.0.0-p247
```

Con esto, ya ruby queda listo para ser utilizado. Ahora necesitamos realizar una instalación de Redis, bastante más sencilla. Se muestra la lista de comandos utilizados para llevar a cabo el proceso, extraídos de la web oficial de Redis.

```
wget http://download.redis.io/redis-stable.tar.gz
tar xvzf redis-stable.tar.gz
cd redis-stable
```

```
make
sudo cp redis-server /usr/local/bin/
sudo cp redis-cli /usr/local/bin/
```

De esta forma Redis queda instalado e incluido dentro del PATH del sistema, de forma que pueda ser iniciado y detenido desde cualquier punto del mismo. Por último ya podemos instalar FnordMetric, disponible como una RubyGem (el nombre que le da ruby a los paquetes o módulos adicionales).

```
gem install fnordmetric
```

Al fin, la herramienta de minería y gestión de datos está instalada, y es momento de integrarla con lo que disponemos actualmente. Gracias a esta herramienta, la base de datos MySQL de la que disponíamos no va a ser necesaria, y la aplicación del subsistema en el hogar interactuará directamente con FnordMetric. Esto presenta una ventaja adicional: los datos no se almacenan en el cliente local a la espera de ser enviados, sino que pasan a estar disponibles de forma inmediata en el servidor.

Debido a este cambio conceptual, es necesario sustituir en la aplicación cliente todos los fragmentos de código dedicados a la generación y envío de logs por unos nuevos cuyo objetivo será enviar la información de audiencia a FnordMetric. Para realizar este envio de información usaremos cURL para comunicarnos con el servidor via HTTP utilizando un sistema de solicitudes personalizadas explicadas en la documentación de FnordMetric.

Sin embargo, antes de entrar en el código explicaremos un poco como trabaja FnordMetric, principalmente sobre las estructuras que utiliza para almacenar los datos. FnordMetric almacena la ocurrencia de determinados eventos en unas variables que ellos llaman **gauges**, que podríamos traducir algo así como indicadores, aunque mantendremos el término original en el resto de la documentación. Estas variables funcionan incrementando y decrementando su valor a través de eventos, y es el propio FnordMetric el que gestiona la parte temporal, conociendo así cuando se producen los incrementos y decrementos, y el valor de las mismas en cualquier momento. Conociendo esta información es capaz de generar gráficas que muestren la información

adecuada.

Estas gauges pueden ser declaradas en el fichero de configuración de fnordmetric, o ser generadas en tiempo real al producirse eventos sobre algunas no declaradas. Sin embargo, aquellas no declardas están más limitadas en su funcionalidad ya que al declararlas es posible realizar varios cambios en su comportamiento. Uno de estos cambios consiste en hacer el almacenamiento constante en lugar por unidad de tiempo, característica que necesitamos para nuestro objetivo. Por lo tanto, únicamente podemos utilizar gauges declaradas en nuestro sistema.

A continuación se muestra un ejemplo de gauge declarada en el fichero de configuración de fnordmetric, **fnord_audimetro.rb**.

```
gauge :Channel1,
    :progressive => true,
    :tick => 1.second
```

En el fragmento anterior podemos observar la declaración de una de estas gauges, la activación del modo progresivo, que significa que el almacenamiento de datos no se reinicia cada cierto tiempo, y el tick que es la granularidad de la información, o cada cuando tiempo aglutina los datos recibidos para comprobar su valor. En el siguiente fragmento de código podemos observar como se envía la información al producirse un cambio en el número de rostros detectados.

```
int numFacesChange(int Faces, int FacesOld){
   channelDataPost = string("curl -X POST -d '{ \"_type
      \": \"_incr\", \"gauge\": \"") + channelName +
      string("\", \"value\": \"") + itos(Faces -
      FacesOld) + string("\"}' localhost:4242/events");
   system(channelDataPost.c_str());
}
```

Cuando FnordMetric está en marcha pone siempre en funcionamiento un Dashboard (accesible a través del puerto 4242 de la máquina donde se ejecu-

ta) en el que muestra los datos que se están recibiendo. Esta herramienta es muy útil para uso interno ya que permite consultar el valor de las gauges en todo momento y representar distintas gráficas con dicha información, aunque su uso no sea apto para el usuario medio debido a lo rudimentario y poco amigable. Sin embargo, fue ampliamente utilizable en labores de desarrollo.

Figura 8.2: Dashboard de FnordMetric

En el próximo prototipo, se sustituirá esta forma de visualizar la información por una más intuitiva para los usuarios. El estado actual del desarrollo, tras tener finalizado este prototipo y los cambios llevados a cabo, es el siguiente:

- Detección y contado de rostros funcional.
- Detección del canal visualizado, conociendo únicamente el número que tiene asignado en Kaffeine.
- El servidor web Apache2 está listo.
- Instalación completa de Ruby, Redis y FnordMetric.
- Symphony2 y JPGraph han sdo eliminados del sistema, siendo sustituidos por las herramientas de FnordMetric.
- MySQL ha sido desinstalado y eliminada su participación en el diseño del sistema.
- Eliminado todo el código relacionado con MySQL en el subsistema cliente, así como la generación de los logs de audiencia.

- Incorporado el código relacionado con el registro de cambios en el estado de audiencia utilizando la API de FnordMetric.
- Ya se pueden consultar los datos de audiencia a través del dashboard de FnordMetric.

8.4. Prototipo 4

El estado actual del software únicamente tiene 2 defectos comparado con los objetivos originales del proyecto: No se produce una distinción de sexos, ni podemos distinguir claramente el canal debido a que el número asignado por Kaffeine puede variar de un equipo a otro. Por tanto, en este último prototipo nos centraremos en estos dos últimos objetivos, junto con otros como un acceso web más sencillo para el mostrado de datos.

Comenzaremos por la detección de sexos. Esta tarea supuso un gran esfuerzo que podría haberse ahorrado admitiendo la incapacidad de resolverlo por mí mismo debido a la falta de documentación del proceso, en lugar de llevar a cabo una inversión de decenas de horas en estudiar el código de Encara2 en busca de las funciones adecuadas. Sin embargo, finalmente fue necesario recurrir a los desarolladores, especialmente al Dr. Modesto Castrillón, que suministró de buen grado un ejemplo de código en el que se llevaba a cabo esta tarea. Este bloque de código resultó no ser trivial al no estar encapsulado en ninguna función de la librería.

El reconocimiento de sexos es realizado por un objeto distinto al de detección de rostros, por lo que es necesario instanciar este objeto. Dicha instanciación requiere del paso de otros objetos como parámetros, obtenidos de una serie de datos que hay que cargar desde distintos ficheros. A continuación se muestran algunas de las líneas de código más relevantes en esta tarea.

```
sprintf(cPath,"ENCARA2data/SVM/GenderDemonolearning/
   Faces_CARAXCARAY/Eigenobjects");//Representation
   space
char absolutepath[256];
realpath(cPath, absolutepath);
IPCAstatic=new CPCA(absolutepath);
```

```
if(IPCAstatic==NULL)
   cout<<"ERROR: IPCAstatic es NULL"<<endl;
sprintf(cPath,"ENCARA2data/SVM/GenderDemonolearning");
realpath (cPath, absolutepath);

ClasifSVMstatic= new    CSVMUtils(absolutepath, //
   Classifier
                IPCAstatic,    //PCA for representation
                100            //number of coefficients
                );
if (ClasifSVMstatic==NULL)
   cout << "ClasifSVMstatic es nulo" << endl;
string audimetroDir = homePath + "/audimetro";

...

int classified=FacialData->Faces[dtindex]->Classify(
   ClasifSVMstatic,label,0,0,false,label1stage,false);
```

Además, fue necesario hacer una pequeña corrección en la librería debido a que en la carga de datos se hace un cambio de carpeta pero en la versión para Linux de Encara2 no se revertía dicho cambio, haciendo que el reconocedor de rostros dejara de funcionar.

```
string audimetroDir = homePath + "/audimetro";
```

Una vez realizados estos cambios, fue necesario incorporar las funciones apropiadas para que los datos de audiencia reflejaran esta nueva información, y que fuera enviada al servidor. Para llevar a cabo esta tarea fue necesario crear dos nuevas gauges por canal, de forma que pudiéramos almacenar el número de hombres y de mujeres espectándolo. No se incluye el código de las funciones de cambio de audiencia debido a su similitud con la de cambio de número de espectadores mostrada anteriormente.

Con la distinción de sexos lista, el siguiente paso es poder distinguir el canal visualizado de forma única. Lo ideal sería utilizar el nombre del canal, aunque recorriendo los ficheros de configuración de kaffeine esta información

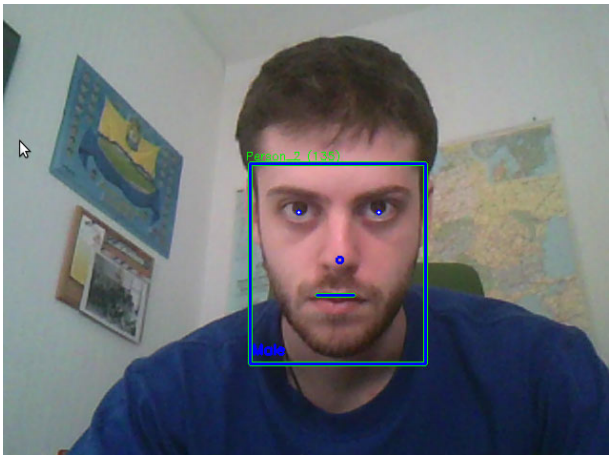

Figura 8.3: Captura de imagen de Encara2, reconociendo al sujeto como hombre

no parece encontrarse en ellos. Tras cierta investigación, se descubre que la configuración de la situación de los canales se encuentra en un fichero **SQLite** en la carpeta del reproductor. SQLite es un sistema de bases de datos inspirado en MySQL, con la diferencia de que toda la información se almacena en fichero. Este formato no es directamente manipulable con el tratamiento de ficheros tradicional y, debido a ello, es necesario instalar una librería adecuada para manejar este tipo de bases de datos.

La librería utilizada es la tercera versión de **SQLiteConnector**. Esta herramienta está disponible en la web oficial de SQLite, y ofrece una API en C para manejar una base de datos de este formato. A continuación se muestran algunas líneas relevantes en la conexión a la base de datos poco después de iniciar el programa.

```
homePath = getenv("HOME");

sqliteFile = (homePath + "/.kde/share/apps/kaffeine/
    sqlite.db");
msg = sqlite3_open(sqliteFile.c_str(),&db);    /* creo el
     archivo para la base de datos */
if (msg!=SQLITE_OK)                        /* verifico si
    hay error */
{
     cout << "Error al crear la base de datos" << endl;
     exit(1);
```

```
}
```

Una vez la conexión está realizada, podemos buscar en la base de datos el nombre del canal asociado al número del que está siendo visualizado actualmente, y comenzar a utilizar este nombre para almacenar los datos de audiencia, en lugar del número como se hacía hasta ahora.

```
sentencia = "SELECT Name FROM Channels WHERE Number=" +
    itos(channel) + ";";
msg = sqlite3_prepare(db,sentencia.c_str(),sentencia.
    length(),&resultado,&siguiente);   /* ejecuto la
    consulta */

...

while (sqlite3_step(resultado)==SQLITE_ROW) /* almaceno
    el resultado de la consulta */
{
     channelName = (const char*)(sqlite3_column_text(
        resultado, 0));
     channelName = spaces2underscores(channelName);
}
```

Además, ahora que ya disponemos de la distinción de sexos, necesitamos crear nuevas estructuras de datos para almacenar esta información. Por ello, en el fichero de ruby que carga la configuración de Fnordmetric se modifican las **gauges** de las que disponiamos, y se crea un nuevo conjunto por triplicado. Por una parte, se crea una gauge con el nombre cada canal TDT detectado en Las Palmas de Gran Canaria, que contendrán los datos de audiencia total para cada canal. A continuación se crean dos nuevos conjuntos con los mismos nombres seguidos de **__male** y **__female**, que representan la audiencia para cada sexo y canal. Se muestra un ejemplo de con las 3 gauges asociadas a la primera cadena de Televisión Española.

```
gauge :La_1,
    :progressive => true,
```

```
    :tick => 1.second

gauge :La_1_male,
    :progressive => true,
    :tick => 1.second

gauge :La_1_female,
    :progressive => true,
    :tick => 1.second
```

Llegados a este punto, ya el servidor de datos y el sistema cliente funcionan de forma completa. A pesar de que los datos obtenidos pueden consultarse en el Dashboard de Fnordmetric, es momento de, como se dijo antes, crear una interfaz web más amigable para el usuario medio, de forma que el proyecto quede completo.

El sistema de funcionamiento de la vista web es el siguiente: Al usuario se le muestran 3 columnas de selectores de canales, una para la audiencia general y una por cada sexo, junto con unos selectores temporales con los que especificar el intervalo temporal de su interés. Los selectores temporales disponibles son los siguientes.

- **Últimas X horas (máximo 24):** Representa las últimas horas de datos de audiencia. El número de horas es elegido por el usuario, con un máximo de 24.
- **Fecha:** Se mostrarán los datos de audiencia desde las 00:00 hasta las 23:59 del día seleccionado.
- **Tiempo real de los últimos 10 minutos:** Se obtiene información de audiencia de los últimos 10 minutos. Sin embargo, a diferencia de las anteriores, esta gráfica se mueve a lo largo del tiempo por lo que el usuario puede observar una evolución constante de la actividad de los espectadores.

Una vez elegidos los datos y el intervalo temporal a mostrar, el selector se desplaza hacia abajo, dejando sitio a la gráfica generada. De esta forma, no es necesario cargar una nueva página y el selector sigue disponible para

Figura 8.4: Interfaz web para la consulta de datos de audiencia

hacer una nueva cosulta, cuyo resultado sustituiría a los datos mostrados actualmente.

Finalmente, se han maquetado todos estos elementos utilizando las reglas CSS de bootstrap, de forma que sean más agradables para la vista y funcionen de forma responsiva, lo que vuelve la página compatible con todas las resoluciones, incluyendo los navegadores móviles.

Con esto, el cuarto y último prototipo queda completo. Veamos las características actuales del proyecto al finalizar el desarrollo del mismo:

- Visualización y control de los canales de televisión visualizados.
- Detección y contado de rostros funcional.
- Distinción de sexos funcional.
- Detección del canal visualizado, conociendo el nombre del canal.
- El servidor web Apache2 está listo.
- Instalación completa de Ruby, Redis y FnordMetric.

Figura 8.5: Selector en calendario para el modo fecha

- Incorporado el código relacionado con el registro de cambios en el estado de audiencia utilizando la API de FnordMetric, teniendo en cuenta la distinción de sexos.
- Los datos de audiencia son consultables en el dashboard de FnordMetric para uso interno.
- La vista web para que los clientes puedan consultar los datos de audiencia está lista, utilizando bootstrap como sistema de estilos y maquetado.

Teniendo un sistema completo y funcional, solo queda someterlo a pruebas y, una vez pasadas, preparar un sistema sencillo de instalación del subsistema cliente, de forma que su preparación sea sencilla en los hogares de los espectadores. Trataremos estos dos puntos finales en la siguiente sección, que aunque no formen parte de ningún prototipo, representan un trabajo adicional.

8.5. Pruebas e instalador

Para llevar a cabo un proceso de pruebas del sistema, es necesario disponer de un sistema de generacion de datos, de forma que se disponga de información que tratar. Para este fin se ha creado un script en ruby, que genera estos datos de forma artificial para los canales La 1 y La 2 de Televisión Española, y se ha acoplado a una copia del fichero de configuración

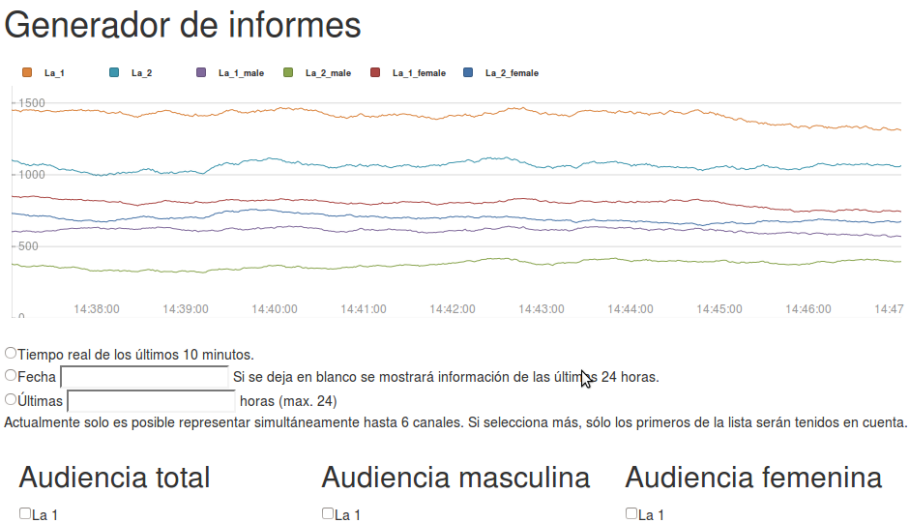

Figura 8.6: Ejemplo de gráfica mostrada en la web

de FnordMetric. Este nuevo archivo de ruby ha sido nombrado **datagen-audimentro.rb**, y a continuación se muestra un fragmento de código del mismo, relevante en la generación de los datos.

```
def start_example_data_generator

  api = FnordMetric::API.new
  Thread.new do
    total = rand(101) + 100;
    males = rand(51) + 50;
    females = total - males;
    api.event(:_type => :_incr, :gauge => :La_1, :value
       => total);
    api.event(:_type => :_incr, :gauge => :La_1_male, :
       value => males);
    api.event(:_type => :_incr, :gauge => :La_1_female, :
       value => females);
    ...
    loop do
      total = rand(1..5);
      males = rand(0..total);
      females = total - males;
      sign = rand(0..1) #increase or decrease
```

```
      if sign == 1
        api.event(:_type => :_incr, :gauge => :La_1, :
           value => total);
        api.event(:_type => :_incr, :gauge => :La_1_male,
            :value => males);
        api.event(:_type => :_incr, :gauge => :
           La_1_female, :value => females);
      else
        api.event(:_type => :_incr, :gauge => :La_1, :
           value => (0 - total));
        api.event(:_type => :_incr, :gauge => :La_1_male,
            :value => (0 - males));
        api.event(:_type => :_incr, :gauge => :
           La_1_female, :value => (0 - females));
      end
      ...
      sleep(0.5)
    end
  end
end
```

Como puede verse, la forma de incorporar datos es a través del evento de FnordMetric **__incr** , de la misma forma que se realiza en la aplicación del subsistema cliente. El proceso se realiza un bucle iterativo, separando cada iteración con un **sleep** de medio segundo, y eligiendo de forma aleatoria, dentro de un máximo y un mínimo, el número de espectadores que se suman o restan a la audiencia de cada canal.

Gracias a este script, es posible probar la inserción, extracción y representación de los datos, incluso cuando éstos alcanzan un gran volumen.

En cuanto al proceso de instalacion en el susbsistema cliente, se ha creado un script shell bajo el nombre **install.sh**, que instala en el sistema todas las librerías necesarias, y otro de nombre **updatelib.sh** que instala o actualiza la librería de Encara2 entre las del sistema. Con esto el proceso de instalación queda bastante simplificado aunque será explicado paso a paso en el manual de usuario (más adelante en este documento).

Llegados a este punto, el proyecto queda finalizado.

Capítulo 9

Resultados y conclusiones

Como resultado del proyecto se ha obtenido un sistema de control de audiencias televisivas que funciona de forma autónoma, sin intervención del espectador. El sistema es estable, controlable y con una precisión aceptable en la detección de los datos (esta precisión depende de Encara2, y no es algo que se puede mejorar sin alterar dicha herramienta).

Este sistema está dividido en tres subsistemas en los que ha habido que trabajar por separado, estando cada uno de ellos basado en tecnologías distintas. Tanto es así, que no tratamos sólo una diferencia de lenguajes de programación, sino que se han estudiado distintas ramas como tecnologías de servidores y programación de escritorio y web, así como una amplia investigación en las novedosas televisiones inteligentes, aunque finalmente no hayan podido utilizarse para el proyecto. Esta gran variedad , aunque por un lado trabajoso, es interesante en un proyecto de origen académico, ya que no sólo conlleva varios campos de aprendizaje, sino que sirve como entrenamiento para ser capaz de adoptar tecnologías en el futuro sin temer las novedades.

En cuanto a los objetivos, el proyecto cumple con los establecidos originalmente, como la detección autónoma del número de espectadores y el canal (y al menos un dato de audiencia adicional, como la distinción de sexos), la gestión de los datos de audiencia y la representación de los mismos en un entorno web.

Durante el desarrollo se han encontrado múltiples problemas, siendo siempre el más importante de ellos la interacción entre componentes. Es-

ta interacción se refiere tanto a los distintos subsistemas (como el envío de información desde el cliente al servidor) como entre distintas aplicaciones dentro del mismo subsistema (como controlar el reproductor desde nuestra aplicación u obtener los datos de sintonización de los canales). Además, se ha encontrado un obstáculo adicional, que es la incorporación de código ajeno al propio proyecto. Esto no siempre es una tarea sencilla, y en esta ocasión ha supuesto una inversión temporal considerable, aunque en gran parte debido a una aproximación inicial errónea.

Finalmente se puede concluir que el resultado del proyecto es satisfactorio y que con muy ligeros ajustes podría instalarse en el hardware que se encontrara apropiado y comenzar su uso a nivel de producción. Por lo tanto, se prueba la posibilidad de la sustitución futura de los audímetros televisivos actuales por unos automatizados, con la emoción que supone participar en los primeros intentos en dicho avance.

Capítulo 10

Trabajo futuro

Está claro que el software desarrollado durante el proyecto, aunque completamente funcional y utilizable, es mejorable en diversos campos. Hablaremos de la mejora en tres áreas: Establecer un subsistema cliente con hardware, fácilmente distribuible y repetible, un aumento en el número de datos de audiencia obtenidos a través del uso de otras librerías de detección de rostros, y la mejora del entorno web, incluyendo una monetización de los datos.

10.1. Hardware del subsistema cliente

Utilizando uno de los miniPCs de los que se hablaba en el capítulo de herramientas, puede prepararse un sistema para ser fácilmente replicable en múltiples unidades mediante los scripts de instalación adecuada, de forma que el sistema sea fácilmente distribuido por los hogares que sea necesario. Para esta tarea sería primero necesario realizar un estudio de viabilidad de los miniPCs hasta encontrar el más adecuado y rentable.

La preparación de estos aparatos incluiría la instalación del software y sus dependencias, el arranque automático del mismo, y la minimización del gasto de recursos innecesario en otras tareas, entre otras tareas. Sería interesante además contemplar la posibilidad de incorporar un mando a distancia al modelo para el control de los canales.

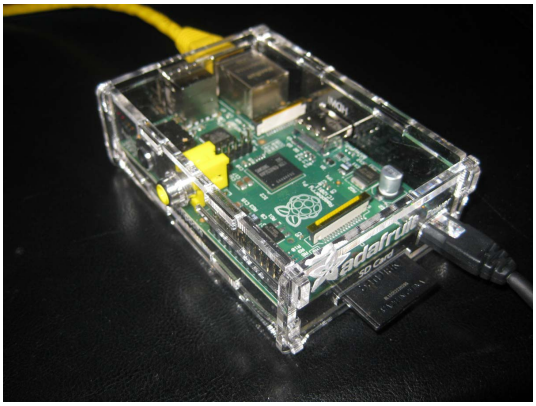

Figura 10.1: Raspberry Pi, un posible modelo final para el subsistema cliente

10.2. Nuevos datos de audiencia

Existen distintas librerías de detección de rostros, algunas de ellas gratuitas y muchas de ellas comerciales, que garantizan una mayor precisión en la detección o una mayor independencia a las condiciones de luz y posición, así como que están preparadas para obtener información adicional de los espectadores, como su raza, grupo de edad, o incluso estado de ánimo.

Todos estos datos enriquecerían muchísimo el proyecto, y lo volvería una competición mucho más fuerte contra los audímetros actuales, ya que ofrecería información novedosa aparte de el hecho de la automatización del proceso de recogida de datos de audiencia.

10.3. Mejoras web

Actualmente, el sistema web para visualizar los datos es muy sencillo, ya que poder mostrar la información es su único objetivo. Sin embargo, es aquí donde se pueden realizar cambios y mejoras enfocados a la monetización del proyecto.

Se puede incorporar un sistema de usuarios y pagos por suscripción, de forma que la información ofrecida no sea gratuita, o incluso se podría contemplar la generación y venta de informes con los datos de audiencia.

Además, esta tarea incluiría un diseño y maquetado de una aplicación web más vistosa y que incluya más información sobre el servicio que se

ofrece, de forma que pueda despertar interés en los clientes potenciales del mismo.

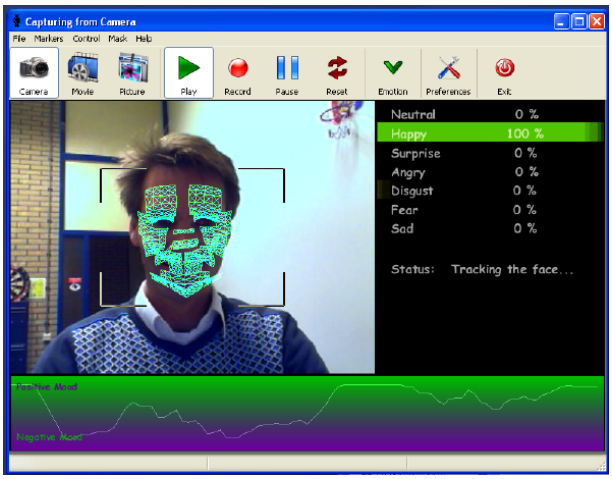

Figura 10.2: eMotion detectando expresiones faciales

Capítulo 11

Manual de Usuario

11.1. Requisitos de instalación

Para una correcta instalación y funcionamiento del subsistema cliente es necesario disponer de:

- Linux distribución Ubuntu versión 12.04 o posterior.
- Cámara web compatible con Linux correctamente instalada.
- Receptor TDT USB compatible con Linux correctamente instalado.
- Conexión a internet.

Para el servidor, la misma máquina puede funcionar. De hecho, el código que se entrega viene preparado para funcionar en la misma máquina, por lo que es esta aproximación la recomendada en caso de querer hacer pruebas.

11.2. Instalación del software del subsistema cliente

Para llevar a cabo la instalación del subsistema cliente es necesario ejecutar los siguientes pasos:

- Copiar la carpeta **audimetro** que se distribuye bajo el **home** del usuario.

- Ejecutar el script **install.sh**.
- Ejecutar el script **updatelib.sh**

Nota: Es posible que para realizar alguna de estas acciones se necesiten permisos de superusuario.

A continuación es necesario abrir Kaffeine y realizar una sintonización de los canales. Para ello, abrimos el programa y en el menú superior elegimos **television->channels** y dentro de la nueva pantalla pulsamos **start scan**. Cuando el proceso finalice podemos cerrar Kaffeine.

11.3. Instalación del software del servidor

La instalación del servidor es más compleja, y requiere de diversas acciones que es difícil recoger en scripts ya preparados como los usados anteriormente. Sin embargo, con la ayuda de internet y algunos ejemplos de código y comandos mostrados en el capítulo de desarrollo no debería ser especialmente difícil seguir los siguientes pasos.

- Instalar el servidor web **Apache2**.
- Copiar como la carpeta raíz de Apache2 la carpeta **www** que se suministra.
- Realizar una instalación manual de **ruby** (explicado en el apartado de desarrollo).
- Instalar Redis (explicado en el apartado de desarrollo).
- Instalar FnordMetric (explicado en el apartado de desarrollo).

Nota: Es posible que para realizar alguna de estas acciones se necesiten permisos de superusuario.

11.4. Ejecución del software del servidor

Debido a que el funcionamiento del software del cliente depende de la correcta ejecución de este software, se explicará primero esta parte.

En primer lugar, deberemos asegurarnos de que el servicio de Apache2 esté funcionando correctamente. Una vez esto esté listo, los pasos a seguir son los siguientes.

- Para iniciar el Sistema de Bases de Datos Redis, es necesario abrir un terminal y escribir **redis-server**. Este terminal debe mantenerse abierto durante todo el proceso.
- A continuación es necesario iniciar el listener de FnordMetric, que es el que conoce el nombre de los canales. Existen dos posibilidades:
  - Para iniciar el listener con generación de datos, nos colocaremos dentro de la carpeta **audimetro** suministrada, y ejecutaremos en un terminal **ruby datagen-audimetro.rb**
  - Para iniciar el listener sin generación de datos y depender únicamente de los datos generados por el sistema cliente, ejecutaremos en un terminal **ruby fnord_audimetro.rb**

  Nota: El terminal correspondiente debe permanecer abierto en cualquiera de los dos casos.

11.5. Ejecución y control del software cliente

Para iniciar el software cliente debemos ejecutar el script **client.sh** suministrado en la carpeta audimetro. Al ejecutar este script, se abrirá el reproductor multimedia **Kaffeine** a pantalla completa, mostrando en una esquina de la pantalla la imagen capturada por la cámara con los resultados de detección. A partir de este momento, los datos de audiencia están siendo enviados al servidor.

El control del software es a través del teclado: con las teclas + y - del teclado alfabético (no las del numérico) se realiza el cambio de canal al siguiente o anterior en la lista de Kaffeine. Para salir del software actualmente se usa la tecla Q, ya que aunque esta opción no es necesaria en un aparato que únicamente se usa como reproductor de televisión, es necesaria durante las labores de desarrollo.

Bibliografía

[1] M. Castrillón, J. Lorenzo, D. Freire, O. Déniz, *Learning to Recognize Gender using Experience.* Hong Kong, 2010.

[2] Michele Hilmes, Jason Jacobs, *The Television History Book*, BFI Publishing, 2008.

[3] Gemma Gimeno, Jordi A. Jauset Berrocal, Miquel Peralta, *Las audiencias en la televisión y El lenguaje de las noticias de televisión*, Editorial UOC, 2008.

[4] Barlovento Comunicación, *Análisis Televisivo 2012*, Madrid, 3 de enero de 2013.

[5] www.linuxtv.org, *Television with Linux*

[6] www.infoadex.es, *Base de datos de publicidad en España*

Apéndice A

Glosario de términos

Como herramienta de consulta durante la lectura y para facilitar la comprensión del documento, se incluye un glosario de términos que incluye desde los términos más comunes hasta los más específicos. De esta forma, se espera que la comprensión pueda ser completa sin necesitar recurrir a fuentes de consulta adicionales.

- **Audiencia.** Público que recibe mensajes a través de un medio de comunicación, habitualmente televisión o radio. Nos centraremos en los tipos de audiencias televisivas, ya que son éstas las de interés para el proyecto.

  - **Audiencia potencial.** Es el conjunto más amplio, el formado por todo aquel individuo que sea susceptible de ver un contenido determinado.
  - **Nicho de audiencia.** Es un subconjunto de la audiencia potencial, formado por todos aquellos que según ciertos precedentes puede convertirse en audiencia real.
  - **Audiencia real.** Número de espectadores de un programa o contenido determinado.
  - **Índice de audiencia.** Porcentaje de la audiencia real respecto a la audicencia potencial para un programa o contenido determinado.

    - **Target.** Se corresponde con el perfil del espactodor buscado por un programa dado. Puede basarse en múltiples criterios como edad, sexo, etnia, profesión o aficiones.
    - **Cuota de pantalla o *share*.** Reparto de las audiencias reales entre todos los canales expresado en porcentajes.

- **Audímetro.** Dispositivo cuya función es registrar los datos de audiencia televisiva, generalmente para el hogar en el que se instala.
- **TDT.** Televisión Digital Terrestre. Transmisión de imágenes en movimiento y su sonido asociado (televisión) mediante una señal digital y a través de una red de repetidores terrestres. Ha sustituido (y sigue en proceso de sustitución) a la televisión analógica.
- **SmartTV.** Televisión inteligente. Se trata de un paso en la convergencia tecnológica entre los ordenadores y las televisiones. Una televisión de estas características es capaz de navegar por internet y ejecutar múltiples tipos de aplicaciones y widgets. Para llevar a cabo estas operaciones, deben contar con su propio sistema operativo.
- **SDK.** Kit de Desarrollo de Software. Son herramientas que las compañías ponen a disposición de los programadores para que éstos puedan crear software para sus dispostivos o sistemas.
- **Servidor.** Un servidor es un nodo que, formando parte de una red, provee servicios a otros nodos denominados clientes.
- **Base de datos.** Colección de información organizada de forma que un programa de ordenador pueda seleccionar rápidamente los fragmentos de datos que necesite. Normalmente el encargado de gestionar esta información es un programa llamado Sistema Gestor de Bases de Datos, o SGBD.
- **Visión por computador.** Es un subcampo de la inteligencia artificial, cuyo propósito es conseguir que un ordenador pueda "comprender" (extraer datos o conclusiones) una escena o las características de una imagen.
- **Minería de datos o *Data Mining*.** Es un campo de las ciencias de la computación cuyo principal objetivo consiste en extraer información de

un conjunto de datos (generalmente de gran volumen) y transformarla en una estructura comprensible para su uso posterior.