UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

DEPARTAMENTO DE INGENIERÍA TELEMÁTICA

INGENIERÍA DE TELECOMUNICACIÓN

PROYECTO FIN DE CARRERA

Integración de la Plataforma NINOS con Twitter en cliente web para la generación autónoma de material audiovisual personalizado

Autor: Raúl Varela Izquierdo

Tutor: Abelardo Pardo Sánchez

Leganés, 26 de septiembre de 2011

Agradecimientos

En primer lugar, me gustaría dar las gracias a Abelardo Pardo, por la oportunidad que me ha dado de realizar este proyecto final de carrera con él, gracias a lo cual he aprendido mucho en campos que eran totalmente desconocidos para mi, como la animación 3D o las aplicaciones web, y que me han abierto un nuevo camino por el que enfocar mi carrera profesional. Gracias por tu paciencia e implicación, sobretodo en estos estresantes últimos días.

Cómo agradecer a mis padres lo que han hecho siempre por mi. Gracias a su esfuerzo, a su coraje, y por qué no, también a sus regañinas, hoy en día soy lo que soy. Siempre han puesto todo de su parte para que no nos faltara de nada y para educarnos de la mejor manera posible. Creo que lo habéis conseguido. Os estaré eternamente agradecido por estar ahí en todo momento. Ah, y aunque muchas veces os lleve la contraria, en el fondo sé que siempre lleváis la razón.

A mi hermano, Roberto, que aunque en los últimos tiempos nos hayamos distanciado un poco, siempre me tendrá ahí para lo que necesite y al que seguiré queriendo tanto en sus momentos buenos como cuando se vuelve un poco "aspero", aunque a ver si empiezan a cambiar las proporciones.

En este momento tan importante para mi, no puedo dejar de acordarme de mis abuelos, Nicasio y Antonia que me hubiera encantado que pudieran ver como su nieto se hacía, ni más ni menos, que Ingeniero; y de Fausto y Bernarda que sé que si hay alguien en el mundo que va a ser más feliz que yo después de acabar esta etapa son ellos, que llevan anhelando este momento durante años. ¡Cómo van a presumir de nieto!

A mi primo Óscar tengo que agradecerle que haya podido acabar esta carrera y no otra, además de su ayuda y consejo en muchísimas ocasiones.

Para agradecerte todo Bea, necesitaría escribir un tomo entero sólo para ti. No sabría por donde empezar. Han sido tantas cosas las que hemos pasado juntos, dentro y fuera de la Universidad. Tantas decepciones seguidas de alegrías. Tantos momentos en los que has sido mi único apoyo, la única que creías en mi, la única que me dabas fuerzas para seguir. Sin duda lo mejor que me ha pasado en esta vida y tengo que agradecérselo a la uc3m. Nunca podría haberme imaginado que alguien pudiera hacerme sentir así. ¡Gracias!

Quiero tener unas palabras para María José, Carlos y Laura porque en tan poco tiempo han conseguido ganarme con su cariño y su apoyo incondicional.

Gracias a Jesús y Fonsi, más que amigos se podrían considerar socios, por su ayuda

y por contar conmigo para la que puede antojarse una de las mayores experiencias que nos toca vivir, seguro que estando juntos, todo nos va genial.

Por último, agradecer a todos mis amigos de la universidad el haber tenido el placer de conocerlos. Han sido tantas cosas las que hemos disfrutado juntos. Desde viajes por Europa y el Caribe, pasando por mil y una cervezas, por cenas y tertulias interminables y sobretodo por hacerme reir una y otra vez sin parar, por sorprenderme día a día por haber encontrado a un grupo de personas tan excepcional. Gracias a Fer, Menda, Kiko, Rober, Elena, Miguel, Santi, Natalia, Rosa, Radigales, Carlos, Manu, Dani, Vaíllo, Miki, Arancha, Marian, Ana, Luis, Álex, Iria, Raúl, Javo. . . sois muy grandes!

Resumen

De un tiempo a esta parte, las imágenes generadas por ordenador han tomado una vital importancia en diversos sectores industriales como son el arte, los videojuegos, las películas y los anuncios, entre otros. Los recursos utilizados para su generación (ya sean imágenes en 2D, 3D CGI, personajes, decorados, o sonidos) son producidos en gran medida de forma independiente ante la ausencia de un marco unificador. Esto provoca que actualmente sea bastante extraña la transferencia de recursos digitales entre una película y un juego, por ejemplo, o la reutilización de los mismos en una nueva producción.

NINOS es una herramienta surgida de un proyecto, cofinanciado por la Unión Europea a través del Sexto Programa Marco, que trata de poner solución a esta problemática. NINOS permite la generación automática de vídeos a partir de un conjunto de objetos y animaciones 3D prediseñadas así como archivos de audio que se componen y renderizan formando una escena tridimensional. La creación del vídeo se realiza en base a una plantilla con formato XML que relaciona, mediante una estructura de etiquetas, a los personajes, los sonidos, las cámaras y otros recursos audiovisuales que aparecerán en la escena, así como las interacciónes entre éstos.

Para comprobar de primera mano todas las características prometidas por NINOS, se construye en el seno de este Proyecto Fin de Carrera un sistema que pretende integrar la generación de animaciones y entornos renderizados de manera automática que proporciona NINOS al funcionamiento general de Twitter. La integración se realiza representando una escenificación de la lectura, por medio de un avatar, de los últimos tweets publicados bien por un mismo usuario, bien contengan un determinado fragmento de texto o bien pertenezcan a una conversación entre distintos usuarios.

El resultado obtenido de la implementación del sistema se recoge en un demostrador en forma de una sencilla aplicación web con un funcionamiento similar a la aplicación real de Twitter, pero que es capaz de a partir de los *tweets* que se seleccionen generar un vídeo de manera automática obteniendo la información de éstos en tiempo real y presentarlo en la interfaz de la aplicación a través de un reproductor embebido.

Abstract

Recently, computer-generated images have acquired much importance in various industrial sectors such as art, video games, movies and advertisements, among others. Resources used for their generation (2D or 3D images, CGI, characters, sets, or sounds) are mainly produced independently in absence of a unifying frame. This situation makes very strange the transfer of digital assets among movies and games, for instance, or their reuse in new productions.

NINOS is a tool emerged from a project funded by the European Union's Sixth Framework Programme, which seeks to bring a solution to this problem. NINOS allows automatic generation of video from a set of objects, predesigned 3D animations and audio files that are composed and rendered to set up a three-dimensional scene. Creation of the video is done based on a XML template that relates characters, sounds, cameras and other audiovisual resources which appear on the scene, and the interactions among them.

To check NINOS's features, a system is developed to integrate generation of animation and rendered environments that NINOS provides, with general functionality of Twitter. Main objective established is the generation of a performance with an avatar who reads last Twitter messages posted either by a single user, or containing a specific piece of text or belonging to a conversation among different users.

System implementation results are contained in a demonstrator with a simple webapp whose operation is similar to real Twitter app. Main difference is that demo is capable of generating video automatically from *tweets* information gathered in real time and showing it in a player embedded in application interface.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Motivación del proyecto

La industria del diseño gráfico abarca infinidad de campos de aplicación hoy en día. El cine, la televisión, los videojuegos, la publicidad e Internet son algunos ejemplos donde el diseño gráfico, y más concretamente la creación de imágenes generadas por ordenador (en inglés CGI, *computer-generated imagery*), está adquiriendo una mayor importancia. Muchas de las imágenes utilizadas en la actualidad en la industria audiovisual se hacen manualmente, comenzando desde niveles básicos (polígonos o píxeles) lo cual conlleva un elevado coste de producción debido al tiempo y experiencia profesional que requiere invertir. En muchas aplicaciones, la existencia de herramientas digitales más sofisticadas en realidad ha provocado la subida de los costes de producción, ya que se dedica más tiempo a complejos procesos secundarios tratando de alcanzar la calidad que el usuario espera.

Tratando de buscar una solución a esta problemática, nace el proyecto SALERO (*Semantic AudiovisuaL Entertainment Reusable Objects*) cofinanciado por la Unión Europea a través del Sexto Programa Marco (FP6, *Sixth Framework Programme*) dentro de la prioridad de Tecnologías de la Sociedad de la Información (IST, *Information Society Technologies*).

El objetivo de SALERO [1] era conseguir la convergencia en la producción de medios para juegos, películas y televisión de un modo más rápido, más efectista y con unos costes menores mediante la combinación de gráficos generados por ordenador, tecnologías de lenguaje natural, tecnologías semánticas y tecnologías de búsqueda y recuperación de material basadas en contenido.

De esta manera, SALERO desarrolló conjuntos de herramientas para crear, administrar, editar, recuperar y presentar contenidos, personajes interactivos, objetos, sonidos, lenguaje escénico y comportamientos, basándose en la investigación de metodologías para la descripción, creación y búsqueda de "contenido inteligente". El "contenido inteligente" debería permitir a los artistas la creación y reutilización de contenidos y

medios complejos sin que éstos tuvieran la necesidad de conocer los aspectos técnicos de cómo funcionan las herramientas que utilizan.

Una de las herramientas experimentales más interesantes que surgen como resultado de este proyecto es la plataforma NINOS. Esta plataforma recoge algunos de los desarrollos llevados acabo por el consorcio que forma SALERO.

NINOS permite la generación automática de vídeos a partir de un conjunto de objetos y animaciones 3D prediseñadas así como archivos de audio que se componen formando una escena tridimensional. La creación del vídeo se realiza en base a un fichero XML que relaciona, mediante una estructura de etiquetas, los objetos que aparecerán, así como las interacciones de éstos. El archivo XML de la escena hace las veces de guión cinematográfico para la composición de la animación.

Al tener control total de qué aparece y cómo aparece en la escena simplemente manejando un archivo de texto XML, se despliegan potencialmente una infinidad de posibilidades para la composición de la misma. Realizando unos pequeños cambios en el guión pueden generarse escenas completamente diferentes. Se podría cambiar un escenario completo o un actor con tan sólo modificar una línea del fichero XML. Esto permitiría incluso la generación de guiones al azar con un sencillo script que aleatorice la elección de los recursos 3D que se manejen. El motor de renderizado que posee NINOS se ocupa de recopilar los recursos que se definen en el guión, agruparlos de la manera especificada y renderizar todo el contenido formando un vídeo con las características que se deseen (formato, codificación, resolución...).

NINOS es capaz de reutilizar los movimientos que se definan para un personaje con cualquier otro sin importar su forma o tamaño, simplemente compartiendo la misma nomenclatura en la definición de la estructura ósea, con el importante ahorro de tiempo que esto conlleva.

Todas estas cualidades que posee, hacen de NINOS una herramienta muy interesante para generar animaciones en tiempo real, sin necesidad de largos periodos de producción, ampliando sobremanera la interactividad y la inmersividad del usuario lo cual trae consigo una mejora sustancial de la calidad de experiencia y por tanto una mejor percepción del producto donde se integre.

La posibilidad de generar escenas animadas al vuelo resulta una característica muy llamativa que rompe totalmente el flujo actual de producción de animaciones 3D. Como prueba del potencial de NINOS se decide hacer uso de una fuente de datos con una frecuencia de actualización grande de modo que el cambio producido en los datos conlleve un cambio en la animación generada automáticamente a partir de éstos. Actualmente, no existe mayor fuente de datos que Internet que, además, posee un caracter inherentemente cambiante.

Esta primera prueba de concepto, de complejidad limitada, se ocupa de simular un programa de previsión del tiempo a partir de animaciones 3D y audio pregrabado. La animación representa una hipotética mujer del tiempo narrando la predicción meteorológica para el día actual de la capital de provincia española elegida. Los datos de la

temperatura máxima y mínima de dicha ciudad se recuperan en tiempo real (aunque sólo se actualizan en origen una vez al día) de la web de la Agencia Estatal de Meteorología (AEMet). En el Anexo A se encuentra ampliada la información sobre esta demo.

Los resultados obtenidos sugieren ampliar el alcance y utilizar una nueva fuente de datos actualizada de manera continua, algo con un flujo constante de información que permita creaciones casi paralelas pero totalmente diferentes. Debido al cumplimiento de estas características y a la importancia adquirida recientemente, se elige Twitter como nueva fuente de datos.

Twitter representa actualmente la inmediatez. Recibe más de 200 millones de tweets de media al día [2] de sus más de 340 millones de usuarios registrados en el momento que se escriben estas líneas con un crecimiento de 9.6 usuarios nuevos por segundo [3]. Además representa una fuente de datos que aúna texto, imágenes, vídeos y enlaces lo cuál permite un mayor número de posibilidades de interacción y creación. A todo esto, habría que añadir la disponibilidad de una API muy completa para el acceso a la información almacenada en Twitter que facilita sobremanera el proceso de integración. De hecho, ya existen un millón de aplicaciones registradas que hacen uso del API [4].

El sistema que se desea construir dentro de este proyecto, pretende integrar la generación de animaciones y entornos renderizados de manera automática que proporciona NINOS al funcionamiento general de Twitter. La integración se realizará representando una escenificación de la lectura, por medio de un avatar, de los últimos *tweets* publicados bien por un mismo usuario, bien contengan un determinado fragmento de texto o bien pertenezcan a una conversación entre distintos usuarios.

Se establece como demostrador de los resultados obtenidos a partir de esta integración entre Twitter y NINOS el despliegue de un cliente web con un funcionamiento similar a la aplicación real de Twitter. La interacción de los actuales usuarios de Twitter con el cliente web a desarrollar conllevará un periodo de reconocimiento de la aplicación prácticamente nulo debido a las similitudes funcionales entre ambas. El cliente desarrollado a tal efecto, deberá recoger la opción de generar y visionar las animaciones a petición. Esta nueva funcionalidad surgida del uso de NINOS proveerá a Twitter de una funcionalidad extra no recogida entre las funciones soportadas en la actualidad, lo cual abre nuevas perspectivas de utilización y unos potenciales nuevos métodos de explotación.

1.2. Objetivos

El objetivo general de este proyecto es desarrollar un sistema que utilice el potencial de NINOS en la generación de vídeos a partir de la composición de recursos 3D para representar un escenario de lectura de mensajes enviados a Twitter. A continuación, se encuentra una lista que recoge de una manera pormenorizada todos los objetivos establecidos para este proyecto:

- Buscar y recuperar los últimos *tweets* escritos o retweeteados por un usuario
- Buscar y recuperar los últimos *tweets* en los que aparezca un término o varios
- Buscar y recuperar todos los *tweets* que pertenezcan a una determinada conversación entre usuarios
- Consultar información y descripción de un usuario
- Consultar número de *tweets*, *followers* y *following* que posee un usuario
- Detectar el idioma con que se ha escrito un *tweet*
- Detectar el idioma en que está escrito cada *tweet* (actualmente español e inglés) y utilizar una voz sintetizada adecuada al idioma
- Generar códigos QR, para ser leídos por dispositivos móviles, que representen las direcciones web contenidas en los *tweets*
- Generar vídeos personalizados con el tipo de búsqueda realizada
- Generar contenidos personalizados para cada vídeo representado, con ciertos parámetros a elegir por el usuario (número de *tweets* contenidos en el vídeo) y otros aleatorizados por el sistema (selección de cámara o personaje)
- Construir aplicación web para la demostración de la prueba de concepto que ofrecezca una interfaz sencilla e intuitiva siguiendo las pautas de funcionamiento establecidas por el cliente web original de Twitter
- Permitir que la aplicación sea accesible desde cualquier dispositivo que se encuentre conectado a Internet, sin la necesidad de la instalación de ningún tipo de software
- Construir aplicación web para la demostración de la prueba de concepto, que sea compatible con diferentes tipos de dispositivos

1.3. Contenido de la memoria

La memoria del proyecto está estructurada en varios capítulos que tratan diferentes aspectos relativos al diseño y la implementación del mismo.

Este primer capítulo se centra en dar una visión general sobre la motivación que ha llevado a la realización de este proyecto así como los objetivos que se quieren alcanzar durante su vida.

En el Capítulo 2 se puede ver una visión del estado de la tecnología actualmente. Se describen brevemente las tecnologías con las que se podría llevar a cabo el desarrollo

de este proyecto, de modo que se tenga una perspectiva global a la hora de seleccionar las herramientas que más se ajusten a las necesidades concretas de éste.

Una vez presentadas las posibles tecnologías que se pueden utilizar, se realiza un análisis de los requisitos que debe poseer la aplicación. El Capítulo 3 se presenta un listado con toda la funcionalidad que se va a desarrollar en la aplicación.

Fijados los requisitos, en el Capítulo 4 se realiza un esquema de la aplicación, en el que aparecen todos los elementos que intervienen así como los distintos interfaces que los conectan, sin entrar en detalle sobre la manera en que éstos serán implementados.

Tras estas primeras secciones, comienzan a describirse los detalles del desarrollo del sistema. El Capítulo 5 presenta los diferentes módulos en los que se subdivide el sistema explicando la funcionalidad que éstos ofrecen además de mostrar en profundidad el método desarrollado para lograr cumplir con los requisitos fijados en capítulos anteriores.

A continuación, en el Capítulo 6 se recogen los componentes hardware y software empleados durante la vida del proyecto. Además se especifican los distintos costes en los que ha incurrido el proyecto.

En el Capítulo 7 se incluyen las conclusiones extraídas de la realización de este proyecto y se esbozan ligeramente unas posibles líneas futuras para completar y mejorar el resultado final del trabajo.

Por último, se añaden unos anexos para complementar la información del proyecto así como un capítulo donde se recogen las referencias bibliográficas sobre las que se apoya buena parte de la labor llevada a cabo. Entre estos anexos, cabe destacar el Anexo A, que recoge una explicación más detallada del funcionamiento de la demo inicial desarrollada como introducción al funcionamiento de NINOS. Finalmente, el Anexo D plantea un glosario de términos para facilitar la lectura de esta memoria y los conceptos que en ella aparecen.

Capítulo 2

Estado del arte

2.1. Introducción

La industria audiovisual (cine, televisión, Internet...) lleva años hablando acerca de la explotación multiplataforma como una forma de producir contenidos más impactantes de manera más rentable. Sin embargo, mientras la tecnología ha ayudado a producir sonidos e imágenes con una mayor calidad, los costes continúan en aumento. Las producciones de calidad para los medios digitales siguen requiriendo un trabajo muy intenso, lo cual lleva ligado un riesgo y un coste para la industria muy elevados.

De un tiempo a esta parte, las imágenes generadas por ordenador han tomado una vital importancia en diversos sectores industriales como son el arte, videojuegos, películas, programas de televisión, anuncios, simuladores de realidad, medios de comunicación, tanto impresos como los nuevos paradigmas comunicativos que surgen en paralelo al despliegue de Internet, y un largo etcétera. Esto ha provocado que se haya puesto especial énfasis en la búsqueda de métodos que logren revertir la situación actual de la producción de contenidos de este tipo, para hacer estos sectores más competitivos.

El proceso de producción de medios digitales, está muy fragmentado (existen infinidad de herramientas y métodos) y además requiere un trabajo altamente cualificado. No existe una tipología clara de recursos que permita gestionarlos con un sistema con consciencia semántica. Los recursos (ya sean imágenes en 2D, 3D CGI, personajes, decorados, o sonidos) son producidos en gran medida de forma independiente ante la ausencia de un marco unificador. La ausencia de formatos estándar de archivos, las diferencias existentes entre los métodos de gestión de datos y de procesamiento y la prevalencia de procesos independientes y propietarios, son algunas de las barreras que frenan la reutilización de estos recursos. Actualmente es bastante extraña la transferencia de recursos digitales entre una película y un juego, por ejemplo, o la reutilización de los mismos en una nueva producción. Incluso que una misma productora pueda volver a utilizar complejas escenas en 3D en secuelas posteriores, es terriblemente difícil, ya que la interconexión en el modelo no son evidentes para cualquiera que no sea el propio

autor. Por no hablar de la dificultad que puede entrañar la búsqueda de los recursos adecuados en el archivo de datos, lo cual puede resultar demasiado difícil y caro.

2.2. SALERO

Buscando una primera aproximación hacia la solución de este panorama, nace en enero de 2006 el proyecto SALERO, [1], cofinanciado por la Unión Europea a través del Sexto Programa Marco, con el objetivo de definir y desarrollar "contenido inteligente".

SALERO no busca inventar nuevos géneros multimedia, para los que pueda no haber un mercado, sino facilitar la producción de los medios actuales para los que ya existe un mercado establecido y probado.

Los principales resultados que se obtienen durante el tiempo de desarrollo de SALERO son:

- Definición de tipos de medios, géneros, estilos y flujos de trabajo para utilizar contenido inteligente en diferentes producciones y plataformas
- Especificación de requisitos de la industria para diferentes medios, así como fijación de las directrices para los investigadores, desarrolladores y productores para llevar a cabo producciones de contenido inteligente multiplataforma
- Desarrollo de un lenguaje de ontologías y metadatos que describen la semántica y el contexto del contenido inteligente
- Implementación de nuevos métodos, basados en contexto, de recuperación de personajes, sonidos, imágenes, movimientos o comportamientos a partir de grandes bases de datos y sistemas de almacenamiento
- Desarrollo de conjuntos de herramientas para crear, administrar, editar, recuperar y presentar contenidos, personajes interactivos, objetos, sonidos, lenguaje escénico y comportamientos
- Creación de aplicaciones para la manipulación de la apariencia, sonido, movimiento y comportamiento de personajes en base a un contexto semántico, y otros objetos para su utilización en diferentes plataformas con variados contextos

Tanto las herramientas como los interfaces desarrollados son compatibles con las prácticas actuales de la industria, lo cual permite el control de las apariencias, sonidos, comportamiento semántico y propiedades de los objetos de contenido inteligente para la producción y postproducción multimedia.

2.3. Plataforma NINOS

De entre la herramientas experimentales surgidas del proyecto SALERO, llama especialmente la atención la plataforma NINOS. Esta plataforma software aúna todas las herramientas desarrolladas por la Fundació Barcelona Media de la Universitat Pompeu Fabra a lo largo del proyecto.

NINOS permite la generación automática de vídeos a partir de un conjunto de objetos y animaciones 3D prediseñadas así como archivos de audio que se componen y renderizan formando una escena tridimensional. La creación del vídeo se realiza en base a una plantilla con formato XML que relaciona, mediante una estructura de etiquetas, personajes, sonidos, cámaras y otros recursos audiovisuales que aparecerán así como las interacciónes entre éstos. El archivo XML de la escena que se desea renderizar tiene, a grandes rasgos, el mismo funcionamiento que un guión cinematográfico.

La creación de un software avanzado que genere guiones puede conseguir que se rendericen vídeos de la misma escena en una amplia variedad de formatos. Por lo tanto, la misma plantilla de guión para una escena se puede utilizar para varias presentaciones a través de diversos dispositivos. Por poner un ejemplo, se pueden dar diferentes pronósticos del tiempo o, en el caso de una de las producciones experimentales que se llevaron a cabo durante el proyecto SALERO, un videojockey virtual introducía diferentes vídeos musicales. Realizando pequeños cambios en el guión que se utiliza de plantilla pueden generarse escenas completamente diferentes. Se puede cambiar un escenario completo o un personaje con tan sólo modificar una línea del fichero XML.

El motor de renderizado que posee NINOS se ocupa de recopilar los recursos que se definen en el guión, agruparlos de la manera especificada y renderizar todo el contenido formando un vídeo con las características que se deseen (formato, codificación, resolución...). La calidad visual está asegurada gracias a que este es un motor en tiempo real con toda la potencia de las características más recientes de OpenGL.

NINOS es capaz de reutilizar los movimientos que se definen para un personaje con cualquier otro sin importar su forma o tamaño, simplemente compartiendo la misma nomenclatura en la definición de la estructura ósea, con el importante ahorro de tiempo que esto conlleva.

Otra de las características que posee, es la gestión del sincronismo labial que dota de mayor realismo a los personajes a la hora de hablar, la animación procedural (parpadeo de ojos y dirección de la mirada hacia el objetivo deseado) y los valores de "Activation/Evaluation" que se utilizan para representar las expresiones y emociones faciales.

2.3.1. Program Editor

La plataforma NINOS incluye una herramienta denominada *Program Editor* que puede facilitar el aprendizaje del funcionamiento de la plataforma, así como hacer más

visuales y rápidos de diseñar los guiones de las escenas. Todas las características que posee la plataforma NINOS son accesibles a través del interfaz del *Program Editor*.

Program Editor es un software de edición en tiempo real que permite la producción de animaciones en base a una línea temporal (Figura 2.1). La producción se decribe a través de una secuencia de fragmentos de animación. Estos fragmentos pueden pertenecer a cámaras, personajes, escenarios, sonidos y objetos de atrezzo, entre otras tantas cosas, y representan sus cambios a lo largo del tiempo.

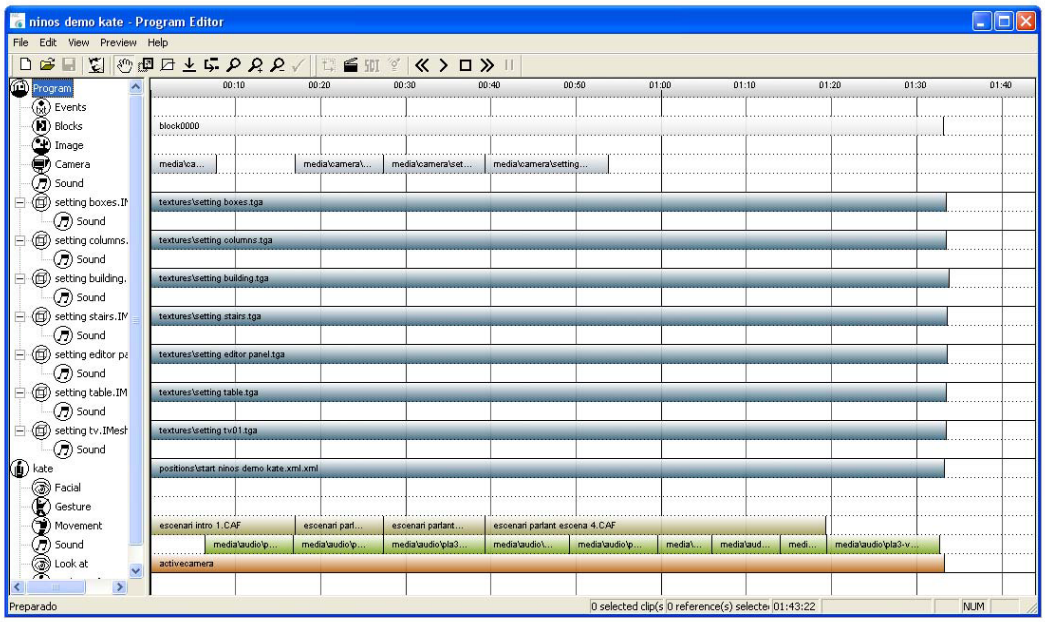


Figura 2.1. Timeline del Program Editor[1]

Program Editor hace posible que los usuarios ensamblen diferentes fragmentos, personajes virtuales, animaciones, cámaras, iluminación y audio a lo largo de una línea de tiempo, y generar vídeos de alta calidad en varios formatos de manera procedimental, de acuerdo con los datos de entrada que se decidan utilizar.

Las plantillas generadas con el *Program Editor* son almacenadas como archivos XML que son interpretados por el generador de programas y renderizados por el motor de NINOS. En los guiones XML se pueden especificar eventos que son lanzados en el tiempo al aparecer o desaparecer determinados recursos.

Como se ve en la Figura 2.2, también se permite la previsualización de la escena diseñada antes de ser renderizada y salvada directamente a un archivo de vídeo del formato elegido.

[1]Fuente: SALERO Project - D11.3.1 Report on Professional Training Programmes

Figura 2.2. Ventana de previsualización del Program Editor[2]

2.3.2. Material audiovisual soportado

El motor de renderizado de la plataforma NINOS está desarrollado para generar la salida audivisual de la interpretación de los guiones XML. Utiliza herramientas que permiten mejorar la calidad del vídeo final y la vista previa interactiva que provee el *Program Editor*. Está desarrollado haciendo uso de OpenGL [6], por lo que existen ciertas restricciones a la hora de crear recursos que deben ser tratadas.

Los archivos utilizados para intercambiar la información entre los editores 3D (como Autodesk 3ds Max) y la plataforma NINOS son formatos estándar que no soportan toda la información específica de cada editor como pueden ser materiales, mallas o nodos atípicos o propietarios, debido a la complejidad que requeriría la compatibilidad con todos los tipos de datos específicos de cada editor existente (p.e. mental ray, Vray, raytrace, etc.). El formato elegido es FBX, propietario de Autodesk, [7], en su edición 2009.3.

Los materiales de los recursos 3D que quieran ser utilizados en NINOS deben mantener los canales separados, soportando los canales *Diffuse*, *Specular Level*, *Bump* (como *normal map*), *Ambient Occlusion* (para mallas estáticas), *Reflection Map*, *Opacity Map*, *Self Illumination Map*. Todas las imágenes que se quieran usar como fuente para un material deben ser mapas en formato TGA. Este formato puede guardar datos de imágenes con 8, 16, 24, o 32 bits (24 bits RGB y 8 bits extra para el canal alpha)

[2]Fuente: SALERO Project - D11.3.1 Report on Professional Training Programmes

de precisión por píxel, además de tener la posibilidad de almacenarse en crudo o utilizando una compresión sin pérdidas PackBits [8] con codificación Run-length [9]. Es un formato con un uso muy extendido debido a su sencillez de implementación y la ausencia de patentes que graven su utilización.

NINOS Renderer posee algoritmos de iluminación y sombreado en tiempo real. Además, interpreta de manera dinámica el comportamiento de la superficie de los materiales bajo distintas condiciones de iluminación. Las fuentes de luz que se pueden utilizar son *Omni*, *Direct* y *Spot*, teniendo en cuenta que las sombras son generadas en tiempo real sólo ante la presencia de luces de tipo *Spot* y *Direct*.

Existe soporte para cualquier tipo de cámara además de poder animar la posición y la orientación de éstas o de cualquier tipo de objeto en el escenario virtual (incluso las luces).

La animación de los personajes se puede realizar mediante el uso de una estructura ósea siempre que no se utilicen deformadores complejos. Las restricciones que debe tener el esqueleto del personajes son que su *root bone* debe llamarse "Bip01", el hueso asignado al ojo izquierdo debe llamarse "el" y el asignado al derecho "er".

NINOS permite que archivos de audio sean asignados a personajes y a objetos de las escenas siempre que éstos sean en formato WAV. Haciendo uso de la herramienta NinosAudio se compone un fichero de audio WAV que aúna todos los fragmentos de sonido incluídos en la escena para posteriormente ser pasado al motor de renderizado que se ocupa de incluir este audio junto con las animaciones en el vídeo final.

2.3.3. Estructura del guión XML

Los guiones utilizados en NINOS están basados en una estructura en forma de árbol implementada en archivos de texto con formato XML. Estos archivos XML están organizados mediante etiquetas. Estos guiones deben seguir un esquema base determinado para que la plataforma funcione correctamente. Se realiza una descripción detallada del formato del guión en el Anexo B.

2.4. Twitter

Twitter es una herramienta que se puede utilizar para enviar y recibir pequeños mensajes de 140 caracteres de los amigos, de algunas organizaciones, de ciertos negocios, de publicaciones, o de desconocidos (incluso infinidad de personajes famosos de todos los ámbitos) que deciden compartir sus experiencias.

Un usuario de Twitter elige las actualizaciones que desea recibir, es decir a la personas que quiere seguir (conocidas como *Following*). A su vez, otros usuarios pueden optar por seguir las actualizaciones de éste (conocidas como *Followers*). Se pueden enviar mensajes públicos para toda la comunidad de Twitter (poseyendo una cuenta pública),

semipúblicos para los usuarios que han sido aprobados previamente para recibirlos (para los *Followers* de una cuenta privada), o privada de un usuario a otro (conocido como *DM, Direct Message*). Se puede ver estos mensajes, llamados actualizaciones o *Tweets*, ya sea en Internet a través de http://www.twitter.com o en dispositivos móviles a través de las distintas aplicaciones oficiales disponibles para diversas plataformas.

Twitter es una red social móvil que combina elementos propios de la mensajería instantánea, las herramientas de comunicación (como AOL Instant Messenger, AIM) y los software de publicación de blogs (como Blogger o WordPress). Tiene en común con los blogs que los *tweets* son publicados, en general, de manera que cualquiera pueda leerlos en Twitter.com (a menos que elija una cuenta privada, de modo que sólo los *followers* aceptados pueden ver sus *tweets*). Sin embargo, le diferencian de los blogs que las entradas entradas que se publican están limitadas a 140 caracteres. Con la mensajería instantánea comparte el hecho de que se permite la comunicación directa y privada (a través de DM's), pero les diferencia que cada *tweet* publicado tiene su propia URL, por lo que cada mensaje es en realidad una página Web. La mensajeria instantánea también carece de la caracterísitica de red social que provee Twitter y de las ideas de publicación-suscripción y difusión de mensajes de uno a muchos.

Twitter ha cambiado y mejorado la forma en que las personas se comunican unas con otras [10], con marcas y empresas, y con los movimientos sociales e iniciativas. Twitter ha permitido a los usuarios recaudar fondos para personas necesitadas [11], coordinar los esfuerzos de rescate en caso de desastre natural [12], y alertar a los cuerpos de seguridad de emergencias y actividades ilícitas tanto a nivel nacional [13] como extranjero [14].

2.4.1. Aplicaciones

Twitter tiene a disposición de los desarrolladores un API que permite la creación de aplicaciones o de otros servicios web para integrarse con las funcionalidades que éste posee, y complementarlas en muchos casos. En el Anexo C se realiza una breve descripción de este API.

Los desarrolladores de aplicaciones tienen un papel fundamental a la hora de ayudar a los usuarios a sacar el máximo partido de Twitter. Se han superado ya el millón de aplicaciones registradas [18] por más de 750.000 desarrolladores que hacen uso del API de Twitter. Sólo en el último año ha habido un aumento de 150.000 aplicaciones. Actualmente se registra una nueva aplicación cada 1,5 segundos, impulsando el crecimiento del ecosistema surgido alrededor de Twitter en diversos áreas. Existen aplicaciones accesibles desde la Web, desde smartphones y tablets, e incluso desde televisiones [19].

Existen infinidad de aplicaciones en áreas muy distintas [20]. Desde clientes web con funcionalidades similares al cliente oficial como Echofon (http://www.echofon.com/), Shareaholic (http://www.shareaholic.com/) o HootSuite (http://hootsuite.com/), o aplicaciones de escritorio como TweetDeck (http://www.tweetdeck.com/) o Twitterrific (http://twitterrific.com/), pasando por clien-

tes para smartphones como Fring (http://www.fring.com/) o UberSocial (http://ubersocial.com/), aplicaciones que facilitan la integración de Twitter con archivos multimedia como Twitpic (http://twitpic.com/) o Twitvid (http://www.twitvid.com/), acortadores de URL para ahorrar caracteres en los *tweets* como Bitly (http://bitly.com/), TinyUrl (http://tinyurl.com/) o el mismo Google URL Shortener (http://goo.gl), aplicaciones para añadir publicidad a los *tweets* y ganar dinero con ello como Twittad (http://www.twittad.com/), aplicaciones que realizan estadísticas sobre la actividad de los usuarios como TwitGraph (http://www.twitgraph.com/) o TwitterStats (http://www.twitterstats.net/), herramientas para gestionar encuestas como Twittpoll (http://twittpoll.com/), herramientas que muestran la localización de los usuarios como TwittEarth (http://www.twittearth.com/) o FollowerSearcher (http://followersearcher.netcom.it.uc3m.es/) y un incontable número de aplicaciones más [21].

Existe un sitio para desarrolladores (https://dev.twitter.com/) donde se recoge amplia documentación e información sobre el mundo Twitter y sobre las distintas herramientas que se encuentra a disposición de éstos. Desde este sitio cualquier persona puede empezar a construir con Twitter, contactar con los miembros del equipo de Twitter, intercambiar ideas con otros desarrolladores, y encontrar todos los recursos que necesita para crear su propio producto o negocio.

2.4.2. Negocio

Los inversores y las empresas están tomando nota del potencial negocio que existe alrededor de las aplicaciones que hacen uso de una manera u otra del API de Twitter. Desde diciembre de 2010 [22], más de 500 millones de dólares han sido invertidos en compañías con aplicaciones o herramientas de este tipo, y se han pagado más de mil millones de dólares en adquisiciones de empresas relacionadas. Este nivel de inversión es un indicativo de la oportunidad para los desarrolladores y empresarios para crear empresas de éxito como parte de la plataforma Twitter.

Dentro de este comentado gran número de adquisiciones de empresas, toman gran importancia las llevadas a cabo por la propia Twitter [23]. En los últimos meses han añadido a su tejido empresarial la empresa Atebits [24] creadores de la aplicación para iPhone que daría lugar posteriormente a la aplicación oficial de Twitter para este sistema, además de TweetDeck [25], cliente de Twitter con gran importancia en distintas plataformas.

Las empresas tienen en Twitter la oportunidad de llegar a millones de personas en cuestión de segundos. Esto abre un sinfín de posibilidades de negocio, que permiten optar al éxito, resultando Twitter una mera ayuda para conseguir los objetivos y las aspiraciones de una gran cantidad de personas. Twitter tiene un enorme potencial empresarial que sólo acaba de comenzar a explorarse.

2.5. Tecnologías web

En la era de las Tecnologías de la Información, han adquirido una gran importancia las aplicaciones cliente-servidor, multihilo, siempre conectadas a Internet y basadas en componentes en detrimento de las antiguas aplicaciones web monolíticas y primitivas [26]. Estas aplicaciones son parte principal de la nueva concepción de Internet, conocida como Web 2.0.

La Web 2.0 facilita la comunicación, el intercambio seguro de información, la interoperabilidad y la colaboración. Los conceptos de la Web 2.0 han dado lugar al desarrollo de una Red basada en comunidades, donde se alojan servicios y aplicaciones tales como redes sociales, sitios para compartir videos, wikis o blogs entre muchas otras.

Hoy en día, las tecnologías web permiten crear un entorno dinámico, centrado en el usuario que fomenta la comunicación en sentido ascendente y descendente [27]. Las aplicaciones web son aplicaciones de Internet enriquecidas (RIAs, en inglés, *Rich Internet Applications*) con características similares a las de una aplicación de escritorio pero ejecutadas por completo en navegadores web donde Internet es la plataforma.

Todos los contenidos alojados en la Web, incluyendo Feeds, RSS, Web Services, Mash-ups o SaaS, están disponibles en cualquier dispositivo. Esto se conoce como Arquitectura Orientada a Servicios (SOA, en inglés, *Service Oriented Architecture*), que es un concepto de arquitectura desacoplada de componentes de software que proveen funciones específicas y que pueden ser invocadas por los usuarios. Este tipo de arquitectura hace que vean modificadas las formas en que los usuarios finales y los desarrolladores de software utilizan la Web.

Todas las funcionalidades surgidas de este nuevo paradigma de concepción y utilización de la Red, han ido desarrollándose en paralelo al enriquecimiento en el diseño y la programación del lado servidor, del lado del cliente y de la presentación de la información.

2.5.1. Tecnologías de presentación de información

Es necesario indicar que se entiende por página web, el *look & feel* (apariencia externa) de un sitio web que contenga sólo información estática o de cada pantalla de una aplicación dinámica más avanzada. Las páginas web [28] pueden estar compuestas por distintos tipos de contenidos, como texto, imágenes, formularios, audio, vídeo o juegos interactivos, entre otros. Sin embargo, independientemente de qué tipo de contenidos contenga una página web, estará creada en HTML (*HyperText Markup Language*) o algún lenguaje similar. El código HTML define la estructura de una página web, con sus gráficos, contenidos y toda la información que ésta incluya. Los CSS (*Cascading Style Sheets*) indican al navegador cómo debe "pintar" esta estructura en pantalla. Estas dos tecnologías son la base de las aplicaciones Web.

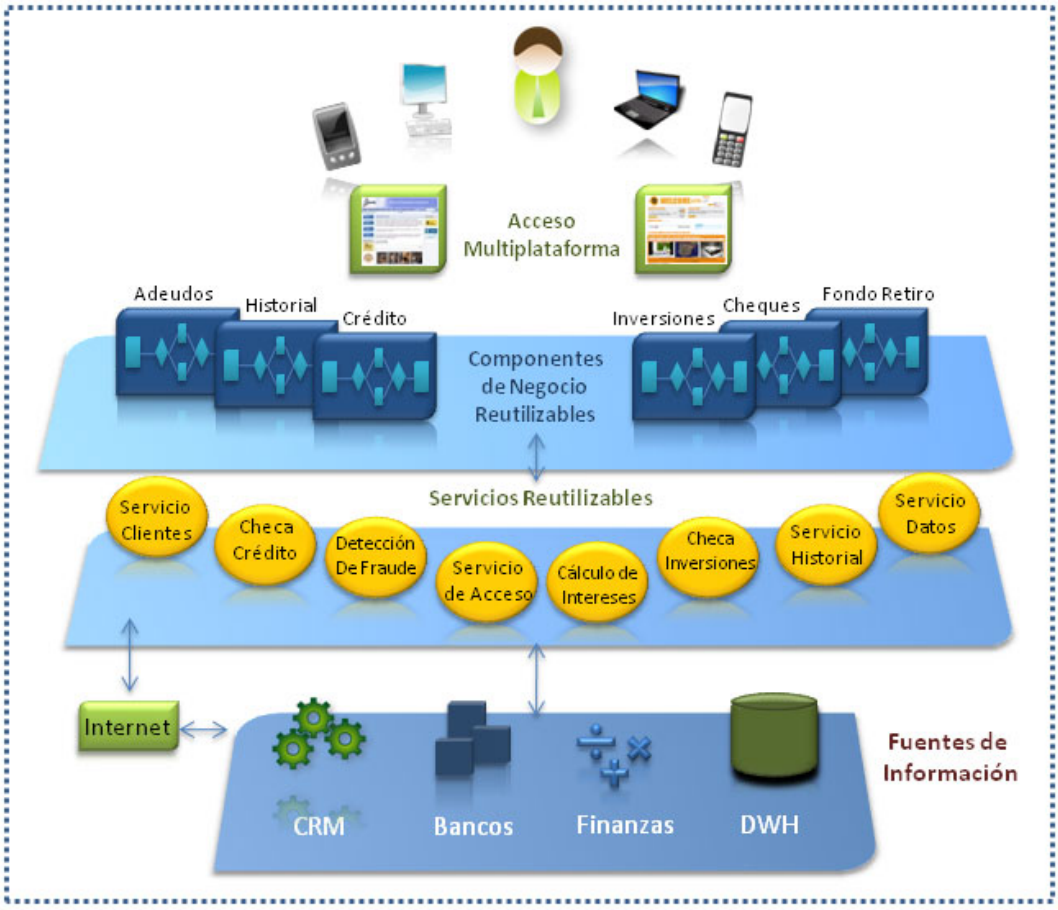


Figura 2.3. Ejemplo de Arquitectura Orientada a Servicios[3]

2.5.1.1. HTML

El lenguaje HTML surge en el año 1989 como un nuevo sistema de hipertexto para el intercambio de información entre investigadores del CERN (Organización Europea de Investigación Nuclear) [29]. En 1995 se publica el estándar HTML2.0, el primer estándar oficial de este lenguaje. Desde finales de 1997 hasta hoy, se utiliza la versión 4. Existen diversas razones por las que esta tecnología está tan extendida [30]:

- Utiliza texto plano,eliminando cualquier tipo de incompatibilidad ya que puede ser leído, interpretado y escrito por cualquier sistema, es un formato universal
- Se centra en describir los contenidos, no qué apariencia tienen éstos o la página que los aloja. La separación entre la definición de los componentes de una página web y la apariencia de los mismos, es una de las características más importantes de HTML

[3]Fuente: SOAction

- Es muy sencillo de manejar, tan sólo hace falta conocer el funcionamiento de los elementos que lo componen
- Es *open source*, no es una tecnología propietaria

En la actualidad, se está llevando a cabo la especificación de HTML5. Esta nueva versión trata de incluir nuevos y más elaborados procesos que permitan y fomenten más implementaciones interoperables entre diversos dispositivos, mejorar y racionalizar el marcado de los contenidos y facilitar APIs para construir aplicaciones web más complejas.

2.5.1.2. CSS

El W3C (*World Wide Web Consortium*) apostó por la estandarización de CSS y lo añadió al grupo de trabajo de HTML en 1995 para publicar la primera versión en 1996 [31]. Actualmente la versión utilizada por los navegadores es CSS2.1, que es una versión aún en desarrollo. Paralelamente a CSS2.1 se está trabajando en la siguiente recomendación, CSS3, que aún se encuentra en una etapa de definición muy temprana.

El diseño de una página web en HTML, está limitado a la inserción de tablas, controles de fuentes, y algunos estilos de letra como son la negrita o la cursiva. Sin embargo, CSS proporciona una gran cantidad de herramientas para dar formato a páginas web con controles precisos. Con las hojas de estilo se puede [28]:

- Controlar minuciosamente cada aspecto de la visualización de la página, incluyendo la la posición de los elementos
- Aplicar los cambios de manera global de manera que se garantice que un diseño es consistente en todo un sitio web mediante la aplicación de una misma hoja de estilo a cada página web que lo compone
- Crear páginas dinámicas utilizando JavaScript u otros lenguajes de programación junto con las hojas de estilo para crear textos y contenidos que responden a la interacción con el usuario

2.5.2. Tecnologías cliente-servidor

Entre las tecnologías web se encuentran también los lenguajes del lado del servidor y los lenguajes del lado del cliente. La diferenciación entre ambos tipos viene definida por el lugar físico desde donde se ejecuta la lógica de la aplicación. En un caso es el servidor el que realiza las funciones implementadas en el código, y en el otro, el navegador web del usuario es el que se ocupa de ejecutar dichas funciones.

También es conveniente tener en cuenta la disimilitudes entre los lenguajes estáticos y los lenguajes dinámicos. En los inicios de Internet era predominante el uso de los lenguajes estáticos, sin embargo con el despliegue de las nuevas plataformas y aplicaciones

web han surgido nuevos lenguajes, que siendo combinados con los estáticos, han conseguido formar un grupo de tecnologías que da sustento al actual Internet. La principal diferencia entre ambos tipos de lenguajes se encuentra en que los estáticos necesitan precompilar el código antes de ser ejecutados mientras que los dinámicos pertenecen al grupo de lenguajes de alto nivel que son interpretados en tiempo de ejecución. Estos lenguajes de alto nivel se combinan con lenguaje HTML de manera que se puedan alcanzar las funcionalidades deseadas.

El flujo de funcionamiento para el código ejecutado en el servidor comienza con una petición realizada a éste por el navegador del cliente. Esta petición es procesada por el servidor que ejecuta las funciones definidas en la lógica de la aplicación. Finalmente se genera una respuesta que es insertada en el código HTML que se envía de vuelta al navegador. Este método hace que la lógica de negocio permanezca más segura ya que el código de la aplicación se queda siempre en el servidor y no alcanza el cliente. Dentro del grupo de tecnologías del lado del servidor tenemos las siguientes:

- CGI (*Common Gateway Interface*) [32] es el sistema más antiguo que existe para presentar contenidos dinámicos en las aplicaciones web. En las aplicaciones CGI, el servidor recibe las peticiones del cliente y las envía a una aplicación externa. Ésta genera la respuesta que se envía al cliente. Los CGI pueden estar escritos en diversos lenguajes como C, C++, Visual Basic o PERL. Su uso se encuentra en decadencia debido a varios motivos entre los que destaca la dificultad de desarrollo de los programas y la carga que genera su ejecución en el servidor.

- ASP.NET [33] se trata de un framework comercializado por Microsoft y utilizado, entre otras funciones, para desarrollar sitios web dinámicos. ASP .NET, es el sucesor de la tecnología ASP (*Active Server Pages*). Para el desarrollo de ASP.NET se pueden utilizar lenguajes como C#, VB.NET o J#. Para que las aplicaciones web desarrolladas mediante ASP.NET puedan funcionar, es necesario tener instalado *Internet Information Server* junto con Microsoft Framework Net.

- Los *servlets* y JSP (*Java Server Pages*) son tecnologías Java [34] desarrolladas por SUN Microsystems y utilizadas para la creación de aplicaciones web dinámicas. JSP es una página web con etiquetas especiales y código Java incrustado (el código JSP se puede embeber en el código HTML), mientras que un *servlet* es un programa Java que recibe peticiones y genera a partir de ellas respuestas en forma de página web, en HTML. Poseen ventajas de Java, como la portabilidad de la plataforma además de tener la facilidad de trabajar tanto con la lógica de negocio como con el acceso a datos.

- PERL (*Practical Extraction and Report Language*) [35] es un potente lenguaje de alto nivel, interpretado y dinámico basado en C, AWK y sed, entre otros. Se puede utilizar para una gran variedad de tareas, destacando entre éstas, el desarrollo Web, la administración de sistemas y la programación de redes. Posee la ventaja de llevar muchos años en funcionamiento, por lo cual existe una gran

cantidad de bibliotecas y módulos que pueden ser usados por los desarrolladores de aplicaciones. Funciona en las principales plataformas como Unix, Mac OS X y Windows. Se distribuye bajo licencia de software libre.

- Python es un lenguaje de programación interpretado cuyo código no tiene necesidad de ser compilado. Se compara habitualmente con PERL. Python es multiplataforma y ha sido portado a las máquinas virtuales de Java y .NET. Es un lenguaje de código abierto que permite la creación de todo tipo de programas, incluyendo aplicaciones web [36].

- Ruby es un lenguaje dinámico orientado a objetos que inspira su sintaxis en Perl, Smalltalk, Eiffel, Ada y Lisp. Es distribuido bajo licencia de software libre. Se puede ejecutar en diversas plataformas. Se trata de un lenguaje dinámico para una programación orientada a objetos rápida y sencilla. El desarrollo de aplicaciones web con esta tecnología se realiza, principalmente, con el framework *Ruby on Rails* [37].

- PHP (PHP Hypertext Preprocessor) [38] es un lenguaje *open source* que permite desarrollar páginas web dinámicas. PHP ejecuta el código en el servidor, generando HTML y enviándolo al cliente. El cliente recibirá los resultados de ejecutar el script, sin ninguna posibilidad de determinar qué código ha producido dichos resultados. PHP es extremadamente simple de utilizar para el principiante, pero a su vez, ofrece muchas características avanzadas para los programadores profesionales. Para la creación de aplicaciones web los desarrolladores se apoyan en algunos frameworks, que facilitan el desarrollo manteniendo una estructura ordenada del código y proveyendo librerías adicionales. Para PHP existen diversos frameworks para programación orientada a objetos y para implementación de la arquitectura MVC (Modelo-Vista-Controlador). Entre ellos cabe destacar Symfony [39] y ZendFramework [40].

En el caso de los lenguajes del lado del cliente, el flujo de ejecución es distinto. Los navegadores interpretan y ejecutan estos lenguajes, sin necesidad de tratamiento previo. Entre este tipo de tecnologías cabe destacar:

- Los applets son pequeñas aplicaciones escritas en Java [41] que son ejecutados dentro de una página web en HTML. El encargado de su ejecución es el navegador web, haciendo uso de la máquina virtual de Java (JVM). Los applets alcanzan el cliente ya precompilados, es por ello que la manera de trabajar de éstos varía un poco con respecto a los lenguajes de script como Javascript. Tienen como ventajas principales la menor dependencia del navegador y la independencia total respecto al sistema operativo de la máquina donde se ejecuten.

- Javascript [42] es un lenguaje de programación utilizado para crear pequeños programas encargados de realizar acciones en el ámbito de una página web, siendo el

navegador el que soporta la carga de procesamiento. Se emplea para generar efectos en las aplicaciones web y para interactuar con el usuario, mediante el control de eventos. Los scripts realizados en Javascript son mas sencillos de programar que los applets. Existen infinidad de librerías de código, como por ejemplo jQuery [43], que simplifican el uso de este lenguaje, pudiéndose generar muchos efectos con pocas líneas de código.

- VBScript (*Visual Basic Script*) [44] es un lenguaje de programación de scripts compatible con Internet Explorer y otros sistemas Microsoft. Se trata de un lenguaje interpretado por el Windows Scripting Host de Microsoft. Sólo puede utilizarse con navegadores de Microsoft, con un funcionamiento similiar a JavaScript.

- ActionScript [45] es el lenguaje de programación de Flash (propiedad de Adobe). Flash permite crear efectos y diseños especiales para páginas web. Para que se puedan visualizar los contenidos generados en Flash, el navegador debe tener instalado un plugin que se lo permita.

- AJAX *Asynchronous JavaScript And XML* es una técnica de desarrollo web para crear aplicaciones interactivas. El concepto es cargar y renderizar una página, luego mantenerse en esa página mientras scripts y rutinas van al servidor buscando, en segundo plano, los datos que son usados para actualizar los gráficos de la página. De esta manera se evita recargar la página cada vez que el usuario genera un evento. AJAX es una tecnología asíncrona que consigue los datos necesarios alojados en el servidor sin interferir en la utilización de la aplicación por parte del usuario. Para que este concepto funcione, se añade un motor AJAX como intermediario entre servidor y usuario. El navegador web carga al inicio de la sesión el motor AJAX que será el encargado de renderizar la interfaz de usuario y comunicarse con el servidor.

Capítulo 3

Análisis del sistema

Durante la etapa de análisis, se realizará la especificación de requisitos software mediante la cual se pretende reflejar las necesidades y deseos, planteados en los apartados 1.1 y 1.2, de forma que éstos queden formulados de manera clara, completa y detallada. El proceso hasta alcanzar el nivel de detalle recogido en esta especificación, se ha basado en el refinamiento de cada requisito de manera incremental.

El objetivo del análisis detallado del sistema realizado en este apartado es constituir la base para el diseño y la implementación del mismo, así como establecer un punto inicial de conocimiento del proyecto.

3.1. Descripción de los objetivos

Este proyecto nace con la idea de sacar el máximo potencial de una herramienta sobresaliente por si misma, como es NINOS, pero con una cantidad inmensa de posibilidades de explotación aún por descubrir. Entre este ingente número de posibilidades surge la integración de NINOS con Twitter.

El principal objetivo que se establece dentro de este proyecto es el de proveer a Twitter de funcionalidades extras no recogidas actualmente en su plataforma, a través de la integración de NINOS en el flujo habitual de funcionamiento. Esta integración consistirá en la aportación de la generación de animaciones y entornos renderizados de manera automática de NINOS a algunos de los casos de uso más habituales dentro de Twitter.

El sistema resultante de este proyecto deberá representar una escenificación de la lectura, por medio de un avatar. Esta lectura podrá ser solicitada para los últimos *tweets* publicados por un mismo usuario, para los últimos *tweets* que contengan un determinado fragmento de texto o para los que pertenezcan a una conversación entre distintos usuarios.

El otro gran objetivo fijado para el desarrollo de este proyecto es la realización de

una sencilla aplicación web que sirva de escaparate para presentar los resultados alcanzados. Se ha considerado que la mejor manera de realizar una demostración será el despliegue de un cliente web con un funcionamiento similar a la aplicación real de Twitter. Esto facilitará la interacción de los actuales usuarios de Twitter ya que el periodo de reconocimiento de la aplicación sería prácticamente nulo debido a las similitudes funcionales entre ambas. El cliente desarrollado a tal efecto, deberá recoger la opción de generar y visionar las animaciones a petición.

3.2. Especificación de requisitos software

Los requisitos software describen principalmente lo que debe de hacer el sistema y tienen un alto grado de importancia en relación con la verificación y el seguimiento del mismo. Estos requisitos deberán cubrir todas y cada una de las necesidades identificadas de modo que se logren cumplir los objetivos planteados en el apartado anterior.

Los requisitos software aquí expuestos no deben ser entendidos como un bloque cerrado, diseñado en las fases iniciales e inamovible ante eventualidades surgidas durante el proyecto. Toman verdadera importancia dentro del conjunto del proyecto gracias a la comunicación constante y bidireccional entre éstos y la definición de los casos de uso en el proceso de diseño del sistema (Apartado 4.2). Esto ha resultado en una especificación de requisitos muy depurada, ajustándose perfectamente a las necesidades reales del proyecto, y unos casos de uso que recogen la funcionalidad adecuada para contemplar todos los requisitos fijados.

Antes de comenzar a enumerar los requisitos es conveniente dar una breve descripción de cada uno de los grupos en los que se engloban.

- Requisitos funcionales: Definen qué tiene que hacer el sistema
- Requisitos de interfaces externas: Definen el modo en que interactúan y se comunican las interfaces y diferentes módulos del sistema
- Requisitos de compatibilidad: Definen las características de los dispositivos finales con los que el sistema debe permitir un correcto funcionamiento
- Requisitos de disponibilidad: Definen la utilización de los recursos para que la aplicación funcione correctamente
- Requisitos de rendimiento: Definen los parámetros que debe cumplir el sistema para no ver degradado su rendimiento y por tanto la calidad de experiencia que se ofrece al usuario
- Requisitos de documentación: Definen las características que debe cumplir la documentación del sistema

- Requisitos de mantenibilidad: Definen los puntos a cumplir para que el correcto funcionamiento del sistema sea sostenido en el tiempo
- Requisitos de testabilidad: Definen las comprobaciones que deberá realizar el sistema para que el funcionamiento sea el adecuado
- Requisitos de soporte y operación: Definen los recursos que deben proveerse para dar soporte a usuarios y poder operar el sistema provisionando nuevos usuarios o dándolos de baja, por ejemplo

3.2.1. Requisitos funcionales

Identificador	**RSF-01**
Descripción	**Consulta *Timeline* de un usuario de Twitter** El sistema presenta los N últimos *tweets* que aparecen en el *timeline* de un usuario de Twitter (si su cuenta no está protegida) a petición del usuario del sistema
Prioridad	Alta

Tabla 3.1. Requisito RSF-01

Identificador	**RSF-02**
Descripción	**Consulta los N últimos *tweets* publicados sobre un tema** El sistema presenta, a petición del usuario, la lista de los N últimos *tweets* que se han publicado en Twitter sobre un tema
Prioridad	Alta

Tabla 3.2. Requisito RSF-02

Identificador	**RSF-03**
Descripción	**Consulta los N últimos *tweets* publicados que contienen un determinado *hashtag*** El sistema presenta, a petición del usuario, la lista de los N últimos *tweets* publicados que contengan un determinado *hashtag*
Prioridad	Alta

Tabla 3.3. Requisito RSF-03

Identificador	**RSF-04**
Descripción	**Información del perfil de un usuario de Twitter** El sistema presenta la información del perfil de un usuario de Twitter, si su cuenta no está protegida. La información del perfil de un usuario comprende el nombre de usuario, el apodo, la descripción del usuario, el número de *tweets* escritos, el número de *following* y el número de *followers*
Prioridad	Media

Tabla 3.4. Requisito RSF-04

Identificador	**RSF-05**
Descripción	**Consulta *Following* de un usuario de Twitter** El sistema presenta la lista de *following* que tiene un determinado usuario de Twitter, si su cuenta no está protegida
Prioridad	Baja

Tabla 3.5. Requisito RSF-05

Identificador	**RSF-06**
Descripción	**Consulta *Followers* de un usuario de Twitter** El sistema presenta la lista de *followers* que tiene un determinado usuario de Twitter, si su cuenta no está protegida
Prioridad	Baja

Tabla 3.6. Requisito RSF-06

Identificador	**RSF-07**
Descripción	**Búsqueda general en Twitter** El sistema presenta los N últimos *tweets* publicados que contengan determinadas palabras, usuarios o *hashtags*
Prioridad	Alta

Tabla 3.7. Requisito RSF-07

Identificador	**RSF-08**
Descripción	**Señalización de conversaciones en Twitter** El sistema señalizará de manera visual los *tweets* que hayan sido publicados como respuesta a otro y que, por tanto, pertenecen a una conversación
Prioridad	Alta

Tabla 3.8. Requisito RSF-08

Identificador	**RSF-09**
Descripción	**Animar mensaje** El sistema generará una animación nueva y única (aleatorizada para cada ejecución) para un mensaje
Prioridad	Alta

Tabla 3.9. Requisito RSF-09

Identificador	**RSF-10**
Descripción	**Animar grupo de mensajes** El sistema procederá a generar una animación nueva y única (aleatorizada para cada ejecución) para un grupo de mensajes publicados por un mismo usuario o que contenga determinados términos o un *hashtag*
Prioridad	Alta

Tabla 3.10. Requisito RSF-10

Identificador	**RSF-11**
Descripción	**Animar conversación** El sistema procederá a generar una animación nueva y única (aleatorizada para cada ejecución) para una conversación entre usuarios
Prioridad	Alta

Tabla 3.11. Requisito RSF-11

Identificador	RSF-12
Descripción	**Configuración de nivel de protección ante palabras malsonantes** El sistema permitirá la configuración de un filtro de palabras malsonantes para los *tweets*. Se puede configurar en nivel leve, grave o mantener desactivado el filtro
Prioridad	Baja

Tabla 3.12. Requisito RSF-12

Identificador	RSF-13
Descripción	**Filtrado de palabras malsonantes (nivel leve)** El sistema detectará la presencia de la palabra malsonante en un *tweet* y procederá a codificarla con asteriscos. El actor que aparezca en el vídeo generado no pronunciará dicha palabra, en su lugar se escuchará un pitido
Prioridad	Baja

Tabla 3.13. Requisito RSF-13

Identificador	RSF-14
Descripción	**Filtrado de palabras malsonantes (nivel grave)** El sistema detectará la presencia de la palabra malsonante en un *tweet* y procederá a descartar dicho *tweet*. Este mensaje no será presentado ni podrá ser animado aunque sí se indicará que se ha descartado
Prioridad	Baja

Tabla 3.14. Requisito RSF-14

Identificador	**RSF-15**
Descripción	**Sustitución de acrónimos** El sistema detectará un acrónimo de los utilizados habitualmente en el lenguaje de la Red en un *tweet* y procederá a sustituirlo por su equivalente en forma desglosada para que el actor de la animación pueda leerlo
Prioridad	Baja

Tabla 3.15. Requisito RSF-15

Identificador	**RSF-16**
Descripción	**Detección de URLs** El sistema detectará las URLs incluídas en los *tweets* que sean procesados por la aplicación de entre todo el texto que componga el mensaje
Prioridad	Alta

Tabla 3.16. Requisito RSF-16

Identificador	**RSF-17**
Descripción	**Generación de código QR representado una URL** El sistema generará un código QR, a partir de una URL detectada, para permitir su inclusión en una animación. De este modo, el usuario mediante un dispositivo provisto de un lector visual adecuado podrá leerlo y obtener acceso a dicha URL
Prioridad	Media

Tabla 3.17. Requisito RSF-17

Identificador	RSF-18
Descripción	**Visualización de vídeo generado** El sistema permitirá la visualización del vídeo generado a petición del usuario a partir de uno o varios *tweets*
Prioridad	Alta

Tabla 3.18. Requisito RSF-18

Identificador	RSF-19
Descripción	**Compartir vídeo generado** El sistema permitirá al usuario publicar en Twitter el vídeo generado
Prioridad	Baja

Tabla 3.19. Requisito RSF-19

Identificador	RSF-20
Descripción	**Detección de idioma de un *tweet*** El sistema detectará el idoma en el que se ha escrito un *tweet*
Prioridad	Media

Tabla 3.20. Requisito RSF-20

Identificador	RSF-21
Descripción	**Generación de voz en idioma detectado** El sistema generará la voz con la que el actor de la escena leerá un *tweet* en el idioma en el que esté
Prioridad	Media

Tabla 3.21. Requisito RSF-21

Identificador	RSF-22
Descripción	**Animación siguiendo diseño de la demo** El sistema generará las animaciones siguiendo el diseño de la demo llevado a cabo en el Apartado 4.4
Prioridad	Alta

Tabla 3.22. Requisito RSF-22

Identificador	RSF-23
Descripción	**Manipulación datos de Twitter** El sistema almacenará de manera temporal los datos que vaya obteniendo de Twitter. De este modo se busca evitar consultas repetidas al API de Twitter que afecten al rendimiento global de la aplicación
Prioridad	Alta

Tabla 3.23. Requisito RSF-23

Identificador	RSF-24
Descripción	**Señalización de *retweets* en Twitter** El sistema señalizará de manera visual los *tweets* que hayan sido reenviados mediante un icono. Se mantiene la información del *tweet* original añadiéndole el ID del usuario que realiza el reenvío
Prioridad	Alta

Tabla 3.24. Requisito RSF-24

3.2.2. Requisitos no funcionales

3.2.2.1. Requisitos de interfaces externas

Identificador	RSIE-01
Descripción	**Interfaz de usuario sencilla e intuitiva** La aplicación debe presentar una interfaz de usuario sencilla e intuitiva que permita interactuar con ella y desplazarse por las distintas vistas de manera adecuada de modo que no se limite el tipo de usuario que pueda hacer uso de ella
Prioridad	Alta

Tabla 3.25. Requisito RSIE-01

Identificador	RSIE-02
Descripción	**Comunicación con Twitter** La comunicación con la aplicación Twitter se realizará a través del API de ésta
Prioridad	Alta

Tabla 3.26. Requisito RSIE-02

3.2.2.2. Requisitos de compatibilidad

Identificador	RSC-01
Descripción	**Independencia del sistema operativo del usuario** La aplicación web de demostración de resultados debe funcionar independientemente del sistema operativo desde el que acceda el usuario. Será por tanto compatible con sistemas Android, Windows, iOS, RIM, Linux, Lion OS X...
Prioridad	Baja

Tabla 3.27. Requisito RSC-01

Identificador	RSC-02
Descripción	**Independencia del navegador web del usuario** La aplicación web de demostración de resultados debe funcionar independientemente del navegador web desde el que acceda el usuario. Será por tanto compatible, siempre que tengan soporte para Flash o HTML5, con navegadores Chrome, Firefox, Internet Explorer, Opera, Safari...
Prioridad	Baja

Tabla 3.28. Requisito RSC-02

Identificador	RSC-03
Descripción	**Características de vídeo óptimas para el dispositivo final** La aplicación web de demostración de resultados debe ser capaz de presentar los vídeos generados con unas características óptimas para cada tipo de plataforma desde donde el usuario ejecute la aplicación. Se deberá utilizar el códec H.264/MPEG-4 AVC de vídeo y el códec AAC modificando la resolución adecuada para cada dispositivo
Prioridad	Baja

Tabla 3.29. Requisito RSC-03

3.2.2.3. Requisitos de disponibilidad

Identificador	RSDI-01
Descripción	**Concurrencia de usuarios** El sistema implementado para la prueba de concepto recogida en este proyecto, sólo debe posibilitar el acceso a un único usuario de manera concurrente
Prioridad	Baja

Tabla 3.30. Requisito RSDI-01

3.2.2.4. Requisitos de rendimiento

Identificador	RSR-01
Descripción	**Tiempo de respuesta máximo a evento de generación de vídeo** Se establece en **300 segundos** el tiempo máximo que puede tardar el servidor de aplicación en realizar el proceso completo de generación de un vídeo a partir de los *tweets* correspondientes
Prioridad	Baja

Tabla 3.31. Requisito RSR-01

Identificador	RSR-02
Descripción	**Tiempo de respuesta máximo a evento de la interfaz de usuario** Se establece en **5 segundos** el tiempo máximo que puede tardar el servidor de aplicación en responder a una interacción del usuario con la interfaz del sistema, excepto para la generación de vídeos
Prioridad	Baja

Tabla 3.32. Requisito RSR-01

Identificador	RSR-03
Descripción	**Número máximo de *tweets* incluidos en el vídeo** Se establece en **diez** el número máximo de *tweets* que se pueden seleccionar para incluir en un vídeo que va a ser generado para no exponer al servidor a una sobrecarga o malfuncionamiento debido al tamaño que ocuparían en disco y el tiempo que conllevaría procesarlos, entrando en conflicto con RSR-01
Prioridad	Baja

Tabla 3.33. Requisito RSR-01

3.2.2.5. Requisitos de documentación

Identificador	RSD-01
Descripción	**Documentación en LaTeX** La documentación generada durante el desarrollo del proyecto debe ser realizada mediante lenguaje LaTeX
Prioridad	Alta

Tabla 3.34. Requisito RSD-01

Identificador	RSD-02
Descripción	**Contenido de la documentación** La documentación generada debe contener detallados los siguientes temas: una breve visión del estado del arte, el análisis del sistema que se va a implementar, la explicación del diseño del mismo y la descripción del proceso de implementación
Prioridad	Alta

Tabla 3.35. Requisito RSD-02

Identificador	RSD-03
Descripción	**Documentación precisa** La documentación generada debe ser precisa en las justificaciones. Se exige la documentación de cada módulo del sistema explicando la funcionalidad que ofrece, métodos o funciones ofrecidas por otros módulos que utiliza, y el modelo de ejecución por parte de otros módulos
Prioridad	Alta

Tabla 3.36. Requisito RSD-03

3.2.2.6. Requisitos de mantenibilidad

Identificador	RSM-01
Descripción	**Facilidad de actualización del sistema** El sistema debe ser diseñado de modo que permita su adaptación a la incorporación de nuevas funcionalidades
Prioridad	Alta

Tabla 3.37. Requisito RSM-01

Identificador	RSM-02
Descripción	**Alta independencia entre módulos del sistema** Los módulos integrantes del sistema deben tener un alto grado de independencia de forma de permitan su implementación y testeo de manera autónoma al resto
Prioridad	Alta

Tabla 3.38. Requisito RSM-02

3.2.2.7. Requisitos de testabilidad

Identificador	RST-01
Descripción	**Idiomas de escritura de los *tweets*** El sistema sólo será testado en su correcto funcionamiento para los *tweets* escritos en inglés y en castellano. Los *tweets* recuperados e interpretados por el personaje que aparecerá en el vídeo podrán estar en cualquier idioma, pero sólamente se garantiza la pronunciación del mismo para los dos idiomas anteriormente señalados
Prioridad	Alta

Tabla 3.39. Requisito RST-01

3.2.2.8. Requisitos de soporte y operación

El ciclo de vida de este proyecto transcurre íntegramente dentro de un entorno de desarrollo, sin contemplar en ningún momento un entorno de producción. Por tanto, tener requisitos de soporte y operación del sistema, no aplica.

3.3. Descripción del usuario

El usuario final debe conocer el funcionamiento de un navegador web, sea cual sea la plataforma, para poder acceder a la dirección web en la que se tendrá acceso a la aplicación, así como saberse desenvolver en el uso de la misma. Además, debe tener conocimientos básicos sobre el funcionamiento de Twitter y las distintas opciones que éste servicio permite. Algunas de estas opciones, vistas anteriormente en el capítulo 2.4, son la respuesta a una publicación (***Reply***), el reenvío de una publicación (***Retweet***) o el etiquetado de temas (***Hashtag***).

3.4. Entorno de trabajo

Esta sección recoge las tecnologías elegidas, de entre algunas de las tratadas en el Capítulo 2, para implementar el sistema buscando el cumplimiento de todos y cada uno de los requisitos software especificados en la Sección 3.2.

3.4.1. Tecnologías web

En la Sección 2.5 se hacía referencia a un gran número de tecnologías utilizadas en la implementación de aplicaciones web. De entre todas, se ha decidido utilizar PHP para programar toda la lógica de negocio que forma el núcleo de funcionamiento del servidor de la aplicación.

3.4.1.1. PHP

La elección de este lenguaje para la programación de la arquitectura del servidor de la aplicación viene sobretodo por lo sencillo que es el desarrollo con él y la velocidad de ejecución. Habría que añadir la cantidad de librerías y métodos que incluye por defecto para el tratamiento de todo tipos de datos y principalmente las funciones de interacción con la Red. Entre otras cosas también provee soporte para expresiones regulares que serán de gran ayuda para el cumplimiento del requisito RSF-16 (Tabla 3.16). Además tiene disponible un manual online muy completo [47] que facilita el soporte de primer nivel durante el desarrollo.

Para la ejecución del código PHP que forma la lógica de la aplicación se ha optado por la utilización del servidor Apache como servidor de aplicaciones.

3.4.1.1.1. Apache HTTP Server

Apache es el servidor web más utilizado del mundo [48]. La característica que más ha ayudado a alcanzar esta posición dominante es que se trata de software libre con una importante comunidad de usuarios a su alrededor que reporta los errores que puedan surgir facilitando una constante actualización del mismo. A esto hay que añadir su soporte multiplataforma que lo hace más accesible.

En el caso de este proyecto, se ha tenido muy en cuenta la numerosa documentación disponible en Internet para atajar cualquier problema que pudiera surgir durante la implementación. Otro punto clave en la elección ha sido la existencia del paquete XAMPP que contiene un completo conjunto de soluciones para desarrollar aplicaciones web entre las que se encuentran PHP y Apache. La principal ventaja de XAMPP viene por la facilidad de configuración inicial y la rapidez con que se consigue poner en marcha por primera vez.

3.4.2. Generación de códigos QR

El cumplimiento del requisito funcional RSF-17 (Tabla 3.17) pasa por encontrar una herramienta capaz de codificar una URL en un código QR. Existen diversas aplicaciones online que realizan esta función buscada y algunas de éstas además proveen un API abierto para realizar peticiones y que devuelva como respuesta el código QR. De entre todas, destacan principalmente Kaywa QR-Code y Google Chart Tools que también

dispone, entre otras herramientas, de un generador de códigos QR.

Se decide elegir herramienta propiedad de Google debido al buen hacer de esta compañía en la gran mayoría de sus aplicaciones dotándolas de una estabilidad y accesibilidad que reduce la incertidumbre que siempre implica utilizar una herramienta externa a la que no se tiene acceso ante eventuales problemas que puedan surgir. A esto hay que añadir que la aplicación de Kaywa se encuentra en estado experimental.

3.4.2.1. Google Chart Tools

El API de Google Chart genera gráficos en formato imagen como respuesta a peticiones HTML. El API permite generar diversos tipos de gráficos como son, gráficos circulares o de barras, mapas, fórmulas e incluso códigos QR. Es posible personalizar toda la información del gráfico que se desea generar (color, tamaño, etiquetas...).

Las peticiones deben incluir obligatoriamente el tipo de gráfico que se desea generar, los datos que va a incluir dicho gráfico, así como su tamaño. Además, cada tipo de gráfico permite añadir distintos parámetros adicionales que completan la información que utiliza el API para su generación. El formato de la petición es el siguiente:

```
http://chart.apis.google.com/chart?cht=<tipo_de_grafico>
 &chd=<datos_del_grafico>&chs=<dimensiones_del_grafico>
              &...parametros_adicionales...
```

3.4.3. Detección de idiomas

El requisito RSF-20 (Tabla 3.20) genera la necesidad de utilizar una herramienta externa que sea capaz de detectar el idioma en que está escrito un *tweet*. Al igual que ocurre con la generación de códigos QR (Sección 3.4.2) existen varios APIs online de uso abierto para todos los usuarios entre los que se encuentran AlchemyAPI y Google Translate API. Como en ese caso, se va a apostar por el uso de la herramienta de Google por los mismos motivos que se esgrimen entonces.

3.4.3.1. Google Translate API v2

Google Translate API permite la detección del idioma de un texto enviado a través de una petición HTTP GET a una URL con el siguiente formato

```
https://www.googleapis.com/language/translate/v2/detect?
                key=GOOGLE_API_KEY&q=TEXTO
```

La respuesta a esta petición HTTP es devuelta en formato JSON y contiene el código del idioma detectado para el texto enviado y la fiabilidad de la predicción realizada.

3.4.4. Sintetizado de voz

En la búsqueda de una herramienta adecuada para realizar el proceso de sintetizado de voz a partir de un texto, no se ha encontrado ningún software disponible entre los *open source*. Entre las aplicaciones propietarias se seleccionó 2nd Speech Center debido principalmente a que era de las pocas que proporcionaban funcionalidad a través de línea de comandos por un coste razonable.

3.4.4.1. 2nd Speech Center

2nd Speech Center permite convertir texto a archivos MP3 o WAV además de ser totalmente compatible con las voces que implementan la familia SAPI5. Estas voces se encuentran entre las más realistas que existen en la actualidad a nivel de usuario.

La aplicación corre sobre sistemas Windows y da la posibilidad de realizar conversiones *text-to-speech* haciendo llamadas a la shell de Windows pasando el texto a convertir y la ruta donde debe situarse el archivo de sonido generado.

3.4.5. Transcodificación

La transcodificación se vuelve imprescidible dentro del sistema debido a la necesidad de transformar el vídeo generado por NINOS de modo que se consiga una reducción drástica en el peso del archivo sin sacrificar la calidad del mismo. A este motivo se añade el cumplimiento de los requisitos RSC-02 (Tabla 3.28) y RSC-03 (Tabla 3.29).

Existen una amplia cantidad de sistemas de transcodificación en la Nube que disponen de APIs para su integración en nuestro sistema. Sin embargo, estas aplicaciones se basan en un modelo de explotación por suscripción mensual que aumenta los gastos, ya de por sí limitados, que se contemplan en el presupuesto del proyecto. Es por esto, que se decide utilizar herramienta de software libre FFmpeg que permite su integración con nuestro sistema mediante el uso de llamadas a línea de comandos.

3.4.5.1. FFmpeg

FFmpeg es un proyecto *open source* compuesto por distintas librería y programas que permiten el tratamiento de datos multimedia. Posee un numeroso grupo de codecs de audio y vídeo que permiten la conversión entre formatos.

La interfaz de línea de comandos permite configurar infinidad de parámetros de las transcodificaciones, aunque también provee de un sistema sencillo que facilita este proceso haciendo necesario tan sólo indicar los codec de salida y el *bitrate* que se desean.

```
ffmpeg [[infile options][-i infile]] [outfile options] outfile
```

3.5. Requisitos del hardware

Los equipos hardware necesarios para el correcto funcionamiento del sistema mantenido en el tiempo, deben cumplir una serie de requisitos de modo que se pueda asegurar un rendimiento adecuado de la aplicación. Por esto, se define una lista de características recomendadas para el servidor que aloja la aplicación web.

- Procesador Intel Core i7 de 4 núcleos o superior
- Sistema operativo Windows 7 o superior
- 4GB de memoria RAM
- Tarjeta gráfica Nvidia o ATI actualizada con los últimos drivers del fabricante, con soporte para OpenGL 2.0 o posterior con al menos 128MB de memoria
- 120GB de disco duro libres para la instalación del servidor y las aplicaciones necesarias, así como el almacenamiento de los vídeos y el material adicional generado durante el funcionamiento normal de la aplicación desarrollada.
- Última versión de la librería OpenAL instalada
- Última versión de las librerías Microsoft VC++ Runtime instaladas

3.6. Restricciones generales

Las principales restricciones que tienen aplicación en el sistema que se va a desarrollar son las relativas al material audivisual soportado por la Plataforma NINOS que se especifican en la Sección 2.3.2.

3.7. Supuestos y dependencias

El funcionamiento correcto del sistema vendrá condicionado por las siguientes características:

- Se debe entender por *tweets* publicados por un usuario aquellos escritos por éste así como los reenvíos de publicaciones de otros usuarios (*retweets*) que haga
- El correcto funcionamiento del sistema operativo, así como del resto de los componentes que componen el servidor de aplicación.
- El correcto funcionamiento de la aplicación de sintetizado de voz, 2nd Speech Center, instalada en el servidor de aplicación.

- La disposición por parte del usuario de una conexión a Internet con la suficiente capacidad para la realización del servicio de manera óptima
- El correcto funcionamiento del servicio provisto por Twitter
- El correcto funcionamiento y disponibilidad de acceso al API para desarrolladores dispuesto por Twitter
- El correcto funcionamiento y disponibilidad de acceso al API de idiomas de Google para detección de idioma
- El correcto funcionamiento y disponibilidad de acceso al API de Google Chart para generación de códigos QR
- Un entorno seguro tanto para los datos almacenados en el servidor como para la información enviada por éste para ser presentada en el navegador web del usuario son asumidas
- La conexión a Internet del usuario será superior a 1Mbps para que la experiencia de usuario alcance niveles adecuados

Capítulo 4

Diseño del sistema

4.1. Introducción

Una vez analizado en profundidad el sistema a implementar en este proyecto se debe diseñar la manera en la que se va a proceder a cumplir los requisitos fijados en la etapa anterior.

A partir de este capítulo se ha de tener en cuenta que el diseño y la implementación del sistema van a ir completamente orientados al despliegue del demostrador mediante el que se validarán los resultados de este proyecto. De este modo, este apartado de diseño se centrará en explicar los casos de uso diseñados para dicha demo así como se describirá el proceso de creación de las estructuras de las distintas animaciones que serán generadas por la aplicación.

4.2. Casos de uso

Los casos de uso se ocupan de especificar el comportamiento del sistema ante las interacciones con los actores. Dicho de otro modo, los casos de uso definen de manera detallada la forma en que los servicios proporcionados por el sistema son utilizados así como la secuencia surgida como respuesta a un evento iniciado por un actor.

Los actores son los personajes que participarán en un caso de uso. En la aplicación que se implementa durante este proyecto tan sólo se recoge la existencia de un actor. Este actor se corresponde con el usuario final de dicha aplicación.

4.2.1. Diagrama

El diagrama de casos de uso que se muestra a continuación proporciona una visión general de la interacción de los actores con el sistema realizando un primer acercamiento a los casos de uso contemplados para esta aplicación.

Como guía para comprender el significado de la Figura 4.1, se debe saber que las líneas que parten desde el actor (**Usuario**) indican que éste tiene interacción directa con los casos de uso destino de las mismas. Las flechas que parten de algunos casos de uso, sin embargo, indican que el caso de uso destino está englobado en el caso de uso origen y por tanto éste da acceso al destino. Estos casos de uso también requieren de la interacción del usuario con el sistema sólo que se realiza a través de otros casos de uso.

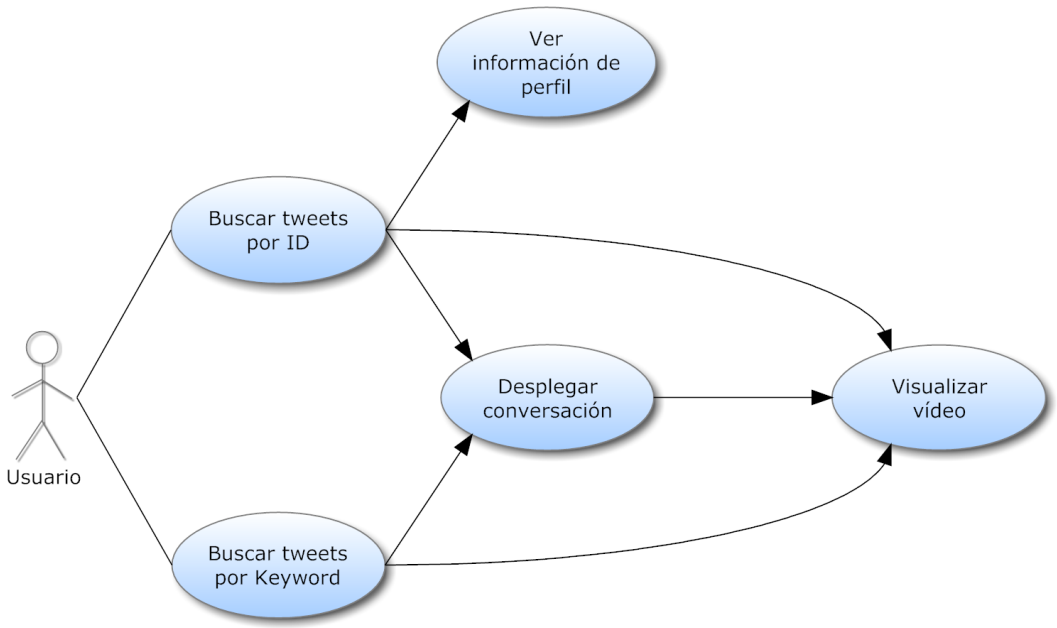


Figura 4.1. Diagrama de casos de uso del sistema

Como se puede ver, las primeras acciones que el usuario puede realizar con la aplicación están relacionadas con las distintas búsquedas de información en Twitter. Se puede realizar una búsqueda por ID (nombre de usuario en Twitter) o por Keyword (palabra, conjunto de palabras, *hashtag*, etc). Una vez se dispone de la lista de *tweets* en pantalla, existe la posibilidad de desplegar una conversación a partir de un *tweet* que esté indicado pertenezca a una o generar un vídeo a partir de un susbconjunto de *tweets* y visualizarlo un vez finalizado el proceso de generación. Otra acción que se puede realizar es el visionado de la información de perfil de un usuario de Twitter después de haber solicitado sus últimos *tweets*.

4.2.2. Descripción

Resulta muy importante, para comprender totalmente los casos de uso representados en la Figura 4.1, realizar una descripción textual de cada uno de ellos que se recogen en las siguientes tablas.

Nombre	**Buscar *tweets* por ID**
Descripción	Permite buscar los N últimos *tweets* publicados por un usuario de Twitter, identificado por su ID, presentándolos en formato lista. Cada *tweet* estará formado por la foto de perfil del usuario, el ID de éste, su nombre, el texto del mensaje publicado y la fecha en que se envió. En el caso de tratarse de una replica a un mensaje o un reenvío de un mensaje de otro usuario, se señalizará visualmente mediante el icono correspondiente. Si se trata de un *retweet*, se presentará con la información original indicando que ha sido reenviado por el usuario que posee el ID de la consulta. Los nombres de usuario de Twitter, las direcciones web y los *hashtag* que aparezcan en un *tweet* estarán identificados como hipervínculos
Precondiciones	El actor debe haber accedido a la página de inicio de la aplicación
Flujo normal	1. El actor selecciona la pestaña ID 2. El sistema muestra una caja de texto para introducir el ID del usuario y otra para introducir el número de los últimos *tweets* que se van a buscar 3. El actor introduce el ID del usuario y el número de *tweets* 4. El sistema devuelve una vista con una lista con la cantidad especificada de los últimos *tweets* publicados por el usuario representado por el ID
Flujo alternativo	3. Si no se introduce el número de *tweets*, el sistema por defecto recuperará 10 4. Si el usuario buscado posee una cuenta privada o no existe, el sistema mostrará la información del error
Poscondiciones	El sistema dispone de toda la información recuperada de la consulta para posteriores acciones
Requisitos	RSF-01, RSF-08, RSF-16, RSF-23, RSF-24

Tabla 4.1. Caso de Uso 1

Nombre	**Buscar *tweets* por Keyword**
Descripción	Permite buscar los N últimos *tweets* publicados por usuarios de Twitter que no posean una cuenta privada, que contengan en el texto el término o términos solicitados, presentándolos en formato lista. Cada *tweet* estará formado por la foto de perfil del usuario, el ID de éste, su nombre, el texto del mensaje publicado y la fecha en que se envió. En el caso de tratarse de una replica a un mensaje de otro usuario, se señalizará visualmente mediante el icono correspondiente. Se pueden realizar búsquedas de *tweets* que contengan una o varias palabras, que mencionen a un usuario o se relacionen con un *hashtag*. Los nombres de usuario de Twitter, las direcciones web y los *hashtag* que aparezcan en un *tweet* estarán identificados como hipervínculos
Precondiciones	El actor debe haber accedido a la página de inicio de la aplicación
Flujo normal	1. El actor selecciona la pestaña KEYWORD 2. El sistema muestra una caja de texto para introducir el término o los términos de los que se vaya a hacer la búsqueda y otra para introducir el número de los últimos *tweets* que se van a buscar 3. El actor introduce el término o el conjunto de términos de búsqueda y el número de *tweets* 4. El sistema devuelve una vista con una lista con la cantidad especificada de los últimos *tweets* publicados que contengan los términos de búsqueda
Flujo alternativo	3. Si no se introduce el número de *tweets*, el sistema por defecto recuperará 10 4. Si la búsqueda no encuentra resultados, el sistema mostrará la información del error
Poscondiciones	El sistema dispone de toda la información recuperada de la consulta para posteriores acciones
Requisitos	RSF-02, RSF-03, RSF-07, RSF-08, RSF-16, RSF-23, RSF-24

Tabla 4.2. Caso de Uso 2

Nombre	Desplegar conversación entre usuarios
Descripción	Permite desplegar y visualizar una lista con los *tweets* que forman una conversación entre varios usuarios de Twitter, siempre que no posean una cuenta privada, a partir de un mensaje marcado como tal mediante el icono correspondiente. Cada *tweet* estará formado por la foto de perfil del usuario, el ID de éste, su nombre, el texto del mensaje publicado y la fecha en que se envió. En el caso de tratarse de una replica a un mensaje de otro usuario, se señalizará visualmente mediante el icono correspondiente. Los nombres de usuario de Twitter, las direcciones web y los *hashtag* que aparezcan en un *tweet* estarán identificados como hipervínculos
Precondiciones	El actor debe haber realizado un búsqueda en la aplicación por cualquiera de los dos criterios disponibles. El sistema debe haber devuelto ante la consulta una lista con *tweets*. Al menos uno de los mensajes debe pertenecer a una conversación
Flujo normal	1. El actor selecciona un *tweet* marcado con el icono de conversación, indicando que pertenece a una 2. El sistema devuelve una vista con una lista con el mensaje o los mensajes que preceden al seleccionado en la conversación
Flujo alternativo	2. Si el sistema no encuentra los mensajes anteriores al seleccionado, mostrará la información del error
Poscondiciones	El sistema dispone de toda la información recuperada de la consulta para posteriores acciones
Requisitos	RSF-08, RSF-16, RSF-23

Tabla 4.3. Caso de Uso 3

Nombre	**Ver información de perfil de un usuario**
Descripción	Permite visualizar la información de perfil de un usuario de Twitter, siempre que no posea una cuenta privada. Dicha información está compuesta por la foto de perfil del usuario y el ID de éste en todos los casos. Siempre que estén definidos, también se verá su nombre y su descripción biográfica. Además aparecerá el recuento de los *tweets* que ha publicado ese usuario y el número de *following followers* que tiene en ese momento.
Precondiciones	El actor debe haber accedido a la página de inicio de la aplicación
Flujo normal	1. El actor selecciona la pestaña ID 2. El sistema muestra una caja de texto para introducir el ID del usuario y otra para introducir el número de los últimos *tweets* que se van a buscar 3. El actor introduce el ID del usuario y el número de *tweets* 4. El sistema devuelve una vista con la información de perfil del usuario representado por el ID
Flujo alternativo	4. Si el usuario buscado posee una cuenta privada o no existe, el sistema mostrará la información del error
Poscondiciones	El sistema dispone de toda la información recuperada de la consulta previa para posteriores acciones
Requisitos	RSF-04, RSF-23

Tabla 4.4. Caso de Uso 4

Nombre	**Visualizar vídeo generado a partir de *tweets***
Descripción	Permite generar y visualizar un vídeo generado a partir de los *tweets* seleccionados para ser incluídos en el mismo. El proceso de generación de este vídeo sigue el diseño descrito en la Sección 4.4 implementándolo de manera aleatoria para cada ejecución
Precondiciones	El actor debe haber realizado un búsqueda en la aplicación por cualquiera de los dos criterios disponibles o haber desplegado una conversación. El sistema debe haber devuelto ante la consulta una lista con *tweets*
Flujo normal	1. El actor selecciona los *tweets* de la lista que desea que aparezcan representados en el vídeo 2. El actor acciona el botón que genera el vídeo 3. El sistema presenta embebido en la vista actual un icono que indica que el proceso de generación del vídeo se encuentra en curso 4. El sistema presenta embebido en la vista actual un reproductor de vídeo cargado con la animación generada de manera aleatoria a partir de los *tweets*
Flujo alternativo	1. Si la lista pertenece a una conversación, este paso no aplica
Poscondiciones	-
Requisitos	RSF-08, RSF-09, RSF-10, RSF-11, RSF-16, RSF-17, RSF-18, RSF-20, RSF-21, RSF-22, RSF-23, RSF-24

Tabla 4.5. Caso de Uso 5

Se ha tratado de recoger la mayor parte de los requisitos posibles en cada uno de los casos de uso descritos. Como se puede observar, dicha descripción muestra como se han contemplado en estos casos de uso todos los requisitos funcionales especificados con una prioridad media o alta. Por problemas de tiempo respecto a la planificación del proyecto, se ha visto pospuesta la implementación de los requisitos cuya prioridad fuera baja. De este modo, el cumplimiento de estos requisitos se dejará para futuras ampliaciones de la funcionalidad de la aplicación.

4.3. Descripción de componentes

Atendiendo a los resquisitos especificados (Sección 3.2) y posteriormente recogidos en la definición de los casos de uso (Sección 4.2), se procede en este apartado al diseño de una arquitectura de sistema que contemple todas estas características.

4.3.1. Perspectiva global

El sistema a construir debe hacer interactuar a la Plataforma NINOS con la información extraída de Twitter mediante su API. Así, se deberá generar material audiovisual de manera automática a partir de uno o varios mensajes extraídos de la aplicación web del demostrador en base a unos ciertos criterios elegidos por el usuario. El material audiovisual generado será presentando para su visualización en diversos dispositivos.

La generación de vídeo, estará apoyada además en distintas fuentes de datos disponibles online, como son la detección de idioma provista por Google Translate y la creación de códigos QR por parte de Google Chart Tools. El centro neurálgico del sistema será el servidor de aplicaciones, que recogerá toda la información externa y la procesará mediante componentes software que correrán en su interior. Por tanto, el medio de comunicación entre los distintos elementos ajenos al servidor de aplicaciones del sistema y éste será Internet. En la Figura 4.2 se representa la perspectiva global de dichos elementos dentro del sistema.

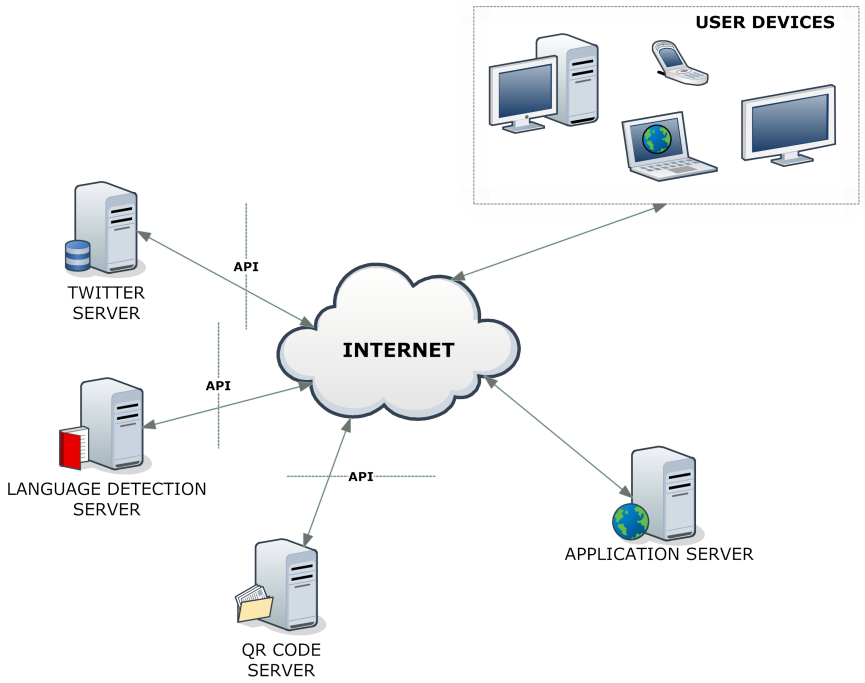


Figura 4.2. Arquitectura global del sistema

4.3.2. Detalle de la arquitectura

El diseño de arquitectura global plasmado en la Figura 4.2 es todavía un diseño a muy alto nivel. El servidor de aplicaciones del sistema va a estar compuesto por una serie de subsistemas que aunarán una cierta funcionalidad del mismo. A continuación, se va a proceder a describir y a mostrar (Figura 4.3) un diseño más en detalle de los componentes lógicos que formarán el servidor de aplicaciones y los modos de interacción de éstos entre ellos y con los elementos externos.

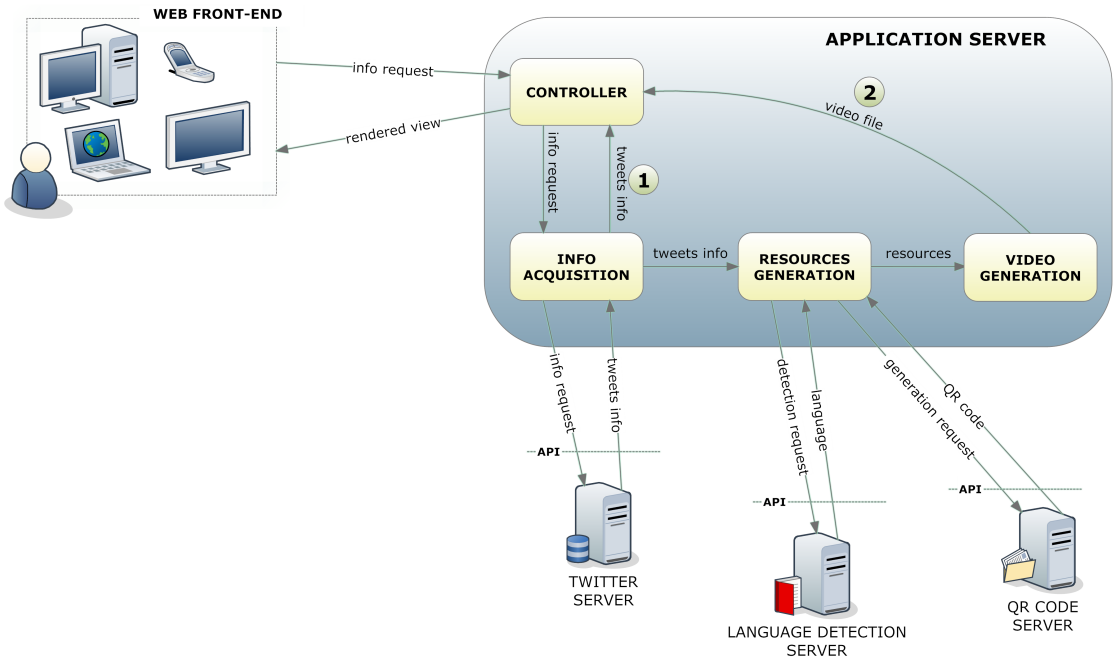


Figura 4.3. Detalle comunicación servidor de aplicaciones

4.3.2.1. Front-end web

El front-end web será el punto de acceso al demostrador del sistema contemplado en este proyecto y por tanto a toda la funcionalidad del mismo. No es más que una página web que muestra las vistas sobre un navegador. Estas vistas permitirán realizar las consultas al sistema y recogerán las respuestas de éste. En la Sección 4.4.3 se puede encontrar más información acerca de las distintas vistas.

El navegador web sobre el que se cargan las vistas del front-end se encargará de enviar las acciones realizadas por el usuario hasta el controller, que hace de puerto de entrada al servidor de aplicaciones. Así mismo, recibirá de éste las vistas renderizadas como respuesta a dichas acciones.

4.3.2.2. Controller

El controller será el subsistema del servidor de aplicaciones encargado de recibir las peticiones de información por parte de la interfaz de usuario, coordinar el funcionamiento interno del servidor para que esta petición sea procesada y presentar la respuesta en el front-end mediante la vista adecuada.

Se podrán considerar dos flujos de proceso diferenciados dependiendo del caso de uso (Sección 4.2) que se esté cubriendo. Para los cuatro primeros casos de usos contemplados en este proyecto, se debe tener en cuenta que la respuesta procede del bloque de adquisición de información, marcado con un número 1 en la Figura 4.3. Sin embargo, en el último caso de uso descrito, la respuesta procede del bloque de generación de vídeo, marcado con un número 2 en la misma imagen.

El controller deberá ser capaz de gestionar la información necesaria para los distintos tipos de consultas recogidos en los casos de uso y trasladarlas al bloque de adquisición de información para que éste las procese. Una vez el controller realiza la petición de información, el resto del flujo de procesado de la información y generación de recursos es totalmente transparente.

Tomando el controller como una caja aislada del resto, éste recibe una petición de información que reenvía. Posteriormente espera a recibir los datos necesarios para generar las vistas de respuesta, o bien en forma de información sobre *tweets* o de vídeo listo para ser embebido, depende del caso de uso que corresponda.

4.3.2.3. Info Acquisition

El bloque de adquisición de información manejará todas la comunicaciones necesarias con Twitter a través de su API. Este bloque será el encargado de recuperar la información solicitada por el usuario en el front-end de la aplicación. Para facilitar el manejo de la gran cantidad de datos obtenidos como respuesta a cada petición sobre el API, se apoyará en un modelo de objetos de datos propio, que además permita su propagación por los demás bloques del servidor de aplicaciones.

4.3.2.4. Resources Generation

El bloque de generación de recursos se ocupará de la creación de todo el material audiovisual o de soporte que sea necesario para la posterior generación del vídeo. Este bloque recibe la información extraída de Twitter, a la cual, realiza un procesado que resulta en una variedad de elementos personalizados y específicos para dicha información.

Algunos de estos elementos son generados internamente en el servidor de aplicaciones, como por ejemplo la voz sintetizada de lectura de los *tweets* o el guión XML (siguiendo el diseño realizado en la Sección 4.4). Sin embargo, otros de los recursos creados por este bloque utilizan servidores externos que los devuelven como respuesta

a una petición a su API, entre los que se encuentran los códigos QR que actúan de representación visual de las URLs contenidas en los *tweets*.

4.3.2.5. Video Generation

El bloque de generación de vídeo reunirá todos los recursos provistos por el bloque anterior y a partir de éstos compondrá un vídeo personalizado para la consulta realizada por el usuario del sistema. Una vez generado este vídeo, se transcodificará a un formato más acorde con los estándares de reproducción de vídeo online y será entregado al controller que se ocupará de embeberlo en una vista que presentará al usuario como respuesta a su petición inicial que disparó todo el proceso.

4.4. Diseño de la demo

El objetivo del demostrador que se va a diseñar es poner de manifiesto el potencial de NINOS a la hora de generar vídeos de manera automática a partir de un guión XML. Si se introduce un cierto carácter aleatorio al proceso de composición del guión, se conseguirá que cada ejecución del algoritmo de generación produzca un resultado distinto. Si bien, el cometido de este proyecto no es centrarse en proporcionar miles de líneas de guión sino más bien hacer ver que dichas posibidades existen y pueden ser explotadas en diferentes grados.

Cada tipo de consulta que se puede realizar en la aplicación, además de la opción de recuperar una conversación, va ligada a un guión que narrará la información obtenida en ella. Todos los guiones que se han escrito tienen una estructura muy sencilla y similar entre ellos, con pequeñas diferencias que veremos en los siguientes apartados.

Realizando un breve resumen, el guión contempla una escena en la cual aparece un personaje ante el que surge una pantalla en la que se van mostrando gráficamente los *tweets* sobre los que se ha aplicado la generación del vídeo mientras éste los narra.

4.4.1. Composición de la escena

Para representar el guión, se ha decidido introducir una serie de elementos que den consistencia a la escena y ayuden a mejorar la calidad de experiencia del usuario final. Todos estos elementos han sido realizados mediante diseño 3D haciendo uso del programa 3ds Max 2010.

El principal elemento dentro de la escena es el actor, que va a representar los mensajes presentes mediante la lectura del contenido de los mismos. Este actor debe estar diseñado de acuerdo a las restricciones que nos impone la utilización de NINOS (ver Sección 2.3.2). En la Figura 4.4 se puede ver un ejemplo de personaje 3D diseñado para funcionar en la aplicación. Este ejemplo ilustra el esqueleto y las texturas que

tendrá el personaje, en una posición neutra.

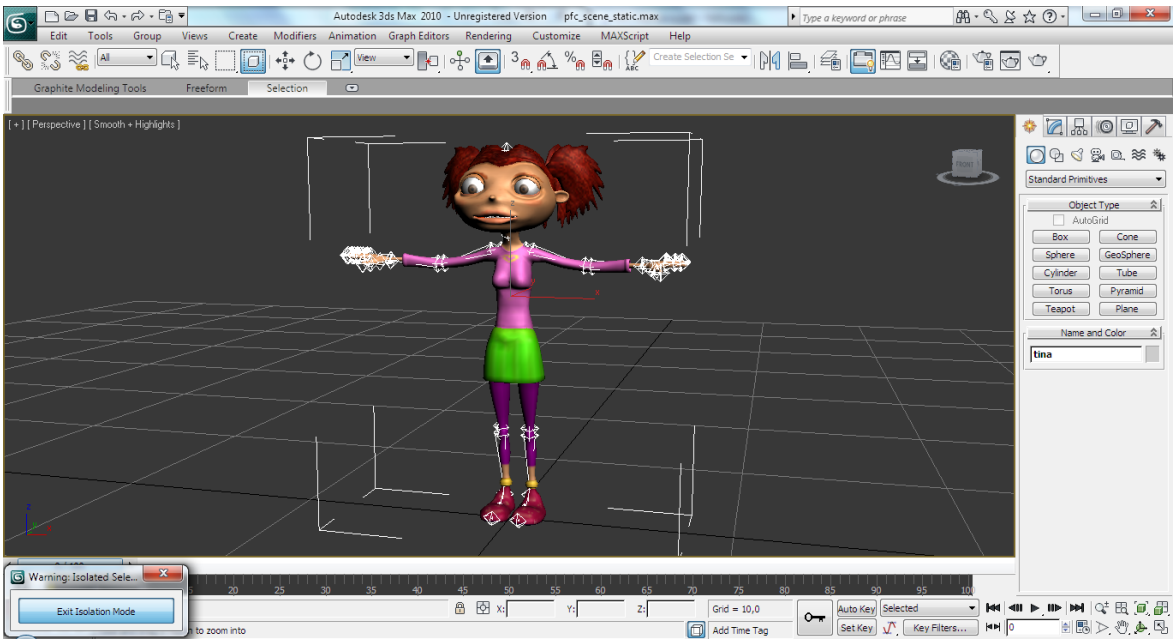


Figura 4.4. Diseño del actor

Se debe tener en cuenta que para que el actor realice movimientos dentro de la escena hay que diseñar también las animaciones que éste podrá representar. De este modo, se ha decidido utilizar una serie de animaciones que dote de mayor realismo al personaje, entre las que se encuentran la posibilidad de caminar, saludar y mantener una posición de *idle*. Las animaciones se diseñan sólo sobre el esqueleto del personaje. Utilizando igual nomenclatura para todos los recursos que represente una misma animación para distintos personajes, se facilita el intercambio de personajes a nivel de guión hasta el punto de sólo tener que modificar el nombre del que se quiera utilizar y directamente se cargaría dicho personaje.

El escenario se forma mediante un suelo y tres paredes en forma de U que sirven de fondo a la representación de los vídeos. Las paredes utilizan una textura similar al ladrillo y el suelo en forma de baldosa, tratando de resultar natural. Se puede ver este escenario en la Figura 4.5

El resto de la escena lo compondrán los paneles de información. Estos paneles serán objetos 3D diseñados para recoger la información sobre lo que se verá en cada vídeo. De este modo, hay un panel, en color rojo, sobre el que irán apareciendo los distintos *tweets* a narrar con toda la información relativa al usuario que lo publica, incluida su foto de perfil y la fecha de publicación. El panel de búsqueda adquiere una gran importancia para situar en contexto cada vídeo, sobretodo, si es visionado a posteriori. Mediante este panel no sólo se sabrá si el vídeo surgió de una búsqueda por ID o por *Keyword* (mediante la selección de la pestaña adecuada) sino, cuál o cuáles fueron los términos concretos objetivo de la búsqueda. Por último, alrededor del panel de *tweets*

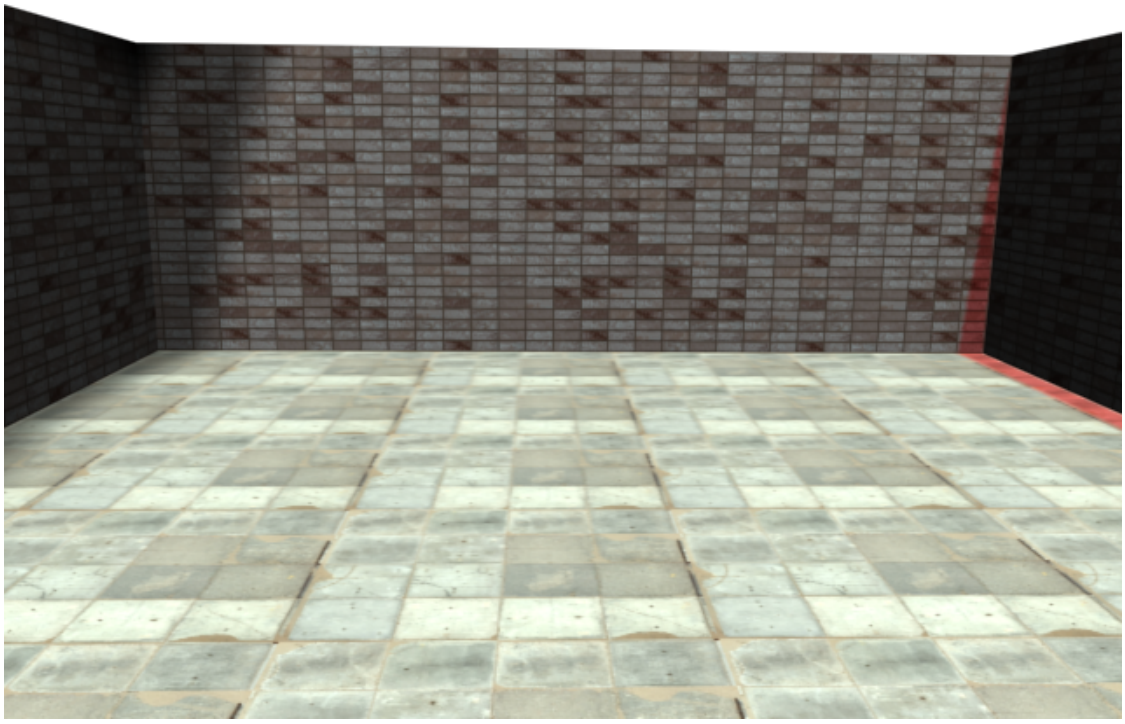


Figura 4.5. Imagen renderizada del escenario

aparecerán tantos paneles como URLs incluya cada mensaje, y en ellos se mostrarán ordenados de izquierda a derecha y de arriba a abajo los códigos QR correspondientes a cada URL. Todos estos objetos están recogidos en la Figura 4.6.

Las luces que iluminan la escena serán dos *spots* situados uno iluminando desde un punto lejano en vertical para dar un relleno lumínico uniforme a todo el espacio dentro del área de visionado y otro enfrente del actor para resaltar la proyección de las sombras de éste y de los objetos presentes sobre el suelo.

Unos de los elementos más importantes de los que conforman la escena son las cámaras. La situación de éstas permitirá, a nivel de generación de guión, modificar y alternar los puntos de vista desde los que se observa la escena. Además, la selección de las cámaras será realizada de manera aleatoria para que cada vídeo resulte único. Para esta demo se han diseñado nueve cámaras como muestra significativa. Estas cámaras están separadas en grupos de tres dependiendo del movimiento que realizan. El primer grupo posee una cámara escorada a la izquierda de la escena, otra centrada y otra más en el lado derecha de la misma, siendo todas estáticas. El siguiente grupo de tres cámaras realizan un movimiento de *travelling* desde una posición alejada hasta alcanzar las posiciones fijas que tienen las del grupo anterior. El último grupo de cámaras realiza un movimiento de *travelling* similar al de las anteriores pero partiendo desde la posicón fija y alejándose progresivamente.

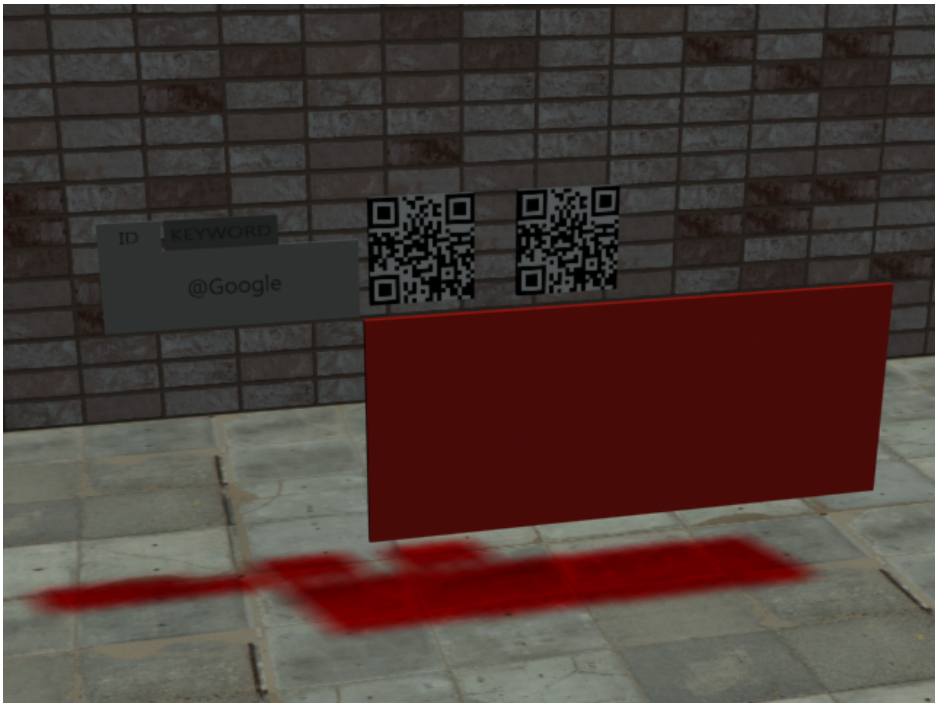


Figura 4.6. Imagen renderizada de los paneles de información

4.4.2. Estructura del guión

El guión se ha decidido estructurar en cuatro bloques bien diferenciados que serán comunes para todos tipos de búsquedas, que sólo se diferenciarán en la elección de algunos elementos presentes en las escenas. A continuación se va a pasar a describir estos bloques de manera que queden claras las interacciones de los elementos con la escena. El diagrama que se muestra en la Figura 4.7 recoge la temporización de los distintos bloques del guión y los elementos que forman parte de éstos.

El primer bloque será la **Introducción**. Este bloque da comienzo a la escena mediante un movimiento de acercamiento de la cámara (seleccionada de manera aleatoria) que ocupa la totalidad del bloque, 4 segundos. Tanto el actor como el panel de *tweets* y el escenario, que estarán presentes durante toda la escena, se encuentrarán estáticos en sus respectivos emplazamientos. Los demás elementos irán entrando o saliendo de la escena a lo largo de la misma. Transcurridos 2 segundos desde el inicio del bloque, el actor dirigirá su vista hacia la cámara y comenzará a saludar.

De inmediato, da comienzo el bloque de **Lectura**, donde se presentarán y serán leídos los *tweets* que correspondan. La duración de este bloque estará definida por la cantidad de *tweets* que se hayan incluído en la generación del vídeo y el tiempo de duración del audio en el que el actor los lee. Por tanto, se puede decir que este bloque estará formado a su vez, en la mayoría de las ocasiones, por pequeños bloques de mensajes de menor tamaño que se corresponderán con cada *tweets*. Estos bloques

de mensajes empiezan con un cambio de cámara dentro del grupo de las que siempre permanecen en una posición estática.

El primer bloque de mensaje tras el bloque de introducción elige la cámara estática cuya posición se corresponda con la de finalización de la cámara de dicho bloque. A partir de éste, el resto de cámaras de los bloques de mensajes son seleccionadas de forma secuencial de izquierda a derecha. Al comenzar el primer bloque de mensaje aparece, en el caso de que no se trate de una conversación, el panel de búsqueda que corresponda, que permanecerá visible hasta finalizar la escena. Dentro de cada uno de estos bloques aparece sobre el panel de *tweet* la información completa del mensaje adecuado, durante el tiempo que el actor tarda en leerlo, desapareciendo al finalizar el bloque. Además, en el caso de que el mensaje contenga alguna URL, aparecerán los paneles con los códigos QR para que estas direcciones web puedan ser capturadas y accedidas desde cualquier dispositivo que disponga de un lector de códigos adecuado.

Tras leerse todos los mensajes, comienza el bloque de **Despedida**. El actor realiza un gesto de despedida y posteriormente retira la mirada de la cámara activa.

Para finalizar, en el bloque de **Cierre** el actor comienza andar hacia delante, abandonando el escenario mientras la cámara aleja su enfoque del mismo.

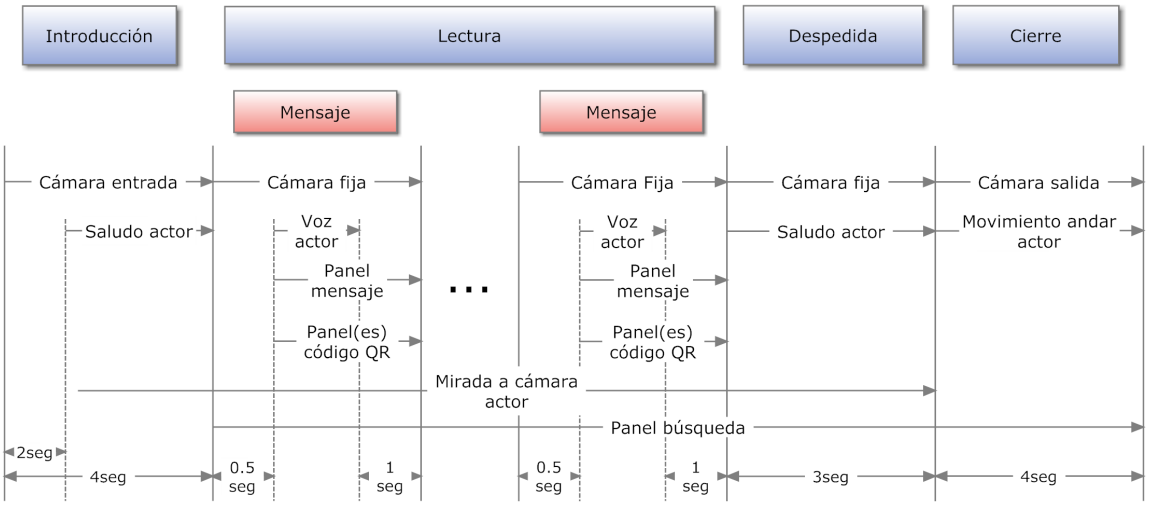


Figura 4.7. Diagrama de la escena

En el diagrama de la Figura 4.7, los tramos de lectura de *tweets* adaptan su duración al tiempo que tarda el actor en leerlos. El resto del guión mantiene una temporización prefijada.

4.4.3. Front-end web

Para realizar la aplicación web que presentará la funcionalidad provista por el grueso de este sistema, de modo más visual y atractivo para un posible usuario, se han utilizado

las tecnologías especificadas en la Sección 3.4.1.

Se ha tomado la decisión de diseño de utilizar un patrón de *look&feel* cercano a la disposición de los elementos presentes en la propia aplicación web de Twitter, simplificando bastante sus funcionalidades.

De este modo, se ha optado por diseñar una página de inicio desde la que se realizan las búsquedas de *tweets* a partir de un **ID** de usuario de Twitter o de algún **Keyword**. Se presenta por defecto la búsqueda por ID, permitiendo acceder a la búsqueda por *Keyword* con sólo seleccionarlo en el enlace proporcionado. El resto de la pantalla incluye la entrada de formulario donde introducir el objetivo de la búsqueda y otra para introducir el número de mensajes que se quieren recuperar. Esta parte es común a ambas opciones de búsqueda aunque se presenta de manera personalizada para cada caso. Para la búsqueda a través del nombre de usuario, también se presentará un *checkbox* para indicar si se quieren devolver también los *retweets* realizados por dicho usuario. La Figura 4.8 recoge un boceto del diseño de esta vista.

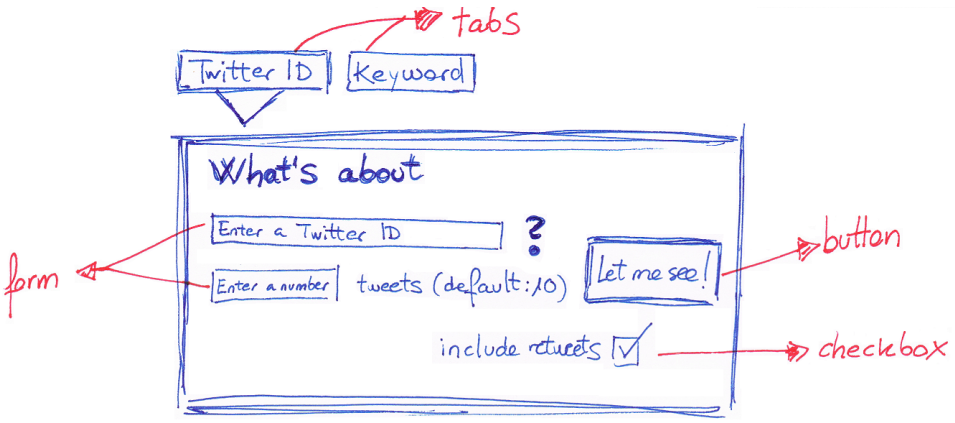


Figura 4.8. Boceto pantalla de inicio

El usuario sólo podrá realizar la búsqueda en el caso de que se haya introducido el término o los términos que se desean buscar, en caso contrario el sistema advertirá de dicha situación mediante un mensaje en pantalla. El sistema también mostrará un aviso cuando no existan resultados para la búsqueda, bien porque no exista el usuario consultado o tenga configurada su cuenta como privada, en el caso de búsqueda por ID o bien porque no haya *tweets* que contengan las palabras buscadas, en el caso de búsqueda por *Keyword*.

El resultado de una búsqueda por ID, en ausencia de los errores comentados anteriormente, es una lista de elementos que contienen la información recuperada. El primer elemento que se presenta en la lista es un resumen del usuario objeto de la consulta. Este resumen incluye la foto del perfil del usuario de Twitter en formato grande, además de su ID, también conocido como *screen name*, su nombre fuera de la aplicación, una

breve *Bio* o descripción de sí mismo además de la información referente al número de *tweets* que lleva publicados y el número de *following* y *followers* que posee en el instante de la consulta. El diseño para esta pantalla, se puede observar en el boceto de la Figura 4.9.

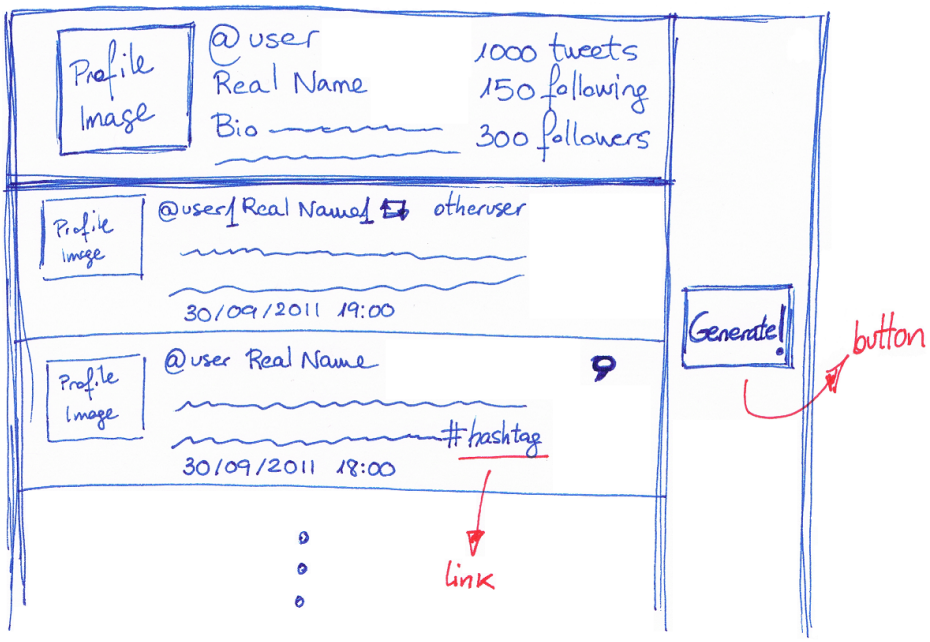


Figura 4.9. Boceto pantalla "Búsqueda por ID"

El resto de elementos estará formado por los mensajes recuperados. Estos *tweets* llevarán un formato idéntico al representado en la aplicación web de Twitter debido a la aceptación que tiene este formato por parte de los usuarios que se encuentran muy familiarizados con el mismo. De hecho, la gran mayoría de aplicaciones diseñadas por terceros respetan siempre este diseño de presentación de los *tweets*. Se puede observar en la Figura 4.10 como el layout realizado por Twitter posee una apariencia similar a la disposición de los recursos que será utilizada por la aplicación desarrollada en este proyecto (Figura 4.9).

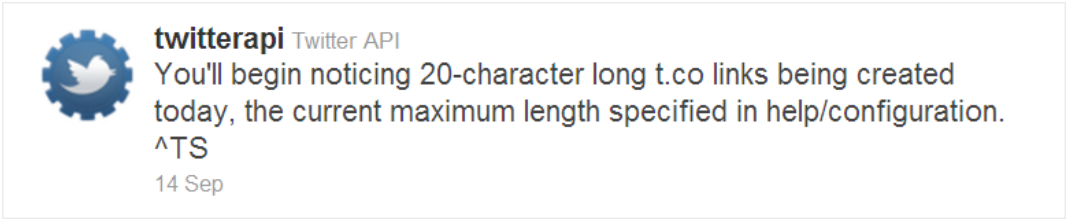


Figura 4.10. Captura del *layout* de la aplicación de Twitter

La siguiente pantalla a analizar es la resultante de la correcta búsqueda por *Keyword*. Ésta es exactamente igual que la comentada anteriormente de la búsqueda por ID, con la única diferencia de no representar el primer elemento con la información del usuario ya que no tiene sentido en este caso.

El diseño realizado para la pantalla donde se mostrarán los *tweets* pertenecientes a una conversación, es calcado al de la pantalla explicada en el parráfo anterior. Sin embargo, en el caso de la conversación, se ha decidido alternar entre izquierda y derecha la justificación de los recursos representados para impulsar la sensación de intercambio de impresiones entre usuarios similar a una charla real en la que se situaría uno enfrente del otro.

Las páginas que recogen la lista de los *tweets* dispondrán de un botón que permitirá lanzar la generación de un vídeo aleatorio basándose en los mensajes que aparecen en ese momento en pantalla.

La última pantalla que se ha diseñado es la correspondiente a la reproducción del vídeo generado a partir de la información de los mensajes elegidos. Esta página no es más que un reproductor de vídeo embebido que permitirá la visualización de éste.

Comentar también que tanto las URLs y los *hashtags* presentes en los mensajes como los nombres de usuario, permitirán ser pulsados. En el caso de las URLs, se abrirá una pestaña nueva en el navegador donde se cargará la web correspondiente. Los *hashtags* devolverán directamente la búsqueda correspondiente al término que representan y los ID de usuario devolverán la búsqueda de los últimos *tweets* publicados por éstos.

Otro punto a tener en cuenta es la señalización mediante los iconos adecuados de los mensajes reenviados (*retweets*) y de las contestaciones a mensajes de otros usuarios (*replies*). Los iconos que serán utilizados para esta tarea se pueden ver en la imagen de la Figura 4.11.

Figura 4.11. Iconos de *retweet* y *reply*, respectivamente

Capítulo 5

Implementación del sistema

5.1. Introducción

Durante el capítulo de implementación, se va a recoger con el mayor detalle posible, todo el trabajo realizado tratando de cumplir los parámetros de diseño fijados en el Capítulo 4. Se ha realizado un importante esfuerzo en la creación de desarrollos propios llevados a cabo íntegramente en el seno de este proyecto, que junto con las herramientas externas disponibles han servido para cubrir las funcionalidades que el sistema debía tener.

Es importante volver a recordar, que en la Sección 3.4 se recoge una breve descripción de las herramientas elegidas para empreder el proceso de creación de la lógica del sistema y de la aplicación web de demostración.

5.2. Descripción de componentes

Realizando una analogía con la estructura del capítulo de diseño, primero se van a explicar las consideraciones globales que se han de conocer para tener una visión general de la implementación del sistema. Posteriormente, se detallará mediante descripciones y diagramas de secuencia el modo concreto en que se han llevado cabo la fabricación de los distintos módulos presentados en la Sección 4.3.2.

5.2.1. Perspectiva global

Existen varios aspectos generales que deben tenerse en cuenta para comprender plenamente todas la tareas que se han llevado a cabo en la etapa de implementación del sistema, para conseguir un funcionamiento acorde a lo especificado.

Debido a las limitaciones de tiempo y de presupuesto inherentes a este proyecto, se ha considerado la utilización de la mayor cantidad posible de funciones existentes

siempre que no tuvieran coste alguno. De este modo se ha tratado de evitar la "reinvención de la rueda" en cada etapa que se quisiera implementar. Mediante la combinación entre la funcionalidad ya existente y la creada íntegramente para este proyecto, unida al principio de desarrollo software conocido como DRY (*Don't repeat yourself*) que apuesta plenamente por la reutilización de código, se busca obtener una implementación ajustada a los requerimientos pero equilibrada en todos los ámbitos de la misma (rendimiento, complejidad, tiempo de desarrollo...).

Un ejemplo de reutilización de software existente que aporta consistencia al sistema además de dotarle de una gran flexibilidad, es la elección de una librería de consulta al API de Twitter escrita en PHP. Esta librería [49] está compartida por su autor como código libre, por lo que puede ser utilizada en este desarrollo. El motivo de la utilización de ésta y no alguna de las otras existentes, es la facilidad de uso que permite además de ser una de las más completas. Principalmente, con esta librería se gestiona de una forma muy eficiente tanto las peticiones HTTP al API como las respuestas de éste. Aún siendo una librería muy completa, se han tenido que realizar algunas modificaciones para añadir funcionalidades no recogidas y que eran parte importante del núcleo de este proyecto. Estas funcionalidades que no incorpora son las provistas por el API de búsqueda que, como se explica en el Anexo C, trabaja de manera autónoma al API principal de consultas de Twitter.

Esta librería escrita en PHP, posee integrada la interacción con el protocolo de seguridad utilizado por Twitter, llamado OAuth [50]. Para que sea completamente funcional tan sólo es necesario descargarse la librería de OAuth para PHP [51]. Este protocolo está comenzando a ser ampliamente utilizado por aplicaciones web que disponen de API público para desarrolladores ya que garantiza a los usuarios que sus datos de acceso nunca son conocidos por *3rd Parties* cuando éstas gestionan el acceso a la información en Twitter de los usuarios. El modo de funcionamiento de este protocolo está extensamente explicado en [50]. En la versión presentada en este documento no se aprovecha la funcionalidad de acceso condicional para los usuarios que poseen cuenta en Twitter, aunque se dispone de ello para futuras ampliaciones de funcionalidad de la aplicación. Sin embargo, sí se hace uso de este protocolo para firmar las peticiones realizadas al API con la *key* y el *secret* provistas por Twitter al registrar la aplicación de este proyecto. De esta manera, Twitter garantiza a las aplicaciones registradas una alta cantidad de peticiones al API. Se puede encontrar más información sobre los números concretos de peticiones permitidas en [52].

Uno de los mayores y más importantes esfuerzos realizados durante todo el proceso de desarrollo de funcionalidades ha sido la creación de la base estructural que forma los guiones XML utilizados por NINOS para generar las escenas en vídeo (se puede ver con más detalle en el Anexo B). Se ha implementado un entramado de objetos y atributos, utilizando PHP, que se encargan de mapear las distintas etiquetas con las que NINOS trabaja. Gestionando los recursos que deben aparecer en una escena mediante objetos PHP se facilita sobremanera la creación de estos recursos y la interacción entre ellos. Para volcar la información en el formato XML que requiere NINOS, se ha provisto a cada objeto de una función que al ser invocada recupera toda la información del objeto

de manera recursiva entre los objetos que pudieran componer a éste imprimiéndola siguiendo el formato XML. La gestión de los guiones se vuelve muy intuitiva ya que primero se crean todos los objetos definiendo las relaciones que guardan entre ellos, para finalmente generar todo el guión con una llamada al método de volcado del objeto de mayor jerarquía, en este caso *Timeline*.

De manera similar a la estructura creada para manejar los objetos de los guiones XML, se han implementado las clases *Tweet* y *User* para recoger y gestionar de un modo eficiente la información que provee Twitter en cada consulta sobre los distintos recursos que componen un mensaje y las diferentes características que posee un usuario. Debido a la diversidad de consultas que se pueden realizar al API y la falta de homogenidad de los campos devueltos en las respuestas, se ha decidido definir los atributos de estas clases de la manera más general posible para que pudieran abarcar todo tipo de consultas y fueran fácilmente adaptables a posibles actualizaciones por parte de Twitter. Obviamente, esto provoca que haya una gran cantidad de datos que actualmente son ignorados en la versión que se recoge en esta memoria ya que no tienen aplicabilidad en la funcionalidad buscada, un ejemplo serían las coordenadas que geolocalizan un *tweet*. Sin embargo, se ha preferido hacer de este modo para establecer una estructura con entidad, que pueda ser reutilizada aún con la inclusión de mejoras sin tener que rehacer el trabajo.

La configuración del servidor Apache, encargado de correr la aplicación, realizada para este proyecto no tiene ninguna modificación especial respecto a la configuración por defecto que se instala con Xampp, quitando la definición habitual de la ruta del disco duro del servidor donde se quiere desplegar el directorio activo del mismo. Sin embargo, se tiene que considerar como un cambio importante el hecho de que el servidor Apache no haya podido ser ejecutado como un servicio debido a la incompatibilidad de los servicios a la hora de realizar llamadas a aplicaciones externas instaladas en local. Esto resultaba un grave problema, ya que era imposible utilizar el software requerido para el correcto funcionamiento del sistema (2nd Speech Center, Ninosaudio, Ninosrender...)

5.2.2. Detalle de la arquitectura

A lo largo de esta sección se van a describir las distintas funciones que están presentes en los subsistemas que forman el servidor de aplicaciones y la manera en que estas funciones implementan toda la lógica de operación diseñada para realizar por cada uno de los subsistemas en la Sección 4.3.2.

Las funciones aparecen nombradas con una breve reseña del proceso que realizan, junto con un identificador, que se utilizará en los diagramas para reconocer unívocamente cada función.

5.2.2.1. Front-end web

Para implementar la interfaz de usuario vista en el navegador web desde el que se acceda, se ha utilizado HTML junto con un *framework* de código abierto para los CSS, llamado 960Grid [53]. La potencia de este *framework* radica en la facilidad con que se sitúan elementos en las vistas. Esto lo consigue mediante la división de cada vista en 12 columnas, las cuales se pueden agrupar o elegir a placer. Se puede dividir también de manera recursiva cada bloque de columnas agrupado. Sin embargo, este sistema tiene una pequeña limitación que repercute en la características de compatibilidad buscadas para el demostrador. Este problema surge debido a que el tamaño total del conjunto de las 12 columnas está prefijado a 960px, lo cual haría que las vistas pudieran no verse de manera óptima en según que dispositivos dependiende de su resolución de pantalla.

Se ha conseguido solventar este inconveniente mediante una versión, conocida como 960 Fluid [54], que realiza unas pequeñas modificaciones en el *framework* de manera que éste adapte la disposición de los elementos de la vista sin que se desmorone la apariencia diseñada. De este modo, cada elemento se redimensiona en proporción al tamaño disponible resultando vistas similares independientes del tamaño. Además, este redimensionamiento se hace en tiempo real, por lo que se puede cambiar el tamaño de la vista mientras los elementos de la misma se adaptan al mismo tiempo. En la Figura 5.1 se puede apreciar como la disposición de los componentes de la vista es idéntica tanto en el navegador de un PC con una resolución de 1366x768 (imagen superior), como en el navegador de un dispositivo Android con una resolución de 800x480 (imagen inferior).

5.2.2.2. Controller

El controlador realiza la validación de los datos introducidos en los campos de formulario tanto para la búsqueda como para el número de *tweets* que se quieran recuperar. Así no se podrá realizar una búsqueda dejando en blanco el campo del objetivo de la misma. En el campo de la cantidad de mensajes si se introducen valores mayores a 10 o menores que 0, o si se introducen caracteres que no sean números, se devolverá el valor de la búsqueda por defecto, que es 10. De esta manera se atajan los posibles errores de forma inicial y no se propagan hacia la petición al API de Twitter, que conllevaría un tiempo extra hasta que éste devolviera la respuesta con el error.

Se ha integrado la posibilidad de pulsar los enlaces que forman las menciones y los *hashtags* que aparecen en el texto de los mensajes recuperados para las búsquedas. El resultado de pulsar una mención es la vista de "Búsqueda por ID" para el usuario mencionado y en el caso de los *hashtags* se verá la vista de "Búsqueda por Keyword" del término concreto al pulsar sobre él. La manera en que se ha implementado este módulo, permite que tanto los casos anteriores como las búsquedas habituales mediante el formulario de la pestaña adecuada obtenga las mismas vistas como resultado. Esto se ha conseguido teniendo en cuenta el modo en que se realiza la petición (GET o POST) lo cual permite al controlador gestionar cada evento.

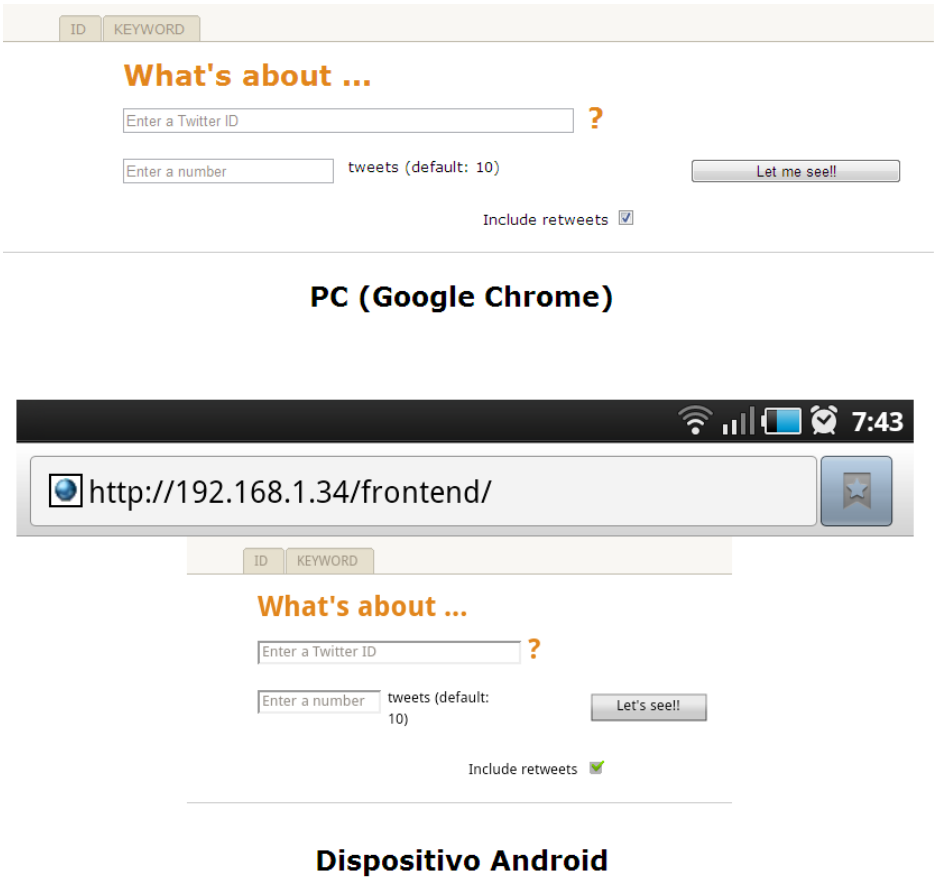


Figura 5.1. Comparación *layout* de una vista con dos resoluciones distintas

Otra de las acciones que maneja el controlador es la carga de la configuración requerida para el correcto funcionamiento del sistema, que se realiza en el momento en que se accede a la aplicación por primera vez y se mantienen disponibles para el resto de la ejecución. Esta configuración recoge las claves provistas para la aplicación por parte de Twitter así como la clave que Google facilita para poder utilizar su API de detección de idiomas.

La inclusión del reproductor de vídeo en la vista de la aplicación una vez se haya solicitado la generación del mismo, también es tarea del controlador. Para embeber el vídeo en la interfaz web, se ha utilizado un reproductor *flash* llamado Flowplayer [55] que dispone de una versión gratuita. Este reproductor permite además, la personalización de la apariencia entre una gran variedad de características. Se puede ver un ejemplo en la Figura 5.2

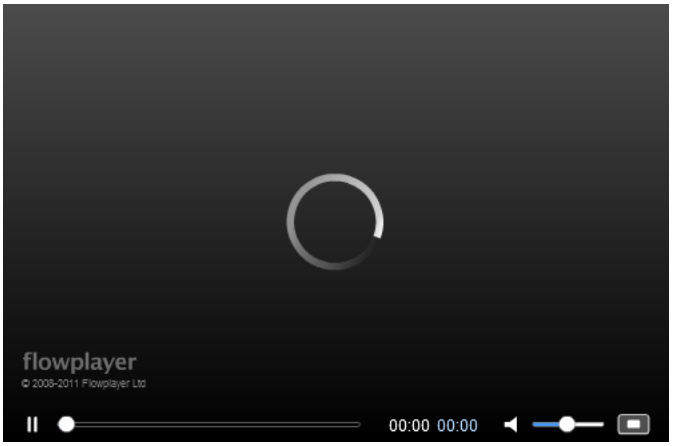


Figura 5.2. Reproductor de vídeo

5.2.2.3. Info Acquisition

El módulo de adquisición de información es el encargado de realizar las peticiones de información a Twitter solicitadas por el usuario desde el front-end web.

Dependiendo del tipo de búsqueda que quiera realizar el usuario, recogidos en el Caso de uso 1 (Tabla 4.1) y en el Caso de uso 2 (Tabla 4.2), se realizará un petición concreta al API de Twitter. Se ha tratado de realizar siempre el menor número de llamadas al API para que el tiempo de respuesta de la aplicación sea el menor posible y se evite que el usuario permanezca a la espera de la respuesta por parte del servidor durante mucho tiempo, lo cual bajaría la calidad de experiencia percibida por éste. Además se ha buscado hacer uso de las llamadas del API que no requerían autenticar al usuario de la aplicación como usuario de Twitter. Se ha requerido que todas las respuestas del API sean devueltas en formato JSON debido a la facilidad y rapidez de manejo con que PHP maneja este formato.

Para las búsquedas por ID se realiza una petición a Twitter mediante la siguiente URL:

```
https://api.twitter.com/1/statuses/user_timeline.json?
   &screen_name=twitterapi&count=2&include_rts=true
```

donde sólo se contemplan los parámetros de los que hace uso este sistema. Estos parámetros son, por orden, el ID de usuario, el número de mensajes que se quieren recuperar y un flag que indica si los reenvíos deben devolverse o no.

Si la búsqueda, por contra, quiere tener por objetivo un término o una serie de ellos, la consulta al API es:

```
http://search.twitter.com/search.json?q=%23Google&rpp=6
```

donde los parámetros corresponden con la búsqueda y el número de *tweets* que serán recogidos.

En el caso de las conversaciones el funcionamiento es algo distinto. Para poder recuperar una conversación primero se ha de haber realizado alguna de las dos búsquedas posibles. Una vez se dispone de una lista de mensajes, se acciona uno que aparezca marcado con el icono que indica que es una respuesta a un *tweet* anterior. La manera que utiliza Twitter para mantener la relación de un mensaje con el mensaje al que respondió, es incluyendo el identificador del mensaje inicial en un campo entre la información que devuelve sobre el mensaje actual. De modo que si se quiere conocer el contenido concreto (usuario, texto, fecha de publicación...) del mensaje original, es necesario consultar a Twitter por esa información. Si a su vez ese mensaje hubiera surgido en respuesta a otro anterior, habría que volver a realizar una nueva consulta a Twitter, y así hasta alcanzar el mensaje que inició la conversación. A la hora de implementar esta funcionalidad, se ha utilizado una función recursiva que recupera toda la información en cascada haciendo peticiones al API de Twitter mediante la siguiente URL:

```
http://api.twitter.com/1/statuses/show/80370961106475251.json
```

donde el número que aparece es el identificador del mensaje buscado.

Una vez se recibe la respuesta en formato JSON del API de Twitter, el servidor de aplicaciones recava toda la información sobre los mensajes y los usuarios de éstos y la vuelca en objetos para su manejo entre los distintos módulos que componen el servidor. En este instante comienza una etapa de procesado de la información, de donde se sacarán dos versiones del texto que compone cada *tweet*.

Una de estas versiones busca convertir el texto plano en texto enriquecido para su presentación en las vistas de la aplicación web, de modo que muestre como enlaces las URLs, las menciones a otros usuarios y los *hashtags*. Las menciones lanzarán una búsqueda sobre los últimos mensajes escritos por ese usuario devolviendo la vista correspondiente, los *hashtags* ejecutarán una consulta sobre el propio término recibiendo como respuesta la vista con los resultados y las URLs serán abiertas en una nueva pestaña o ventana (dependiendo el método de apertura por defecto que tenga configurado cada navegador). La otra versión del texto está destinada a ser interpretada por el motor de síntesis de voz por lo que se busca introducir ciertas pausas, antes o después de los elementos nombrados, para que la lectura del texto resulte lo más realista posible. Estas pausas se introducen entre el texto mediante un mecanismo de SAPI5 conocido como "XML Control Tags" que utiliza etiquetas XML, como por ejemplo, `<silence msec="500/>` (al alcanzar esta etiqueta, se pausa la lectura 500 milisegundos tras lo cual, se reanuda).

Para realizar la sustitución de los elementos comentados por lo que corresponde según la versión, se utilizan expresiones regulares. Una expresión regular es un patrón de formación de texto, una regla que describe un conjunto de posibles cadenas de

caracteres que pueden cumplirla. De este modo, se utiliza ampliamente para validar entradas de datos o, como será en este caso, para buscar coincidencias dentro de un conjunto más grande de caracteres.

Se aprovecha la funcionalidad provista por las expresiones regulares debido a la presencia de interesantes funciones dentro de las librería de PHP. Para la tarea que se quiere realizar, la función que mejor se adapta a las necesidades es la conocida como `preg_replace_callback`, la cual busca coincidencias de la expresión en un texto, realizando una llamada a una función externa en el momento que encuentra alguna coincidencia. Esto permite colocar la lógica de la sustitución de los elementos buscados dentro de esta función externa.

Las expresiones regulares se deben diseñar de una manera cuidadosa buscando recoger la mayoría de los casos que se quieren contemplar, tratando de que ninguno de los supuestos que debieran arrojar coincidencias, no sea detectado. El caso del formato que deben seguir los nombres de usuario registrados en Twitter es sencillo, ya que puede tratarse de un nombre compuesto tanto por letras como por números y con una extensión comprendida entre uno y veinte caracteres, siempre precedidos por @. La expresión regular elegida para buscar menciones dentro de un texto es:

```
@(\w{1,20})
```

Para el caso de los *hashtag* se ha comprobado que pueden estar formados tanto por letras como por números y el símbolo barra baja (_), pero siempre debe haber al menos una letra en el conjunto. Además deben comenzar por el símbolo # y no tienen límite máximo de caracteres por lo cual podría darse el caso de uno con 140 caracteres (máximo permitido por Twitter para los mensajes). La expresión regular utilizada para buscar coincidencias y detectar *hashtags* es:

```
(#(\w*[a-zA-Z_]+\w*))
```

Por último, el caso de las URLs es el más delicado de todos, debido a la enorme cantidad de combinaciones y tamaños que se pueden dar debido a las reglas complejas que especifican la formación de las mismas. Se han tratado de recoger las máximas URLs posibles de las que suelen utilizarse habitualmente. Lo que resulta imprescindible para que sean detectadas las URLs es que esté definido su protoco de conexión (FTP, HTTP o HTTPS), de cualquier otra manera, no habrá coincidencia.

```
(?:ftp|https?):(?:(?:[\w.-+%!$&'()*+,;=]+:)*
[\w.-+%!$&'()*+,;=]+)?.[.]+(?:[a-z0-9-.%]+)
(?::[0-9]+)?(?:[|?][\w#!:?+=&%!$' *,;()[]-]*)?\b?/xi
```

5.2.2.4. Resources Generation

El módulo de generación de recursos del servidor de aplicaciones tiene algunos de los procesos más grandes en términos de memoria requerida y de tiempo dedicado a la ejecución de los mismos. Esto es debido a que la mayoría de los ficheros con los que trabaja este módulo son bastante pesados (imágenes, audio, diseños 3D) y por tanto, su manejo repercute de manera importante en el consumo de memoria RAM ya que en muchos casos deben ser cargados por completo en ella y en capacidad de proceso que queda libre para el resto de servicios.

Uno de los recursos generados en este módulo son las imágenes que van formando las texturas de los paneles de *tweets* que se ven en la escena. Estas imágenes son generadas para cada uno de los mensajes que leerá el personaje durante la escena. La función que se ha desarrollado para hacerse cargo de este objetivo se llama **textImgGen**. Esta función genera a partir de la información recibida del módulo de adquisición sobre cada mensaje una imagen local, alojada en el servidor con una apariencia similar al *layout* visto en la aplicación de Twitter (Figura 4.10).

Esta imagen se genera con un mismo tamaño especificado siempre de modo que tenga el suficiente tamaño para ser legible cuando se incluya en el vídeo. A partir de esas medidas fijas, se comienza a situar de manera dinámica los elementos que contendrá (foto de perfil del usuario que publica el mensaje, nombre del usuario, texto del mensaje, fecha de publicación y los iconos de *retweet* y *reply* siempre que proceda) mediante la librería para manejar imágenes que proporciona PHP. En la Figura 5.3 se puede observar un ejemplo de una imagen generada mediante esta función.

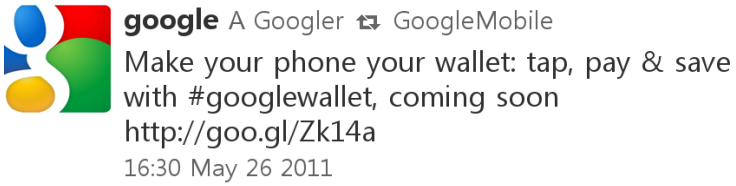


Figura 5.3. Textura del panel de *tweet*

Esta función ha requerido la implementación de un algoritmo que realizara la tarea de colocación de estos elementos de manera dinámica y siguiendo siempre la misma estructura, independientemente de la información que sea. De este modo, el texto del mensaje es colocado sin salirse de los límites de la imagen, incluso llegando al nivel de realizar una separación de los caracteres de una palabra para que ocupe distintas líneas en el caso de que fuera demasiado grande para ocupar una sola. Esto que parece algo trivial a la hora de presentar texto en una página web, por ejemplo, tratándose de una imagen que se está generando en caliente de manera automática sin intervención ninguna de herramientas de diseño, adquiere la necesidad de una lógica que vigile la correcta disposición de los elementos de una manera cuidadosa. También se ha dotado a la función de la posibilidad de orientar la disposición de los elementos con la imagen

de perfil del usuario a la derecha para que la simulación de la conversación sea más intuitiva al ir alternando los mensajes orientación derecha y orientación izquierda para los distintos usuarios que participan en ella.

Como apoyo a esta función y a alguna más que veremos más adelante, ha sido necesario realizar el desarrollo de un conversor de imágenes, que partiendo de las imágenes en los formatos que maneja de origen PHP (GIF, JPG, PNG, entre otros), devuelva una imagen en formato TGA, el único formato de imágenes para las texturas que soporta NINOS (ver Sección 2.3.2). Esta función de conversión de formatos ha sido desarrollada a nivel de la especificación de la estructura de los archivos TGA [56] realizando la conversión entre escalas de colores de la composición de cada píxel de la imagen.

Una vez creada la imagen, la manera de llevar a cabo la personalización del panel es accediendo al fichero que contiene el modelo 3D del mismo y modificando por completo la textura que tiene aplicada. Para realizar esta tarea, también se ha necesitado construir un algoritmo que permita trabajar con el formato binario propio de los ficheros FBX, navegar a través de la información codificada que contiene y modificar tan sólo la parte del modelo 3D que define la información de la textura que tiene aplicada para intercambiarlo por la nueva información. Se realiza una copia de cada instancia al realizar cada intercambio de textura para mantener disponible el archivo de plantilla que se utilizará para cada mensaje. El resultado, en comparación con la Figura 4.6 donde se podía ver el panel aún sin información, es un panel totalmente legible con la información provista en tiempo real por Twitter y empotrado en un vídeo (ver Figura 5.4), junto con otros tantos paneles distintos para cada mensaje.

Figura 5.4. Panel de *tweet* en el vídeo

Las imágenes de perfil en Twitter se obtienen gracias a la URL que proporciona el API dentro de la información de cada mensaje. Twitter dispone de varios tamaños de las imágenes de modo que pueda ser elegido el tamaño más adecuado dependiendo de las necesidades de calidad que se busque y el ancho de banda del que se disponga para descargarlas en cada situación. En la Figura 5.5, se aprecia la diferencia entre los distintos tamaños que existen. En el sistema que se está implementando en este proyecto, se ha contemplado siempre la descarga de la imagen de tamaño 128x128 por motivos de calidad a la hora de ser utilizada, principalmente, en las texturas de los paneles de *tweet*. Para poder utilizar una imagen de perfil en las funciones que la requieran, es descargada mediante un pequeño código que se encarga de, a partir de la URL donde se encuentran (apunta siempre al tamaño normal de las imágenes), modificar la terminación del nombre del fichero para apuntar a la imagen de tamaño *reasonably_small* y guardarla en el servidor para que pueda ser utilizada. De esta manera siempre se trabaja con la imagen de mejor calidad independientemente del tamaño que se necesite presentar, que en caso de que fuera menor sería reducida en ese momento concreto.

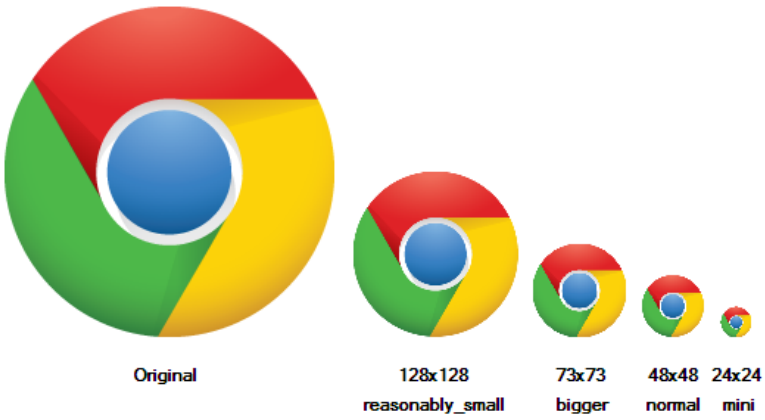


Figura 5.5. Tamaños de imagen de perfil

Otro de los recursos que es generado dentro de este módulo es la imagen que actúa como textura en el panel de búsqueda que aparece en el vídeo. Se ha realizado el desarrollo de una función, llamada **searchImgGen**, que posee un algoritmo similar al utilizado para la imagen de la textura del panel de *tweet*, pero en este caso se realiza una adaptación del tamaño de la fuente y un centrado del texto tanto de izquierda a derecha como de arriba a abajo. La función recibe los parámetros de la búsqueda y el tipo de búsqueda y ejecuta el algoritmo que genera la imagen en formato PNG. Posteriormente se hace uso de la función de conversión hacia TGA y se integra como textura dentro del archivo 3D del panel. Se puede ver un ejemplo del resultado de la integración ya realizada de la textura en la Figura 5.4 en la esquina superior izquierda para el caso de una búsqueda por *keyword*. Para este caso, ha sido necesario realizar una mejora a la función de intercambio de texturas para contemplar el caso en que existan varias texturas dentro de un objeto 3D, que no era el caso del panel de *tweet* que

sólo tenía una, ya que en este caso el panel de búsqueda tiene tres texturas diferentes (etiqueta ID, etiqueda *keyword* y términos de búsqueda) y hay que intercambiar sólo una de ellas.

Como se vio en el capítulo anterior, se hace uso de códigos QR como representación visual de las URLs que pueden aparecer en los vídeos, de manera que los usuarios puedan interactuar con estos códigos y acceder las URLs a través de un dispositivo de lectura de códigos que además se encuentre conectado a la Red. La forma en que los códigos QR son introducidos en los vídeos, es similar a la de los paneles que hemos visto anteriormente. Se utiliza una imagen, en este caso un código QR generado online a través del API Chart de Google (ver Sección 3.4.2.1), que es asignado como textura a un panel concreto durante la lectura en la escena del *tweet* donde está recogida la URL que representa. La Figura 5.6 recoge un ejemplo de un código QR.

Figura 5.6. Código QR

El audio de lectura de los *tweets* es otro de los recursos multimedia generados en este módulo. El software utilizado para sintetizar voz a partir de texto, es conocido como 2nd Speech Center. Este servicio se ejecuta en local en servidor de aplicaciones. Este programa requiere tener instalado, también en local, los paquetes de voces que quieran ser utilizados para la síntesis. Para realizar la prueba de concepto de este proyecto, se ha decidido hacer uso de dos voces femeninas, una dedicada a sintetizar texto en inglés con acento del Reino Unido y otra destinada a trabajar con texto en español, ambas compatibles con SAPI5. El funcionamiento de este programa es muy sencillo aunque no así serán los procesos internos que realice para completar la conversión texto-voz. Se le pasan como entradas el texto a convertir, ya procesado para resultar más realista mediante la introducción de pausas, y la voz que se pretenda utilizar. El resultado es un archivo WAV que recoge la lectura de un mensaje de Twitter.

Para realizar el sintetizado de voz de la mejor manera posible, se debe asignar la voz más adecuada para cada texto que se quiera convertir. De este modo, se estableció la necesidad de detectar el idioma de cada *tweet* para elegir la voz adecuada. Debido al limitado número de voces disponibles, se ha considerado establecer la voz en inglés como voz por defecto para cualquier idioma excepto el español, para el cual se utilizará la otra voz. La detección de idioma de un texto realizada por el API Translate de Google, no es 100 % fiable, de hecho, presenta siempre los resultados mediante un indicador de

la certeza con que se realiza la predicción. Cuantas más palabras tenga el texto, más fácil le resultará al API arrojar un resultado fidedigno.

El último de los recursos que se genera en este módulo, una vez han sido generados todos los demás, es el guión XML. Para la generación de éste, se ha realizado un desarrollo que se encarga de componer la escena siguiendo el esquema fijado en la Sección 4.4.2, que se puede observar gráficamente junto a la temporización en la Figura 4.7. Va creando los distintos objetos, según la jerarquía diseñada en PHP para modelar el funcionamiento de NINOS, que estarán presentes en la escena y añadiendo los diversos recursos que se han generado, desplegando las dependencias temporales entre todos, de modo que la aparición de cada uno de ellos en la escena sea justo el momento planificado. Se introducen ciertas aleatorizaciones a nivel lógico que junto con los recursos personalizados para cada composición del guión, conseguirán que cada guión se único. El resultado final es un fichero XML con la estructura jerárquica requerida por los procesos de composición y renderizado de NINOS.

5.2.2.5. Video Generation

La generación del vídeo parte de la generación de una sola pista de audio que comprenda los distintos instantes de audio definidos para la narración de los *tweets* a partir de los especificado en el guión XML generado. Este archivo de audio tiene un formato .WAV y es generado a partir de una llamada a una aplicación externa que se ocupa de recuperar los eventos de audio descritos en el guión y componerlos. Esta aplicación pertenece a la plataforma software que compone NINOS. Para obtener el resultado esperado, se debe hacer una llamada del tipo:

```
ninosaudio -xml SCREENPLAY.XML -d AUDIOOUTPUT.WAV -lipsync -w
```

donde *-lipsync* fuerza la generación de la sincronización labial de los actores que aparecerán en la escena y *-w* permite que se sobrescriba el fichero de destino en caso de existir con anterioridad.

Una vez se dispone del audio resultante de la llamada a ninosaudio, se realiza una ejecución de ninosrender utilizando como entradas el mismo guión XML, la pista de audio global y los codec a utilizar, de la siguiente manera:

```
ninosrenderer -xml SCREENPLAY.XML -root ROOTPATH -wav AUDIOOUTPUT.WAV
                    -d VIDEOFILE.AVI -vc VIDEOCODEC
```

donde *-root* indica el directorio raíz donde se encuentran los recursos que se han definido en el guión que va a utilizar NINOS para generar el vídeo.

Se ha decidido hacer uso de los códecs de 3ivx para codificar en formato MPEG-4/H.264 para la imagen y AAC para el audio. Esta decisión adoptada se ha debido al hecho de que un vídeo de 3 minutos generado por NINOS en crudo, sin recibir una

transcodificación previa a su almacenamiento en disco, puede ocupar varios gigabytes en disco duro. Esto resulta un importante problema debido a la dificultad de manejar archivos de este tamaño para el equipo utilizado en el entorno de demostración y a la alta capacidad que se antoja necesaria para poder realizar algunas consultas, también inabordable por parte del servidor utilizado en este entorno. El hecho de elegir los códecs comentados atiende al buen ratio calidad-compresión que poseen, lo cual permitirá una posible posterior transcodificación sin que se aprecie un degradado grave en la calidad del vídeo.

5.2.3. Diagramas de secuencia

Tras realizar una explicación detallada de todos los módulos que forman el servidor de aplicaciones, y la forma en que éstos han sido implementados, se van a representar los diagramas de secuencia de las llamadas entre módulos a las distintas funciones implementadas para cada uno de los casos de uso recogidos en la Sección 4.2. El diagrama de secuencia del caso de uso 4 es exactamente igual al del caso de uso 1, incluso presentan la misma vista al usuario.

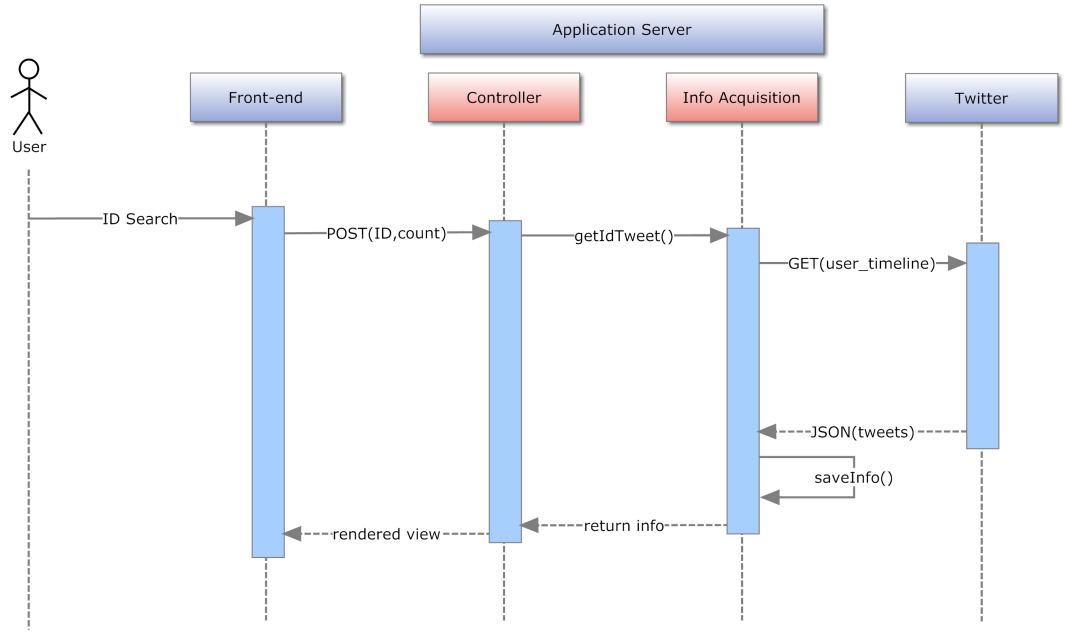


Figura 5.7. Caso de uso 1. Buscar *tweets* por ID

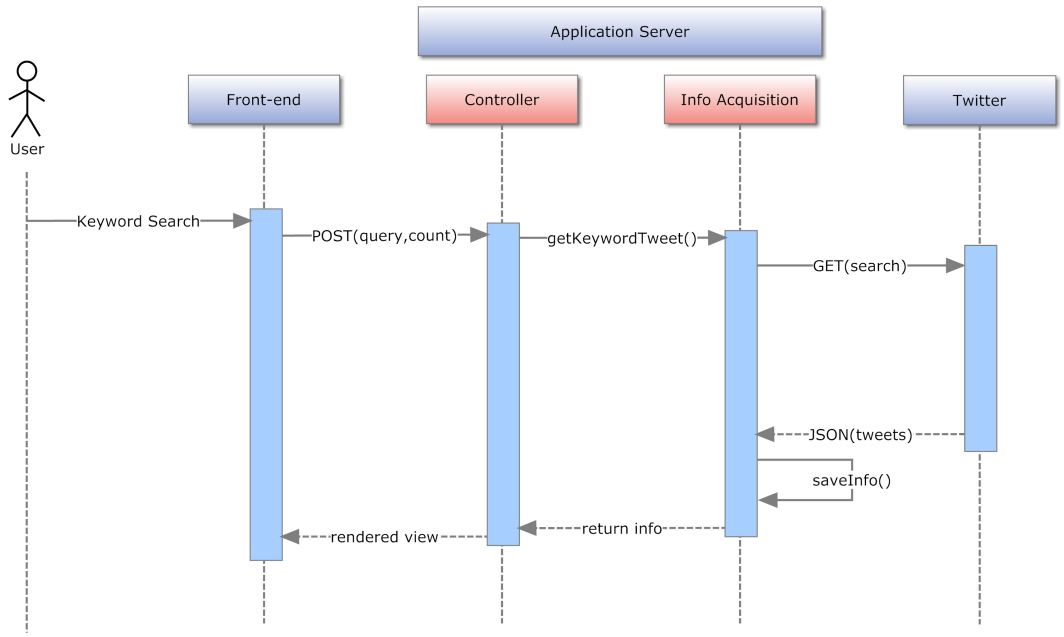


Figura 5.8. Caso de uso 2. Buscar *tweets* por Keyword

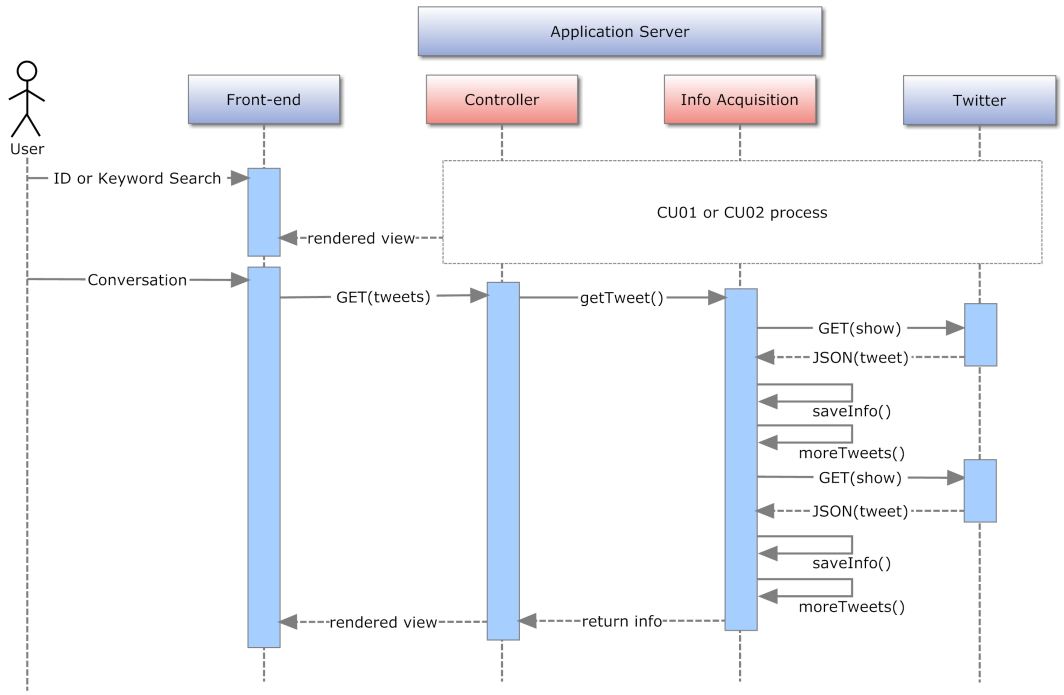


Figura 5.9. Caso de uso 3. Desplegar conversación entre usuarios

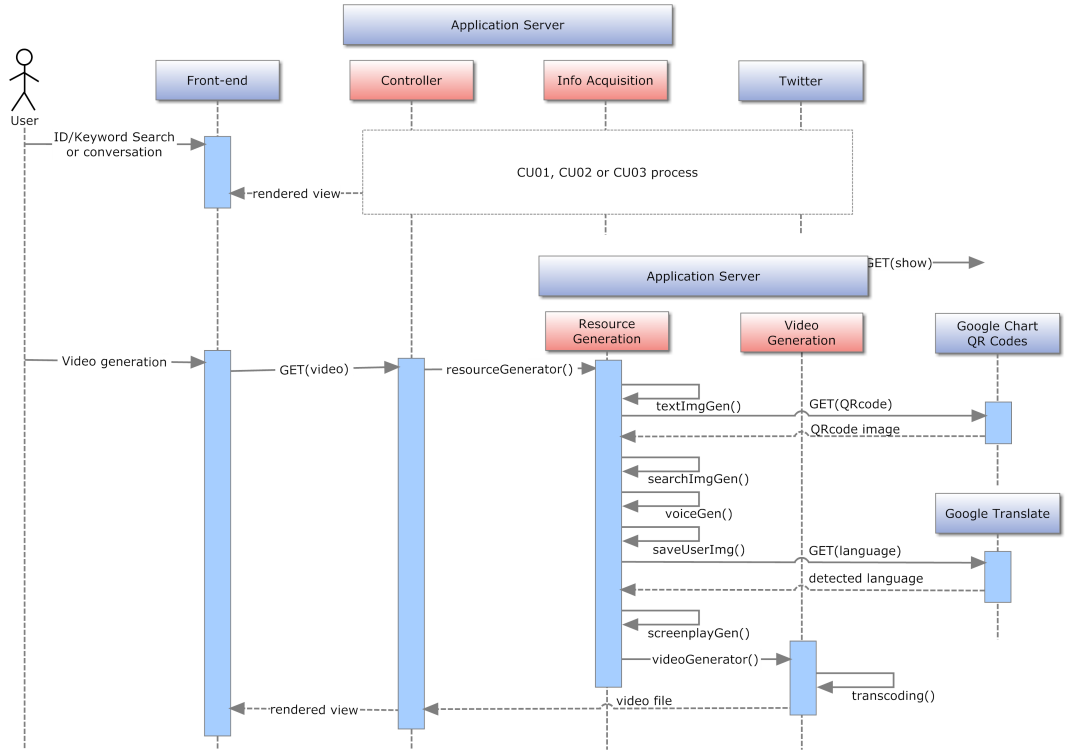


Figura 5.10. Caso de uso 5. Visualizar vídeo generado a partir de *tweets*

5.3. Presentación de la demo

La implementación de la lógica del servidor de aplicaciones que se encarga de la transformación de la información extraída de Twitter en un vídeo que representa, de manera automática, esa información; es el núcleo principal de este proyecto. En este núcleo se han introducido requisitos de diseño para que el desarrollo del sistema estuviera guiado por el demostrador diseñado en la Sección 4.4. Además, se ha tenido que desarrollar un sencillo frontal web que sirviera de punto de acceso a este demostrador de modo que se facilitara la presentación de los resultados obtenidos del grueso del proyecto.

5.3.1. Escenas resultantes

En este apartado se recogen algunas capturas obtenidas a partir de vídeos generados mediante los distintos módulos implementados en este proyecto.

La Figura 5.11 muestra una imagen extraída de un vídeo generado a partir de una búsqueda por ID. Como se puede observar en el panel de búsqueda, se señala el tipo de búsqueda que ha producido el vídeo junto con el término concreto objeto de la misma.

En este caso, se puede ver como se ha realizado una consulta de los últimos *tweets* publicados por el usuario "GoogleMobile". También se puede observar que aparece un panel con un código QR que está conectado con la URL que se aprecia en el texto del mensaje.

Figura 5.11. Vídeo de búsqueda por ID

En la Figura 5.12, por contra, se recoge una captura del vídeo generado para una búsqueda por *Keyword*. En este caso, el panel de búsqueda tiene marcada la pestaña de *Keyword* y muestra el término de la búsqueda, que se trata del *hashtag* "#Android". Obviamente, la información del panel de *tweet* es otra completamente distinta, así como el código QR que representa la URL de este mensaje.

Como último ejemplo de las escenas resultantes de la implementación completa del sistema, se puede ver en la Figura 5.13 la captura de un vídeo generado para una conversación. En esta ocasión, se prescinde del panel de búsqueda ya que no tiene aplicación ninguna. Se puede observar como la distribución de la información del panel de *tweet* tiene orientación derecha, con la imagen del perfil del usuario que escribió el mensaje en posición opuesta a los ejemplos que se acaban de ver. Como se ha comentado anteriormente, esto busca remarcar la alternancia entre los usuarios que participan en la conversación. Otro detalle que se puede extraer de este ejemplo es la ausencia de código QR, debido a que en este caso el mensaje no contiene ningúna URL. También se puede ver como está presente el icono de *reply* que indica que el mensaje ha sido publicado en respuesta a uno anterior.

Figura 5.12. Vídeo de búsqueda por *Keyword*

5.3.2. Front-end web

El front-end desarrollado para este sistema tiene el único cometido de ser utilizado para las pruebas de demostración, nunca debe ser contemplado como una aplicación en pre-producción. De momento, ni siquiera se puede considerar una primera versión de un producto, sino que se debe ver como una herramienta interna al proceso de implementación con una utilidad y una funcionalidad concreta y limitada. De todos modos, se va a proceder a mostrar una serie de capturas de algunas vistas que posee la aplicación. El navegador sobre el que se han realizado las capturas es Google Chrome en su versión 14.0.835.186 para sistemas Windows.

La primera captura que se puede ver en la Figura 5.14 pertenece a la vista que arroja la aplicación al acceder a ella inicialmente. Se trata de un redireccionamiento por defecto a la pestaña de búsqueda por ID. En esta vista se debe meter el nombre del usuario sobre el que se quiere realizar la consulta así como el número de mensajes que se quieren recuperar. También permite especificar si se quieren ver los mensajes que este usuario de Twitter a reenviado o sólo los publicados por si mismo.

El resultado de introducir un nombre de usuario en el formulario correspondiente y el número de mensajes que se quiere en el otro y pulsar el botón, es la captura que se ve en la Figura 5.15. En esta vista se observa el término de la búsqueda y el número de mensajes que se han recuperado previo a la presentación de la información de perfil del usuario objeto de la búsqueda. A continuación se muestran los mensajes recuperados de la consulta. Conviene tener en cuenta que se ha decidido mantener el diálogo de búsqueda en la posición superior de la vista para facilitar búsquedas posteriores ya que

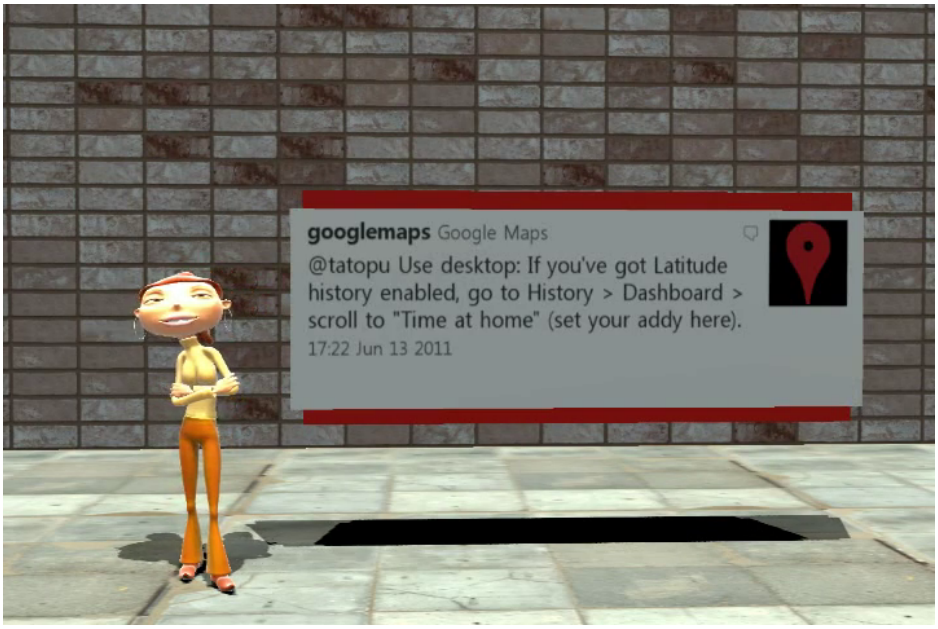


Figura 5.13. Vídeo de conversación

se ha considerado que podría resultar más cómodo para el usuario. Además, sobre la lista de mensajes, se puede ver cómo está colocado el botón que arranca el proceso de generación del vídeo, que culmina con el visionado del mismo en el reproductor comentado en la Sección 5.2.2.2. Como se observa en el texto de los mensajes, están habilitados los enlaces, para ser accedidos, de las URLs, los *hashtags* y las menciones.

La Figura 5.16 es el resultado de pulsar con el cursor sobre la pestaña de *Keyword*, que lleva directamente a la vista de búsqueda por *Keyword*. Esta vista es similar a la de la búsqueda por ID, con la única diferencia de la desaparición de la opción de incluir *retweets*, que no se contempla para este tipo de búsqueda.

Si se introduce un *Keyword* y un número de mensajes a leer y se pulsa el botón, se accede a la vista de la Figura 5.17 con el resultado de la búsqueda. La disposición general de la vista es similar al caso del resultado de la búsqueda por ID, con la salvedad de la desaparición de la información del perfil del usuario, que en este caso no tiene sentido presentar.

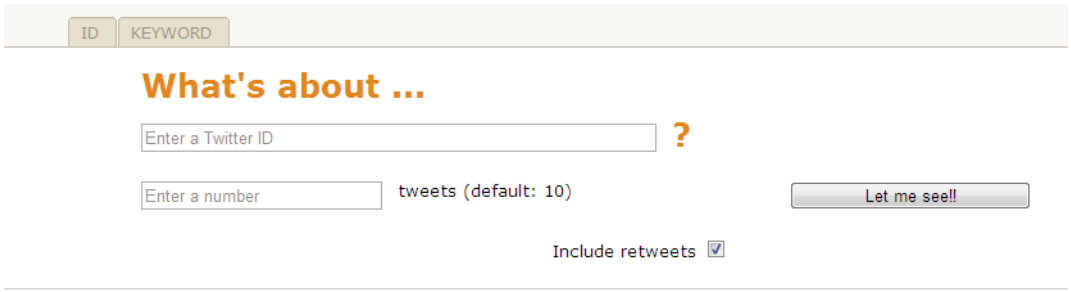


Figura 5.14. Vista inicial del front-end

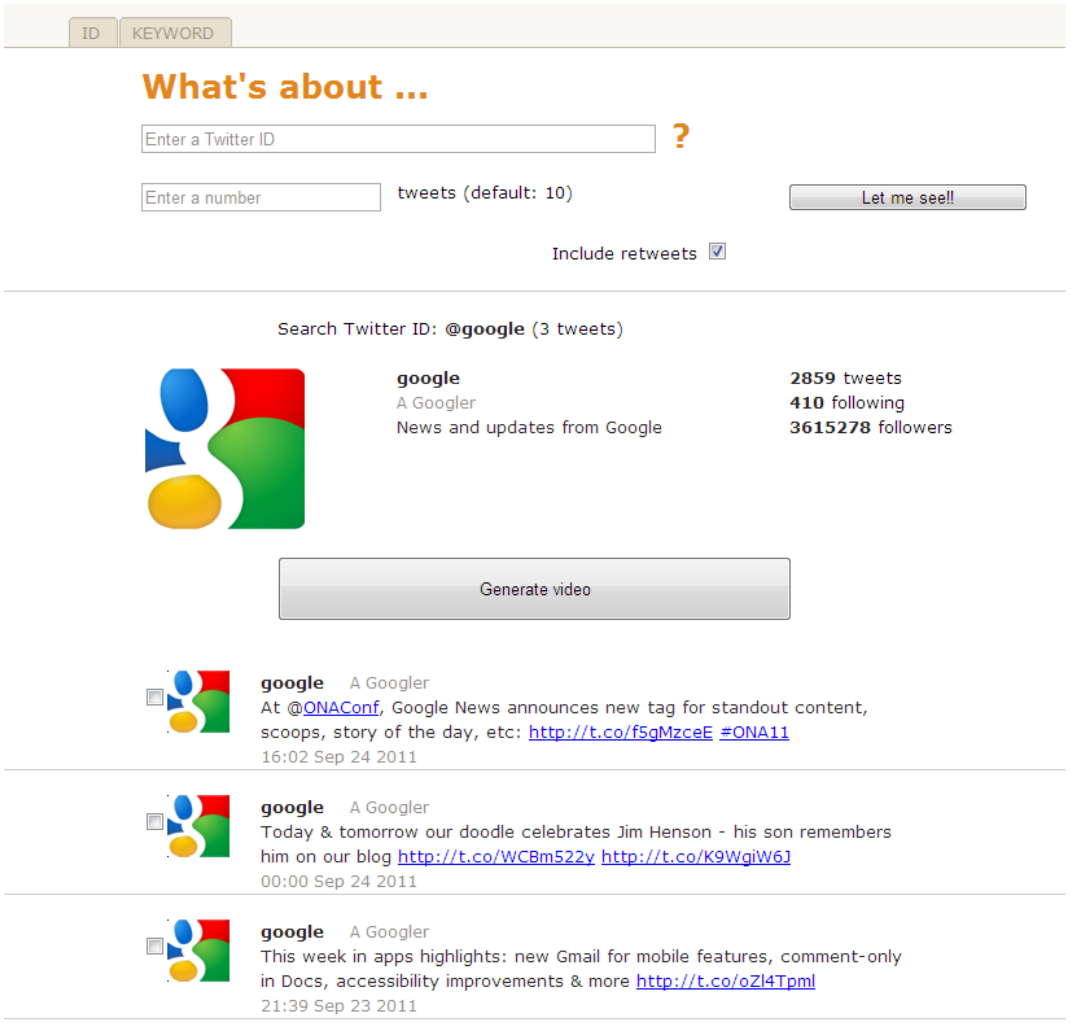


Figura 5.15. Vista del resultado de búsqueda por ID

Figura 5.16. Vista de la pestaña de búsqueda por *Keyword*

Figura 5.17. Vista del resultado de búsqueda por *Keyword*

Capítulo 6

Gestión del proyecto

En este capítulo se presentan distintos aspectos relativos a la gestión del proyecto. Primero, se desglosarán los medios técnicos empleados para la realización del sistema, tanto el hardware como el software. Posteriormente, se finalizará con una breve descripción de los gastos que forman el presupuesto del proyecto.

6.1. Medios técnicos empleados

Aquí se presentan las diferentes herramientas hardware y software que han sido requeridas durante la realización del proyecto. Aunque algunos de los programas citados a continuación no son estrictamente imprescindibles para la realización del proyecto, han sido de gran ayuda y se ha decidido incluirlos también en este apartado.

6.1.1. Hardware

En la Tabla 6.1 se pueden ver los distintos dispositivos hardware que se han utilizado a lo largo de la realización del proyecto.

6.1.2. Software

En la Tabla 6.2 se muestra un listado que recoge todo el software utilizado para la realización del presente Proyecto Fin de Carrera.

6.2. Presupuesto

Para la elaboración del presupuesto del proyecto, se ha considerado, debido a las características especiales de desarrollo del mismo a tiempo parcial, que los días de

Servidor	Portátil HP Pavilion DV6-2080es
[Servidor] Procesador	Intel Core i7-720QM Processor (6M Cache, 1.60 GHz)
[Servidor] Memoria RAM	4GB DDR3
[Servidor] Tarjeta gráfica	NVIDIA GeForce GT 230M (1GB)
[Servidor] Disco duro	500GB SATA 7200 rpm
Dispositivo Android	Samsung Galaxy S - Android 2.2.1

Tabla 6.1. Hardware utilizado

Sistema operativo	Windows 7 Home Premium (64 bits)
Entorno de desarrollo integrado	Eclipse IDE for PHP Developers
Backup y control de versiones	Dropbox
Diseño 3D	Autodesk 3ds Max 2010 (32bit)
Síntesis de voz	2nd Speech Center
Diseño de diagramasl	SmartDraw 2010
Reproducción de vídeo	VLC Media Player 1.1.11
Editor de texto avanzado	gedit 2.30.1
Distribución de LaTeX	MiKTeX 2.8

Tabla 6.2. Software utilizado

trabajo tienen 3 horas (en media), las semanas 5 días y los meses 4 semanas. De este modo, teniendo en cuenta que el tiempo de vida del proyecto ha sido de 12 meses, el número de horas totales de dedicación al mismo asciende a 720 horas. Este número de horas puede ser dividido en 9 meses dedicados a la parte de diseño y desarrollo de la aplicación (540 horas) y los 3 meses restantes realizando el proceso de documentación del trabajo (180 horas) principalmente invertido en la escritura de esta memoria.

El coste imputado al proyecto en concepto de personal se debe a la cantidad de horas de duración del proyecto y al precio/hora que cobra, en este caso, un Ingeniero de Telecomunicaciones. Se ha considerado un precio de la hora de un Ingeniero, ajustado a la situación actual del mercado para el sector de las Telecomunicaciones, de 50 €/hora. Se pueden ver estos datos en la Tabla 6.3.

El coste derivado del hardware es amortizable, con lo cual tan sólo tendrá repercusión sobre el presupuesto del proyecto el tiempo en que cada equipo sea utilizado en

Concepto	Horas	Honorarios	Coste RRHH
Ingeniero de Telecomunicaciones (Diseño y desarrollo)	540 horas	50 €/hora	27.000 €
Ingeniero de Telecomunicaciones (Documentación)	180 horas	50 €/hora	9.000 €
			36.000 €

Tabla 6.3. Costes de personal

el mismo sobre el total de su tiempo de vida. Esta información está contenida en la Tabla 6.4.

Concepto	Precio	Vida útil	Tiempo de uso	Coste
Portátil HP Pavilion DV6-2080es	999 €	36 meses	12 meses	333 €
Samsung Galaxy S (Android 2.2.1)	449 €	36 meses	6 meses	74,83 €
				407,83 €

Tabla 6.4. Costes del hardware

La manera de imputar el coste del software es similar a como se ha realizado con el hardware, ya que también se consideran amortizables las licencias adquiridas para poder utilizar un determinado producto software. En la Tabla 6.5 se recogen los costes derivados de este concepto.

Es complicado establecer una lista de todos los costes indirectos aplicables al proyecto. Se intentarán estimar a través de la suma del coste de la conexión a Internet y el gasto de luz. Para la conexión a Internet, se ha considerado una tarificación mensual por parte del proveedor de servicios de Internet de un total de 56,05 €/mes. En cuanto a la luz, se aplicará un gasto mensual de 50 €/mes. La Tabla 6.6 recoge estos datos.

Por último, se recoge en la Tabla 6.7 el coste total en que se ha incurrido, que es la suma de los costes calculados previamente.

Concepto	Precio	Vida útil	Tiempo de uso	Coste
Eclipse IDE for PHP Developers	0 €	-	-	0 €
Dropbox	0 €	-	-	0 €
Autodesk 3ds Max 2010 (32bit)	2.602 €	18 meses	12 meses	1.734,67 €
2nd Speech Center + voz español + voz inglés	84,82 €	12 meses	12 meses	84,82 €
SmartDraw 2010	146,64 €	12 meses	6 meses	73,32 €
VLC Media Player 1.1.11	0 €	-	-	0 €
gedit 2.30.1	0 €	-	-	0 €
MiKTeX 2.8	0 €	-	-	0 €
				1.892,81 €

Tabla 6.5. Costes del software

Concepto	Precio mensual	Tiempo de uso	Coste
Conexión a Internet	56,05 €	12 meses	672,6 €
Consumo de luz	50 €	12 meses	600 €
			1.272,6 €

Tabla 6.6. Costes indirectos

Concepto	Coste
Coste de personal	36.000 €
Coste del hardware	407,83 €
Coste del software	1.892,81 €
Costes indirectos	1.272,6 €
	39.573,24 €

Tabla 6.7. Costes totales

Capítulo 7

Conclusiones y líneas futuras

7.1. Conclusiones

Tras ver los resultados obtenidos tras implementar el sistema global que integra Twitter y NINOS, se puede concluir que NINOS es una herramienta más potente e interesante incluso de lo que se esperaba en un primer momento. La unión de ambas tecnologías realizada en este proyecto ha conseguido, contando con unos recursos humanos y técnicos muy limitados, establecer una serie de funcionalidades completamente innovadoras respecto al estado del arte del momento.

El hecho de elevar NINOS a la Red y aprovechar sus características en un entorno cambiante en tiempo real como es Twitter, y potencialmente accesible por millones de personas, abre una inmensa cantidad de nuevos usos y caminos para explorar. Resulta además muy acertada la usabilidad alcanzada con la aplicación web ligera desplegada para el demostrador, que marca la posible tendencia, en cuanto a presentación, de sistemas similares que pudieran explotar la funcionalidad provista por NINOS.

Cierto es que la versión de NINOS utilizada para este proyecto tiene aún mucho trabajo por delante, para mejorar un producto que ya de por sí es muy bueno. Sin embargo, existen aspectos relativos al rendimiento y los tiempos de los procesos de generación de los vídeos que deben ser pulidos ya que tienen una repercusión importante en la percepción que se obtiene del funcionamiento global de la herramienta. Si bien es cierto, que el entorno de despliegue del sistema implementado no ha sido el más adecuado, al utilizar sistemas hardware de bajo perfil que para nada tendrían que ver con los que se utilizarían en un entorno de producción.

El concepto del demostrador elegido para este proyecto basado en la Red parece que está en línea con las tendencias actuales de la llamada Web 2.0 en la que triunfan importantes cantidades de servicios ofrecidos íntegramente sobre Internet. Es por esto que se cree que, realizando un replantamiento de la idea de la presentación de los *tweets* de los usuarios mediante vídeos compuestos en tiempo real y desarrollando ésta un poco más, podría obtenerse una aplicación web ligada a Twitter con posibilidades

de hacerse un hueco en el ecosistema de aplicaciones que hacen uso de su API y proveer una funcionalidad extra al sitio de *microblogging* que llamara realmente la atención de los usuarios. Además, pensando brevemente se pueden encontrar varias opciones que podría ofrecer la implementación planteada en este proyecto dentro de un despliegue real del servicio, como por ejemplo la personalización por parte de los usuarios de los actores o la publicidad insertada en los escenarios, con las cuales conseguir ingresos. Mediante un acuerdo, esto podría abrir a Twitter una nueva línea de explotación de su servicio que podría resultarles interesante sobretodo ahora que está intentando conseguir nuevos medios de obtener beneficios con su sistema [57].

7.2. Líneas futuras

En este apartado se tratan algunas de las posibles líneas futuras sobre las que se podría trabajar para ampliar la funcionalidad del sistema implementado.

7.2.1. Acceso privado

Permitir el acceso de los usuarios de Twitter a su propia cuenta a través de la aplicación web del demostrador. Esto se realizaría mediante la autenticación a través de OAuth. De este modo, se podrían ofrecer un gran número de nuevas funciones y recursos recogidos por el API de Twitter pero que requieren estar autenticados como usuario, como por ejemplo publicar *tweets*.

7.2.2. Geolocalización

Aprovechar la geolocalización para plantear el escenario del vídeo. De este modo se pueden tener pequeñas recreaciones de ciudades importantes y utilizarlas como escenario para la animación de un *tweet* geolocalizado en ese lugar. También se puede cargar la bandera del país desde donde se envía un *tweet* geolocalizado como otro modo de presentar la información de localización de los usuarios.

Otro posible escenario que se puede utilizar es la vista de StreetView, facilitada por el API de Google Maps, desde donde se esté geolocalizado. Se puede utilizar la geolocalización también para descargar imágenes del sitio o de lugares cercanos (a través del API de Google Maps) y utilizarlas para personalizar el escenario (colocándolas en cuadros, posters...)

Utilizar API de un servicio meteorológico para, a partir de la geolocalización de un *tweet*, representar el escenario con las condiciones meteorlógicas del lugar desde donde se ha enviado.

7.2.3. Filtro de palabras malsonantes

Actualmente Twitter no realiza una clasificación de los *tweets* según el tipo de lengüaje utilizado en ellos. Esto podría suponer que usuarios con poca edad se encuentren con palabras malsonantes al hacer uso de la aplicación desarrollada, lo cual no sería conveniente para su desarrollo cognitivo además de exponer esta aplicación a una valoración negativa por parte de los padres.

Sería relativamente sencillo y abarcable realizar un filtro de palabras y expresiones malsonantes en las etapas posteriores a la finalización de este proyecto, logrando un resultado efectivo que dotaría a la aplicación de un valor añadido. Este filtro se podría desarrollar con varios niveles en los que se optara, por ejemplo, por eliminar visualmente la palabra, reproducir un sonido en vez de ser dicha la expresión por el personaje, codificar sólo algunas de las letras...

7.2.4. Interpretación de acrónimos

El lenguaje utilizado en los servicios de mensajería instantánea y en Twitter está cargado de acrónimos. Estos acrónimos se utilizan en lugar de las expresiones a las que representan para ahorrar tiempo en su escritura y espacio en los mensajes (especialmente importante en Twitter que limita sus *tweets* a 140 caracteres).

Podría resultar valioso desarrollar un sistema de interpretación multilíngüe de estos acrónimos que guarde una relación de los más utilizados junto con su significado y sean sustituídos a la hora de narrarse los *tweets* para facilitar su entendimiento. De esta manera se evita que los mensajes se conviertan en ocasiones en una sucesión de letras aparentemente carentes de significado.

7.2.5. Humanización

Evitar que el avatar lea los símbolos que componen los emoticonos y hacer que éste realice una representación facial del emoticono correspondiente mediante el uso del tag "activation evaluation" incluido en la Plataforma NINOS.

7.2.6. Personalización

Uno de los campos que se podría desarrollar para hacer más atractivo el uso del sistema por parte de los usuarios sería dar la posibilidad de personalizar los diferentes objetos que aparecen en la escena. Posibilitando la edición del narrador que aparece en la escena o creando nuevos personajes para asignarlos a la interpretación de los *tweets* de un determinado usuario de Twitter. De esta manera, se podría conseguir una mayor implicación de los usuarios finales del sistema y por tanto una mayor difusión y alcance de la aplicación.

Permitir la interacción con otras aplicaciones online de modo que se aumente las posibilidades interacción del usuario. Permitir subir los vídeos generados al perfil de Youtube, publicar en Twitter el vídeo resultante de una búsqueda o publicarlo en el muro de Facebook son algunas de las mejoras que se podrían incluir en este aspecto.

7.2.7. HTML5

Modificar la estructura del sistema para probar la potencia de WebGL en HTML5 en lugar de la Plataforma NINOS, para la generación de las animaciones de manera automática. Las posibles ventajas de esta nueva implementación parecen claras ya que se aligeraría la carga soportada por el servidor al realizarse el proceso de generación de cada escena 3D en el navegador propio de cada usuario.

7.2.8. Dispositivos portátiles

Sería interesante tratar de obtener formatos y resoluciones óptimas para los distintos sistemas móviles (Android, iOS y RIM, entre otros) y dispositivos portátiles existentes en la actualidad. Se realizaría una detección de las características del dispositivo utilizado por el usuario y a partir de éstas, habría que llevar a cabo una transcodificación concreta que adapte el vídeo al dispositivo.

7.2.9. Computación distribuída

El principal problema que tiene actualmente el sistema diseñado, es el tiempo que lleva el procesado de las animaciones hasta crear un vídeo en crudo. Esto supone una limitación a la hora de utilizar este sistema para generar vídeos en un tiempo asumible para una aplicación web. Este problema radica en la utilización de una sola máquina, no demasiado potente, para el desarrollo de la demo que se presenta como prueba de concepto en este proyecto final de carrera.

Una posible solución para mejorar el rendimiento y la respuesta de la aplicación sería cambiar el actual servidor utilizado por una máquina más potente, como podría ser una estación de trabajo, con una mayor capacidad de proceso que permita acortar los tiempo de creación de los vídeos con la Plataforma NINOS. Aún así, nos encontraríamos con el problema de la degradación del rendimiento de la nueva máquina al recibir accesos simultáneos a la aplicación y por tanto, peticiones de generación de videos al mismo tiempo. Esto nos llevaría de nuevo a tener una aplicación web que no reporta un vídeo al usuario dentro de los límites lógicos de respuesta que cabría esperar de una aplicación de este tipo.

Por todo esto, se ha pensado que la mejor solución para evitar que el acceso simultáneo de usuarios a la aplicación haga resentir el rendimiento de la misma, es montar una arquitectura de computación distribuída que descentralice la tarea de ge-

neración de vídeos a más estaciones de trabajo conectadas dentro de una misma red. De esta manera, se conseguiría repartir las distintas peticiones de generación de vídeos logrando que los distintos usuarios perciban un rendimiento adecuado de la aplicación. Dependiendo del número de usuarios simultáneos que reciba la aplicación, habría que realizar un estudio para dimensionar de manera correcta la red de máquinas que formarían parte de la arquitectura distribuída.

Otra de las tareas que afecta de mayor manera al rendimiento final de la aplicación es la transcodificación que se realiza desde el vídeo en crudo arrojado por la Plataforma NINOS a los distintos formatos (además de calidad y tamaños del mismo) que se puedan necesitar dependiendo del dispositivo desde el que se acceda a la aplicación o con los que se quiera compartir dicho vídeo. Actualmente es el propio servidor de la aplicación el que se ocupa de realizar esta tarea de forma secuencial con tantos perfiles de vídeo distintos como se quieran tener.

Aprovechando la red de computación distribuída que comentabamos antes, se podría hacer que estos equipos utilizaran sus períodos de desocupación para realizar el proceso de transcodificación. De este modo, distintos equipos realizarían la transcodificación de los vídeos en paralelo, con la resultante mejora en el rendimiento de la aplicación. Una opción interesante para realizar esta tarea dentro del software opensource podría ser Apache Hadoop [58], mantenido y distribuido por The Apache Software Foundation.

Anexo A

Demo: Servicio Meteorológico

Anterior al desarrollo de este proyecto, se prepara un demostrador para comprobar el funcionamiento de NINOS y observar de un modo más cercano su potencial. Esta primera prueba de concepto, de complejidad limitada, se ocupa de simular un programa de previsión del tiempo a partir de animaciones 3D y audio pregrabado. La animación representa una hipotética mujer del tiempo narrando la predicción meteorológica para el día actual de la capital de provincia española elegida. Los datos de la temperatura máxima y mínima de dicha ciudad se recuperan en tiempo real (aunque sólo se actualizan en origen una vez al día) de la web de la Agencia Estatal de Meteorología (AEMet), ver Figura A.1. El resultado buscado para la escena de predicción es similar al mostrado en las Figuras A.2(a) y A.2(b).

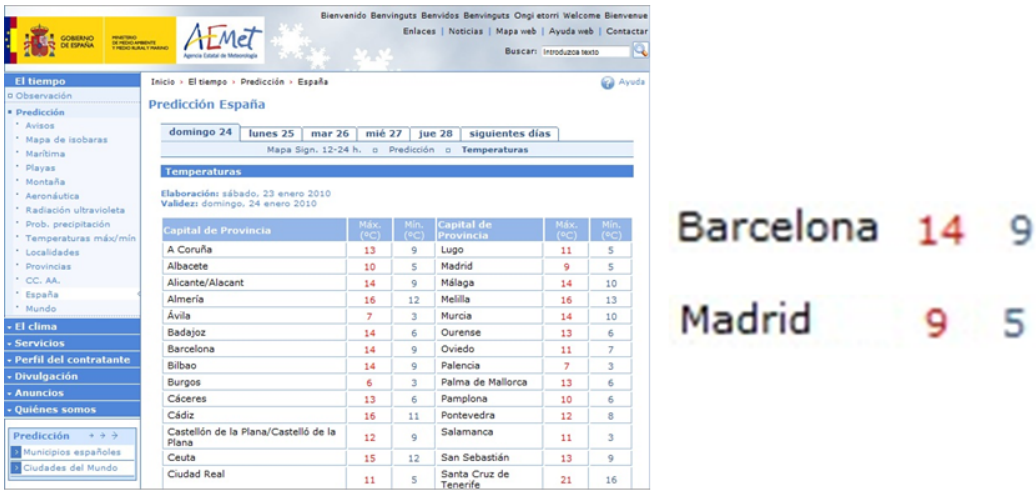
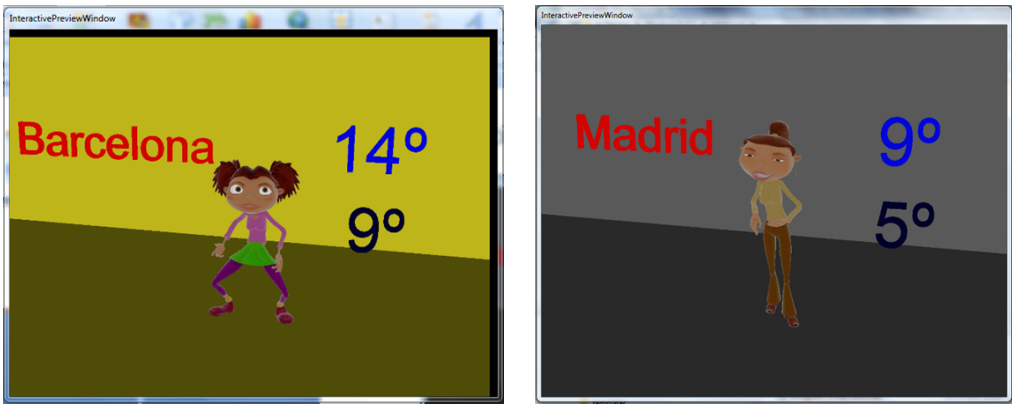


Figura A.1. Web AEMet - Predicción temperatura diaria

La ejecución de esta demo se gestiona mediante un archivo por lotes de Windows lanzado desde la línea de comando. Este archivo (conocido como *batch*) está compuesto por texto sin formato que recoge comandos de MS-DOS. El archivo solicita al usuario



(a) (b)

Figura A.2. Ejemplos de predicción de Barcelona (a) y Madrid (b)

el nombre de la provincia de la cual se desea consultar las temperaturas máxima y mínima para el día en curso. Al introducir la provincia se lanza un programa escrito en PHP al que se le pasa ésta. Este sencillo script se ocupa de realizar extracción de los datos (*web scraping*) de la página de la AEMet donde se recoge las temperaturas diarias organizadas por provincia (http://www.aemet.es/es/eltiempo/prediccion/espana?w=13).

La información obtenidada a partir de la extracción de la web, es parseada buscando la provincia solicitada para extraer las temperaturas máxima y mínima de ésta. Una vez se conocen todos los datos de entrada que necesita la función de generación de guión XML (provincia, temperatura máxima y temperatura mínima) se lanza el proceso.

La escena diseñada trata de parecer un programa meteorológico como los que se pueden ver en televisión pero en formato muy reducido. Se ha decidio dividir la escena en seis bloques para manejar su planificación. Estos bloques son la introducción, el saludo, la narración de la temperatura máxima, la narración de la temperatura mínima, la despedida y la salida. Durante el bloque de introducción, la cámara (elegida de manera aleatoria) realiza un *travelling* de acercamiento hasta alcanzar la posición del actor, quedánsose entonces fija enfocándolo. Mientras, suena una sintonía de cabecera simulando el comienzo de un programa de televisión.

A continuación comienza el bloque de saludo, en el cual el actor pronuncia una frase (elegida aleatoriamente entre tres ya pregrabadas) de bienvenida al servicio meteorológico, indicando la provincia sobre la que se va a realizar la previsión. En el siguiente bloque aparecen en la escena sobreimpresos el nombre de la provincia y la temperatura máxima que se alcanzará durante el día (ver Figura A.3). El actor realiza un comentario sobre esta temperatura. La frase que utiliza de comentario está elegida de manera aleatoria entre varias que han sido pregrabadas para rangos de temperatura. De este modo, existen un par de frases entre las que se elegirá si la temperatura es menor a 0ºC, otras para una temperatura entre 0ºC y 15ºC y otras tantas si la tem-

peratura supera los 15°C. El bloque que le sigue, el actor realiza algo similar para la temperatura mínima (que aparece en este momento sobre la escena), también con sus correpondientes frase separadas en rangos. En la A.4 se puede apreciar la disposición de la escena durante este bloque. Para acabar, en el bloque de despedida, el actor saluda y se despide con una frase elegida al azar entre varias para pasar a caminar saliendo de la escena mientras se oye la misma melodía del comienzo (bloque de salida).

Figura A.3. Predicción de la temperatura en Segovia (Máxima)

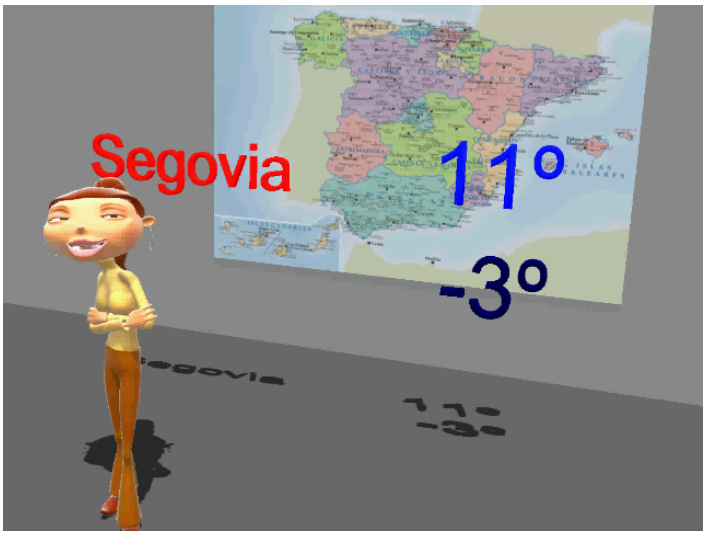


Figura A.4. Predicción de la temperatura en Segovia (Mínima)

La generación del guión XML que forma la disposición diseñada para la escena, está implementado en PHP. Para apoyar la generación del guión se ha creado la base estructural que forma los guiones XML utilizados por NINOS para generar las escenas en vídeo (se puede ver con más detalle en el Anexo B). Se ha implementado un entramado de objetos y atributos, utilizando PHP, que se encargan de mapear las distintas etiquetas con las que NINOS trabaja.

Anexo B

Descripción estructura guión XML

Los guiones utilizados por la plataforma NINOS deben tener un esquema determinado de etiquetas. Existen dos tipos de etiquetas, las que son propiedades básicas que sólo pueden contener un valor, y las que son objetos que a su vez pueden contener una o varias etiquetas de cualquiera de los dos tipos.

A continuación se describe la estructura de etiquetas de los guiones. Las etiquetas subrayadas se corresponden con etiquetas de tipo objeto y el resto son de tipo propiedad.

- ***Timeline*** - Etiqueta raíz del guión. Contiene la definición temporal de todos los recursos que aparecen en la escena
  - **Id** - Identificador
  - **Config** - Objeto que contiene la información de configuracion del vídeo
  - **Program_block** - Delimita el bloque de cada escena
  - **Generic** - Objeto genérico (presente por defecto)
  - **Object3d** - Objeto 3D que aparece en escena
  - **Audio** - Pista de audio que suena en escena
  - **Actor** - Personaje que aparece en escena
  - **Camera** - Cámara presente en escena
- ***Config*** - Contiene la información de configuracion del vídeo
  - **Id** - Identificador
  - **Width** - Anchura de resolución del vídeo
  - **Height** - Altura de resolución del vídeo
  - **Fps** - Tasa de imágenes por segundo del vídeo

- ***Program_block*** - Delimita el bloque de cada escena
  - **Id** - Identificador
  - **Begin** - Referencia temporal de inicio del recurso
  - **End** - Referencia temporal de fin del recurso
- ***Generic*** - Objeto genérico
  - **Id** - Identificador
  - **Src** - Archivo de origen
- ***Object3d*** - Objeto 3D que aparece en escena
  - **Id** - Nombre del fichero donde se encuentra el objeto 3D
  - **Src** - Directorio donde se encuentra
  - **State** - Estado del objeto
  - **Transform** - Transformación del objeto
  - **Audio** - Pista de audio perteneciente al objeto
- ***Audio*** - Pista de audio que suena en escena
  - **Id** - Identificador
  - **Begin** - Referencia temporal de inicio del recurso
  - **End** - Referencia temporal de fin del recurso
  - **Src** - Directorio completo donde se encuentra la pista de audio
  - **Fade_in** - Duración del fundido de entrada
  - **Fade_out** - Duración del fundido de salida
  - **Volume** - Volumen del audio (en tanto por uno)
- ***State*** - Estado del objeto o del actor durante la escena
  - **Id** - Identificador
  - **Begin** - Referencia temporal de inicio del recurso
  - **End** - Referencia temporal de fin del recurso
  - **Fade_in** - Duración del fundido de entrada
  - **Fade_out** - Duración del fundido de salida
  - **Src** - Directorio completo donde se encuentra el recurso
  - **Offset_begin** - Offset respecto a la referencia temporal de inicio
  - **Offset_end** -Offset respecto a la referencia temporal de fin

- ***Actor*** - Personaje que aparece en escena
  - **Id** - Nombre del actor
  - **Src** - Directorio completo donde se encuentra el actor
  - **Lipsync_horizontal_sensitivity** - Sensibilidad de la sincronización labial del actor (en horizontal)
  - **Lipsync_vertical_sensitivity** - Sensibilidad de la sincronización labial del actor (en vertical)
  - **Lipsync_horizontal_bias** - Desplazamiento de la sincronización labial del actor (en horizontal)
  - **Lipsync_vertical_bias** - Desplazamiento de la sincronización labial del actor (en vertical)
  - **Auto_lipsync** - Flag de activación de lipsync automático (1=¿Activado, 0=¿Desactivado)
  - **State** - Estado del actor
  - **Gesture** - Gesto del actor
  - **Movement** - Movimiento del actor
  - **Audio** - Pista de audio perteneciente al actor
  - **Lookat** - Evento durante el que el actor mira a algún lugar determinado
  - **Bodyconfig** - Configuración del cuerpo del actor
  - **Activationevaluation** - Expresión de la boca del actor
- ***Gesture*** - Gesto del actor
  - **Id** - Identificador
  - **Begin** - Referencia temporal de inicio del recurso
  - **End** - Referencia temporal de fin del recurso
  - **Src** - Directorio completo donde se encuentra el recurso
  - **Fade_in** - Duración del fundido de entrada
  - **Fade_out** - Duración del fundido de salida
- ***Movement*** - Movimiento del actor
  - **Id** - Identificador
  - **Begin** - Referencia temporal de inicio del recurso
  - **End** - Referencia temporal de fin del recurso
  - **Src** - Directorio completo donde se encuentra el recurso
  - **Fade_in** - Duración del fundido de entrada

    - **Fade_out** - Duración del fundido de salida

- ***Lookat*** - Evento durante el que el actor mira a algún lugar determinado
    - **Id** - Identificador
    - **Begin** - Referencia temporal de inicio del recurso
    - **End** - Referencia temporal de fin del recurso
    - **Src** - Identificador del recurso hacia el que mira el actor
    - **Fade_in** - Duración del fundido de entrada
    - **Fade_out** - Duración del fundido de salida
- ***Bodyconfig*** - Configuración del cuerpo del actor
    - **Id** - Identificador
    - **Begin** - Referencia temporal de inicio del recurso
    - **End** - Referencia temporal de fin del recurso
    - **Src** - Nombre de la configuración corporal del actor
    - **Fade_in** - Duración del fundido de entrada
    - **Fade_out** - Duración del fundido de salida
- ***Activationevaluation*** - Expresión de la boca del actor
    - **Id** - Identificador
    - **Begin** - Referencia temporal de inicio del recurso
    - **End** - Referencia temporal de fin del recurso
    - **Src** - Posición de la boca del actor definida por un vector con dos posiciones
    - **Fade_in** - Duración del fundido de entrada
    - **Fade_out** - Duración del fundido de salida
- ***Morphing*** - Expresión facial del actor
- ***Camera*** - Cámara desde la que se observa la escena
    - **Id** - Identificador
    - **Begin** - Referencia temporal de inicio del recurso
    - **End** - Referencia temporal de fin del recurso
    - **Src** - Directorio completo donde se encuentra el recurso
    - **Name** - Nombre que identifica a la cámara dentro del fichero
    - **Fade_in** - Duración del fundido de entrada
    - **Fade_out** - Duración del fundido de salida
    - **Target** - Posición del objetivo hacia el que apunta la cámara

A continuación se puede observar un ejemplo de la estructura de un guión XML:

```
<timeline id="">
      <config id="default" width="640" height="480" fps="25" />
      <program_block id="block0000" begin="0" end="audio0008.end" />
      <generic id="" src="" />
      <object3d id="predictionStage.FBX"
            src="meshes\predictionStage.FBX">
            <state id="state0000" begin="0" end="audio0008.end"
            fade_in="0" fade_out="0" src="" offset_begin="0"
            offset_end="0" />
            <audio id="audio0000" begin="0" end="begin+9"
                  src="media\audio\weatherOpening.wav" fade_in="0"
                  fade_out="2" volume="1.00" />
            <audio id="audio0008" begin="audio0006.end" end="begin+9"
                  src="media\audio\weatherOpening.wav" fade_in="0"
                  fade_out="2" volume="1.00" />
      </object3d>
      <object3d id="max9.FBX" src="meshes\max9.FBX">
            <state id="state0001" begin="audio0004.begin"
            end="audio0008.end" fade_in="0" fade_out="0" src=""
            offset_begin="0" offset_end="0" />
      </object3d>
      <object3d id="min-4.FBX" src="meshes\min-4.FBX">
            <state id="state0002" begin="audio0005.begin"
            end="audio0008.end" fade_in="0" fade_out="0" src=""
            offset_begin="0" offset_end="0" />
      </object3d>
      <object3d id="avila.FBX" src="meshes\avila.FBX">
            <state id="state0003" begin="audio0004.begin"
            end="audio0008.end" fade_in="0" fade_out="0" src=""
            offset_begin="0" offset_end="0" />
      </object3d>
      <actor id="irina" src="actors\irina"
            lipsync_horizontal_sensitivity="1.00"
            lipsync_vertical_sensitivity="1.00"
            lipsync_horizontal_bias="0.00" lipsync_vertical_bias="0.00"
            auto_lipsync="1">
            <state id="state0004" begin="0" end="audio0008.end"
            fade_in="0" fade_out="0"
            src="actors\irina\movement\idle.fbx" offset_begin="0"
                  offset_end="0" />
            <gesture id="gesture0000" begin="audio0001.begin"
            end="begin+3" src="gesture_activeGreeting.fbx" fade_in="0"
            fade_out="0" />
            <gesture id="gesture0001" begin="audio0004.begin"
            end="begin+1.3" src="gesture_neutralTalk_front.fbx"
            fade_in="0" fade_out="0" />
            <gesture id="gesture0002" begin="audio0005.begin"
            end="begin+1.3" src="gesture_passiveTalk_front.fbx"
            fade_in="0" fade_out="0" />
            <gesture id="gesture0003" begin="audio0006.end"
            end="begin+3" src="gesture_activeGreeting.fbx" fade_in="0"
            fade_out="0" />
            <movement id="movement0000" begin="3" end="begin+3.3"
                  src="movement_walking.fbx" fade_in="1" fade_out="2"
                  loop_media="1" />
            <movement id="movement0001" begin="gesture0003.end"
            end="begin+10" src="movement_walking.fbx" fade_in="1"
            fade_out="2" loop_media="1" />
            <audio id="audio0001" begin="audio0000.end" end="begin+7"
                  src="media\audio\intro1.wav" fade_in="0" fade_out="0"
                  volume="1.00" />
```

```
            <audio id="audio0003" begin="audio0001.end-1.4"
            end="begin+2" src="media\audio\avila.wav" fade_in="0"
            fade_out="0" volume="1.00" />
            <audio id="audio0004" begin="audio0003.end" end="begin+9"
                  src="media\audio\day1_1.wav" fade_in="0" fade_out="0"
                  volume="1.00" />
            <audio id="audio0005" begin="audio0004.end" end="begin+8"
                  src="media\audio\night0_2.wav" fade_in="0"
                  fade_out="0" volume="1.00" />
            <audio id="audio0006" begin="audio0005.end" end="begin+4"
                  src="media\audio\farewell2.wav" fade_in="0"
                  fade_out="0" volume="1.00" />
            <lookat id="lookat0000" begin="audio0001.begin"
            end="audio0003.end" src="activecamera" fade_in="1"
            fade_out="0" />
            <lookat id="lookat0001" begin="audio0004.begin"
            end="audio0004.end" src="activecamera" fade_in="1"
            fade_out="0" />
            <lookat id="lookat0002" begin="audio0005.begin"
            end="audio0005.end" src="activecamera" fade_in="1"
            fade_out="0" />
            <lookat id="lookat0003" begin="audio0006.begin"
            end="audio0006.end" src="activecamera" fade_in="1"
            fade_out="0" />
            <lookat id="lookat0004" begin="audio0006.end" end="begin+5"
                  src="activecamera" fade_in="1" fade_out="1" />
            <track type="bodyconfig" erasable="1">
                  <bodyconfig id="bodyconfig0000" begin="0"
                  end="audio0008.end" src="basic" fade_in="0"
                  fade_out="0" />
            </track>
            <track type="activationevaluation" erasable="1">
                  <activationevaluation id="activationevaluation0000"
                        begin="movement0000.end" end="audio0008.end-5"
                        src="-0.48 0.37" fade_in="2" fade_out="2" />
            </track>
      </actor>
      <camera id="camera0000" begin="0" end="begin+6"
            src="media\camera\Camera06.fbx" name="Camera06" fade_in="0"
            fade_out="0" target="" />
      <camera id="camera0001" begin="audio0004.begin"
      end="audio0004.end" src="media\camera\Camera01.fbx"
      name="Camera01" fade_in="0" fade_out="0" target="" />
      <camera id="camera0002" begin="audio0005.begin"
      end="audio0005.end" src="media\camera\Camera03.fbx"
      name="Camera03" fade_in="0" fade_out="0" target="" />
      <camera id="camera0003" begin="audio0006.begin"
      end="audio0006.end" src="media\camera\Camera02.fbx"
      name="Camera02" fade_in="0" fade_out="0" target="" />
      <camera id="camera0004" begin="audio0006.end" end="begin+6"
            src="media\camera\Camera08.fbx" name="Camera08" fade_in="0"
            fade_out="0" target="" />
</timeline>
```

Anexo C

API Twitter

El API de Twitter para desarrolladores se compone de tres partes: dos APIs REST y un API de streaming. Las dos APIs REST se encuentran separadas históricamente ya que la que proporcionaba la capacidad de búsqueda para los datos de Twitter estaba gestionada por Summize Inc., compañía independiente de Twitter, que fue adquirida más tarde y rebautizada como Twitter Search. El API de streaming es diferente de las dos APIs REST ya que el streaming debe soportar conexiones de larga duración sobre una arquitectura diferente.

Los métodos del API REST de Twitter permiten a los desarrolladores acceder a los datos del núcleo de Twitter. Esto incluye las actualizaciones del *timeline*, los datos de estado y la información del usuario. Los métodos del API de búsqueda proveen a los desarrolladores de un mecanismo de interacción con Twitter Search y los datos de tendencias. El API de streaming proporciona un acceso a un alto volumen de *tweets* casi en tiempo real en forma de muestreo y filtrado.

Twitter establece el estándar abierto OAuth para la autenticación a la hora de utilizar sus APIs . Sin embargo, no todas las APIs de Twitter requieren el uso de la autenticación. Mientras el API de búsqueda no requiere autenticación, tanto el API REST de Twitter como el API de streaming requieren autenticarse para poder ser utilizados. El API de streaming soporta autenticación mediante OAuth y autenticación básica (usuario y contraseña). El API REST de Twitter además de OAuth, permite el modo de autenticación out-of-band/PIN code y xAuth, que sigue el flujo de autenticación de OAuth pero con algunas variaciones.

Se puede encontrar más información sobre los procesos de autenticación y las funciones de consulta al API en [59].

Anexo D

Glosario de términos

Esta sección pretende servir como punto de referencia a una persona no vinculada al sector de las tecnologías de la información y familiarizarlo con los términos y conceptos utilizados. Por este motivo no se pretende que la misma tenga rigor docente sino facilitar el entendimiento de esta documentación en forma precisa y correcta de acuerdo a lo que se desea transmitir.

API Interfaz de Programación de Aplicaciones (en inglés, *Application Programming Interface*). Grupo de rutinas que provee un sistema operativo, una aplicación o una biblioteca, que define cómo invocar desde un programa un servicio que éstos prestan.

CGI Imagen generada por ordenador (en inglés, *Computer Generated Imagery*). Imágenes artificiales creadas mediante el uso de herramientas gráficas de ordenador. Pueden ser íntegramente realizadas de manera artificial o bien integrando también imágenes reales.

Código QR *Quick Response Barcode*. Código de barras bidimensional que permite codificar texto, direcciones web, eventos de calendario, información de contactos, direcciones de correo electrónico, datos de geolocalización y números de teléfono entre otros. El código consta de módulos de color negro dispuestos en un patrón de forma cuadrada sobre fondo blanco. Dependiendo del tamaño que tengan estos códigos, pueden almacenar hasta 4.296 caracteres alfanuméricos.

CSS Hojas de estilo en cascada (en inglés, *Cascading Style Sheets*)

DM *Direct Message*. Mensaje privado enviado a través de Twitter por un usuario a uno de sus *followers*.

FBX Formato de archivo (.fbx) propiedad de Autodesk, utilizado para proveer interoperabilidad entre aplicaciones de creación y diseño de contenido digital.

Follower Usuario de Twitter que recibe en su *timeline* las notificaciones de los *tweets* publicados por otro usuario.

Following Usuario de Twitter al que otro usuario (conocido como *follower*) está suscrito para recibir las notificaciones de sus *tweets* publicados. Este usuario puede enviar DMs a sus *followers*.

Hashtag Término utilizado en Twitter para describir toda palabra o frase precedida del prefijo # que permite a los usuarios remarcar palabras clave o temas en los *tweet* además de servir para categorizar mensajes y facilitar el acceso directo a mensajes por tema.

HTML Lenguaje de marcado de hipertexto (en inglés, *HyperText Markup Language*)

HTTP *HyperText Transfer Protocol*. Protocolo de transferencia de hipertexto, orientado a transacciones que sigue el esquema petición-respuesta entre un cliente y un servidor.

JSON *JavaScript Object Notation*. Estándar abierto basado en texto diseñado para el intercambio de datos legible por humanos. Se deriva del lenguaje de programación JavaScript para representar las estructuras de datos simples y matrices asociativas, llamadas objetos. A pesar de su relación con JavaScript, es independiente de este lenguaje y dispone de analizadores sintácticos para casi cualquier lenguaje de programación.

OAuth Estándar abierto para la autorización utilizado en Twitter, que permite a los usuarios compartir sus recursos privados (por ejemplo fotos, vídeos, listas de contactos) almacenados en un sitio, con otro sitio sin tener que entregar sus credenciales, por lo general nombre de usuario y contraseña.

OpenGL *Open Graphics Library*. Estándar abierto para la creación de aplicaciones que generen gráficos 2D y 3D a partir de primitivas geométricas simples (puntos, líneas y triángulos) definidas en una API multilenguaje y multiplataforma.

PHP *PHP Hypertext Preprocessor*. Lenguaje de scripting ampliamente utilizado para fines generales, especialmente adecuado para el desarrollo web ya puede ser embebido en páginas HTML.

Plataforma NINOS Conjunto de herramientas para la generación automática y semiautomática de piezas audiovisuales a partir de un guión escrito en XML.

REST Transferencia de Estado Representacional (en inglés, *Representational State Transfer*). Estilo de arquitectura software para sistemas hipermedia distribuidos como World Wide Web. Las arquitecturas de estilo REST están formadas por clientes y servidores. Los clientes inician peticiones a los servidores, los cuales procesan dichas solicitudes y devuelven las respuestas apropiadas. Las solicitudes y las respuestas se construyen alrededor de la transferencia de las representaciones de los recursos. Un recurso puede ser prácticamente cualquier concepto coherente y significativo que pueda ser representado por una dirección. Una representación de un recurso suele ser un documento que recoge el estado actual o previsto de un recurso.

TGA *Truevision Graphics Adapter*, también conocido como TARGA (*Truevision Advanced Raster Graphics Adapter*). Formato de archivo (.tga) de mapa de bits creado por Truevision Inc.

Timeline Término utilizado en Twitter para describir un flujo de *tweets* listados en orden temporal. El *timeline* de un usuario es un listado que recoge todos los *tweets* publicados por sus *followers*.

Tweet Término utilizado en Twitter para describir los mensajes de texto de hasta 140 caracteres que publican sus usuarios.

Twitter Sitio web que ofrece servicio de red social y microblogging (sus usuarios pueden enviar y publicar mensajes breves).

URL Localizador uniforme de recursos (en inglés, *Uniform Resource Locator*). Cadena de caracteres que especifica la localización de un recurso disponible en Internet.

Youtube Sitio web para compartir vídeos en los que los usuarios pueden subir, compartir y ver vídeos.

XML *Extensible Markup Language*. Llenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C) que proporciona una manera de definir lenguajes para diferentes necesidades. Algunos de estos lenguajes que usan XML para su definición son SAML, XHTML, etc.

Bibliografía

[1] SALERO Project. http://www.salero.info accedido 20/06/2011. 1, 8

[2] Twitter Surpasses 200 Million Tweets Per Day. Mashable. http://mashable.com/2011/06/30/twitter-200-million accedido 28/07/2011. 3

[3] Twitter Statistics http://twopcharts.com/twitter300million?source=nl accedido 28/07/2011. 3

[4] One Million Registered Twitter Apps. Twitter Blog. http://blog.twitter.com/2011/07/one-million-registered-twitter-apps.html accedido 28/07/2011 3

[5] Publications. SALERO Project. http://salero.info/b53b71c8df7dc268e455694de7bc3ca2/en/resources/index.html accedido 20/08/2011.

[6] OpenGL. http://www.opengl.org accedido 21/08/2011. 11

[7] Autodesk FBX. Platform-Independent 3D Data Interchange Technology. http://usa.autodesk.com/adsk/servlet/pc/index?id=6837478&siteID=123112 accedido 21/08/2011 11

[8] Apple Developer Connection. Technical Note TN1023. Understanding PackBits. http://web.archive.org/web/20080705155158/http://developer.apple.com/technotes/tn/tn1023.html accedido 21/08/2011 12

[9] Run Length Encoding (RLE). The Data Compression Resource on the Internet http://www.data-compression.info/Algorithms/RLE/index.htm accedido 21/08/2011 12

[10] L. Fitton, M.E. Gruen, L. Poston. *Twitter for Dummies*. 2009. Wiley Publishing, Inc. 13

[11] Texting and Twitter Campaigns Helping Haiti. suite101.com. http://www.suite101.com/content/texting-and-twitter-campaigns-helping-haiti-a190432 accedido 24/08/2011 13

[12] Japan earthquake: how Twitter and Facebook helped. The Telegraph. http://www.telegraph.co.uk/technology/twitter/8379101/Japan-earthquake-how-Twitter-and-Facebook-helped.html accedido 24/08/2011 13

[13] El Twitter de la Policía Nacional empieza a usarse como el teléfono 091. Expansión.com http://www.expansion.com/agencia/efe/2011/08/20/16433815.html accedido 24/08/2011 13

[14] Essex Police considers enlisting Twitter as means to fight crime. Mirror. http://www.mirror.co.uk/news/top-stories/2011/06/14/essex-police-considers-enlisting-twitter-as-means-to-fight-crime-115875-23199903 accedido 24/08/2011 13

[15] Odeo Releases Twttr. Techcrunch.com http://techcrunch.com/2006/07/15/is-twttr-interesting/ accedido 26/08/2011

[16] Odeo RIP, Hello Obvious Corp. GigaOM. http://gigaom.com/2006/10/25/odeo-rip-hello-obvious-corp/ accedido 26/08/2011

[17] A Conversation With Twitter Co-Founder Jack Dorsey. The Daily Anchor. http://www.thedailyanchor.com/2009/02/12/a-conversation-with-twitter-co-founder-jack-dorsey/ accedido 26/08/2011

[18] One Million Registered Twitter Apps. Twitter Blog. http://blog.twitter.com/2011/07/one-million-registered-twitter-apps.html accedido 27/08/2011 13

[19] Twitter On Your TV! Business Insider. http://www.businessinsider.com/twitter-on-your-tv-2009-3 accedido 27/08/2011 13

[20] The Only Twitter Applications List You'll Ever Need. Squidoo. http://www.squidoo.com/twitterapps accedido 27/08/2011 13

[21] Twitter Fan Wiki. http://twitter.pbworks.com/w/page/1779726/Apps accedido 27/08/2011 14

[22] One Million Registered Twitter Apps. Twitter Blog. http://blog.twitter.com/2011/07/one-million-registered-twitter-apps.html accedido 27/08/2011 14

[23] Twitter Acquisitions. Seo by the Sea. http://www.seobythesea.com/2011/01/twitter-acquisitions accedido 27/08/2011 14

[24] Twitter Acquires Atebits, Maker of Tweetie. The New York Times. http://bits.blogs.nytimes.com/2010/04/09/twitter-acquires-atebits-maker-of-tweetie accedido 27/08/2011 14

[25] Twitter Buys TweetDeck For $40 Million. TechCrunch. http://techcrunch.com/2011/05/23/twitter-buys-tweetdeck-for-40-million accedido 27/08/2011 14

[26] A. A. Puntambekar. *Web Technologies*. Technical Publications. 2009 15

[27] B. Livingston. *Using Web 2.0 Technologies*. ASTD Press. 2010 15

[28] E. Tittel, Jeff Noble. *HTML, XHTML & CSS For Dummies*. Wiley Publishing. 2008 15, 17

[29] Where the web was born. CERN. http://public.web.cern.ch/public/en/about/web-en.html accedido 06/09/2011 16

[30] A. Harris, C. McCulloh. *HTML, XHTML, and CSS All-in-One Desk Reference For Dummies*. Wiley Publishing. 2008 16

[31] Cascading Style Sheets, level 1. W3C. http://www.w3.org/TR/CSS1/ accedido 06/09/2011 17

[32] The Common Gateway Interface (CGI) Version 1.1. http://www.rfc-editor.org/rfc/rfc3875.txt. Octubre 2004 18

[33] ASP.NET. http://www.asp.net/ 18

[34] JavaServer Pages Technology. Oracle. http://www.oracle.com/technetwork/java/javaee/jsp/index.html accedido 06/09/2011 18

[35] The Perl Programming Language. http://www.perl.org/ 18

[36] Web Programming in Python. Python. http://wiki.python.org/moin/WebProgramming accedido 06/09/2011 19

[37] Ruby on Rails. http://rubyonrails.org 19

[38] PHP: Hypertext Preprocessor. http://www.php.net/ 19

[39] Open-Source PHP Web Framework. Symfony. http://www.symfony-project.org/ 19

[40] Zend Framework. http://framework.zend.com/ 19

[41] Applets. http://java.sun.com/applets/ 19

[42] Introducción a Javascript. Librosweb.es. http://www.librosweb.es/javascript/ accedido 07/09/2011 19

[43] jQuery. http://jquery.com/ 20

[44] VBScript. http://msdn.microsoft.com/en-us/library/t0aew7h6(v=vs.85).aspx 20

[45] ActionScript. Adobe. http://www.adobe.com/devnet/actionscript.html 20

[46] Ajax: A New Approach to Web Applications. Adaptative Path. http://adaptivepath.com/ideas/ajax-new-approach-web-applications accedido 07/09/2011

[47] PHP Manual. http://www.php.net/manual/en/ 38

[48] May 2011 Web Server Survey. Netcraft. http://news.netcraft.com/archives/2011/05/02/may-2011-web-server-survey.html accedido 09/09/2011 38

[49] TwitterOAuth by Abraham Williams. https://github.com/abraham/twitteroauth 62

[50] The OAuth 1.0 Protocol Specification. http://oauth.net accedido 19/10/2010 62

[51] OAuth code repository. Google Code. http://code.google.com/p/oauth/ accedido 25/10/2011 62

[52] Rate Limit. Twitter Developers. https://dev.twitter.com/docs/rate-limiting accedido 25/09/2011 62

[53] 960 Grid System. http://960.gs 64

[54] Variable Grid System. http://www.spry-soft.com/grids/ accedido 23/09/2011 64

[55] Flowplayer. http://flowplayer.org/index.html 65

[56] TGA format specification. http://paulbourke.net/dataformats/tga/ accedido 23/09/2011 70

[57] Twitter Looking For More Revenue, Advertising Moves One More Notch Up. FasTake. http://fastake.com/2011/07/looking-for-more-revenue-twitter-moves-advertising-one-more-notch-up/ accedido 25/09/2011 90

[58] Apache Hadoop. http://hadoop.apache.org/ 93

[59] Twitter Developers. https://dev.twitter.com/ 105