FACULTAD DE INFORMÁTICA

Traballo fin de grao
Grao en Enxenaría Informática

Mención en Enxeñería do Software

Sistema de grabación automática de la emisión en directo de una emisora de radio comunitaria con Software Libre

Autor: Veloso Abalo, Iago

Director: Casanova Crespo, José María

A Coruña, Septiembre 2014

A mi familia, y amigos.

Agradecimientos:

Quiero agradecer a mi director José María Casanova y a la gente de Cuac FM en especial a Fernando Souto, su apoyo y dedicación en la elaboración de este proyecto.

También quiero agradecer el apoyo mostrado en todo momento por mi primo Pablo, mi familia y mis compañeros y la confianza que han depositado en mí.

A todos ellos, muchas gracias.

Resumen:

Sistema de grabación automática de la emisión en directo de una emisora de radio comunitaria con Software Libre

Autor: Iago Veloso Abalo.
Director: José María Casanova Crespo.

En este proyecto clásico de ingeniería se desarrolla una aplicación web utilizando Python en conjunto con Django (entorno de desarrollo web), además de un programa de escritorio que se encargará de realizar la captura de audio mediante la entrada de micrófono y su posterior subida al servidor web.

El proyecto consiste en crear un sistema que simplifique la gestión de una radio comunitaria, facilitando la programación de su parrilla, la grabación automática de sus programas y la difusión de podcast e información de la gente que participa en los programas.

Este proyecto está destinado a crear una nueva herramienta de software libre. Al disponer en el ámbito de la Universidad de A Coruña de una radio comunitaria como Cuac FM se realizaron evaluaciones y se desarrolló teniendo en cuenta sus necesidades.

El sistema desarrollado se compone de dos partes, por un lado la aplicación web que será la parte con la que interactúen los usuarios y por otro un programa de grabación de audio que será instalado en la emisora y se comunicará con la aplicación web de manera automática.

El proyecto supuso el despliegue de la aplicación web en un entorno de producción junto con la instalación del programa grabador en una Raspberry Pi, un dispositivo económico y de prestaciones limitadas.

Palabras clave

- Radio Comunitaria
- Programación emisora
- CUAC FM
- Grabación audio
- Raspberry Pi
- Software Libre
- Django
- Python

Índice general

Índice de figuras

Capítulo 1

Introducción

1.1. Contexto

Una radio comunitaria es una estación de transmisión con fines no lucrativos que no se somete a la lógica del dinero, está caracterizada por la participación y la defensa de los intereses de la comunidad.

Este tipo de estaciones de radio se han convertido en una herramienta indispensable para el desarrollo de las comunidades. Debido a sus características muchas veces carecen de apoyo gubernamental y se financian de pequeños patrocinadores para su mantenimiento.

Las radios comunitarias actualmente cuentan con pocos o casi ningún medio que les permiten automatizar las actividades que tienen que llevar a cabo y se ven obligadas a realizar manualmente tareas repetitivas y redundantes.

Al ser organizaciones abiertas y participativas tienen un problema añadido respecto a las radios convencionales puesto que su número de miembros suele ser mayor y mucho más cambiante.

Además, por lo general, su personal carece de formación técnica lo cual hace más complicado el uso de programas que le ayuden a automatizar tareas.

A mayores el proyecto será publicado bajo una licencia de software libre debido a que la gran mayoría de radios comunitarias no podrían hacer frente al desembolso que

supondría una licencia privativa de los programas de este estilo.

Por todo ello, el objetivo de este proyecto es la creación de una herramienta libre que le permita facilitar estas labores a Cuac FM y a otras radios comunitarias puesto que en la actualidad no existe una herramienta centrada en esta problemática.

1.2. Situación actual de Cuac FM

Cuac FM[1] es una asociación que gestiona una emisora comunitaria situada en A Coruña. En sus inicios fue concebida por sus fundadores como una radio universitaria, una emisora dependiente de la universidad donde sus alumnos pudieran expresarse; sin embargo, la oposición de la Universidad provocó que sus miembros crearan una asociación juvenil independiente.

Con el tiempo, Cuac FM se fue transformando hasta convertirse en 1996 en una radio comunitaria que invita a la participación y promueve el impulso de este tipo de medios mediante la celebración y acogida de encuentros, tales como Encuentro14[2].

Figura 1.1: Cuac FM

Actualmente Cuac FM realiza la programación de la parrilla mensualmente en una hoja de cálculo, una vez acaban el laborioso proceso de asignar y recolocar los horarios

[1] http://cuacfm.org/
[2] http://encuentro14.org/

a los programas que se emitirán, le realizan una captura de pantalla y la cuelgan en su página web (Figura 1.2).

Esta parrilla lleva asociado un código de color que representa si se trata de un programa en directo, una retransmisión[3] o una redifusión[4] además del idioma en el que se emite.

Con ello proporcionan una información más o menos exacta de la programación de ese mes a sus oyentes, con lo cual si surge un imprevisto durante ese período tendrían que volver a realizar dicho proceso que en la mayoría de los casos queda sin corregir hasta el mes siguiente debido a la gran cantidad de trabajo que conlleva.

Figura 1.2: Programación Cuac FM

Una vez tienen la hoja de cálculo con las horas asignadas hay una persona encargada de introducir las horas y la duración de los programas en un script para que estos sean grabados.

[3]Contenido en diferido.
[4]Contenido externo

La grabación de audio está parcialmente automatizada lo que permite despreocuparse de la grabación de los programas una vez configurado. Sin embargo, el script grabador es independiente al resto del sistema y es necesario actualizarlo cuando la programación cambia editando una serie de ficheros escasamente intuitivos.

El script realiza la subida mediante el protocolo FTP a un servidor externo y ahí termina su función. Los socios del programa tienen acceso a esa web donde descargan el archivo de audio y si quieren distribuir el podcast de su programa, deberán buscarse algún servicio externo que se lo permita.

El presente proyecto se encargara de dar solución a estos problemas e integrar en un mismo sistema la programación de la parrilla, la grabación de los programas y la generación de podcast.

1.3. Objetivos

Los objetivos definidos para este proyecto son:

- Gestionar la programación de la parrilla de programas de radio de una emisora comunitaria, permitiendo su visualización asociada a la información de los programas.
- Sustituir el actual script grabador por un programa que cubra las nuevas necesidades de la organización.
- Realizar la grabación empleando la interfaz de entrada de sonido en base al horario de las emisiones.
- Gestionar las grabaciones y asociar los metadatos a los archivos de audio.
- Publicar en formato podcast las grabaciones de los programas.
- El software resultado dispondrá de una licencia de software libre compatible con la definición de la Free Software Foundation, para facilitar la reutilización del proyecto por otras emisoras comunitarias.

1.4. Estructura de la memoria

1. **Introducción**: Da una visión general sobre el contexto en el que se enmarca el proyecto, introduce la problemática a tratar, detalla el alcance y los objetivos del proyecto.
2. **Estado del arte**: Estudio de las aplicaciones existentes en el mercado.
3. **Herramientas y tecnologías utilizadas**: Detalle de las herramientas y tecnologías utilizadas en el transcurso del proyecto.
4. **Metodología**: En esta parte de la memoria se describe la metodología utilizada en la elaboración del proyecto.
5. **Desarrollo**: Se muestra la planificación temporal del proyecto junto con su desarrollo.
6. **Evaluación de costes**: Se lleva a cabo una estimación de su coste final.
7. **Diseño del sistema**: Se describe la estructura de la aplicación web y el programa grabador.
8. **Implementación**: Se expone información referente a la implementación del proyecto explicando las partes de mayor complejidad.
9. **Pruebas del sistema**: Se explica el proceso llevado a cabo en la elaboración de las pruebas.
10. **Conclusión y futuras líneas de trabajo**: Conclusiones sobre la realización del proyecto contando si se han cumplido los objetivos fijados y posibles mejoras o ampliaciones del sistema en el futuro.

Apéndices:

A. **Licencia**: Contiene un estudio sobre las dependencias que usa el proyecto y la licencia escogida.

B. **Manual de usuario**: Se trata de una guía para el usuario final que le permite aprender a usar el sistema de forma efectiva

C. **Instalación del sistema**: Se indica el software necesario para utilizar el sistema así como los pasos necesarios para su instalación.

D. **Glosario de Términos**: Definición de conceptos usados a lo largo de la memoria que son necesarios para entender el funcionamiento del sistema.

E. **Acrónimos**: Listado de siglas utilizadas a lo largo de la presente memoria.

F. **Bibliografía**: Listado de referencias bibliográficas consultadas para la realización del presente trabajo.

Capítulo 2

Estado del arte

Actualmente en el mercado no existen herramientas que abarquen toda la funcionalidad que pretende alcanzar este proyecto. La gran mayoría de herramientas existentes se centran en la transmisión en directo de audio, pero faltan otras que ayuden a la gestión de la programación de la parrilla e informen a los oyentes proporcionándoles contenido en diferido.

A continuación se muestra una serie de herramientas que proporcionan parcialmente la funcionalidad que se quiere alcanzar en este proyecto.

Airtime

Airtime es un software para la transmisión de audio que permite gestionar una estación de radio. Permite acceder y cargar archivos desde el propio navegador web además de proporcionar un calendario para editar la emisión de los programas. También ofrece la posibilidad de grabar audio en directo si se utiliza en conjunto con Ecasound en la misma máquina. Se distribuye bajo una licencia GNU.

Como desventajas este sistema no proporciona información a los oyentes. Serviría para gestionar la parrilla y automatizar la grabación.

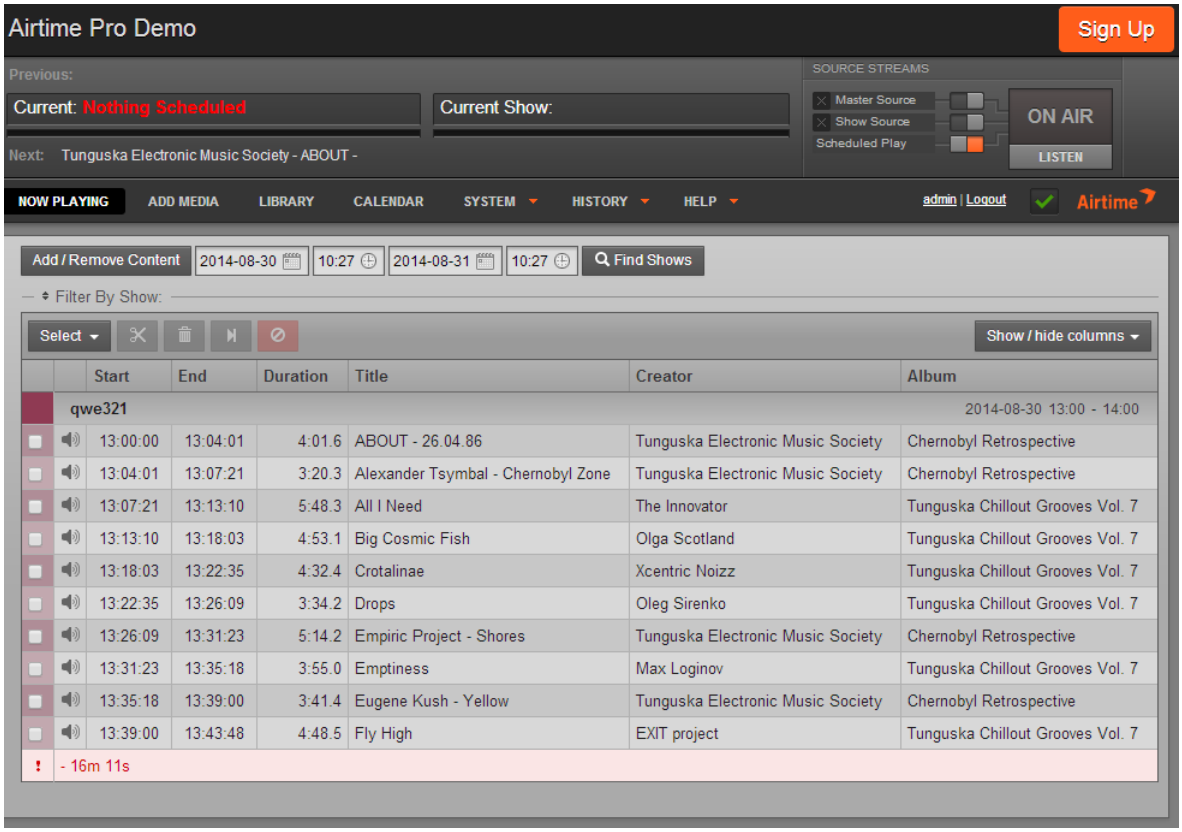

Figura 2.1: Airtime

Zara Radio

Es un automatizador de programación de emisoras de radio, su principal característica es la emisión automática de listas de reproducción. El programa se distribuye como freeware pero no es software libre y funciona únicamente sobre Windows, concretamente sólo es compatible hasta Windows XP.

El software tuvo una gran popularidad en su época y todavía se sigue usando, actualmente el desarrollo de este producto se ha reemplazado por una versión de pago.

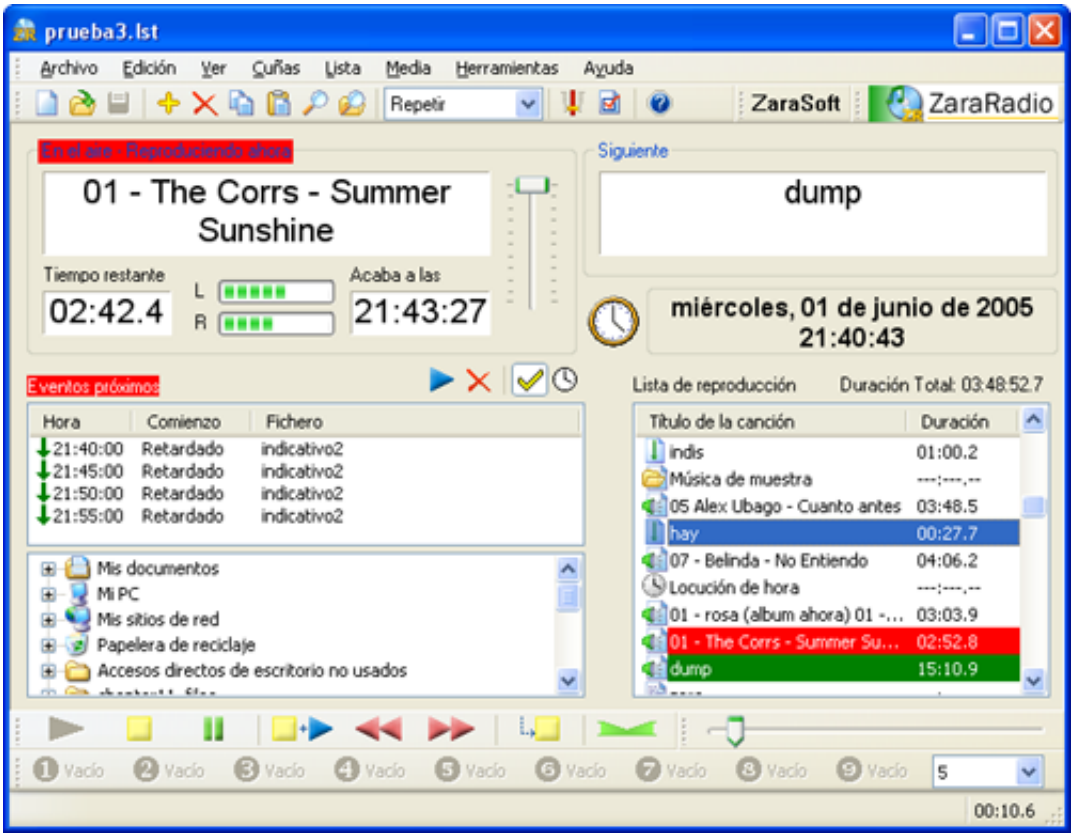

Figura 2.2: Zara Radio

iVoox

iVoox es una plataforma web donde los usuarios pueden escuchar, subir y compartir todo tipo de audios. Cuenta con una comunidad de oyentes y dispone de categorías para localizar programas de todo tipo.

El servicio es gratuito tanto para subir audios como para escucharlos aunque se ofrece la posibilidad de hacerse premium y eliminar la publicidad.

Las desventajas son la inclusión de publicidad y que el sistema sólo acepta ficheros de audio en formato MP3. Además para cuentas gratuitas la longitud del RSS se restringe a 20 elementos y los programas no pueden durar más de 2 horas.

Por tanto este sistema sólo valdría para la distribución del audio una vez grabado teniendo en cuenta las restricciones explicadas anteriormente.

Audacity

Audacity es software libre y está centrado en el manejo de archivos de audio, entre sus funciones más destacables esta la grabación mediante micrófono, importación y exportación de audios, edición y uso de efectos. Es multiplataforma y cuenta con una Licencia Pública General de GNU (GPL).

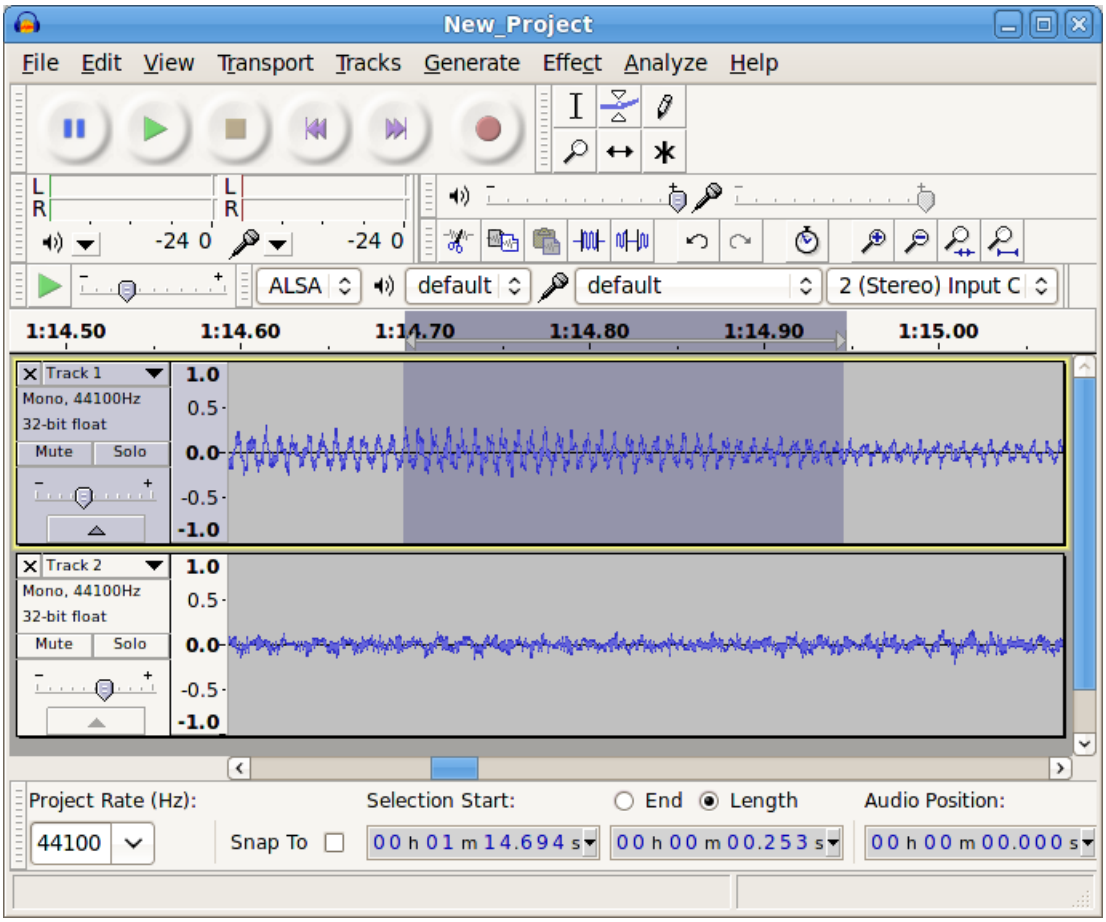

Figura 2.3: Audacity

Como contrapartida no soporta la programación de eventos y el soporte para scripts es experimental a día de hoy.

Ecasound

Ecasound es un paquete libre de herramientas destinado para procesar audio, puede usarse para reproducir, grabar y convertir audio. No cuenta con interfaz gráfica y permite su uso a través de línea de comandos.

Conclusión

Analizadas las diferentes opciones existentes en el mercado, queda manifestada la necesidad de un sistema informático que agrupe la gestión de las parrillas de programación, la grabación de los programas y la gestión de emisiones en un único software.

Capítulo 3

Herramientas y tecnologías utilizadas

Para la elaboración de este proyecto se han utilizado herramientas y tecnologías libres debido a que el objetivo principal es desarrollar una nueva herramienta que facilite la reutilización de este proyecto por otras emisoras comunitarias; por todo ello, es necesario que este sistema cuente con una licencia de software libre.

3.1. Plataformas de desarrollo de aplicaciones web

Una aplicación web es cualquier aplicación software que funciona en un navegador web y por lo tanto tiene que estar en un lenguaje conocido por el navegador. Se suele enviar una combinación de JavaScript, HTML y CSS al navegador para que lo procese. Esto no significa que el programador tenga forzosamente que usar dichas tecnologías para desarrollar la aplicación si no que el resultado final debe ser proporcionado en dichos lenguajes.

Existen múltiples tecnologías y plataformas para el desarrollo de aplicaciones Web. Las que presentan una mayor relevancia hoy en día son Django, Java EE, .NET, PHP y Ruby on Rails.

A pesar de que Django y Ruby on Rails son más recientes, todas estas plataformas tienen un muy buen grado de madurez, lo que les permite acometer la mayoría de proyectos Web actuales, cada uno tiene unas características que lo harán más apropiado para un tipo determinado de aplicación y situación de negocio.

Veamos una breve introducción a cada uno:

Django

Este framework utiliza Python como lenguaje de programación en la vista y el modelo. Python cuenta con tipado dinámico y una sintaxis fácil de comprender. Usado en conjunto con Django hace que facilite un desarrollo rápido, limpio y sencillo.

Java

Java es la plataforma más extendida en el entorno corporativo. Se trata de una tecnología muy madura y popular que cuenta con innumerables librerías de todo tipo. Los IDEs[1] más usados son Eclipse o Netbeans, ambos muy potentes y de código abierto.

PHP

PHP (acrónimo recursivo de Hypertext Pre-processor) es de la misma época pero a diferencia de Java, estaba pensado desde el principio como un lenguaje que se pudiera incorporar en documentos HTML. La gran ventaja de PHP es que resulta sencillo empezar con él y existe mucha documentación online.

.NET

.NET se destaca por la utilización de varios lenguajes de programación, mientras que Django, Java EE, PHP y Ruby on Rails sólo permiten un único lenguaje. Es ampliamente usado en el panorama de software empresarial y al ser de Microsoft cuenta con una alta integración con sus productos.

En contrapartida el entorno de desarrollo también es de su propiedad y es necesario utilizar su sistema tanto para el desarrollo como para la puesta en funcionamiento del servidor, no siendo ninguno de ellos software libre.

[1] Entornos de desarrollo

Ruby on Rails

Al igual que Django se trata de un framework que se pensó desde el principio para el diseño de aplicaciones web. Actualmente Django y Ruby on Rails ofrecen prácticamente las mismas prestaciones y no hay unos criterios muy definidos para escoger uno u otro.

3.2. Selección del framework

Debido a la metodología que utilizamos y el tipo de proyecto nos decantamos por utilizar Django.

Este entorno nos proporciona la rapidez que necesitamos para implementar las necesidades del cliente sin perder el tiempo en la configuración como ocurre con otras tecnologías.

Además al ser desarrollado en Python disponemos de la ventaja de utilizar un mismo lenguaje para la aplicación web y el programa grabador.

A continuación se describe todas las propiedades de este framework en detalle:

Django

Django es un framework web de alto nivel para Python que fomenta un desarrollo rápido y limpio, se centra lo máximo posible en la automatización y sigue el principio DRY[2] que hace énfasis en evitar la repetición de tareas y código. Sus puntos fuertes son:

- Dispone de un ORM[3] con una completa API que te abstrae de la base de datos permitiendo crear, consultar, actualizar y borrar objetos.
- Crea automáticamente una interfaz administrativa que permite a usuarios autenticados añadir, cambiar y borrar objetos.
- Muestra URLs limpias y elegantes gracias al uso de expresiones regulares que facilitan el SEO[4].

[2] Don't Repeat Yourself.
[3] Object-relational mapping.
[4] Search Engine Optimization.

- Utiliza un patrón parecido a MVC[5], llamado Model Template View[6] que separa la lógica de la vista.
- Provee múltiples sistemas de protección frente a ataques.
- Tiene un robusto sistema de internacionalización que facilita las tareas de traducción y formateado de fechas y números según las diferentes regiones del mundo.
- Es open source y cuenta con una licencia BSD.

3.3. Otras herramientas y tecnologías utilizadas

Python

Python[7] es un lenguaje de programación interpretado que usa tipado dinámico, orientación a objetos y es multiplataforma, posee una licencia de código abierto compatible con la licencia pública general de GNU.

Gunicorn

Gunicorn es un servidor multihilo WSGI[8] HTTP abierto para Python, es compatible con varios web frameworks de forma nativa, Django entre ellos.

Nginx

Nginx[9] es un servidor web/proxy inverso ligero multiplataforma de alto rendimiento, entre sus múltiples características cuenta con un servidor de archivos estáticos, se distribuye bajo una licencia BSD.

[5] Model–view–controller.
[6] El controlador es manejado principalmente por el framework.
[7] https://www.python.org/
[8] Web Server Gateway Interface
[9] http://nginx.org/

Virtualenv

Virtualenv[10] es una herramienta para la creación de entornos virtuales de Python, un entorno virtual es un espacio independiente de los paquetes instalados en el sistema, por lo que permite tener diferentes librerías instaladas sin afectar al resto del sistema.

Pip

Pip[11] es un gestor de paquetes de Python que permite instalar, actualizar y borrar paquetes. Es ampliamente utilizado junto a Virtualenv.

Redmine

Redmine[12] es una herramienta para la gestión de proyectos, incluye una serie de herramientas de las que podemos destacar: seguimiento de errores, calendarios de actividades y diagramas de Gantt. Es código abierto y está disponible bajo una licencia GNU.

Git

Git[13] es un sistema de control de versiones diseñado por Linus Torvald. Se trata del gestor de versiones moderno más empleado y cuenta con una gran comunidad de desarrolladores a su alrededor lo que favorece la integración de Git con múltiples herramientas. Es distribuido bajo una licencia GNU.

GitHub

GitHub[14] es un portal que ofrece servicios de hosting para proyectos usando Git. Ofrece cuentas gratuitamente a proyectos de tipo open source y es ampliamente utilizado por la comunidad.

[10] http://virtualenv.pypa.io/
[11] http://pip.pypa.io/
[12] http://www.redmine.org/
[13] http://git-scm.com/
[14] http://github.com/

JSON

JSON[15] es un formato ligero para el intercambio de datos y un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.

La simplicidad de JSON ha dado lugar a la generalización de su uso, especialmente como alternativa a XML en AJAX. Una de las supuestas ventajas de JSON sobre XML como formato de intercambio de datos en este contexto es que es mucho más sencillo escribir un analizador sintáctico (parser) de JSON. En JavaScript, un texto JSON se puede analizar fácilmente usando la función eval(), lo cual ha sido fundamental para que JSON haya sido aceptado por parte de la comunidad de desarrolladores AJAX, debido a la ubicuidad de JavaScript en casi cualquier navegador web.

CSS (Cascading Style Sheets)

CSS [2] es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML). El W3C (World Wide Web Consortium) es el encargado de postular las especificaciones de las hojas de estilo que servirán de estándar para los navegadores. La idea que se encuentra detrás del desarrollo de CSS es separar la estructura del documento de su presentación.

StarUML

StarUML[16] es una herramienta UML open source que permite crear todo tipo de diagramas.

Dia

Dia[17] es una aplicación informática de propósito general para la creación de diagramas, desarrollada como parte del proyecto GNOME. Está concebido de forma modular, con diferentes paquetes de formas para diferentes necesidades. Actualmente se incluyen

[15] http://json.org/
[16] http://staruml.io/
[17] http://dia-installer.de/

entre otros, diagramas entidad-relación, diagramas UML, diagramas de flujo o diagramas de redes.

Eclipse

Eclipse[18] es un IDE de código abierto y multiplataforma para el desarrollo de proyectos software. Está compuesto por un conjunto de herramientas que proporcionan al programador un entorno adecuado para su trabajo, incorporando editor de código con resaltado de sintaxis, compilador, pruebas unitarias, control de versiones, etc. Además Eclipse empleada módulos (plugins en inglés) para proporcionar toda su funcionalidad a diferencia de otros entornos donde todas las funcionalidades están incluidos, las necesite el usuario o no.

FileZilla

FileZilla[19] es un cliente FTP[20] multiplataforma de código abierto y software libre, licenciado bajo la licencia pública general de GNU. Soporta los protocolos FTP, SFTP y FTP sobre SSL/TLS (FTPS).

GIMP

GIMP[21] (GNU Image Manipulation Program) es un programa de edición de imágenes digitales en forma de mapa de bits, tanto dibujos como fotografías. Es un programa libre y gratuito que forma parte del proyecto GNU.

HTML

HTML [9], siglas de HyperText Markup Language, es el lenguaje de marcado predominante para la elaboración de páginas web. Es usado para describir la estructura y el

[18] http://www.eclipse.org/
[19] http://filezilla-project.org/
[20] File Transfer Protocol.
[21] http://www.gimp.org/

contenido en forma de texto, así como para completar el texto con objetos tales como imágenes. HTML se escribe en forma de etiquetas rodeadas por corchetes angulares (<, >).

Javascript

JavaScript [10][11] es un lenguaje de programación interpretado. Se define como orientación a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Se utiliza principalmente del lado del cliente implementado como parte de un navegador web, permitiendo mejoras en la interfaz de usuario y páginas web dinámicas. Todos los navegadores modernos interpretan el código de JavaScript integrado en las páginas web. Para interactuar con una página web se provee al lenguaje de JavaScript de una implementación del DOM[22].

JQuery

JQuery[23] es una biblioteca o framework de JavaScript que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, general animaciones y agregar iteracción con la técnica AJAX a páginas web. JQuery, al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menor tiempo y espacio.

Twitter Bootstrap

Twitter Bootstrap[24] es un framework para el desarrollo de interfaces web basado en HTML, CSS y con extensiones opcionales en JavaScript.

[22] Document Object Model.
[23] http://jquery.com/
[24] http://getbootstrap.com/

LaTeX

Latex[25] es un sistema de composición de textos orientado especialmente a la creación de libros, documentos científicos y técnicos que contengan fórmulas matemáticas. La idea principal de Latex es que el autor se centre en el contenido y no en la forma del documento. Para lograr esto, Latex está provisto de una serie de macros predefinidos.

Modelo Entidad-Relación

Un diagrama o modelo Entidad-Relación (denominado también modelo E-R) es una herramienta para el modelado de datos de un sistema de información. Estos modelos expresan entidades relevantes para el sistema así como sus interrelaciones y propiedades.

Chromium

Chromium[26] es un navegador web de código abierto bajo una licencia BSD a partir del cual se basa el código fuente de Google Chrome.

Mozilla Firefox

Mozilla Firefox[27] es un navegador libre y de código abierto descendiente de Mozilla Aplicattion Suite y desarrollado por la Fundación Mozilla.

MySQL

MySQL[28] es un sistema de gestión de bases de datos relacional, multihilo y multiusuario. MySQL AB (subsidiaria de Sun Microsystems y esta a su vez de Oracle Corporation) desarrolla MySQL como sofware libre en un esquema de licenciamiento dual.

[25] http://www.latex-project.org/
[26] http://www.chromium.org/
[27] http://www.mozilla.org/
[28] http://www.mysql.com/

OpenOffice.org Writer

OpenOffice.org Writer[29] es el procesador de textos de la suite informática OpenOffice.org. Permite exportar archivos de texto a los formatos PDF y HTML sin software adicional, posibilitando su uso como un editor WYSIWYG[30].

TexStudio

TexStudio[31] es un editor gratuito distribuido bajo licencia GPL, para escribir documentos de texto, multiplataforma, que integra muchas herramientas necesarias para desarrollar documentos con Latex, en una sóla aplicación. TexStudio incluye soporte Unicode, correción ortográfica, auto-completado y un visor incorporado en PDF[32].

Ubuntu

Ubuntu[33] es un sistema operativo que utiliza un núcleo Linux y su origen está basado en Debian. Al igual que otros sistemas operativos GNU/Linux, está compuesto de múltiple software normalmente distribuido bajo una licencia libre o de código abierto.

Wordpress

WordPress[34] es un sistema de gestión de contenido o CMS[35] enfocado a la creación de blogs. Se distribuye bajo una licencia GPL.

[29] http://www.openoffice.org/
[30] What You See Is What You Get.
[31] http://texstudio.sourceforge.net/
[32] Portable Document Format.
[33] http://www.ubuntu.com/
[34] http://wordpress.org/
[35] Content Management System

Capítulo 4

Metodología

Hemos utilizado una metodología agil basada en el desarrollo iterativo e incremental aplicando algunas prácticas de XP[1] aplicables a un trabajo individual:

En el desarrollo iterativo e incremental el software no se elabora todo de una sóla vez, se divide en partes, estas partes reciben el nombre de iteraciones, en cada iteracion se realizan una serie de actividades que producen un incremento del producto desarrollado que progresivamente va añadiendo o mejorando funcionalidades del sistema.

XP [21] [22] se basa en los siguientes valores[2]:

- **Comunicación**: Clientes, desarrolladores, directores del proyecto trabajan en una misma habitación. En nuestro caso al ser una única persona la que realiza el proyecto esto se cumple, sin embargo no era posible que el cliente estuviera físicamente presente durante todo el desarrollo, en caso de surgir alguna duda o problema realizábamos la comunicación vía internet.

- **Simplicidad**: Se simplifica el diseño para agilizar el desarrollo y facilitar el mantenimiento, esto hace necesario llevar a cabo refactorizaciones de código a medida que el código crece. También se aplica a la documentación, evitando proporcionar información innecesaria o de escaso valor, para ello se deben elegir adecuadamente los nombres de las variables, métodos y clases.

[1] eXtreme Programming.

[2] http://www.extremeprogramming.org/values.html

- **Retroalimentación**: El cliente está involucrado en todo el desarrollo del proyecto, al realizar el proyecto en ciclos cortos se conoce su opinión en tiempo real minimizando el tener que rehacer partes que no cumplen los requisitos y ayuda a priorizar los de más valor para el cliente.
- **Respeto**: Todos los miembros del equipo dan y reciben el respeto que merecen. Los desarrolladores respectan la experiencia de los clientes y viceversa.
- **Coraje o valentía**: Esta técnica consiste en diseñar y codificar para el presente no para el futuro, esto intenta evitar la pérdida de tiempo empleado en diseños muy severos e implementaciones pensando en cosas que podrían llegar a ocurrir. También permite a los desarrolladores libertad para refactorizar el código cuando lo crean necesario permitiendo desechar código obsoleto sin importar el esfuerzo invertido en él.

En cuanto a las prácticas que propone XP se han seguido:

- **Historias de usuario**: Una historia de usuario es un recordatorio de una conversación sobre una característica del software recogida de la información del cliente. En cada reunión se recogieron estas historias de usuario y se introdujeron en redmine.
- **Ciclos cortos**: La planificación de iteraciones cortas es imprescindible en nuestro caso para mantener al cliente informado del estado del producto y nos permite planear y priorizar la funcionalidad que se va a entregar desarrollando las partes de mayor valor para él.
- **Integración continua**: Este proceso tiene como objetivo comprobar que cada actualización del software no genere problemas, para ello se unen los cambios y se ejecutan las pruebas.
- **Diseño simple**: Elegir el camino más fácil primero evitando hacer el diseño especulando posibles cambios futuros.
- **Refactorizar**: A medida que se desarrolla el código tiende a desordenarse, por eso es necesario ir realizando refactorizaciones.
- **Abrazar el cambio**: Realizar los cambios si son necesarios. Por lo general cuando hay que cambiar o reemplazar una parte del sistema que ya está desarrollada se suele evitar modificarla debido a que sería desechar trabajo.

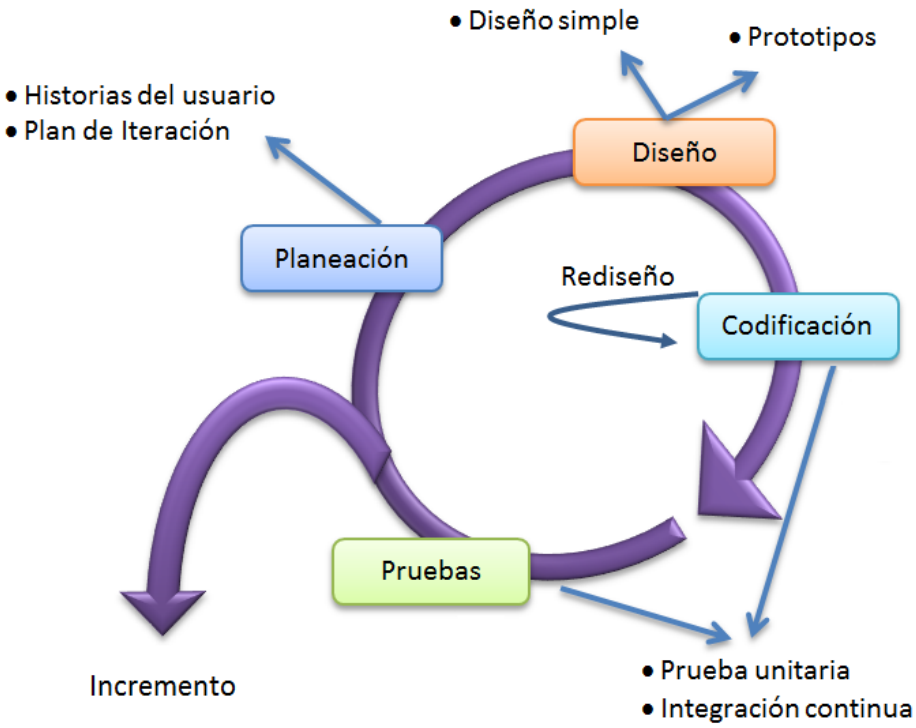

Figura 4.1: Etapas de XP

La toma de requisitos se ha realizado con el cliente, se han planificado iteraciones cortas en la medida de lo posible para tenerlo involucrado evitando que esto le ocasione algún trastorno y poder añadir funcionalidad priorizada según sus intereses. Para ello tras cada iteración se despliega la aplicación en un entorno de producción y se recoge el feedback por parte de los posibles usuarios, en nuestro caso de Cuac FM.

Una vez recogidos los resultados se analizan, priorizando y planificando los objetivos de la próxima iteración. Cabe señalar que todos los datos se han ido introduciendo en redmine a lo largo del desarrollo, por tanto contamos con datos reales.

Capítulo 5

Planificación y ejecución del proyecto

En este capítulo se describirá el proceso del proyecto, los problemas que surgieron y las soluciones llevadas a cabo.

5.1. Definición de las iteraciones

Teniendo en cuenta que el trabajo fin de grado está comprendido por un total de 12 créditos ECTS y cada uno supone unas 25-30 horas de carga de trabajo. Se realizó la estimación de la siguiente forma:

12 créditos * 30 horas/crédito = 360 horas

Se establecieron alrededor de 60 horas por iteración dando a lugar a 6 iteraciones. Tras cada iteración se realizó una reunión donde se recogió el feedback del cliente, y se priorizaron requisitos para el siguiente incremento.

Una vez completada la iteración se llevó a cabo un seguimiento y se tomaron medidas correctivas en el caso de que fueran necesarias.

5.2. Flujo de trabajo en Git

A lo largo del desarrollo hemos mantenido dos ramas principales, una rama en Git es un apuntador a un commit (una confirmación de cambios) que se va moviendo conforme

vamos añadiendo commits a esa rama.

Las ramas principales de nuestro proyecto son la master o principal y la develop o desarrollo. La rama principal refleja el estado en producción, los cambios son realizados sobre la rama de desarrollo y cuando alcanza un punto estable y está listo para ser publicado todos los cambios se pasan a la rama principal. Además dependiendo de la funcionalidad que vamos a agregar se crea una nueva rama.

Figura 5.1: Flujo de trabajo en Git - develop y master

5.3. Reunión inicial

Se celebró una reunión inicial con el cliente para que nos pusiera al tanto de las necesidades que tiene la organización Cuac FM y cómo funcionaba el sistema que tenían en aquel momento. La Figura 5.2 representa un diagrama hecho a partir de un boceto proporcionado por el cliente.

5.3.1. Infraestructura

En ese momento la organización contaba con una maquina corriendo una versión de Windows XP y un script que realizaba la grabación del audio de los programas, su manejo no era muy intuitivo puesto que para cambiar los horarios de grabación era necesario editar unos ficheros de texto de donde leía la información necesaria para la grabación de las emisiones. Además sólo grababa una hora o dos según el tipo de programa.

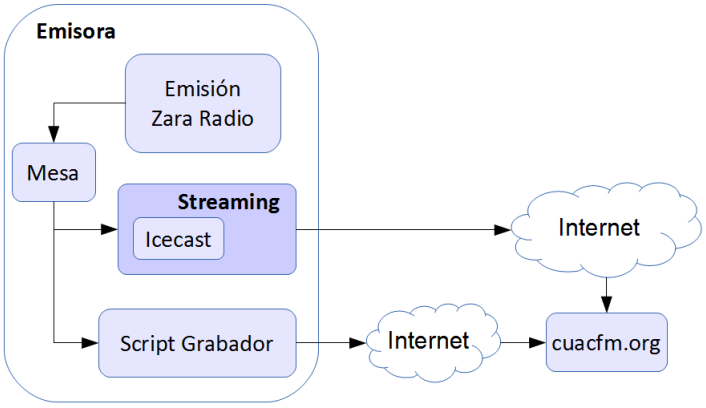

Figura 5.2: Estructura Cuac FM

Se evaluó que era inviable reaprovechar el programa puesto que su código fuente se había perdido y además era necesario reiniciar el equipo cada determinado tiempo debido a un bug de memoria.

El cliente también nos informó que junto con el programa convivían varios virus y que prefería cambiar el sistema operativo. Se acordó que el software grabador correría sobre alguna distribución de GNU/LINUX.

5.3.2. Usuarios

En Cuac FM se distinguen cuatro tipos de usuarios, los pertenecientes al departamento de tecnología, los pertenecientes al departamento de contenidos, los socios del programa y los usuarios no autentificados.

Los usuarios del departamento de tecnología se encargan de los aspectos técnicos de configuración de la aplicación, además de ser los encargados de solucionar cualquier problema relacionado con el funcionamiento. Tendrían por tanto el papel de administrador.

Los responsables de contenidos tienen como objetivo principal la gestión de la parrilla y la creación de usuarios socios de programa.

Los usuarios del programa tienen menos privilegios que los anteriores y sólo podrán editar su información personal y la de los programas en los que participen.

Por último los usuarios no autentificados carecen de privilegios y sólo tendrán acceso

a la parte pública de la web, en donde podrán ver la programación de la emisora así como la información de los programas junto con su podcast y la información de las personas que colaboran en los programas.

Todo ello hace ver que las organizaciones comunitarias pueden llegar a tener varios roles de usuarios, como la idea de esta herramienta es que funcione en cualquier tipo de organización se tomó la decisión de diseñarla de tal manera que el cliente final pudiera definir los roles que quisiera. Esto se consiguió permitiendo crear grupos de usuarios dentro de la aplicación y asignando permisos a estos grupos.

5.3.3. Aspectos importantes

Con respecto a los objetivos generales del proyecto el cliente recalcó la importancia de que el software debía ser usado por personal sin cualidades técnicas y que debía cubrir las funcionalidades del sistema actual.

5.4. Primera iteración

Para la primera versión se planificó la creación del esqueleto básico de la web y el despliegue en un entorno de producción, para ello se usó un servidor gratuito durante un año en Amazon Web Services. Para alojar el proyecto se optó por GitHub puesto que es uno de los sitios más usados para proyectos open source.

5.4.1. Seguimiento

El desarrollo se completó en 65.60 horas, 5.6 horas más de lo establecido, no se tomó ninguna medida puesto que era un tiempo aceptable.

5.4.2. Reunión

En la reunión se revisó el trabajo realizado y se acordaron llevar a cabo ciertos cambios referentes a la gestión del horario de los programas.

Figura 5.3: Iteración 1

Era necesario que los programas estuvieran relacionados de algún modo con su lista de horarios para hacer más fácil la gestión de la parrilla de programación. Se acordó mejorar este aspecto lo máximo posible puesto que era de vital importancia que la asignación de horarios se pueda hacer de una manera cómoda.

A continuación se muestran las historias de usuario recogidas priorizadas en orden descendente:

- Facilitar la gestión de la parrilla de programación.
- Mostrar el horario de emisión de los programas.
- Permitir que un usuario pueda estar en un programa bajo un rol.
- Generar las fechas de emisión de los programas.
- Hacer que cada episodio tenga participantes.
- Permitir que los programas tengan temporadas.
- Crear un rol de socio administrador que pueda añadir y quitar gente de los programas
- Poder añadir información a priori en episodios
- Los episodios pueden cancelarse.

Se planearon las 4 primeras para la siguiente iteración y se concretó la fecha de la próxima reunión en la que asistirían además del cliente el resto de usuarios de Cuac FM.

5.5. Segunda iteración

Esta iteración tenía como objetivo cambiar la creación de la parrilla de programación, para que fuera más amigable con el usuario se usó un plugin de jQuery llamado FullCalendar[1] que proporciona un calendario y la posibilidad de arrastrar y soltar eventos. Además se tenían que visualizar los próximos programas a emitir e implementar el tipo de usuario socio de programa. Este tipo de usuario tiene privilegios limitados que lo restringen a editar su perfil y los programas y episodios en los que aparece.

Se tomó la decisión de crear una página de administración personalizada puesto que la zona de administración por defecto de Django no dispone de estas opciones, en cambio aprovechamos su sistema de permisos para poder crear nuevos roles dinámicamente y con ello permitir que el software se adapte a cualquier tipo de organización, no sólo a Cuac FM.

Figura 5.4: Iteración 2

[1] http://arshaw.com/fullcalendar/

5.5.1. Seguimiento

Aparecieron problemas durante el transcurso de la iteración, principalmente debido a la incorporación de FullCalendar y ello conllevó el retraso de la tarea de "creación automática de las emisiones", se decidió invertir más horas y posponer la entrega de esta funcionalidad para la siguiente iteración.

5.5.2. Reunión con los usuarios

Esta reunión tuvo lugar en las instalaciones de la emisora y se realizó una pequeña exposición del producto y de las funciones que llegaría a tener. El esfuerzo invertido en la creación de la herramienta de edición visual de la parrilla de programación valió la pena y se convirtió en una de las características más deseadas del software.

Para la próxima iteración se acordó realizar el desarrollo del programa grabador, además de los usuarios se recogieron las siguientes historias:

- Poder almacenar varias configuraciones de parrillas
- Permitir clonar parrillas
- Añadir un buscador y filtros para facilitar la búsqueda donde haya un gran número de elementos.
- Separar los programas que están actualmente en emisión de los que terminaron

También se discutió la posibilidad de presentar el proyecto en un congreso que se iba a celebrar en Coruña, donde Cuac FM era uno de los organizadores. Este congreso denominado Encuentro14[2] reuniría no solo a las radios comunitarias de España si no de Europa.

5.6. Tercera iteración

En esta iteración se llevó a cabo el desarrollo del programa grabador, el lenguaje elegido fue Python puesto que la aplicación web también se estaba desarrollando en este

[2]http://encuentro14.org/

leguaje.

5.6.1. Programa grabador

Primero debíamos encontrar las librerías necesarias para capturar el audio de la entrada de sonido, se llevó a cabo una búsqueda y análisis de librerías para la grabación de audio, se encontraron las siguientes:

- **PyMedia:** Una librería muy completa que acepta múltiples formatos de audio entre los que se destacan OGG y MP3. Está disponible en Linux y Windows. Sin embargo este proyecto está abandonado siendo su última actualización en 2006 y aparecieron incompatibilidades a la hora de la instalación.
- **PyAudio:**Proporciona una interfaz a PortAudio, que es una librería multiplataforma y funciona sobre Windows, Macintosh OS X, y Unix. Es código abierto y se distribuye sobre una licencia MIT.
- **Pyalsaaudio:** Provee una api para acceder a ALSA[3]. Funciona únicamente en Linux y cuenta con una licencia libre, concretamente PSF[4], la misma usada en la mayoría de las distribuciones de Python.

En un primer momento la opción elegida fue PyAudio por ser multiplataforma y contar con actualizaciones recientes junto con una buena documentación. Sin embargo esta librería resulto que grababa únicamente en formato wav[5] un formato de audio digital sin comprensión dando lugar a tamaños de archivo exageradamente grandes, además de ser un formato en propiedad de Microsoft.

Se decidió entonces delegar en un programa externo a Python la captura de audio, avconv[6] permite grabar audio en múltiples formatos incluyendo OGG, además de la posibilidad de incluir metadatos en el archivo de grabación. La librería se distribuye bajo una licencia LGPL[7]

[3]Advanced Linux Sound Architecture.
[4]Python Software Foundation.
[5]Waveform Audio File Format
[6]Avconv se reemplazó en la iteración 6 por cuestiones de rendimiento.
[7]GNU Lesser General Public License.

En el programa grabador se creó un archivo de configuración con múltiples opciones personalizables para permitir la configuración a usuarios con pocos conocimientos técnicos, entre las opciones del programa de grabación se incluye la llamada a avconv por lo que es posible cambiar el formato o la calidad del audio a grabar sin necesidad de modificar ninguna línea de código.

Figura 5.5: Iteración 3

5.6.2. Seguimiento

Debido a los problemas encontrados en el desarrollo del programa de grabación se volvió a tomar la decisión de invertir más horas y aplazar funcionalidad para la siguiente iteración.

5.6.3. Exposición en Encuentro14

Este encuentro celebrado el 11, 12 y 13 de abril en A Coruña fue organizado por Cuac FM y ReMC[8] y albergó tres eventos:

- **Jornada Europea de Redes de Medios Comunitarios**: Representantes de las principales redes Europeas se reunieron en este evento. Solamente estas redes representan a 1104 medios comunitarios.

[8]Red de Medios Comunitarios

- **Encuentro de la Red de Medios Comunitarios**: La ReMC[9] es la entidad más representativa del sector en el Estado. Cuenta con cerca de 40 entidades asociadas y celebrará en A Coruña su Asamblea General.
- **Reunión del European Board de AMARC**[10]: El Consejo Europeo de la Asociación Mundial de Radios Comunitarias (AMARC) celebró su reunión anual en A Coruña. AMARC agrupa a 4.000 miembros y asociados en más de 130 países.

En la exposición se presentó el producto desarrollado hasta la fecha y se explicaron las funcionalidades que tendría. El público mostró interés por saber cómo funcionaba y cuando estaría disponible.

Figura 5.6: Encuentro 14

Tras esta exposición se vio la necesidad de crear una web que contenga toda la información del producto así como la de buscar un nombre para el sistema.

[9] http://www.medioscomunitarios.net/
[10] http://amarceurope.eu/

5.7. Cuarta iteración

En la reunión de la iteración 2 se contempló la necesidad de cambiar el sistema actual de programación de la parrilla puesto que hasta el momento sólo permitía una única parrilla y el cliente deseaba tener guardadas varias y la posibilidad de crear y clonar nuevas.

Por lo tanto esta iteración abordaba la comunicación del programa grabador con la aplicación web y la generación de un podcast por programa junto con este nuevo cambio. A mayores se pensaría el nombre del producto y se pondría en funcionamiento su web.

5.7.1. Comunicación entre aplicaciones

Para la comunicación del programa grabador con la aplicación web se optó por el uso de un token de autentificación, la aplicación web proporciona este token que debe introducirse en las opciones de configuración del programa grabador la primera vez que se usa. Con esto evitamos que cualquier usuario pueda enviar información que debería ser proporcionada únicamente por el programa grabador, aunque para mayor seguridad debe consultarse sobre https.

5.7.2. Generación del podcast

Para poner en funcionamiento un podcast tuvimos que crear un canal RSS (un archivo XML) que fuera compatible con la especificación RSS 2.0[11], y tuviera etiquetas recomendadas por iTunes[12]. Puesto que cada programa contiene un canal RSS compatible con iTunes es posible darlos de alta para que aparezcan en la iTunes Store. En la Figura 5.7 vemos uno de los programas en la aplicación iTunes.

5.7.3. Elección del nombre y puesta en marcha de la web

El nombre elegido para el sistema fue **RadioCo** y para alojar la web se decidió contratar un servidor y un dominio. Para presentar el contenido de la web se optó por instalar

[11] http://cyber.law.harvard.edu/rss/rss.html
[12] http://www.apple.com/es/itunes/podcasts/specs.html

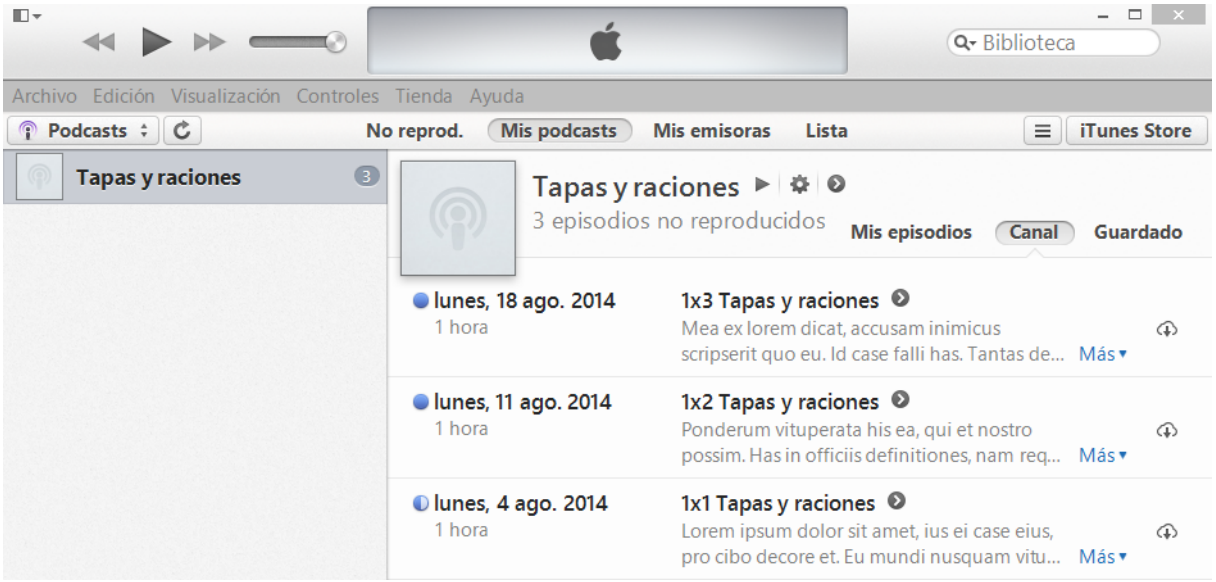

Figura 5.7: Vista previa del podcast con iTunes

WordPress[13], un sistema de gestión de contenido o CMS[14] que proporciona una interfaz gráfica para su instalación y gestión.

La página web está accesible en www.radioco.org y en la Figura 5.8 se muestra una vista previa.

5.7.4. Seguimiento

Los problemas encontrados principalmente en la generación del podcast dieron lugar a nuevos retrasos en la realización de la iteración. Esta vez no se pospuso la entrega de nueva funcionalidad y la opción tomada fue la asignación de más horas al único recurso del que dispone el proyecto.

Además en este momento se tomó la decisión de posponer la entrega y aprovechar el tiempo extra para añadir nuevas funcionalidades.

[13] http://wordpress.org/
[14] Content Management System

Figura 5.8: Página web de RadioCo

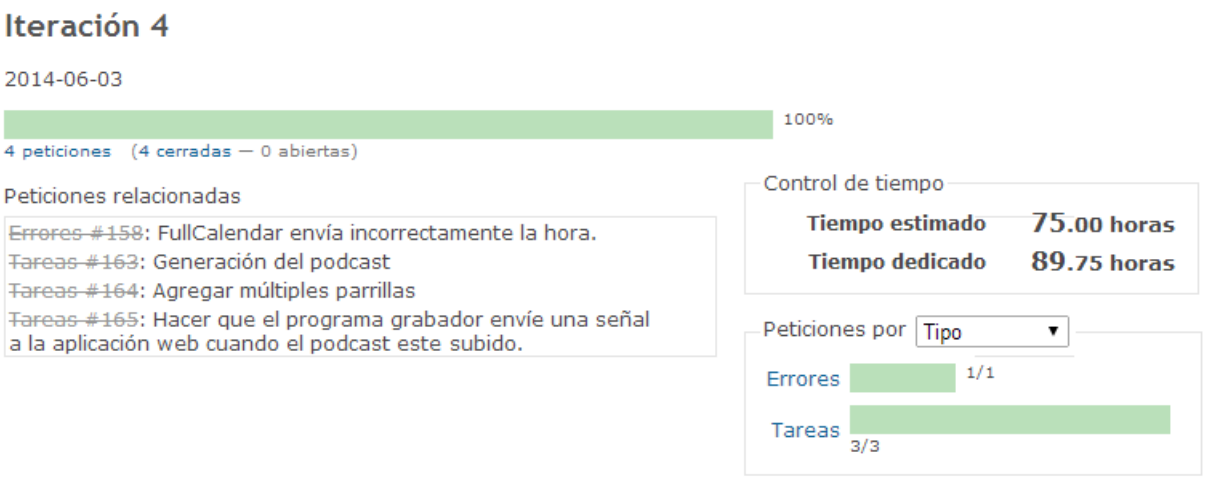

Figura 5.9: Iteración 4

5.8. Replanificación

Tras la cuarta iteración llevábamos realizadas un total de 285,75 horas frente a las casi 240 horas estimadas.

Esta desviación se había solucionado principalmente trabajando más horas, a cambio se había entregado a tiempo la funcionalidad requerida pero puesto que se trataba de un proyecto final de carrera se decidió posponer la entrega para mejorar el sistema, por ello se amplió el alcance del proyecto añadiendo unos nuevos objetivos:

- **Mejorar el sitio de administración para los usuarios registrados:** La zona de administración que tenían estos usuarios era funcional pero carecía de ayudas tales como la búsqueda y el filtrado de elementos.
- **Poder ver la programación de la semana entera:** Actualmente la interfaz sólo mostraba día por día los programas que se iban a emitir.
- **Desplegar el sistema en un sistema empotrado:** Adquirir hardware y desplegar en él el programa grabador.

Esta nueva funcionalidad se repartió en 2 nuevas iteraciones de unas 75 horas cada una, debido a que coincidía con el periodo de vacacional era muy posible que el cliente no estuviera disponible para realizar a cabo las reuniones tras cada iteración.

5.9. Quinta iteración

En esta iteración se llevó a cabo la mejora de la parte administrativa, se tomó la decisión de aprovechar el sistema de administración que provee Django para beneficiarse de las opciones de filtrado y búsqueda que posee. Por otra parte se decidió usar FullCalendar para mostrar la vista de la programación semanal.

5.9.1. Seguimiento

Por coincidir en verano no se pudo celebrar la reunión con el cliente pero los objetivos quedaron fijados en la última reunión. El cambio de la parte administrativa de la web

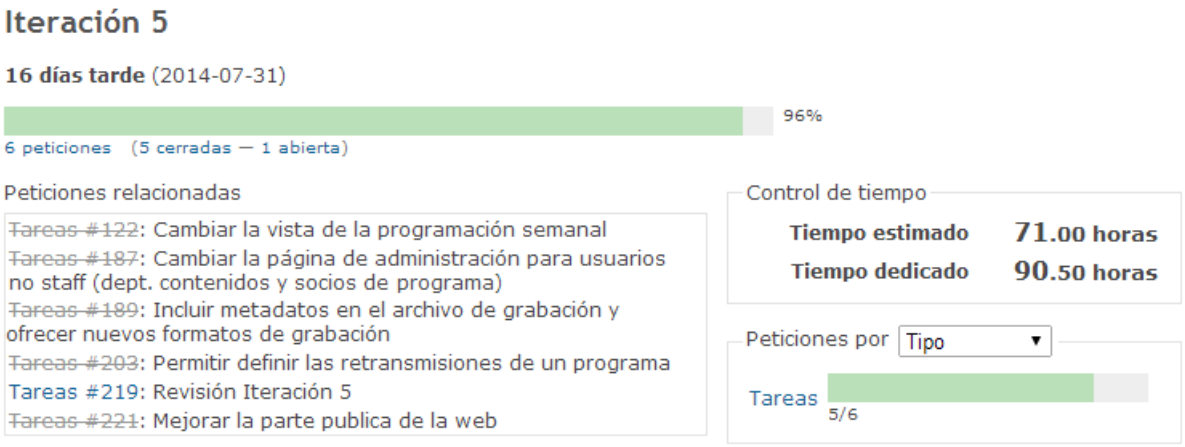

Figura 5.10: Iteración 5

llevó más tiempo del planificado puesto que requería entrar a modificar partes del propio framework. Se tomó la medida de dedicarle más horas y cumplir con los plazos previstos.

5.10. Sexta iteración

Esta última iteración aborda la búsqueda, adquisición y puesta en funcionamiento de un sistema embebido con el programa de grabación. Se valoraron dos opciones:

- **Raspberry Pi** Un ordenador de placa reducida de bajo coste, no cuenta con entrada de audio por lo que habría que buscar un complemento que nos proporcione dicha funcionalidad. Un punto más a favor es que cuenta con una gran comunidad detrás.
- **Cubieboard** En comparación con la Raspberry es superior en cuanto rendimiento pero también lo es en coste. Al igual que la Raspberry carece de entrada de audio y es menos conocida.

Por ser más conocida y barata se eligió la Raspberry Pi, a continuación se hizo un estudio para elegir el complemento para la grabación de audio, las dos opciones encontradas más económicas fueron:

- **U-CONTROL UCA202**[15]: Un grabador usb que soporta la grabación de audio de

[15] www.behringer.com

hasta 16 bits a 48kHz.

- **Wolfson Pi Audio Card**[16]: Una tarjeta de sonido que permite la grabacion de audio de hasta 24 bits a 192kHz. Cuenta con unas prestaciones muy superiores a las otras opciones con un precio levemente superior al dispositivo de Behringer. Sin embargo este modelo no es compatible con la última versión de Raspberry Pi (model B+) y es muy complicado conseguir una carcasa que se ajuste a este complemento.

Se optó entonces por el U-CONTROL UCA202 a pesar de ofrecer unas prestaciones más limitadas pero suficientes para la grabación de audio. En la Figura 5.12 podemos ver el sistema funcionando.

Durante las pruebas de grabación se detectó que la Raspberry no disponía de una capacidad de procesamiento suficiente para el comando Avconv puesto que consumía toda la CPU disponible y ello provocaba que se perdieran datos de sonido.

Con la esperanza de solucionarlo se buscaron optativas para la grabación de audio que consumieran menos recursos. La solución llego por parte de arecord, un comando para grabar sonido usando el driver ALSA, y oggenc un codificador de audio en OGG.

Con estos nuevos comandos el consumo de CPU se sitúa alrededor del 60 % y se realiza la grabación sin cortes.

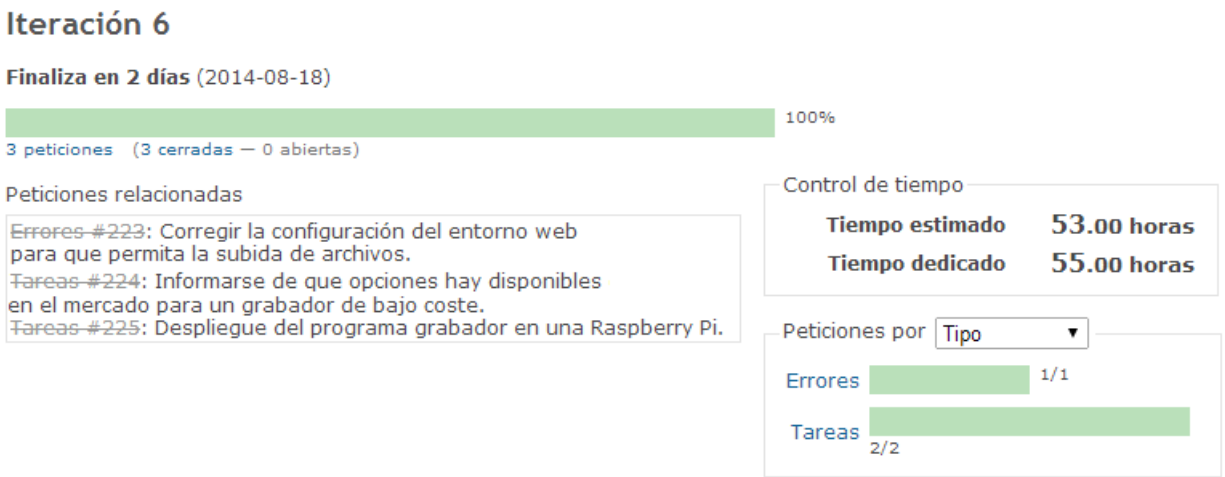

Figura 5.11: Iteración 6

[16] www.wolfsonmicro.com

Figura 5.12: Raspberry Pi y U-CONTROL UCA202

5.10.1. Seguimiento

Esta iteración acabó en el tiempo estimado a pesar de encontrarse con problemas inesperados gracias en parte a la decisión de diseño tomada en la iteración 3 de delegar la grabación a otro programa.

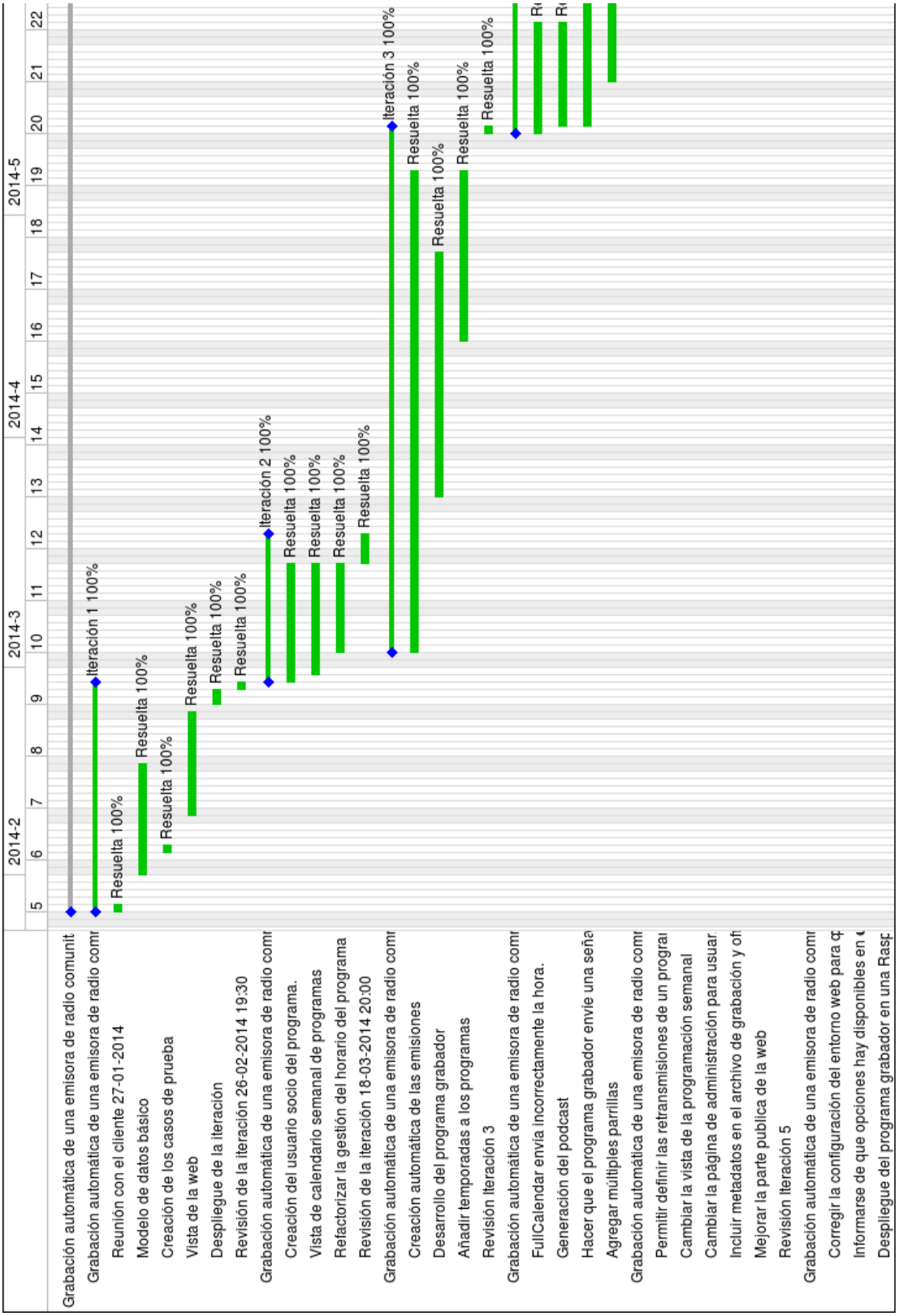

Figura 5.13: Diagrama de Gantt, parte 1

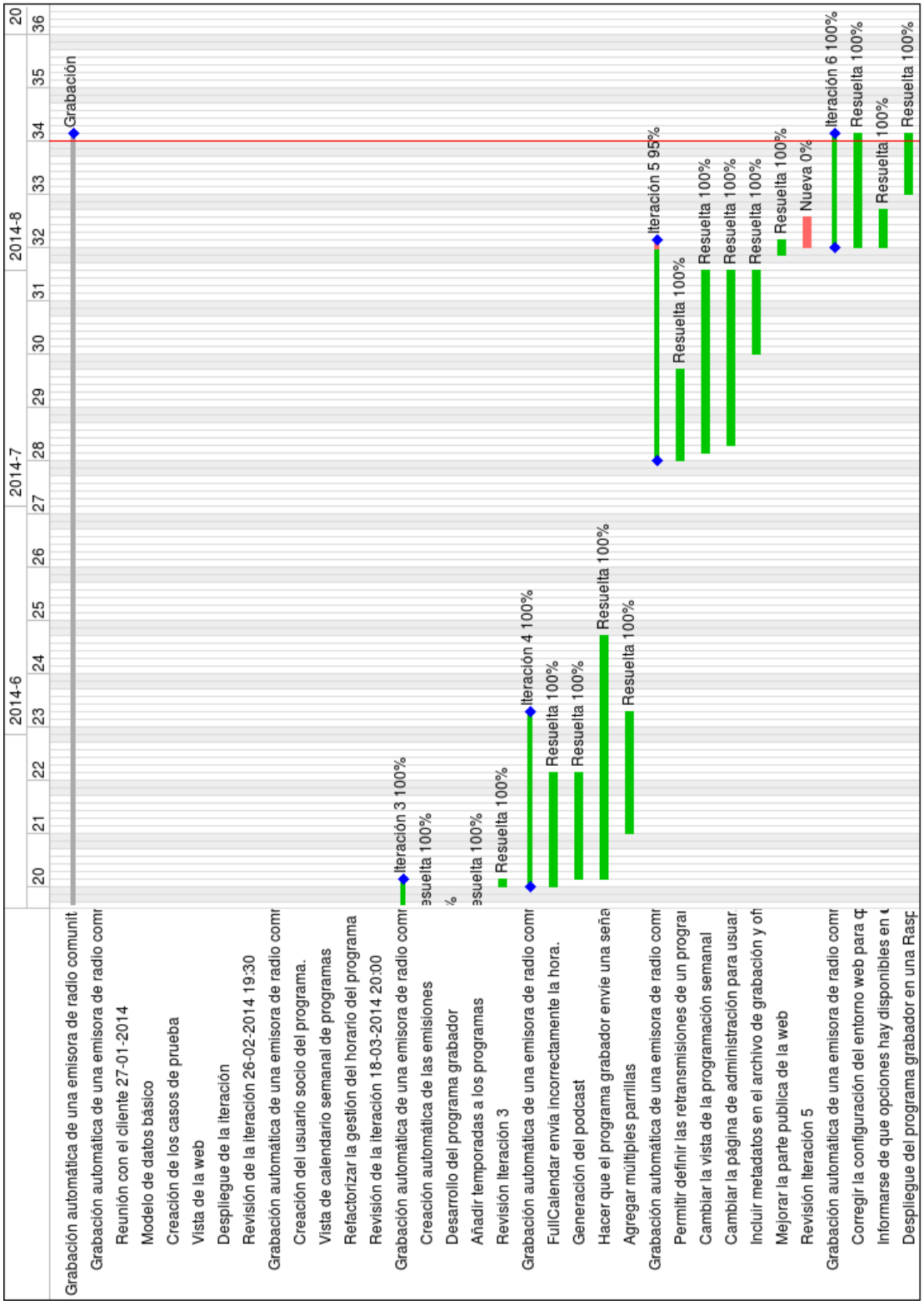

Figura 5.14: Diagrama de Gantt, parte 2

Capítulo 6

Evaluación de costes

El coste del proyecto se realiza teniendo en cuenta los recursos necesarios para su elaboración. Hay que tener en cuenta que este cálculo no incluye margen de beneficio.

6.1. Coste final

Por un lado tenemos a los recursos humanos, en nuestro caso una misma persona desempeñó múltiples roles:

Actividad	Horas	Coste por hora	Total
Análisis	73.50	23€/hora	1690.5€
Diseño	6	18€/hora	108€
Prototipado	2.5	14€/hora	35€
Desarrollo	313.15	14€/hora	4384.1€
Pruebas	17	14€/hora	238€
Despliegue	17.10	19€/hora	324.9€

Con estos datos obtenemos que el coste de los recursos humanos es de 6780.5€.

Coste de productos y servicios:

Producto / Servicio	Coste
Contratación de un servidor web con dominio incluido durante 12 meses	28.89€
Contratación de un servidor de aplicaciones	0*€
Ordenador Raspberry Pi B+	33.71€
Grabador de sonido Behringer UCA202	27.89€
Adaptador de corriente	10.61€
Carcasa Raspberry Pi	7.96€
Adaptador wifi usb	9.11€
Tarjeta micro SD	6.06€

* 12 meses de prueba en AWS[1] restringido a 750 horas/mes.

La suma del coste de los productos y servicios nos da como resultado 124.23€

En cuanto al software utilizado para su desarrollo ha tenido un coste nulo puesto que se han utilizado únicamente herramientas de software libre.

Por tanto el coste total del proyecto es de: 6780.5€+ 124.23€+ 0€= 6904.73€

6.2. Diferencia entre el coste real y el coste planificado

Para calcular el desfase entre coste estimado y el real necesitamos conocer las horas planificadas y dedicadas en cada iteración. En nuestro caso recurrimos a Redmine para obtener dichos valores:

Iteración	Horas estimadas	Horas dedicadas	Diferencia
Iteración 1	60	65.5	5.5
Iteración 2	48	67.9	19.9
Iteración 3	50	62.5	12.5
Iteración 4	75	89.75	14.75
Iteración 5	71	90.5	19.5
Iteración 6	53	55	2
Total	357	431.15	74.15

[1] Amazon Web Services

Teniendo la diferencia entre las horas estimadas y las dedicadas podemos calcular el sobrecoste multiplicándolas por un precio medio por hora, siendo este precio 15.73€/hora nos da un sobrecoste de 1166.38€.

6.3. Reflexiones

Con los datos anteriores observamos que el proyecto se desvió un 20 %. Las principales causas de esta desviación se deben a una planificación demasiado optimista unido a la falta de experiencia en la estimación y en las tecnologías en las que se desarrolló el proyecto.

Capítulo 7

Diseño del sistema

En este capítulo se describe la arquitectura del sistema diferenciando la aplicación web y el programa grabador.

7.1. Arquitectura general

Hay dos subsistemas claramente distinguibles en este trabajo fin de grado.

- **Aplicación web**, desarrollada en Django y Python.
- **Programa grabador**, implementado en Python.

La aplicación web es la parte visible al exterior, los usuarios realizan en ella toda su actividad, satisface las necesidades de información de los oyentes proporcionando la información de los programas y la gente que participa en ellos. La programación de la parrilla cuenta con enlaces que llevan directamente a la información del episodio, cada uno contiene un reproductor HTML5 en caso de que el audio esté disponible. También es posible subscribirse a un programa, puesto que cada programa tiene su propio podcast.

Para los usuarios registrados en la aplicación dispone de una zona de administración donde se encuentran todas las funcionalidades de la aplicación.

Por otra parte el programa grabador una vez instalado no es necesario manipularlo, se comunica de forma periódica con la aplicación web para recoger la información de los

programas a grabar. Además cuando acaba de grabar y subir un programa le envía los datos del audio para que esté disponible en la web.

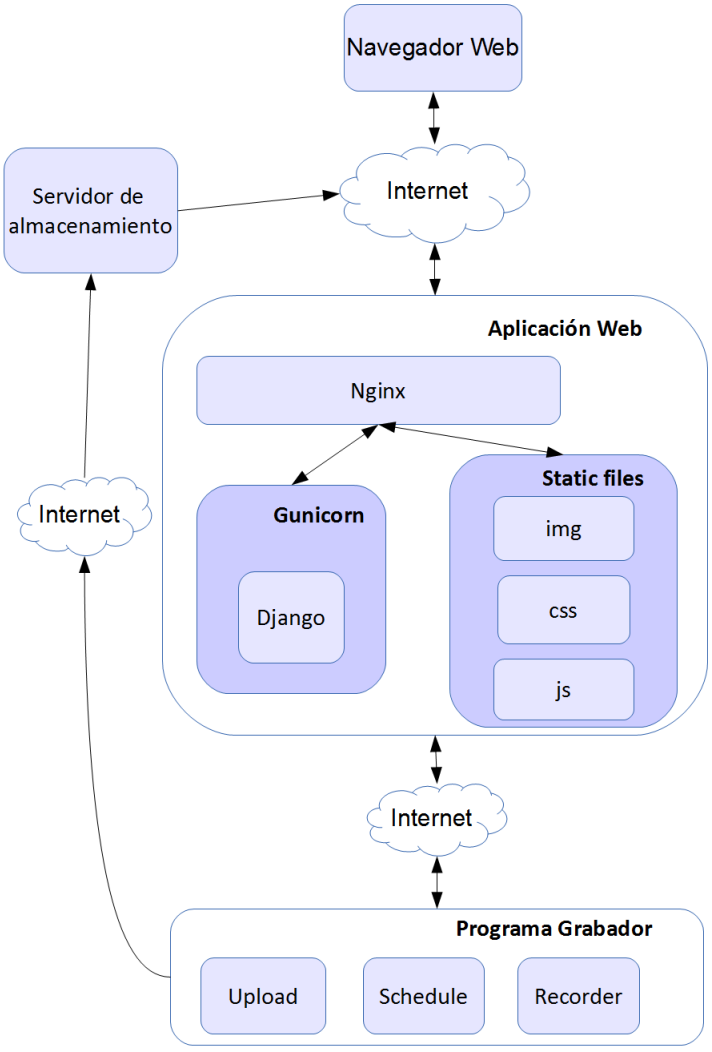

Figura 7.1: Arquitectura global del Sistema

7.2. Subsistema Aplicación Web

Este sistema corresponde con la parte visible por los usuarios.

7.2.1. Arquitectura

La arquitectura de este subsistema viene determinada por el framework usado, Django usa una arquitectura inspirada en el patrón MVC[1] llamada Model Template View, veamos sus diferencias:

- La vista en Django representa que datos serán representados pero no el cómo.
- El Template representa como los datos son representados.
- El controlador del MVC estaría representado por el propio framework puesto que se encarga de enviar la petición a la vista correspondiente de acuerdo a la configuración de URL de Django.
- El concepto de modelo no varía.

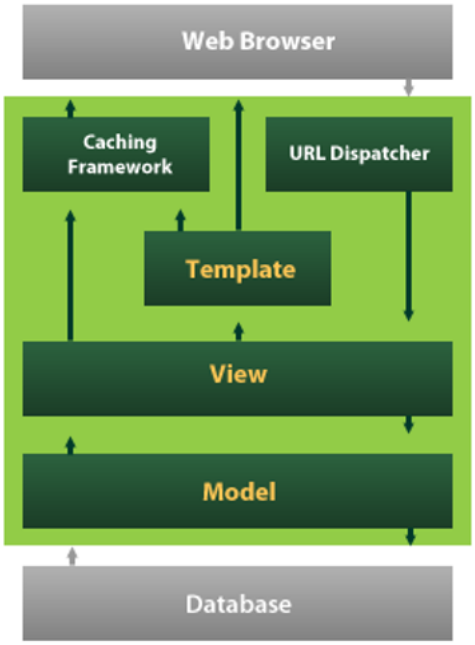

Figura 7.2: Esquema Model Template View

[1] Model–view–controller.

7.2.1.1. Estructura

El paquete principal del proyecto se llama radio y contiene el resto de subpaquetes y módulos.

Un paquete cuenta con sus propios archivos y la idea de esto sería poder reusarlo completamente en otra aplicación sin realizar cambios. Por tanto tienen una alta cohesión y un acoplamiento mínimo.

Una carpeta si contiene un archivo __init__.py es considerado un paquete de Python, cada archivo Python que contiene un paquete se considera un módulo. Por convenio se suelen adoptar los siguientes nombres:

- **models.py:** contiene las entidades persistentes. Un modelo de Django contiene los campos y comportamientos de los datos que estas guardando, el objetivo es definir los datos del modelo en un único sitio y derivar cosas de él.
- **urls.py:** contiene las declaraciones de las direcciones URL . El esquema URL de Django te permite diseñar tus propias URLs sin limitaciones. Este módulo hace un mapeo entre el patrón URL y la función a llamar, la vista.
- **views.py:** contiene los funciones de la vista. Generalmente devuelven un template renderizado con los datos obtenidos en dicha función.
- **tests.py:** Contiene las funciones de prueba.
- **forms.py:** Contiene los formularios usados.
- **admin.py:** Es el encargado de todo lo relativo a las entidades que aparecerán en la zona de administración.

El proyecto se dividió en paquetes agrupados por funcionalidad, cada paquete contiene su modelo, funciones de prueba, configuración url, vista y template.

- **Usuarios:** Donde se agrupan la funcionalidad relativa al perfil de usuario, Django proporciona un modelo de usuario con un sistema de autentificación y permisos que hemos extendido para añadirle atributos propios de nuestra aplicación.

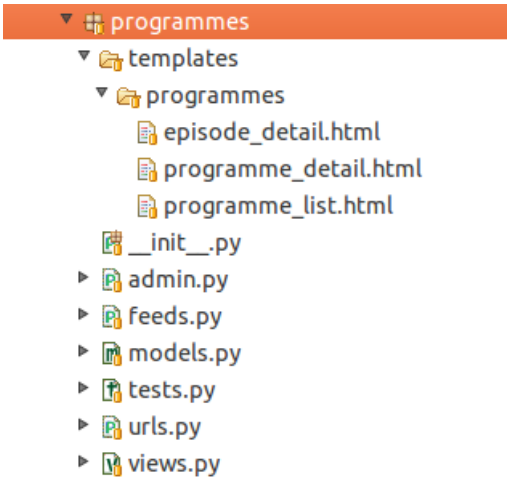

Figura 7.3: Ejemplo de un paquete en Python

- **Programas:** Este paquete contiene los modelos del programa, sus episodios y su podcast.
- **Horarios:** Se compone del horario y de la parrilla de programación
- **Zona de administración personalizada:** Una zona de administración personalizada para los usuarios registrados en la aplicación.
- **Opciones globales:** Configuraciones globales de la aplicación sin tener que modificar el código fuente. Son editables a través de la zona de administración.

7.2.2. Modelo de datos

La aplicación web almacena toda la información en un modelo de base de datos relacional, para el desarrollo de esta aplicación se ha elegido MySQL por ser una de las opciones más utilizadas en el mundo del software libre y ya se tenía experiencia previa en este entorno.

Paquete Usuario Django proporciona un sistema de autentificación que maneja cuentas de usuario, grupos, permisos y sesiones. Para nuestra aplicación es necesario extender el modelo User, la documentación oficial[2] recomienda que si sólo necesitamos añadir

[2]http://www.djangoproject.com/

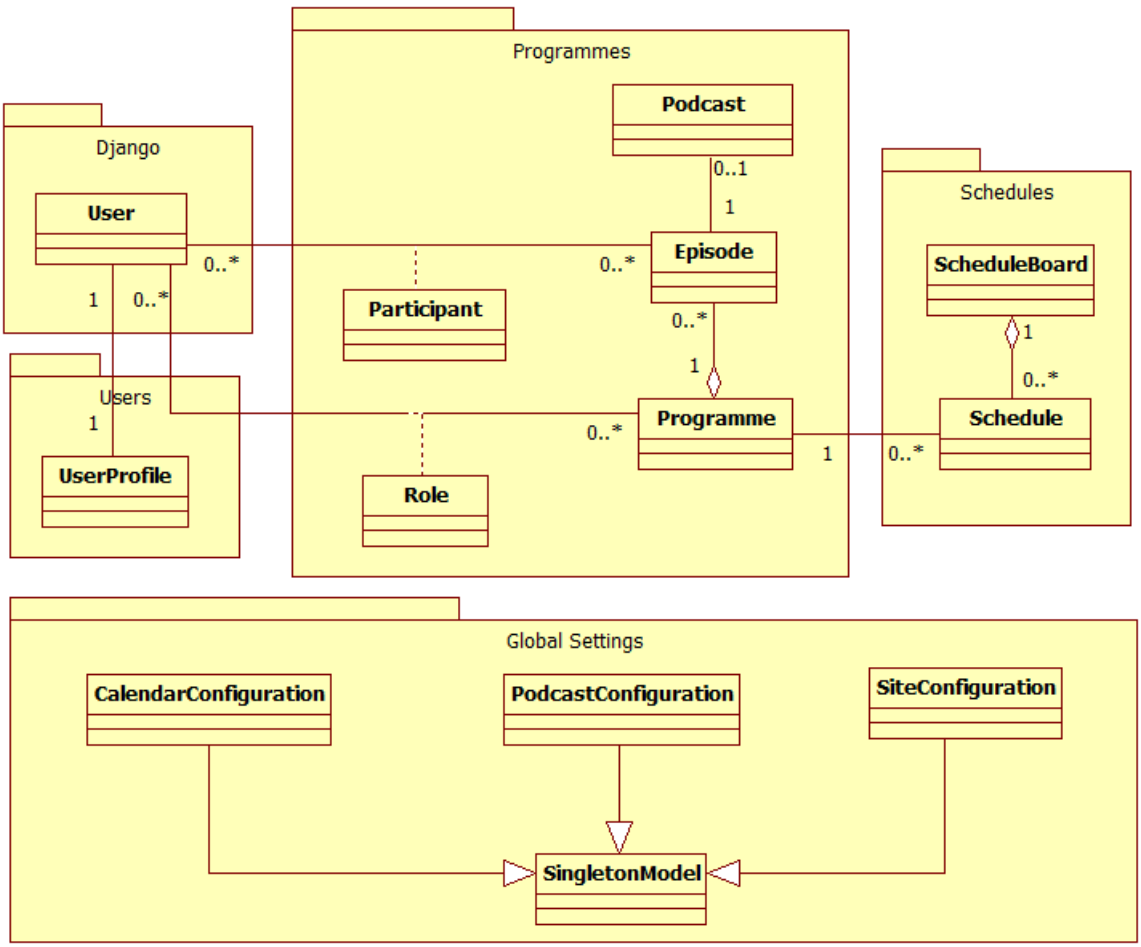

Figura 7.4: Diagrama de paquetes de la aplicación web

información añadamos una relación uno a uno con el modelo proporcionado por Django, esto es lo que hemos hecho en la entidad UserProfile (Figura 7.5).

Paquete Programas En la Figura 7.6 se muestran los programas junto con sus episodios, un programa y un episodio tiene por lo general varios usuarios y un usuario puede estar en ninguno, uno o varios programas, para ello se crean las tablas intermedias que contienen información adicional como que papel realizan y la fecha en la que se unieron al programa. Además un episodio, puede tener un podcast asociado.

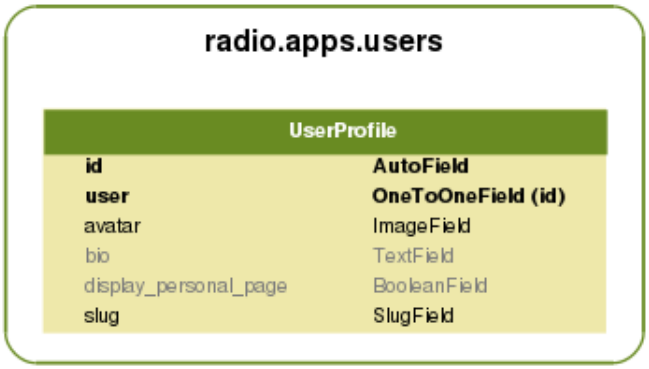

Figura 7.5: Diagrama de entidades del paquete usuarios

Paquete Horarios Las entidades de la Figura 7.7 representan las parrillas y los horarios de emisión de los programas. Una parrilla puede contener varios horarios, cada horario sólo pertenece a una parrilla y está asociado a un programa. Un horario tiene una hora de inicio y un día de la semana, la duración viene dada por la duración del programa, de esta forma es necesario calcular las fechas de emisión concretas.

Configuración global En la Figura 7.8 están las opciones de configuración a nivel global de la aplicación, se han separado por áreas: calendario, podcast y el sitio web. Sólo puede existe una instancia de cada una en la base de datos.

7.2.3. Funcionamiento

Cuando llega una petición el servidor construye una HttpRequest que se pasa a los componentes, el manejador a continuación intenta encontrar una vista a la que enviar la petición a través de la configuración URL que no es más que un archivo con una lista de urls con expresiones regulares a examinar. En caso de que encaje se envía a la vista y esta consigue los datos del modelo, los renderiza en un template y envía la respuesta (ver Figura 7.9).

Durante este proceso se pueden ejecutar middlewares, un middleware es una clase de Python que se ejecuta durante el procesado de la petición para añadir una determinada funcionalidad, Django permite añadir middlewares de forma transparente como si se tratara de un sistema de plugins.

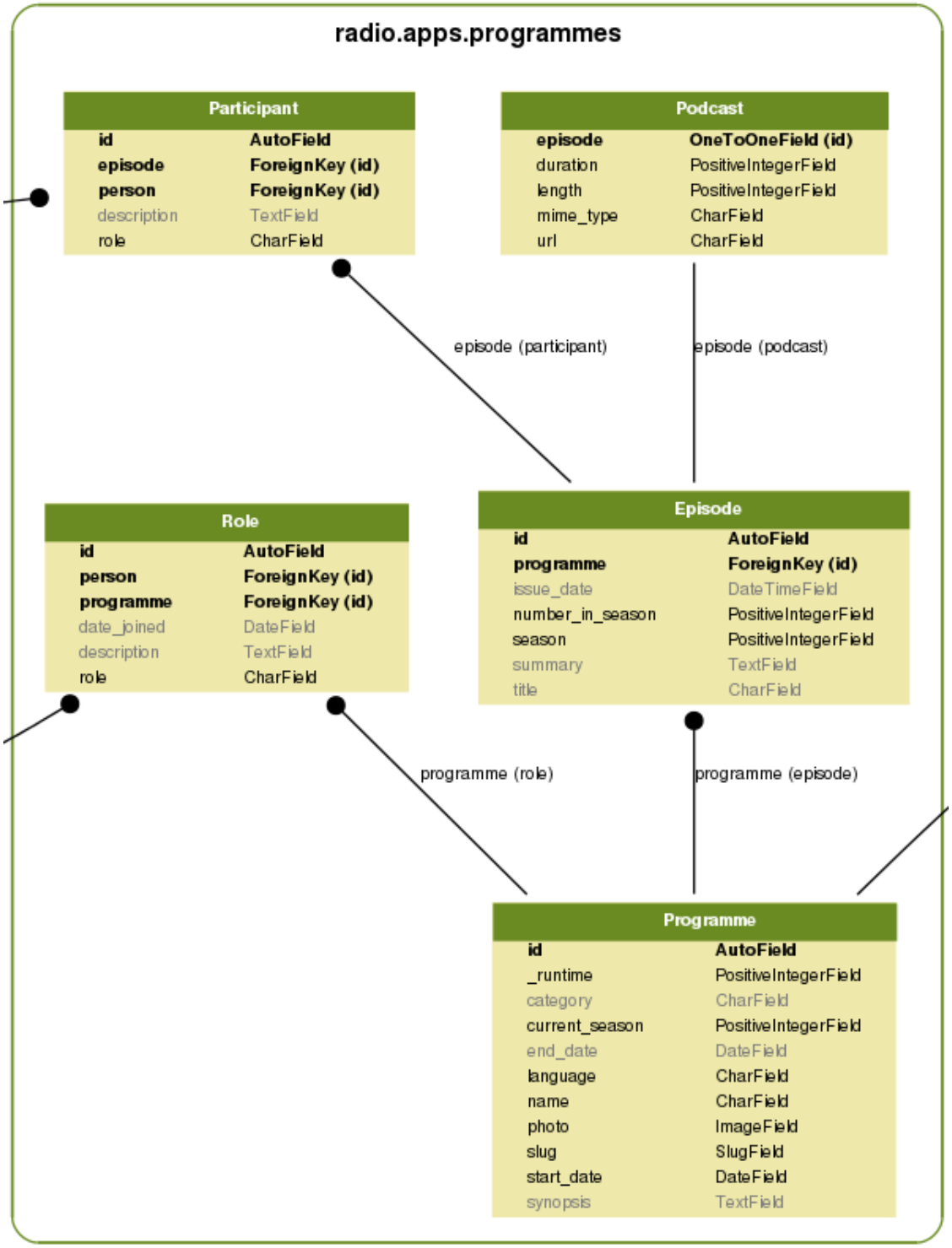

Figura 7.6: Diagrama de entidades del paquete programas

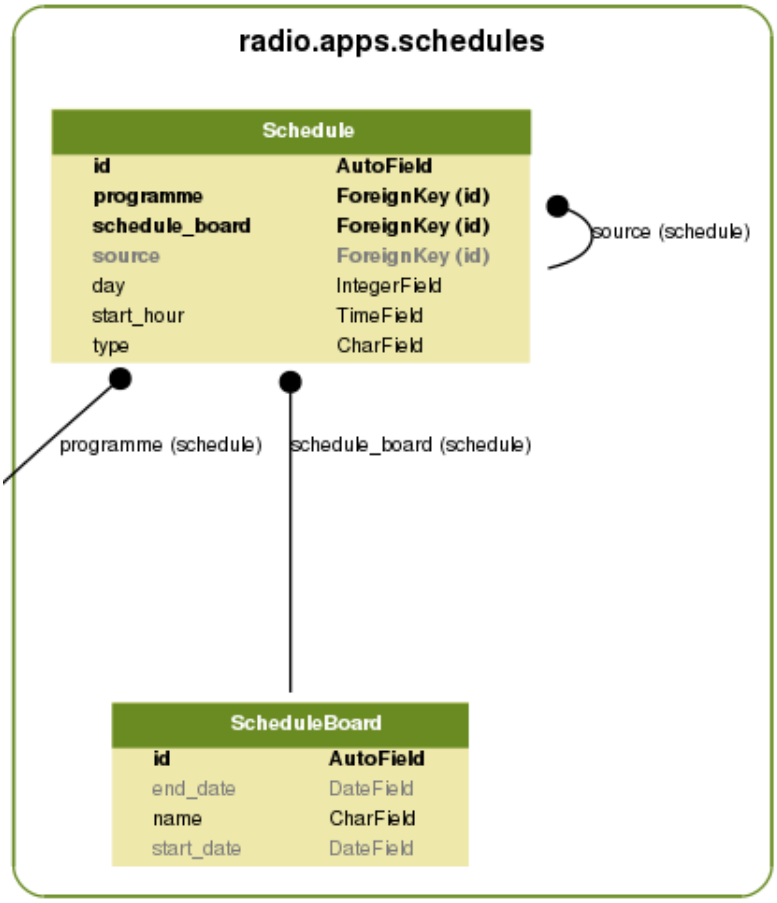

Figura 7.7: Diagrama de entidades del paquete horarios

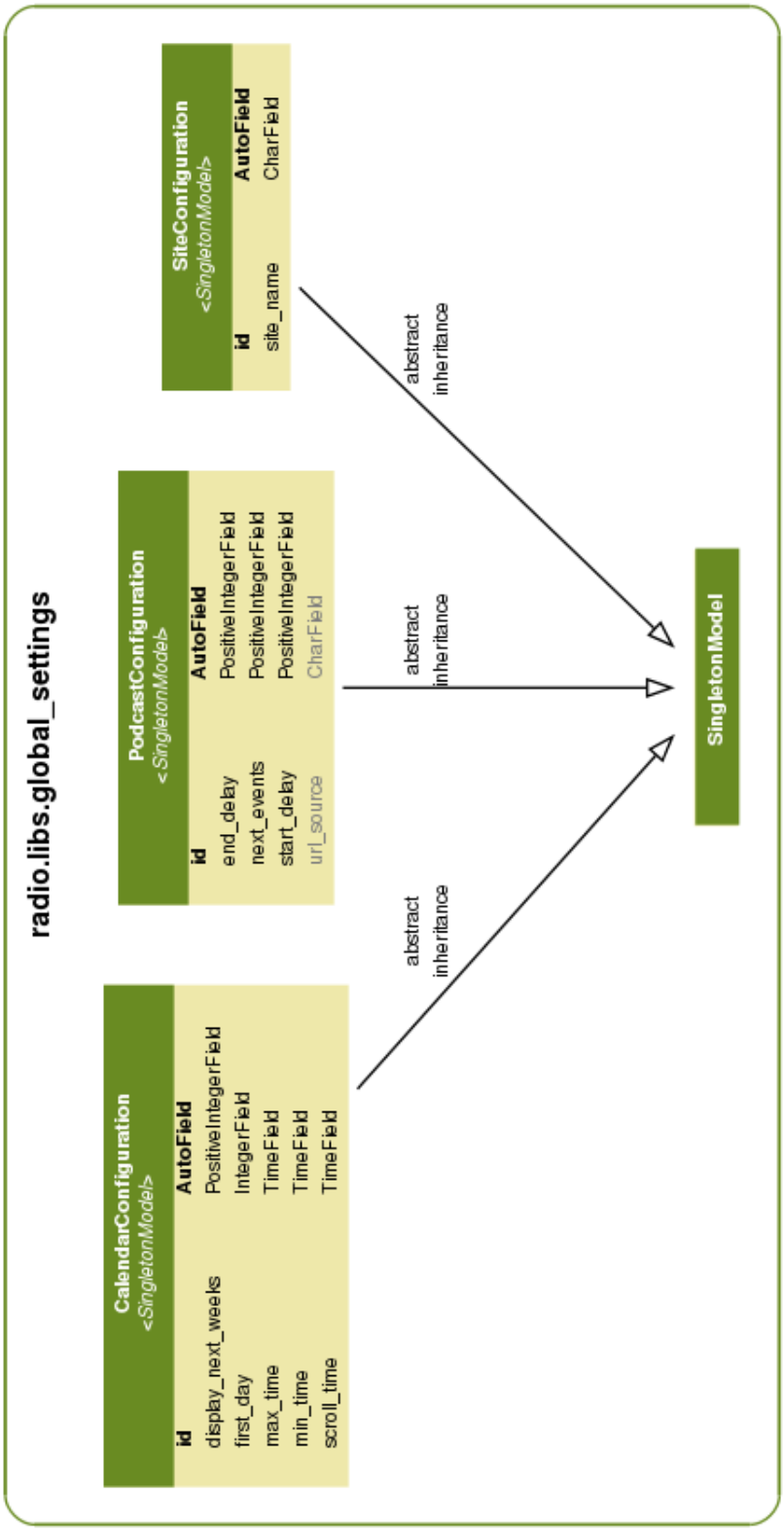

Figura 7.8: Diagrama de entidades del paquete de configuraciones globales

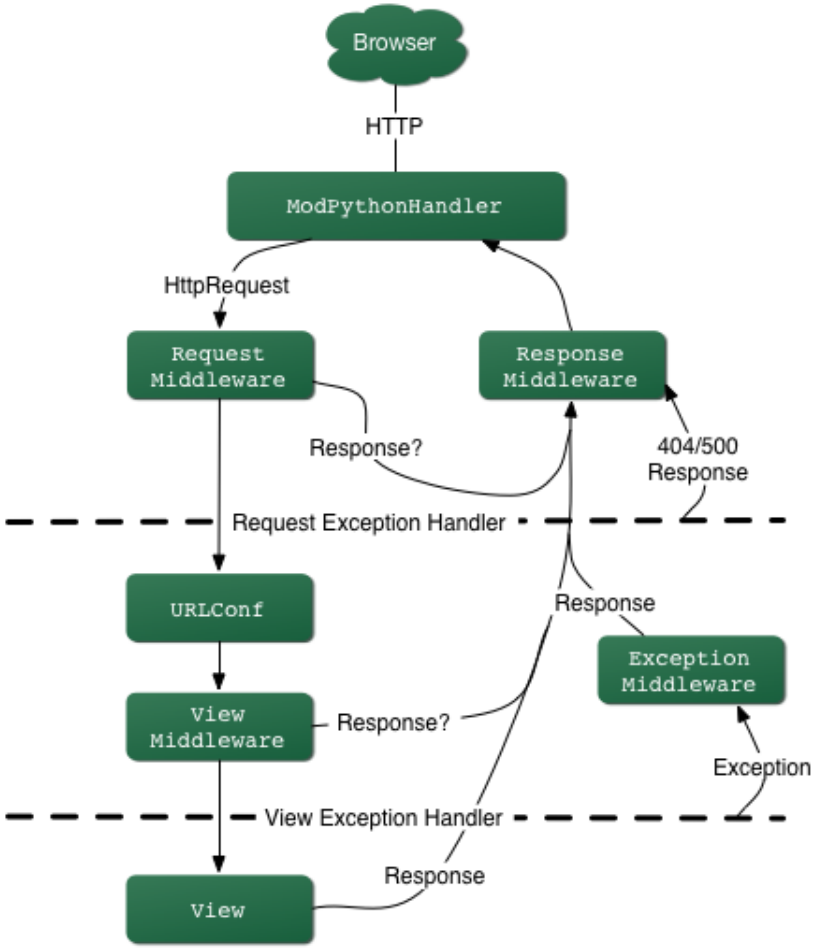

Figura 7.9: Diagrama de flujo de una petición

7.3. Subsistema Programa Grabador

Es el programa encargado de llevar a cabo la captura del audio, se ha escrito completamente en Python y realiza los siguientes pasos:

- Obtener los horarios de grabación del servidor.
- Realizar la grabación de audio según los horarios obtenidos.
- Comprimir el audio.
- Añadir metadatos al archivo de audio.
- Subida a un servidor ftp.
- Comunicar a la aplicación web que la grabación se ha realizado.

7.3.1. Arquitectura

No se ha seguido ningún patrón arquitectónico específico para la realización de este programa. El programa no cuenta con base de datos, almacena la información de los programas a grabar en un fichero con formato JSON y los archivos de audios siguen un flujo de carpetas. Esto reduce los pasos de instalación y ayuda a usuarios sin conocimientos técnicos a instalar el programa.

7.3.1.1. Estructura

Este programa se compone de 4 módulos claramente diferenciados:

- **Horarios:** El encargado de obtener y actualizar los próximos programas a grabar.
- **Subida:** Realiza la subida de archivos por FTP y se lo comunica a la aplicación web.
- **Grabacion:** Captura el audio en el formato preestablecido en el fichero de configuración.
- **Pruebas:** Contiene las funciones de prueba.

Para dar una mayor tolerancia a fallos el programa cuenta con un sistema de log donde se recogen los errores, avisos e información. Se han tenido en cuento otros problemas como una caída de tensión o desconexiones durante la subida de los archivos. Para ello los archivos pasan por una serie de carpetas:

- **Incompleta:** Es donde se graba el archivo, en caso de una caída de tensión u otro problema inesperado durante la grabación aquí se encontraría el audio parcialmente grabado.
- **Completa:** Aquí se encuentran los archivos grabados listos para su subida, esto permite que en caso de problemas de conexión se almacenen aquí los archivos y se suban cuando la conexión vuelva a estar disponible incluso si se cierra el programa.
- **Subidos:** Cuando se sube un archivo el programa mueve el audio de la carpeta de archivos completados a subidos. Es posible cambiar este comportamiento simplemente editando el archivo de configuración y activando la opción de que se borren tras su subida.

El nombre de las carpetas así como su ruta es configurable a través del archivo de configuración.

7.3.2. Funcionamiento

Cuando se inicia el programa lee un fichero de configuración llamado setting.ini en el que se encuentran los diferentes parámetros de configuración, una vez leídos se crean las carpetas de grabación en caso de que no existan y se inician varios hilos, cada uno de estos hilos tiene una función específica:

- **Hilo principal:** Se queda a la espera hasta que llega la hora de empezar la grabación, cuando llega el momento de grabar crea otro hilo y se queda a la espera.
- **Hilo de grabación:** Recibe los parámetros a grabar y lanza los subprocesos de grabación especificados en el archivo de configuración, una vez acaba crea un hilo que se encargara de las acciones post grabación y finaliza. El sentido de hacer esto es liberar lo antes posible este hilo puesto que no se podrá grabar hasta que este termine.

- **Hilo de acciones post grabación:** La finalidad de este hilo es llevar a cabo tareas que pueden requerir tiempo como es la compresión de audio o añadir metadatos, en nuestro caso todo esto lo conseguimos hacer directamente al lanzar el comando de grabación y sólo usamos este hilo para mover el fichero a otra ubicación.
- **Hilo de actualización de horarios:** Es el encargado de realizar la petición de consulta de horarios al servidor y almacenarla en disco. El motivo de hacerlo así es evitar problemas de conexión y la posibilidad de modificar manualmente el archivo.
- **Hilo de subida:** Se encarga de subir los archivos al servidor FTP y comunicar a la aplicación web de que se ha llevado a cabo la grabación.

Capítulo 8

Implementación

En este apartado se explicarán las partes de mayor complejidad del sistema y se proporcionan ejemplos de implementación.

8.1. Aplicación web

8.1.1. Perfil de usuario: Ejemplo completo

En esta sección se muestran los pasos necesarios para crear en Django una página que muestre la información de los usuarios.

Para ello creamos un nuevo paquete que contendrá todo lo relacionado con este ámbito, se sigue la estructura explicada en la apartado 7.2.1.1.

8.1.1.1. Modelo

Aprovechamos el usuario genérico proporcionado por Django y lo extendemos añadiendo la información que necesitamos:

```
from django.contrib.auth.models import User
from django.core.urlresolvers import reverse
from django.db import models
from django.db.models.signals import post_save
from django.template.defaultfilters import slugify
```

```
from django.utils.translation import ugettext_lazy as _

class UserProfile(models.Model):
    user = models.OneToOneField(User)
    bio = models.TextField(blank=True, verbose_name=_("biography"))
    avatar = models.ImageField(upload_to='avatars/', default='/static/
        radio/images/default-userprofile-avatar.jpg', verbose_name=_("
        avatar"))
    display_personal_page = models.BooleanField(default=False,
        verbose_name=_("display personal page"))
    slug = models.SlugField(max_length=30)

    def get_absolute_url(self):
        return reverse('users:detail', args=[self.slug])

    def save(self, *args, **kwargs):
        if not self.pk:
            try:
                p = UserProfile.objects.get(user=self.user)
                self.pk = p.pk
            except UserProfile.DoesNotExist:
                pass
        self.slug = slugify(self.user.username)
        super(UserProfile, self).save(*args, **kwargs)

    class Meta:
        verbose_name = _('user profile')
        verbose_name_plural = _('user profile')

    def __unicode__(self):
        return "%s's profile" % self.user
```

En el código anterior extendemos el usuario genérico de Django con una relación y añadimos 4 campos, uno de ellos no será visible ni editable por el usuario y es generado cuando se guarda el objeto. Este campo llamado slug transforma una cadena a caracteres ASCII eliminando y transformando acentos u otros símbolos no permitidos en ASCII. Esto es necesario para poder mostrar una URL legible.

También usamos ugettext para proporcionar internacionalización al sistema, "_('user profile')" será el string a traducir, este módulo estándar de Python se encargara de buscar

estas cadenas y generar un archivo de traducción para el idioma deseado.

Django dispone de unos middlewares para elegir a que idioma traducir las cadenas automáticamente:

```
MIDDLEWARE_CLASSES = (
    'django.middleware.common.CommonMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.locale.LocaleMiddleware'
)
```

Con esto conseguimos proporcionar a cada usuario individual el idioma que prefiere según sus preferencias basadas en los datos de la petición. Si no queremos este comportamiento podemos establecer un idioma por defecto en el fichero de configuración de Django.

8.1.1.2. Zona de administración

Una vez tenemos el modelo podemos darlo de alta para que aparezca en la zona de administración y los usuarios puedan cambiarlo. Puesto que Django ya cuenta con un usuario y sería muy incómodo que el perfil apareciera por separado podemos hacer que aparezcan juntos usando la opción “Inline”:

```
from django.contrib import admin
from django.contrib.auth.admin import UserAdmin
from django.contrib.auth.models import User
from radio.apps.users.models import UserProfile

class UserProfileInline(admin.StackedInline):
    model = UserProfile
    can_delete = False
    exclude = ('slug',)

class UserProfileAdmin(UserAdmin):
    inlines = (UserProfileInline,)

# Re-register UserAdmin
admin.site.unregister(User)
admin.site.register(User, UserProfileAdmin)
```

Con esto ocultamos el campo slug y no permitimos el borrado por separado de un perfil de usuario sin borrar el objeto usuario. En la Figura 8.1 podemos ver el resultado.

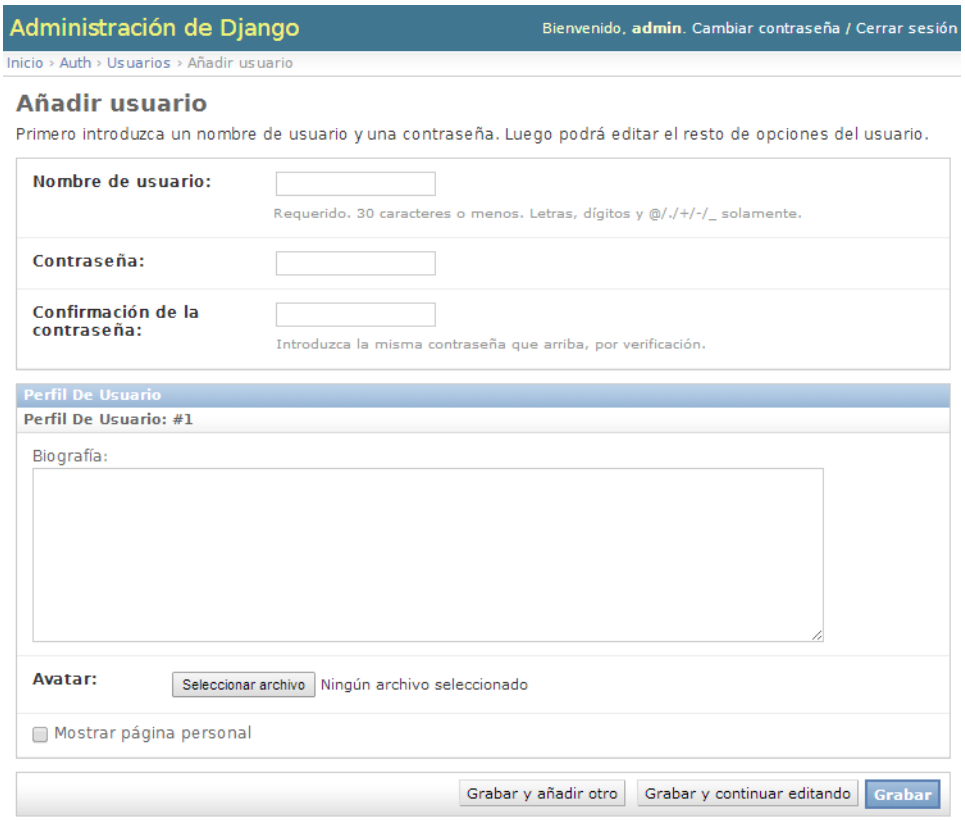

Figura 8.1: Página de administración – Crear usuario

8.1.1.3. Vista y template

Si queremos mostrar la información en la parte pública de la web debemos crear la plantilla (más conocido en ingles por template) y la función correspondiente:

La plantilla es el código html con etiquetas que serán sustituidas con los valores del contexto. Puesto que todas las páginas tienen áreas comunes es conveniente definir una plantilla base y extender de ella las partes que varíen. En el siguiente código extendemos de la plantilla llamada base.html sobrescribiendo el bloque content.

```
{% extends "home/base.html" %}
{% load staticfiles i18n %}
{% block users_class %}class="active"{% endblock %}
```

```
{% block content %}
<div class="row">
   {% if userprofile %}
      <h1>{% firstof userprofile.user.get_full_name|upper userprofile.
          user|upper %}</h1>
      <div class="media">
         <img src="{{ userprofile.avatar.url }}">
      </div>
      <p>{{userprofile.bio}}</p>
      {% if role_list %}
         <h2>{% trans "Contributions"|upper %}</h2>
         <dl id="tabulate">
         {% for role in role_list %}
            <dt><a href="{% url 'programmes:detail' role.programme.slug
                %}">{{ role.programme }}
            {% if unspecified != role.role %}
               ({{ role.get_role_display }})
            {% endif %}
            </a>
            </dt>
            <dd>
               {% if role.description %}
                  {{role.description}}
               {% else %}
                  .
               {% endif %}
            </dd>
         {% endfor %}
         </dl>
      {% endif %}
      {% else %}
         <h1>{% trans "No user information available"|upper %}</h1>
         <div class="media">
         <img src="{% static "radio/images/default-userprofile-avatar.
             jpg" %}">
         </div>
      {% endif %}
</div>
{% endblock %}
```

Django nos proporciona de un sistema de internacionalización a nivel de plantilla, al igual que en el código estas cadenas aparecerán en los correspondientes ficheros para ser traducidas.

Una vez tenemos la plantilla necesitamos la función en la vista para que obtenga los datos y mediante la plantilla procese y envíe el resultado:

```
from django.shortcuts import render, get_object_or_404
from radio.apps.programmes.models import  Role, NOT_SPECIFIED
from radio.apps.users.models import UserProfile

def userprofile_detail(request, slug):
    userprofile = get_object_or_404(UserProfile.objects.select_related(
        'user'), slug=slug, display_personal_page=True)
    context = {'userprofile': userprofile,
        'unspecified': NOT_SPECIFIED,
        'role_list':Role.objects.filter(person=userprofile.user).
            select_related('programme')}
    return render(request, 'users/userprofile_detail.html', context)
```

Esta función recibe como parámetros la petición y el slug que será con el que consultaremos la base de datos. Django nos proporciona un atajo para que en caso de que no exista el objeto que buscamos produzca un error HTTP 404, fijémonos que también se le indica que el perfil de usuario tenga la opción de mostrar su perfil al público.

Una vez obtenidos los datos se crea el contexto que no son más que información clave valor, con este contexto se renderiza el template y se envía el resultado.

8.1.1.4. Configuración URL

Por último nos falta asociar la vista a una URL, esto se hace mediante una expresión regular que nos permite pasar parámetros:

```
from django.conf.urls import patterns, url
from radio.apps.users import views

urlpatterns = patterns('',
    url(r'^(?P<slug>[-\w]+)/$', views.userprofile_detail, name='detail'
        ),
)
```

Django comprobará si la url proporcionada encaja con algún patrón y a continuación llama a la función correspondiente, en caso de que no encuentre ninguna produce una excepción HTTP 404.

8.1.1.5. Diseño web adaptable

La aplicación se adapta a diferentes tamaños de pantalla, para ello Bootstrap nos facilita la tarea proporcionando clases CSS y otros componentes que usaremos para que se ajuste a las dimensiones de la pantalla personalizándolas si es necesario. En la Figura 8.2 y 8.3 podemos ver este comportamiento.

Figura 8.2: Vista de un usuario

Figura 8.3: Vista de un usuario – Versión móvil

8.1.2. Gestión de horarios

La parte más complicada de este proyecto tiene que ver con la gestión de las fechas.

No podemos almacenar las fechas de emisión de los programas en base de datos debido a que esta información podría ser infinita en caso de que el programa y la parrilla no tuvieran una fecha de finalización. Una solución podría ser generar y guardar las emisiones de las próximas x semanas pero esto ocuparía un espacio innecesario debido a que se repiten cada semana además de que nos generaría problemas en caso de que las emisiones cambiaran puesto que sería necesario actualizar sus horarios.

Por ello se decidió que las fechas de emisión de un programa tienen que ser calculadas a partir de los horarios en la parrilla de programación. Cada horario de la parrilla cuenta con el día de la semana y la hora, esto es debido a que la programación de la parrilla es semanal (se repite cada semana). Con esta información podemos generar las próximas fechas de emisión, Python nos proporciona un módulo para llamado rrule para ayudarnos en esta tarea.

```
rrule.rrule(rrule.WEEKLY, byweekday=[day], dtstart=datetime.datetime.
    combine(start_date, start_hour))
```

Con esto construimos un objeto rrule siendo: day y start_hour la información proporcionada por el horario y start_day el día actual por ejemplo. Si quisiéramos listar este objeto obtendríamos fechas infinitas puesto que no le hemos indicado una fecha de finalización.

En nuestro caso el problema se complica puesto que tenemos que añadirle la restricción de fechas del programa y la de la parrilla a la que esté asociado el horario, el método para obtener el objeto rrule sería el siguiente:

```
def __get_rrule(self):
        start_date = self.programme.start_date
        if self.schedule_board.start_date and start_date < self.
            schedule_board.start_date:
            start_date = self.schedule_board.start_date
        if self.programme.end_date:
            end_date = self.programme.end_date
            if self.schedule_board.end_date and end_date > self.
                schedule_board.end_date:
                end_date = self.schedule_board.end_date
```

```
            # Due to rrule we need to add 1 day
            end_date = end_date + datetime.timedelta(days=1)
            return rrule.rrule(rrule.WEEKLY, byweekday=[self.day],
                dtstart=datetime.datetime.combine(start_date, self.
                start_hour), until=end_date)
        else:
            end_date = self.schedule_board.end_date
            if end_date:
                # Due to rrule we need to add 1 day
                end_date = end_date + datetime.timedelta(days=1)
                return rrule.rrule(rrule.WEEKLY, byweekday=[self.day],
                    dtstart=datetime.datetime.combine(start_date, self.
                    start_hour), until=end_date)
            else:
                return rrule.rrule(rrule.WEEKLY, byweekday=[self.day],
                    dtstart=datetime.datetime.combine(start_date, self.
                    start_hour))
```

A parte de poder obtener la lista de fechas, rrule nos permite realizar 4 operaciones:

- **rrule.before(dt)**: Devuelve la última fecha generada antes de la fecha pasada como parámetro (dt).
- **rrule.after(dt)**: Devuelve la primera fecha generada después de la fecha pasada como parámetro (dt).
- **rrule.between(after, before)**: Devuelve las fechas generadas entre las fechas pasadas como parámetro (after y before).
- **rrule.count()**: Devuelve el número de fechas generadas si es posible.

Sobre estos métodos tuvimos que construir las operaciones más sofisticadas como por ejemplo obtener el próximo horario de emisión de un programa.

```
    @classmethod
    def get_next_date(cls, programme, after):
        list_schedules = cls.objects.filter(programme=programme, type='
            L')
        list_schedules = list_schedules.filter(
            programme__end_date__isnull=True) | list_schedules.filter(
            programme__end_date__gte=after)
```

```
        list_schedules = list_schedules.filter(
            schedule_board__start_date__isnull=False,
            schedule_board__end_date__isnull=True) | list_schedules.
            filter(schedule_board__end_date__gte=after)
        closer_date = None
        closer_schedule = None
        for schedule in list_schedules:
            date = schedule.date_after(after)
            if date and (closer_date is None or date < closer_date):
                closer_date = date
                closer_schedule = schedule
        if closer_schedule is None:
            return None, None
        return closer_schedule, closer_date
```

Esta operación recibe el programa a consultar (programme) y una fecha (after) que será la actual, el método solicita a la base de datos los horarios asociados al programa que queremos consultar, restringiendo ya en la misma consulta la fecha de inicio y finalización del programa y de la parrilla de programación para traer la menor información posible.

Una vez obtenidos los horarios llama al método date_after, que internamente utiliza rrule, para obtener la próxima fecha de emisión de ese horario y hace lo mismo con el resto de horarios para quedarse con la fecha más próxima.

8.2. Programa Grabador

En el programa grabador las partes más destacables tienen que ver con la gestión de los hilos, la gestión del proceso grabador y la captura de señales.

8.2.1. Hilos

El programa fue diseñado de forma que pese a surgir problemas de conexión no se interrumpiera la grabación de los programas. Por ello cuenta con varios hilos de ejecución para realizar las diferentes tareas.

8.2.1.1. Gestión de memoria

Algunos de estos hilos son creados para realizar una tarea concreta y tras ello finalizan su ejecución, si no nos preocupamos de ellos estaremos generando hilos zombis que consumirán memoria y en una aplicación de este estilo que está continuamente en ejecución provocara que se agote la memoria del sistema.

Tenemos 2 tipos de hilos que se crean y se destruyen, el hilo de grabación y el hilo de acciones post grabación.

- **Hilo de grabación:** El hilo es relativamente fácil de gestionar puesto que sólo hay uno en ejecución y no se crea el siguiente hasta que este termina.

- **Hilo de post acciones:** Este hilo es el que se ocupa de mover archivos y puede ocurrir que se encuentren en ejecución varios a la vez. Por ello es necesario almacenar los hilos en una lista y liberarlos una vez terminen.

El hilo se crea y se inicia, a continuación se mete en la lista:

```
post_actions_thread = PostRecorderThread(config = config, file_path =
    recorder_thread.file_path, file_name = recorder_thread.file_name)
post_actions_thread.start()
post_actions_threads_list.append(post_actions_thread)
```

El bucle principal del programa comprobara periódicamente si hay hilos que han acabado su ejecución y los liberara haciendo una operación join sobre ellos y a continuación los elimina de la lista:

```
def join_post_actions():
    global post_actions_threads_list
    for thread in post_actions_threads_list:
        if not thread.is_alive():
            thread.join(1)
            post_actions_threads_list.remove(thread)
            return True
    return False
```

8.2.1.2. Gestión de señales

El programa se ha pensado también teniendo en cuenta la posibilidad de que sea cerrado en cualquier momento, esto influye en la forma en la que los hilos son controlados.

El hilo principal asocia a cada señal un manejador para capturar la señal y poder llevar a cabo acciones antes de cerrar el programa:

```
...
if sys.platform == 'win32':
        signals = [signal.SIGTERM, signal.SIGINT]
    else:
        signals = [signal.SIGTERM, signal.SIGINT, signal.SIGHUP, signal
            .SIGQUIT]
    for sig in signals:
        signal.signal(sig, handler)
...

def close_all():
    if main_stop:
        main_stop.set()
    if schedules_stop:
        schedules_stop.set()
    if schedules_thread:
        schedules_thread.join()
    if recorder_thread is not None and recorder_thread.is_alive():
        logging.error('Recorder of ' + str(recorder_thread.file_name) +
            ' aborted')
        recorder_stop.set()
        recorder_thread.join()
    if upload_thread is not None:
        upload_stop.set()
        upload_thread.join()
    logging.info('Programm closed')
    sys.exit(0)

def handler(signum = None, frame = None):
    logging.debug('Signal handler called with signal:' + str(signum))
    close_all()
```

Python cuenta con un módulo llamado threading que nos proporciona todo lo necesa-

rio para gestionar hilos, concretamente usamos el método set() del objeto Event para que los hilos salgan del bucle y terminen su ejecución. Gracias a esto podemos parar los hilos y realizar un cierre correcto liberando la memoria y cerrando archivos.

8.2.1.3. Excepciones

Por lo general cada hilo captura sus propias excepciones y las documenta en el archivo de log, sin embargo no tiene mucho sentido mantener el programa en ejecución en caso de que surja algún problema en el hilo de grabación, por ello es necesario pasar esta excepción al hilo principal.

El hilo grabador captura las excepciones y las introduce en una variable que será comprobada periódicamente por el hilo principal:

```
def run(self):
        try:
                ...
        except OSError:
                e = RecorderException(msg = 'Recorder failed: Please
                    check your libraries and your command in settings.
                    ini')
                self.exceptions.put(e)
        except RecorderException as e:
                self.exceptions.put(e)
```

8.2.1.4. Pausar ejecución

Cada hilo, excepto el de grabación y de tareas post grabación, ejecuta un bucle que no termina a menos que se pare la ejecución del programa. Si no pausamos la ejecución de estos hilos tras cada vuelta en el bucle provocaremos que consuman toda la CPU disponible.

El bucle principal del programa comprueba si ha llegado la hora de empezar a grabar, en caso negativo se detiene 0.3 segundos con la función “time.sleep(0.3)”. Sin embargo el hilo de subida de archivos y el de actualización de horarios no necesitan un tiempo de espera tan bajo, en la configuración por defecto está establecido un minuto y una hora respectivamente.

El problema ocurre que si dormimos un hilo, a diferencia del hilo principal no podemos usar la opción time.sleep puesto que ignoraría la señal de cierre. Para ello Python nos proporciona un objeto llamado "Event" que sirve para la comunicación entre hilos, dispone de un estado interno que puede estar activado o desactivado y un método de espera, similar a time.sleep() con la diferencia de que si el estado interno está activado el hilo sale inmediatamente de la espera. Por tanto el bucle nos quedaría de la siguiente forma:

```
def run(self):
        while (not self.stop_event.is_set()):
                try:
                        if self.offline == False:
                                self.update_recoder_list()
                        self.load_recorder_list()
                        # time.sleep(int(self.config.get('SETTINGS', '
                            update_time')))
                        self.stop_event.wait(self.config.getint('
                            SETTINGS', 'update_time'))
                except Exception as e:
                        msg = 'Error at schedules ' + str(type(e)) + '
                            - ' + str(e)
                        print msg
                        logging.error(msg)
                        self.stop_event.wait(self.config.getint('
                            SETTINGS', 'retry_time'))
```

8.2.2. Grabación de audio

La grabación de audio se realiza en un subproceso, el comando es proporcionado desde el archivo de configuración. Además la salida del comando grabador de audio se redirige a otro comando que será el encargado de comprimir el audio y añadirle los metadatos en tiempo real.

8.2.2.1. Comandos

Los comandos por defecto en el archivo de configuración son los siguientes:

```
recorder_command:arecord --buffer-size=192000 -f S16_LE -c 2 -r 48000 -
    t raw
```

```
recorder_command_2:oggenc - -r -B 16 -C 2 -R 48000 -q 3 --title [TITLE]
    --artist [AUTHOR] --album [ALBUM] --tracknum [TRACK] --genre [
    GENRE] -c comment=[COMMENT] -o [OUTPUT]
```

Las variables están en mayúsculas entre corchetes y son sustituidas por el programa antes de ejecutar el subproceso. Esto da mucha flexibilidad a la hora de cambiar el comando de grabación.

8.2.2.2. Comunicación

Como ya se dijo anteriormente el primer comando es el que realiza la captura de audio y el segundo el que le agrega los metadatos y lo comprime. Es necesario por tanto redirigir la salida del primer comando al segundo de la siguiente forma:

```
process_1 = subprocess.Popen(self.command_1, stdout = subprocess.PIPE)
process_2 = subprocess.Popen(self.command_2, stdin = process_1.stdout)
```

Cuando llegue el momento enviamos una señal al primer proceso para que pare la grabación de audio, el segundo proceso termina automáticamente su ejecución cuando acaba de leer los datos de entrada:

```
process_1.terminate()

process_1.wait()
process_2.wait()
```

Capítulo 9

Pruebas del sistema

Las pruebas constituyen una parte fundamental del proceso de desarrollo y nos permiten verificar y validar un sistema. Las metodologías de pruebas están orientadas a probar la existencia de errores y no la ausencia de ellos, se persigue detectar el mayor número de fallos posibles.

A continuación se describen las herramientas utilizadas para llevar a cabo dichas pruebas

9.1. Aplicación Web

Django utiliza la librería estándar de Python para la automatización de las pruebas. django.test.TestCase, es una subclase de la librería de Python unittest.TestCase que corre cada test dentro de una transacción para proporcionar aislamiento.

9.1.1. Casos de prueba

Como se ha explicado anteriormente en el apartado 7.2.1.1 cada paquete contiene su módulo de pruebas.

Las pruebas no usan la base de datos real o de producción, si no que se crea una específica para correr las pruebas y se destruye una vez que estas acaban. Esto garantiza que las pruebas no estén condicionadas por datos preexistentes, también es posible crear

datos antes de cada una de ellas en el método setUp():

```
import datetime
from django.core.exceptions import ValidationError
from django.test import TestCase
from radio.apps.schedules.models import ScheduleBoard

class ScheduleBoardMethodTests(TestCase):
    def setUp(self):
        ScheduleBoard.objects.create(name="jan",
                start_date=datetime.datetime(2014, 1, 1),
                end_date=datetime.datetime(2014, 1, 31))
        ScheduleBoard.objects.create(name="1_14_feb",
                start_date=datetime.datetime(2014, 2, 1),
                end_date=datetime.datetime(2014, 2, 14))
        ScheduleBoard.objects.create(name="after_14_feb",
                start_date=datetime.datetime(2014, 2, 15))
        for schedule_board in ScheduleBoard.objects.all():
            schedule_board.clean()

    def test_runtime(self):
        jan_board = ScheduleBoard.objects.get(name="jan")
        feb_board = ScheduleBoard.objects.get(name="1_14_feb")
        after_board = ScheduleBoard.objects.get(name="after_14_feb")

        self.assertEqual(None,
        ScheduleBoard.get_current(datetime.datetime(2013, 12, 1, 0, 0,
            0, 0)))
        self.assertEqual(jan_board, ScheduleBoard.get_current(datetime.
            datetime(2014, 1, 1, 0, 0, 0, 0)))
        self.assertEqual(jan_board, ScheduleBoard.get_current(datetime.
            datetime(2014, 1, 31, 0, 0, 0, 0)))
        self.assertEqual(jan_board, ScheduleBoard.get_current(datetime.
            datetime(2014, 1, 31, 12, 0, 0, 0)))
        self.assertEqual(feb_board, ScheduleBoard.get_current(datetime.
            datetime(2014, 2, 1, 0, 0, 0, 0)))
        self.assertEqual(feb_board, ScheduleBoard.get_current(datetime.
            datetime(2014, 2, 14, 0, 0, 0, 0)))
        self.assertEqual(after_board, ScheduleBoard.get_current(
            datetime.datetime(2014, 2, 15, 0, 0, 0, 0)))
        self.assertEqual(after_board, ScheduleBoard.get_current(
            datetime.datetime(2014, 6, 1, 0, 0, 0, 0)))
```

```
def test_validation_exception_1(self):
    schedule_board = ScheduleBoard.objects.create(
            name="2_14_feb",
            start_date=datetime.datetime(2014, 2, 2),
            end_date=datetime.datetime(2014, 2, 14))
    self.assertRaises(ValidationError, schedule_board.clean)

def test_validation_exception_2(self):
    schedule_board = ScheduleBoard.objects.create(
            name="after_18_feb",
            start_date=datetime.datetime(2014, 2, 18))
    self.assertRaises(ValidationError, schedule_board.clean)
```

9.1.2. Cobertura

La cobertura del código nos permite comprobar el porcentaje de código al que accedemos desde los test. Es decir, nos permite saber cuánto código estamos realmente probando con nuestros test. Esto es fundamental para detectar zonas de código que nunca se ejecutan debido a errores de programación.

Python dispone de herramientas para realizar esta tarea. En las pruebas se ha usado el módulo coverage que se puede obtener desde pip, el sistema de gestión de paquetes de Python.

Coverage for **radio/apps/users/models** : 94%

33 statements | 31 run | 2 missing | 0 excluded

```
1  from django.contrib.auth.models import User
2  from django.core.urlresolvers import reverse
3  from django.db import models
4  from django.db.models.signals import post_save
5  from django.template.defaultfilters import slugify
6  from django.utils.translation import ugettext_lazy as _
7
8
9  class UserProfile(models.Model):
10     user = models.OneToOneField(User)
11     bio = models.TextField(blank=True, verbose_name=_("biography"))
12     avatar = models.ImageField(upload_to='avatars/', default='/static/radio/images/default-userprofi...
13     display_personal_page = models.BooleanField(default=False, verbose_name=_("display personal page"))
14     slug = models.SlugField(max_length=30)
15
16     def get_absolute_url(self):
17         return reverse('users:detail', args=[self.slug])
18
19     def save(self, *args, **kwargs):
20         if not self.pk:
21             try:
22                 p = UserProfile.objects.get(user=self.user)
23                 self.pk = p.pk
24             except UserProfile.DoesNotExist:
25                 pass
26         self.slug = slugify(self.user.username)
27         super(UserProfile, self).save(*args, **kwargs)
28
29     class Meta:
30         verbose_name = _('user profile')
31         verbose_name_plural = _('user profile')
32
33     def __unicode__(self):
34         return "%s's profile" % self.user
35
36
37 def save_slug(sender, instance=None, **kwargs):
38     if instance is not None and not kwargs.get('raw', False):
39         userprofile, created = UserProfile.objects.get_or_create(user=instance)
40         userprofile.save()
41
42
43 # Signal while saving user
44 post_save.connect(save_slug, sender=User)
```

index coverage.py v3.7.1

Figura 9.1: Cobertura del modelo

9.2. Programa grabador

Al estar desarrollado en Python las técnicas para el desarrollo de las pruebas son las mismas que las explicadas en la aplicación web.

Capítulo 10

Conclusión y futuras líneas de trabajo

Este capítulo recoge una breve reflexión sobre el trabajo realizado y cómo podría seguir mejorándose en el futuro.

10.1. Conclusiones

El producto final desarrollado cumple con todos los objetivos planteados inicialmente en el apartado 1.3:

- El sistema dispone de un editor con el que es posible gestionar la parrilla de programación, además de mostrar la información de la que dispone cada programa.
- El nuevo programa de grabación soporta toda la funcionalidad del script que la organización utilizaba.
- La grabación de los programas se realiza por la interfaz de audio elegida y a la hora proporcionada por la aplicación web.
- Las grabaciones se graban, se le asocian los metadatos, se comprimen y se suben al servidor automáticamente.
- Cada programa en la aplicación web dispone de su propio podcast.
- El software dispone de una licencia de software libre compatible con la definición de la Free Software Foundation y se distribuye bajo la licencia GPLv3.

A mayores se han conseguido unos nuevos objetivos:

- Se ha creado una página de administración que dispone de filtros y buscador.
- La parrilla de programación que se les proporciona a los clientes cuenta con la opción de visualizar por días o semanas los horarios, permitiendo ver la programación de las semanas posteriores.
- El programa grabador se ha desplegado con éxito en una Raspberry Pi.
- Se ha creado una web para el proyecto donde se puede encontrar toda la información referente a él.

La creación de este nuevo producto supone para Cuac FM y el resto de radios comunitarias una gran ayuda a la hora de automatizar el proceso de grabación de sus programas. Además la implantación del software en una Raspberry Pi ofrece nuevas posibilidades puesto que se puede adquirir este sistema de grabación automatizado por menos de 80€.

En lo que respecta a la obtención de conocimientos que se han experimentado con las tecnologías y herramientas descritas en el capítulo 3, resaltaría especialmente las adquiridas en Django, Python y JavaScript puesto que considero que me serán las más útiles en un futuro próximo.

Por otra parte enfrentarme a la planificación de este proyecto, su desarrollo y su puesta en práctica junto con el conjunto de decisiones tomadas que influyeron directamente sobre el mismo, unido a la interacción con usuarios finales ha sido una de experiencias más valiosas de este proyecto.

10.2. Futuras líneas de trabajo

De las historias de usuario que no se realizaron obtenemos la siguiente lista de mejoras:

- **Permitir cancelar episodios** Cancelar una emisión de un programa e informar de ello a los oyentes.

- **Catalogar los programas** Separar los programas que están actualmente en emisión de los que terminaron

A continuación se presenta una lista de mejoras que podrían incorporarse en el futuro:

- **Aumentar la seguridad** Securizar con https la aplicación web.
- **Ofrecer más opciones de subida** Aparte de la subida de archivos por FTP añadir otros protocolos.
- **Integrar aún más el programa grabador con la aplicación web** Hacer que el log y toda la configuración del programa grabador aparezca en la aplicación web.
- **Añadir más opciones de búsqueda** Crear un buscador de programas para usuarios no autentificados.
- **Permitir cancelar episodios** Cancelar una emisión de un programa e informar de ello a los oyentes.
- **Mejorar la interfaz de la aplicación web**

Apéndices

Apéndice A

Licencia

Para la elección de la licencia de este proyecto hay que tener en cuenta las dependencias que tiene con otras librerías y la licencia sobre que estas se distribuyen.

A.1. Introducción

La Licencia Pública General GNU es una licencia de Software Libre, esto es, una licencia que está diseñada para darle cuatro libertades esenciales, las cuatro libertades que definen al Software Libre.

Un programa es Software Libre si tú, usuario, tienes estas cuatro libertades:

- La libertad cero es la libertad de ejecutar el programa como se desee, con cualquier propósito.
- La libertad uno es la libertad para estudiar el código fuente y modificarlo para que haga lo que tu deseas o necesitas que haga.
- La libertad dos es la libertad para ayudar a otras personas, es la libertad de hacer copias y distribuirlas a otros en cualquier momento.
- La libertad tres es la libertad para ayudar a la comunidad: la libertad de publicar o distribuir las versiones modificadas cuando tú quieras.

A.2. Estudio de las dependencias del proyecto

A.2.1. Aplicación web

Gracias a que las dependencias están puestas por escrito en la carpeta requirements obtenemos la siguiente lista:

- **Python v2.7.8**[1]: Se distribuye bajo una licecia PSF[2] que es compatible con la GPL.
- **Django v1.6**[3]: Se distribuye bajo una licencia BSD, que permite su uso hasta en proyectos de software privativo.
- **python-dateutil**[4]: También se distribuye bajo una licencia BSD.
- **pytz**[5]: Dispone de una licencia MIT, muy parecida a la BSD
- **django_extensions**[6]: Cuenta con una licencia MIT.
- **MySQL-python**[7]: Se distribuye bajo una licencia GPL.
- **django-bootstrap3**[8]: Dispone de una licencia Apache Versión 2.0, compatible con la GPL.
- **djangorestframework**[9]: Cuenta con una licencia BSD compatible con la GPL
- **Pillow**[10]: Utiliza una licencia MIT para su distribución.

[1] https://docs.python.org/2/
[2] http://www.python.org/download/releases/2.7/license/
[3] https://www.djangoproject.com/
[4] https://labix.org/python-dateutil
[5] http://pytz.sourceforge.net/
[6] http://django-extensions.readthedocs.org/
[7] https://github.com/farcepest/MySQLdb1
[8] http://django-bootstrap3.readthedocs.org/
[9] http://www.django-rest-framework.org/
[10] https://pillow.readthedocs.org

A.2.2. Programa grabador

El programa grabador también cuenta con un fichero con sus dependencias:

- **Python v2.7.8**[11]: Se distribuye bajo una licecia PSF[12] que es compatible con la GPL.
- **requests**[13]: Dispone de una licencia Apache Versión 2.0, compatible con la GPL.
- **python-magic**[14]: Cuenta con una licencia MIT.
- **psutil**[15]: Cuenta con una licencia BSD compatible con la GPL

Además para la grabación de audio se usan las siguientes librerías:

- **vorbis-tools**[16]: Este paquete cuenta con una licencia GNU GPL aunque algunas librerías son distribuidas bajo BSD, por lo tanto es completamente compatible con la GPL
- **alsa-utils**[17]: Cuenta con una licencia LGPL[18] que además de ser compatible con la GPL permite su uso en programas privativos.

[11] https://docs.python.org/2/
[12] http://www.python.org/download/releases/2.7/license/
[13] http://docs.python-requests.org/
[14] https://github.com/ahupp/python-magic
[15] https://github.com/giampaolo/psutil
[16] http://www.vorbis.com/
[17] http://www.alsa-project.org/
[18] Lesser GPL

A.3. Licencia final

Este trabajo se distribuye bajo GFDL[19], la licencia de documentación libre de GNU.

Figura A.1: Logotipo de la licencia GFDL

Esta licencia, a diferencia de otras, asegura que el material licenciado bajo la misma esté disponible de forma completamente libre, pudiendo ser copiado, redistribuido, modificado e incluso vendido siempre y cuando el material se mantenga bajo los términos de esta misma licencia (GNU GFDL).

Para el software; la aplicación web y el programa grabador se eligió la licencia GPLv3, la versión 3 de la Licencia Pública General de GNU.

Figura A.2: Logotipo de la licencia GPLv3

Esta es la primera licencia copyleft para uso general, lo que significa que los trabajos derivados sólo pueden ser distribuidos bajo los términos de la misma licencia. Bajo esta filosofía, la licencia GPL garantiza a los destinatarios de un programa de computador los derechos y libertades reunidos en definición de software libre y usa copyleft para asegurar que el software está protegido cada vez que el trabajo es distribuido, modificado o ampliado.

En resumen estas licencias garantizan que este software pueda seguir mejorándose y se siga distribuyendo como software libre.

[19]GNU Free Documentation License

Apéndice B

Manual de usuario

En este apéndice se explicara el funcionamiento de la aplicación web exponiendo al lector toda la funcionalidad del sistema. Para el ejemplo se usara como dirección de la aplicación web la ruta http://demo.radioco.org/, esta ruta debe ser reemplazada por la dirección de su aplicación web.

B.1. Zona pública

Esta zona es accesible a todos los usuarios y su objetivo es informar al público de los programas que tiene lugar en la emisora, los horarios en los que se emiten, las personas que forman parte de él y los podcast que tienen.

B.1.1. Página principal

En la página principal se muestra el estado del programa actual junto con una barra de progreso que indica la duración restante. Además se listan los siguientes programas a emitir.

Figura B.1: Página principal

B.1.2. Horarios

En esta página están los horarios de emisión de los programas, cada horario redirige a la información de su episodio en caso de que no haya información de ese episodio se le envía a la página del programa.

También dispone de unos controles para cambiar entre la vista semanal o diaria y avanzar o retroceder de fecha.

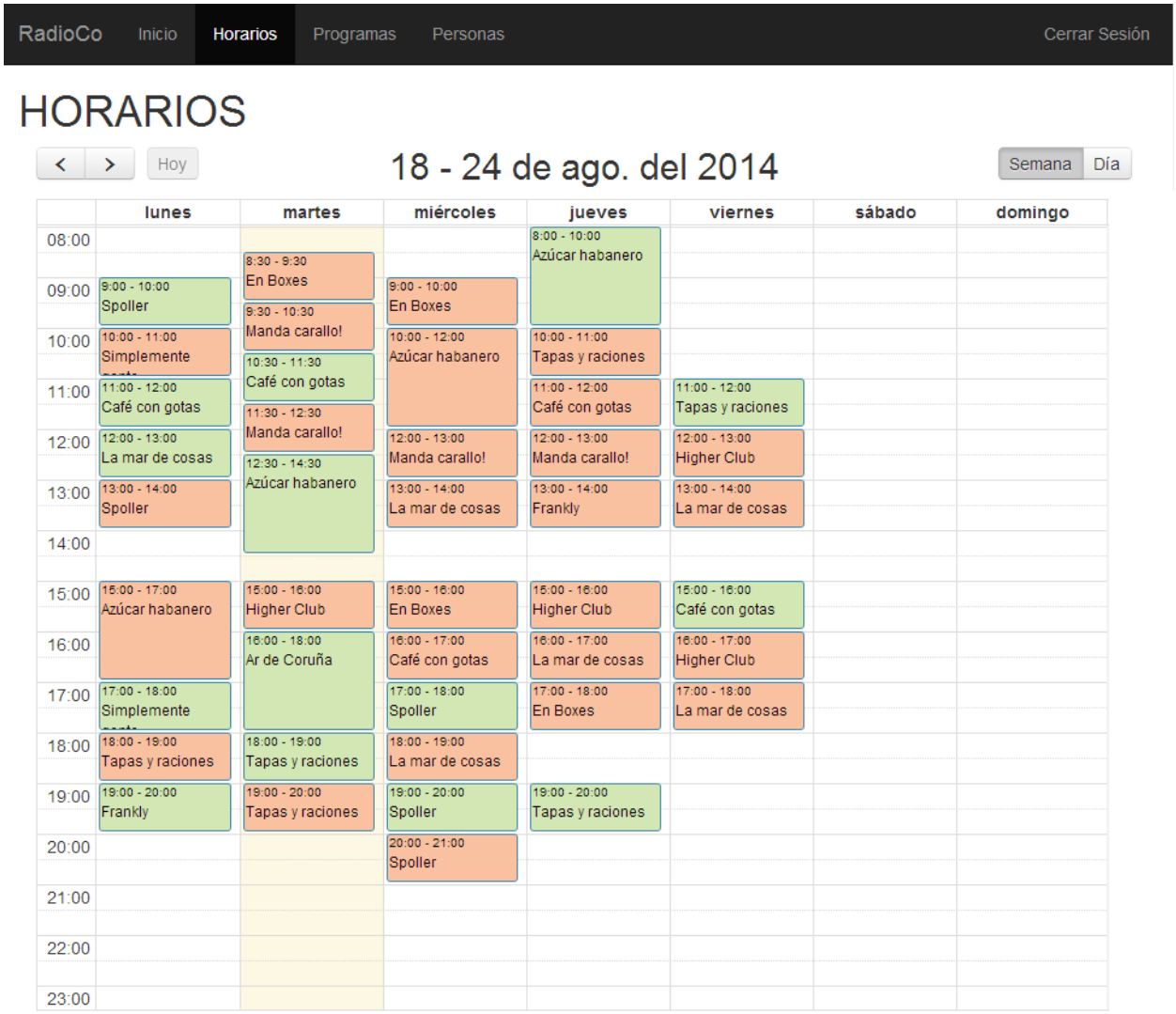

Figura B.2: Página de horarios

B.1.3. Programa

Muestra una lista con todos los programas, cada programa se acompaña de su foto junto con un resumen de su descripción.

B.1.3.1. Vista de programa

Un programa muestra la descripción del mismo, sus episodios, las personas que actualmente realizan el programa, el idioma en el que se emite además de un enlace rss

compatible con iTunes.

B.1.3.2. Vista de episodio

Un episodio contiene muestra una descripción de los temas tratados en él, además de los participantes. En caso de que esté disponible el audio introduce un reproductor html5 para permitir su reproducción directamente en la web. Si el podcast aún no se emitió informa al usuario del tiempo restante que falta para estar disponible.

B.1.4. Personal

Una lista con los perfiles de los usuarios. Para aparecer en ella el usuario tiene que activar la opción de mostrar su perfil al público (apartado B.2.1.3).

B.1.4.1. Perfil de usuario

Contiene una biografía, su foto y una descripción de lo que hace en los programas.

RadioCo Inicio Horarios Programas Personas Cerrar Sesión

Tapas y raciones

Ei aperiri assueverit vim. Ex minim delicatissimi conclusionemque est, pri dicunt malorum ex. Te quem autem usu, stet eripuit accommodare duo cu. Ne eos dicam detracto deseruisse. Cu sea error viderer mandamus. Dolor quando alterum sit ad. Nam indoctum similique quaerendum ne, est no porro semper. Elitr partiendo ne qui, ad rebum soluta option nam. Pri an vero duis soluta, atomorum hendrerit nec et, vis affert virtute indoctum cu. Unum lorem eam in, nobis tincidunt et vel. Commodo habemus ea mel, dolorum laboramus expetendis an mel, alia falli cu has. Id dicunt nonumes commune ius, tibique petentium mea an.

EPISODIOS

Temporada 1

- 1x2 Tapas y raciones
- 1x1 Tapas y raciones

PLANTILLA

Pedro Pérez

Ius altera vidisse suavitate no, in aliquando intellegebat reprehendunt qui, pro ea fabellas invenire. Ei sit dico mandamus. Ea omittantur definitionem mel, ei dicta periculis consulatu eam, an usu adhuc reprimique. Vidit veritus repudiandae ad vel, nibh ridens nonumes te mel. Sed in feugiat nostrum fabellas, dolor labitur sit ad. Alia dicant explicari eu est, vel novum possit oportere ad.

Juana Vila (Presentador)

Sed in feugiat nostrum fabellas, dolor labitur sit ad. Alia dicant explicari eu est, vel novum possit oportere ad.

ACERCA DE

Este programa lleva emitiendo desde el 1 de Julio de 2014

Idioma:

Rss:

RadioCo

Figura B.3: Programa Tapas y raciones

B.2. Zonas de administración

El sistema cuenta con dos zonas de administración

- http://demo.radioco.org/admin/ - (Figura B.10).
- http://demo.radioco.org/configuration/ - (Figura B.4).

La primera de ellas está destinada para uso exclusivo del administrador. Proporciona la creación de usuarios, grupos de permisos y configuración. La creación de programas, episodios y horarios debe hacerse a través de la segunda página de administración.

En esta página los permisos descritos en la sección B.2.2.4 funcionan de manera diferente, solo afectan los de creación, modificación y borrado además de que se muestran todos los elementos de la base de datos.

La otra página de administración es a la que redirige el sistema tras la autentificación y proporciona la funcionalidad relativa a los programas, episodios, horarios, y perfil de usuario. Cuenta con un sistema de permisos más avanzado (ver sección B.2.2.4) y un editor gráfico para la edición de horarios.

B.2.1. Zona de administración principal

Dependiendo de los permisos otorgados al usuario que se encuentre en la página es posible que se encuentren menos apartados que en la Figura B.4.

B.2.1.1. Episodio

En este apartado se encuentran todos los episodios en los que has participado. Dispone de un buscador en la parte de arriba y la opción de filtrar por la fecha de emisión y programa en el panel de la derecha.

Si un episodio de la lista aparece sin fecha de emisión se debe a que los horarios de emisión del programa han cambiado. El sistema mantiene en todo momento las horas de emisión de los episodios acorde a los horarios, esto quiere decir que si se modifican los horarios de la parrilla, las fechas de la parrilla o las fechas del programa los episodios

Figura B.4: Página principal de administración

Figura B.5: Página administración – Listado de episodios

que aún no fueron emitidos cambiaran su hora de emisión para que cuadre con la nueva programación.

En la Figura B.5 el episodio 7 no tiene fecha de emisión debido a alguno de los cambios anteriormente descritos.

Para crear un nuevo episodio tienes que estar realizando un papel en el programa.

Además no se da la opción para editar la fecha de emisión, la temporada o el número de episodio puesto que se generará automáticamente. En caso de que no haya una fecha disponible para su emisión mostrará un error.

Figura B.6: Página administración – Creación de un episodio

B.2.1.2. Programa

Dependiendo de los permisos que tengas podrás ver tus programas o todos. En este apartado puedes cambiar la foto de cabecera de tu programa, la descripción del mismo y añadir y editar la información de las personas que colaboran con el programa.

B.2.1.3. Perfil de usuario

En este lugar podrás escribir información acerca de ti. Recuerda que para que tu perfil sea público necesitas activar la casilla “Mostrar página personal”. También es el sitio indicado si quieres cambiar la contraseña.

B.2.1.4. Parrilla

Aquí puedes crear, modificar, borrar y clonar parrillas de programación. Una parrilla se considera que esta desactivada si no tiene fecha inicio ni fecha fin esto puede ser útil para conservar parrillas que ya no se usan o que se usarán en un futuro pero aún no se tienen definidas las fechas.

En caso de estar varias activas las fechas no pueden solaparse. Una parrilla sin fecha de fin significa que nunca acaba, exactamente igual que un programa sin fecha de finalización.

Para clonar una o varias parrillas tenemos que seleccionarlas en la lista y seleccionar en el desplegable de acciones la opción clonar (Figura B.7).

Figura B.7: Página administración – Clonación de una parrilla

B.2.1.5. Horarios

La edición de horarios se realiza a través de una herramienta de edición gráfica que permite crear, modificar y borrar horarios de una forma cómoda (Figura B.8).

En la parte superior se muestra el nombre de la parrilla de programación que se está editando, a la izquierda solo aparecen los programas válidos en el rango de fechas de la parrilla.

Para crear un nuevo horario basta con elegir el tipo de horario y arrastrar un programa de la izquierda al calendario. Para borrar hay que realizar el paso inverso.

Figura B.8: Editor de horarios

El sistema permite también definir de que día son las retransmisiones que se van a emitir, para ello debemos clicar en el horario que queremos y nos saldrá un menú como el de la Figura B.9 con los horarios en directo de ese programa. Tan sólo tenemos que elegir el horario deseado y guardar cambios.

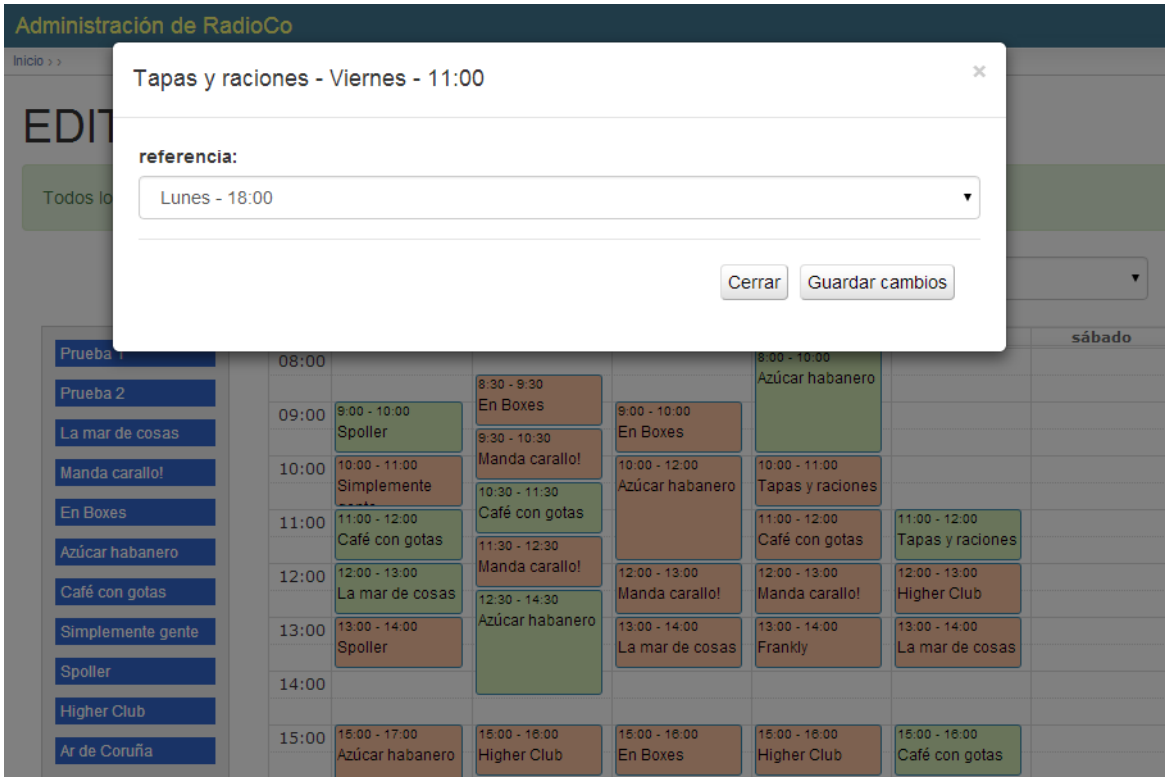

Figura B.9: Editor de horarios – Definiendo una retransmisión

B.2.2. Zona de administración privada

El administrador debe configurar las opciones globales de las que dispone la aplicación a través del panel de administración (Figura B.10).

B.2.2.1. Configuración del calendario

En este apartado están las opciones de configuración del calendario:

- **Tiempo de desplazamiento**: En caso de que el calendario completo no entre en la ventana esta opción muestra el calendario a partir de la hora introducida con el scroll ya realizado. Esto es útil si por ejemplo tus emisiones empiezan a las 8 de la mañana.

- **Día de inicio**: Permite cambiar el día en el que empieza la semana.

Figura B.10: Página privada de administración

- **Hora mínima**: En caso de que tus programas solo emitan a partir de cierta hora esta opción permite acortar el calendario mostrado.
- **Hora máxima**: Mismo caso que la opción anterior
- **Número de próximas semanas a mostrar**: El número de semanas que quieres mostrar los horarios de emisión de tus programas.

B.2.2.2. Configuración del podcast

En este apartado reúne opciones de grabación y emisión del podcast:

- **Dirección URL**: La dirección URL pública donde los archivos de audio estarán disponibles tras su subida.
- **Retraso inicial de la grabación**: El tiempo que el programa grabador debe esperar para empezar a grabar. Este tiempo se aplica a todos los programas, pospone el inicio de la grabación y descuenta este tiempo de la duración.
- **Retraso final de la grabación**: El tiempo que el programa descuenta del final del programa. Este tiempo se aplica a todos los programas y descuenta este tiempo de la duración.

- **Próximos eventos**: Los próximos horarios a grabar que serán proporcionados al programa grabador. Esta acción también provoca la creación de los que no existan.
- **Token para el programa grabador**: Un token necesario para autentificar el programa grabador, en caso de que no exista el usuario "RadioCo_Recorder" el sistema pide que se creé. No es necesario asignarle ningún permiso especial.

B.2.2.3. Configuración Global

El nombre que se muestra en la web se puede editar en este apartado.

B.2.2.4. Gestión de permisos

Una de las características que proporciona este sistema es la personalización de la zona de administración. Dependiendo de los permisos que se le asignen a un usuario podrá llevar a cabo unas acciones u otras.

Gestión de grupos de permisos El sistema permite la creación de grupos de permisos, esto significa que puede crear un grupo llamado administradores y añadirle permisos en vez de asignárselos directamente a un usuario.

Por cada objeto aparecen como mínimo 3 permisos, crear, modificar y borrar. Algunos que aparecen en esta lista son irrelevantes, los que debe tener en cuenta son los relativos a:

- **Programa**: Además del permiso para crear, editar y borrar aparece uno nuevo llamado "ver todos los programas". Dependiendo de si el usuario tiene este permiso podrá cambiar más atributos al editar un programa (ver Figura B.14). Esto sirve para evitar que un usuario pueda editar la información de su programa sin interferir con otras opciones más sensibles tales como es la duración y fecha de emisión de sus programas.
- **Episodio**: Aparecen los 3 permisos básicos, según los que se le asignen a un usuario podrá borrar y editar sus propios episodios (episodios en los que el usuario se encuentra entre los participantes).

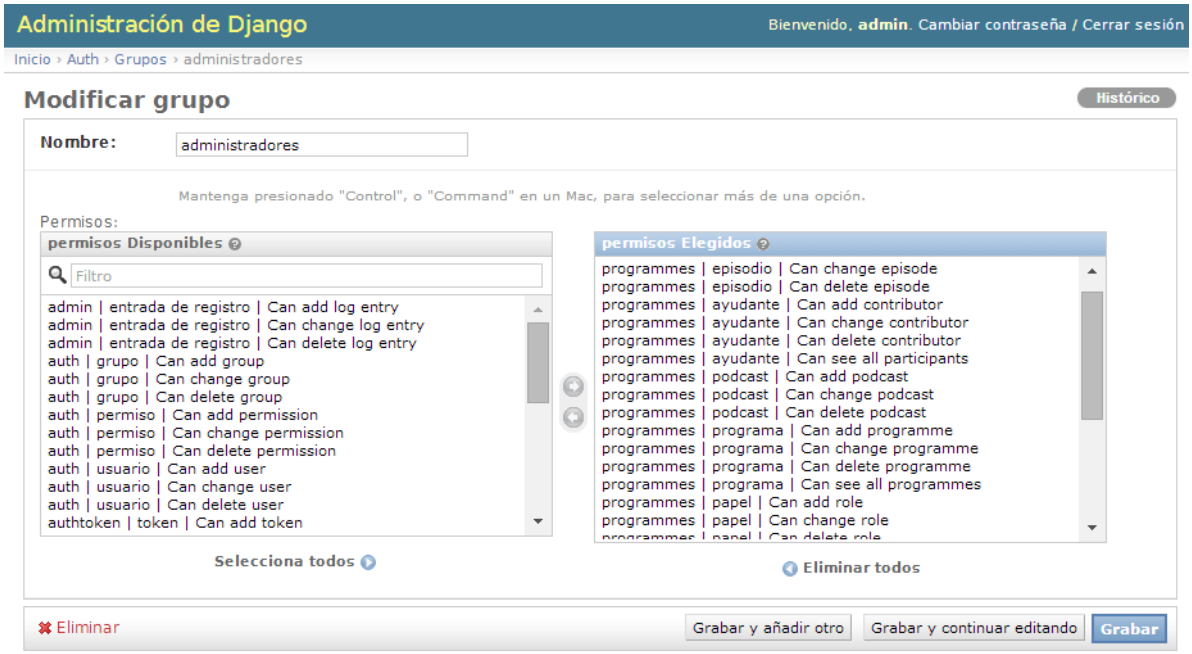

Figura B.11: Página administración – Grupo administradores

Figura B.12: Página administración – Listado de grupos

- **Papel**: Tiene un permiso adicional llamado "ver todos los papeles". Dependiendo de si el usuario tiene este permiso aparecerá un campo a mayores con la persona que hace este papel permitiendo que pueda añadir a otros usuarios a sus programas (programas en los que ese usuario tiene un papel).

  Esta información aparece asociada a cada programa.

- **Ayudante**: Funciona igual que el caso anterior solo que se refiere a un episodio,

si tiene el permiso de ver el resto de ayudantes podrá añadir nuevos usuarios a sus episodios.

- **Horario**: Aparecen los 3 permisos básicos, es necesario concederle los 3 para el uso de la herramienta de edición gráfica. Permite gestionar los horarios de las parrillas.
- **Parrilla**: Aparecen los 3 permisos básicos. Permite gestionar las parrillas de programación.
- **Perfil de Usuario**: Aparecen los 3 permisos básicos, pero sólo es necesario conceder el permiso de editar para permitir la modificación del perfil.

A continuación se muestran los permisos que sólo afectan a la página de administración del administrador:

- **Usuario**: Crear, modificar y borrar usuarios. En este apartado es donde se lleva a cabo la asignación de permisos. En la Figura B.13 se muestran la asignación de un usuario al grupo contenidos. Todos los usuarios registrados marcados como "Activo" tienen permiso para entrar en la página de administración general pero sólo los que tienen la casilla "staff" se les permiten entrar en la página de administración del administrador.
- **Configuración del calendario**: Aparecen los 3 permisos básicos, pero sólo es necesario conceder el permiso de editar. Afecta a la configuración del calendario.
- **Configuración del podcast**: Igual que el caso anterior. Afecta a la configuración de la generación del podcast y de las opciones de grabación.
- **Configuracion Global**: Opción para cambiar el nombre la web.

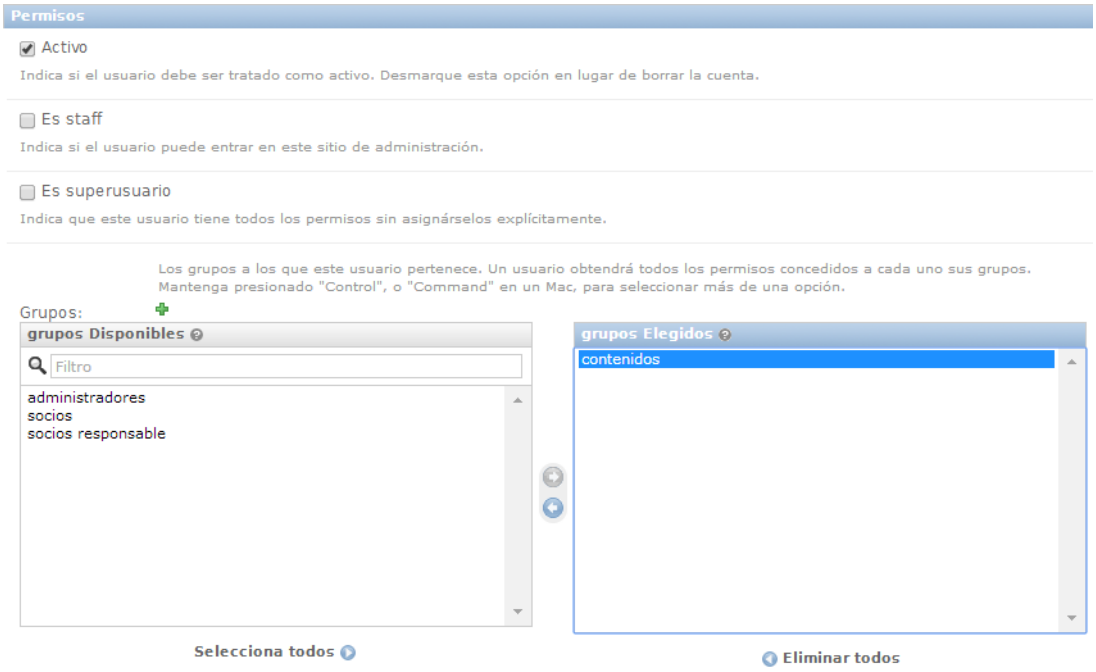

Figura B.13: Página administración – Asignación de permisos

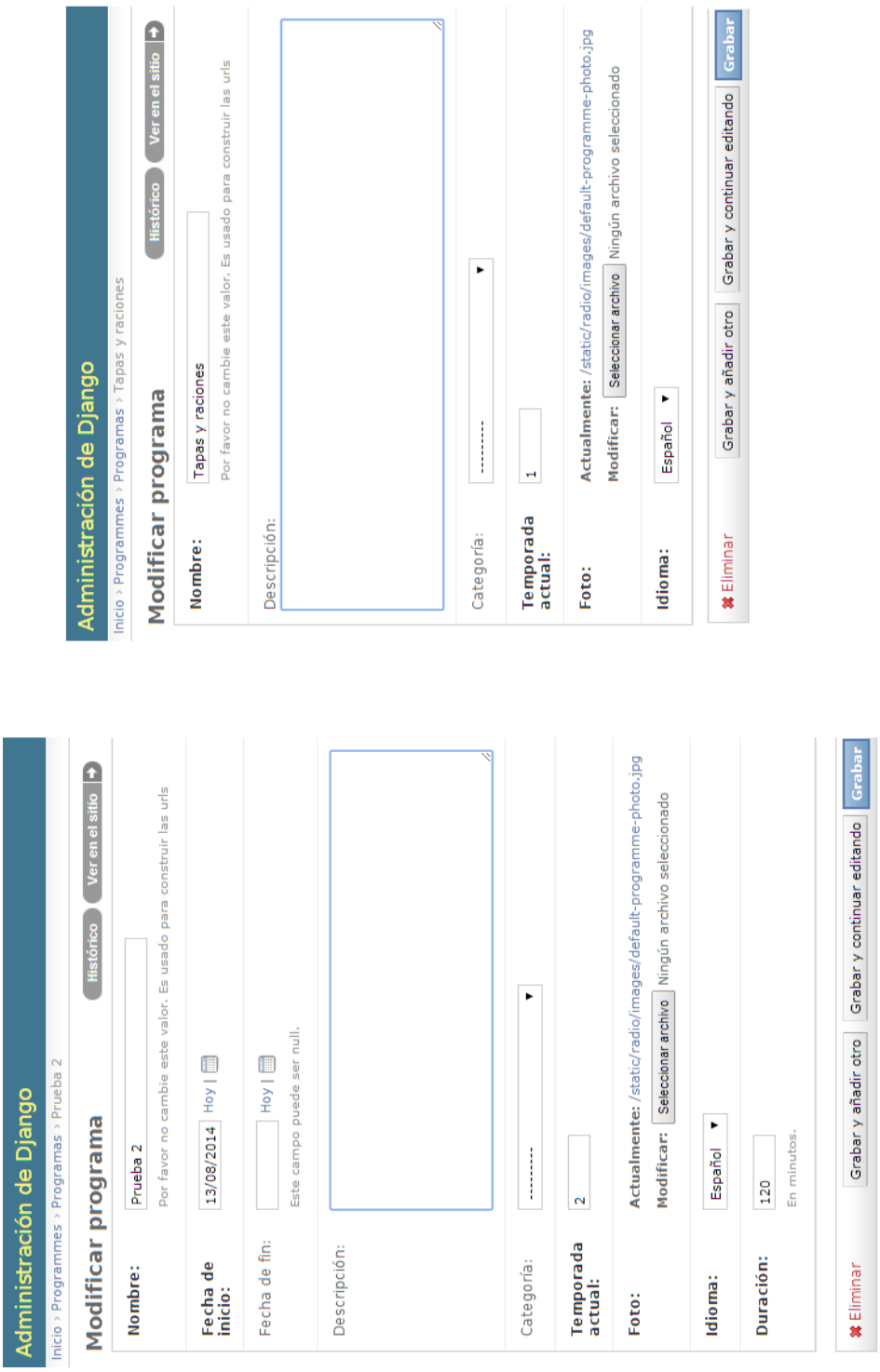

Figura B.14: Comparativa del formulario programa

Apéndice C

Instalación del Sistema

Este manual pretende describir los pasos para la instalación en las distribuciones Debian y Ubuntu.

C.1. Instalación de la aplicación web

A continuación se describirá el proceso para llevar a cabo la instalación de la aplicación web, el directorio principal parte de la raíz del sistema siendo la ruta completa "/webapps/". Esta guía no pretende explicar el uso de cada herramienta.

Para empezar creamos la carpeta donde se instalara el proyecto y nos situamos en ella:

```
mkdir /webapps
cd /webapps
```

C.1.1. Configuración de la base de datos

Primero debemos configurar la base de datos, para ello hemos elegido MySQL:

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

Creamos un nuevo usuario para la aplicación web:

```
mysqladmin -u root create djangoRadio -p
```

Nos conectamos y creamos una nueva base de datos asignándole permisos al nuevo usuario, donde “djangoRadio” es la base de datos y el nombre de usuario y “’v3rySTR0NGp4ssw0rd” la contraseña elegida al crear el usuario:

```
CREATE DATABASE djangoRadio;
GRANT ALL PRIVILEGES ON djangoRadio.* to djangoRadio@localhost
    IDENTIFIED BY 'v3rySTR0NGp4ssw0rd';
```

C.1.2. Creacion del usuario que correrá la aplicación

Creación del usuario llamado radio y del grupo llamado webapps:

```
sudo groupadd --system webapps
sudo useradd --system --gid webapps --home /webapps/django-radio radio
```

Cambiamos el propietario de la carpeta y le asignamos permisos al grupo sobre la carpeta:

```
sudo chown -R radio:users /webapps/django-radio
sudo chmod -R g+w /webapps/django-radio
```

C.1.3. Instalación del servidor web

Para instalar la aplicación web en un entorno de producción se ha usado Nginx y un entorno virtual de Python creado con virtualenv, para ello tuvimos que instalarlo de la siguiente manera:

```
sudo apt-get install nginx
sudo apt-get install python-virtualenv
```

Una vez instalados los paquetes procedemos a la descarga de la aplicación web, por ejemplo desde GitHub:

```
sudo apt-get install git-core
git clone https://github.com/iago1460/django-radio.git
```

Tenemos que editar el archivo de configuración para añadir los servidores permitidos, en este ejemplo “demo.radioco.org” y cambiar la configuración de la base de datos con la información introducida en la sección C.1.1.

```
nano /webapps/django-radio/radio/radio/settings.py
```

```
...
ALLOWED_HOSTS = ['demo.radioco.org', '127.0.0.1', 'localhost']
...
DATABASES = {
    'default': {
        'ENGINE':'django.db.backends.mysql',
        'NAME': 'djangoRadio',
        'USER': 'djangoRadio',
        'PASSWORD': 'v3rySTR0NGp4ssw0rd',
        'HOST': '127.0.0.1',
        'PORT': '3306',
    }
}
...
```

A continuación entramos en la carpeta y creamos el entorno virtual:

```
cd django-radio
virtualenv .
```

Activamos el entorno virtual e instalamos las dependencias

```
source bin/activate
sudo apt-get install python-pip
pip install -r requirements.txt
pip install gunicorn
```

C.1.3.1. Configuración de gunicorn

Creamos y editamos el archivo de configuración:

```
nano /webapps/django-radio/bin/gunicorn_start
```

```
#!/bin/bash

NAME="radio_app"                                  # Name of the
    application
DJANGODIR=/webapps/django-radio/radio             # Django project
    directory
SOCKFILE=/webapps/django-radio/run/gunicorn.sock  # we will communicte
    using this unix socket
```

```
USER=radio                                          # the user to run as
GROUP=webapps                                       # the group to run as
NUM_WORKERS=3                                       # how many worker
    processes should Gunicorn spawn
DJANGO_SETTINGS_MODULE=radio.settings               # which settings file
     should Django use
DJANGO_WSGI_MODULE=radio.wsgi                       # WSGI module name

echo "Starting $NAME as `whoami`"

# Activate the virtual environment
cd $DJANGODIR
source /webapps/django-radio/bin/activate
export DJANGO_SETTINGS_MODULE=$DJANGO_SETTINGS_MODULE
export PYTHONPATH=$DJANGODIR:$PYTHONPATH

# Create the run directory if it doesn't exist
RUNDIR=$(dirname $SOCKFILE)
test -d $RUNDIR || mkdir -p $RUNDIR

# Start your Django Unicorn
# Programs meant to be run under supervisor should not daemonize
    themselves (do not use --daemon)
exec /webapps/django-radio/bin/gunicorn ${DJANGO_WSGI_MODULE}:
    application \
  --name $NAME \
  --workers $NUM_WORKERS \
  --user=$USER --group=$GROUP \
  --log-level=debug \
  --bind=unix:$SOCKFILE
```

Damos permisos de ejecución al archivo:

```
sudo chmod u+x /webapps/django-radio/bin/gunicorn_start
```

C.1.3.2. Configuración del sistema de log

```
sudo apt-get install python-dev
pip install setproctitle
sudo apt-get install supervisor
```

Creamos y editamos el archivo de configuración:

```
sudo nano /etc/supervisor/conf.d/radio.conf
```

```
[program:radio]
command = /webapps/django-radio/bin/gunicorn_start                    ;
    Command to start app
user = radio                                                           ;
    User to run as
stdout_logfile = /webapps/django-radio/logs/gunicorn_supervisor.log    ;
    Where to write log messages
redirect_stderr = true                                                 ;
    Save stderr in the same log
```

Creación de la carpeta y archive que almacenará los logs:

```
mkdir -p /webapps/django-radio/logs/
touch /webapps/django-radio/logs/gunicorn_supervisor.log
```

C.1.3.3. Configuración de Nginx

Creamos y editamos el archivo de configuración:

```
sudo nano /etc/nginx/sites-available/radio
```

```
upstream hello_app_server {
  # fail_timeout=0 means we always retry an upstream even if it failed
  # to return a good HTTP response (in case the Unicorn master nukes a
  # single worker for timing out).

  server unix:/webapps/django-radio/run/gunicorn.sock fail_timeout=0;
}

server {

    listen   80;
    server_name ec2-54-213-150-139.us-west-2.compute.amazonaws.com;

    client_max_body_size 5M;

    access_log /webapps/django-radio/logs/nginx-access.log;
    error_log /webapps/django-radio/logs/nginx-error.log;
```

```
    location /static/ {
        alias   /webapps/django-radio/static/;
    }

    location /media/ {
        alias   /webapps/django-radio/media/;
    }

    location / {
        # an HTTP header important enough to have its own Wikipedia
            entry:
        #   http://en.wikipedia.org/wiki/X-Forwarded-For
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

        # enable this if and only if you use HTTPS, this helps Rack
        # set the proper protocol for doing redirects:
        # proxy_set_header X-Forwarded-Proto https;

        # pass the Host: header from the client right along so
            redirects
        # can be set properly within the Rack application
        proxy_set_header Host $http_host;

        # we don't want nginx trying to do something clever with
        # redirects, we set the Host: header above already.
        proxy_redirect off;

        # set "proxy_buffering off" *only* for Rainbows! when doing
        # Comet/long-poll stuff.  It's also safe to set if you're
        # using only serving fast clients with Unicorn + nginx.
        # Otherwise you _want_ nginx to buffer responses to slow
        # clients, really.
        # proxy_buffering off;

        # Try to serve static files from nginx, no point in making an
        # *application* server like Unicorn/Rainbows! serve static
            files.
        if (!-f $request_filename) {
            proxy_pass http://hello_app_server;
            break;
        }
```

```
    }

    # Error pages
    error_page 500 502 503 504 /500.html;
    location = /500.html {
        root /webapps/django-radio/static/;
    }
}
```

Habilitamos el sitio web:

```
sudo ln -s /etc/nginx/sites-available/radio /etc/nginx/sites-enabled/
    radio
```

C.1.4. Iniciando el sistema

Por último sólo nos queda compilar los idiomas y crear los archivos estáticos con los siguientes comandos:

```
/webapps/django-radio/radio/manage.py compilemessages
/webapps/django-radio/radio/manage.py collectstatic
```

Iniciando los servicios:

```
sudo supervisorctl start radio
sudo service nginx start
```

Si todo va bien tendríamos la aplicación web funcionando.

C.1.5. Problemas conocidos

Al configurar el entorno en un servidor de amazon nos encontramos con que era necesario ampliar el buffer del nombre del dominio:

```
sudo nano /etc/nginx/nginx.conf
```

```
...
server_names_hash_bucket_size 128;
...
```

Reiniciamos y debería solucionarse:

```
sudo supervisorctl restart radio
sudo service nginx restart
```

C.2. Instalación del programa grabador

Nos situamos en el directorio deseado y comenzamos la instalación.

C.2.1. Obtención del programa

Descargamos el programa de los repositorios y entramos en la carpeta:

```
sudo apt-get install git-core
git clone https://github.com/iago1460/django-radio-recorder.git
cd django-radio-recorder
```

C.2.2. Creación del entorno virtual

La instalación de un entorno virtual es opcional, a continuación se proporcionan los comandos para instalar y crear el entorno virtual:

```
sudo apt-get install python-virtualenv
virtualenv .
source bin/activate
```

C.2.3. Instalación de requisitos

Instalamos los requisitos:

```
sudo apt-get install python-dev
pip install -r requirements.txt
sudo apt-get install alsa-utils
sudo apt-get install vorbis-tools
```

C.2.4. Configuración del programa

Toda la configuración se realiza a través del archivo de configuración llamado “settings.ini”:

C.2.4.1. Configuración típica

Para realizar la conexión del programa con la aplicación web debe proporcionarle los siguientes datos:

- **url**: La dirección URL pública de la aplicación web y junto con la ruta del api, por ejemplo: "http://demo.radioco.org:80/api/1/". Es necesario mantener el mismo formato que el ejemplo.
- **token**: El token de autentificación proporcionado por la aplicación web en la página de administración.

Con los pasos anteriores el programa ya es capaz de obtener la información a grabar y comunicarle cuando ha grabado el archivo. Veamos ahora como configurar la subida del archivo mediante FTP:

- **enable**: Esta opción debemos establecerla a cierta escribiendo "True"
- **server**: La dirección del servidor ftp, respete el formato "servidorftp.com"
- **username**: El nombre de usuario del servidor FTP
- **password**: La contraseña del servidor FTP

Si no se quieren conservar los archivos grabados una vez subidos al servidor se debe activar la opción "delete_files_after_upload".

A continuación se muestra el archivo de configuración por defecto con el resto de opciones:

```
nano recorder/settings.ini
```

```
#
# Units of measurement for time: seconds
# Boolean values: 'True', 'False'
#
[SETTINGS]
offline_mode:False
url:http://localhost:8000/api/1/
token:8fdde6d703c05773084ea83e5ec2da62637666a0
```

```
# Name of files
schedule_file:schedules.txt
log_file:status.log

# Folders
recording_folder:recorders/incomplete/
complete_folder:recorders/complete/
uploaded_folder:recorders/uploaded/

# Recording
metadata_language:en
file_extension:ogg
recorder_command:arecord --buffer-size=192000 -f S16_LE -c 2 -r 48000 -
    t raw
recorder_command_2:oggenc - -r -B 16 -C 2 -R 48000 -q 3 --title [TITLE]
    --artist [AUTHOR] --album [ALBUM] --tracknum [TRACK] --genre [
    GENRE] -c comment=[COMMENT] -o [OUTPUT]

# Time is represented in seconds
update_time:3600
retry_time:120
upload_time:60
socket_timeout:300

# Upload configuration
delete_files_after_upload:False

[FTP]
enable:False
server:server.radioco.org
username:username
password:password
```

C.2.4.2. Configuración avanzada del programa

Configuración de la grabación de audio Por defecto el programa graba a 16 bits a una frecuencia de 48kHz en estéreo. Para cambiar esto es necesario consultar el manual del comando “arecord”

```
man arecord
```

Si se modifican los parámetros de grabación del comando “arecord” es necesario cambiar la configuración de “oggenc” que comprime el fichero a ogg.

```
man oggenc
```

Si quiere grabar en otro formato diferente de ogg como por ejemplo mp3 puede consultar la librería lame[1]

Configuración offline El programa grabador puede funcionar sin conexión a internet, aunque esta configuración es algo atípica permite realizar la grabación de audio si se le proporcionan los horarios, esta información se puede obtener de la siguiente dirección:

http://demo.radioco.org:80/api/1/

recording_schedules/?start=2014-09-01+00:00:00&next_hours=320

Debes sustituir “demo.radioco.org:80” por la dirección del servidor. La fecha a partir de la que quieres la información start=2014-09-01+00:00:00, siendo en este ejemplo el uno de septiembre de 2014 a las 00:00:00. Opcionalmente el parámetro “next_hours=320” sobrescribe la configuración del servidor para obtener los horarios a grabar en las próximas 320 horas.

Una vez introducida correctamente la dirección URL deberá autentificarse con las credenciales del programa grabador. La información obtenida debe ser guardada en el archivo definido en la configuración, por defecto “schedules.txt”, dentro de la carpeta del programa.

C.2.5. Ejecución del programa

Para ejecutar el programa:

```
python main.py
```

Si lo queremos dejar en Segundo plano:

```
nohup python main.py &
```

[1] http://lame.sourceforge.net/

El programa intentará seguir en ejecución a pesar de que ocurran errores. En la carpeta se puede encontrar un archivo de log, por defecto llamado “status.log”, donde se registran los errores y avisos de la aplicación.

Para finalizar la ejecución del programa basta con pulsar “Control + C” o enviarle una señal de cierre al proceso.

Apéndice D

Glosario de términos

- **AMARC**: es una organización no gubernamental internacional al servicio del movimiento de la radio comunitaria, que agrupa cerca de 4.000 miembros y asociados en más de 110 países. Su objetivo es apoyar y contribuir al desarrollo de la radio comunitaria y participativa de acuerdo con los principios de solidaridad y la cooperación internacional. En el Consejo Internacional de AMARC se encuentran representados todos los continentes.

- **Biblioteca**: Una biblioteca (usado también el término librería del inglés library) es un conjunto de subprogramas utilizados para desarrollar software. Las bibliotecas contienen código y datos, que proporcionan servicios a programas independientes, es decir, pasan a formar parte de estos.

- **BSD**: Es una licencia de software libre permisiva como la licencia de OpenSSL o la MIT License. Esta licencia tiene menos restricciones en comparación con otras como la GPL estando muy cercana al dominio público. La licencia BSD al contrario que la GPL permite el uso del código fuente en software no libre.

- **CMS**: Es un sistema de gestión de contenidos que permite crear una estructura de soporte (framework) para la creación y administración de contenidos, principalmente en páginas web, por parte de los administradores, editores, participantes y demás usuarios.

- **Entorno de desarrollo**: Un entorno de desarrollo integrado (IDE) es un programa informático compuesto por un conjunto de herramientas que permiten facilitar la

programación.

- **Framework**: Un framework es una estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, con base en la cual, otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros programas para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

- **GNU GPL**: Licencia Pública General de GNU es la licencia más ampliamente usada en el mundo del software libre y garantiza a los usuarios finales (personas, organizaciones, compañías) la libertad de usar, estudiar, compartir (copiar) y modificar el software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios. Esta licencia fue creada originalmente por Richard Stallman fundador de la Free Software Foundation (FSF) para el proyecto GNU.

- **GNU LGPL**: Licencia Pública General Reducida de GNU es una licencia de software creada por la Free Software Foundation. La principal diferencia entre la GPL y la LGPL es que la última puede enlazarse a (en el caso de una biblioteca, śer utilizada poŕ) un programa no-GPL, que puede ser software libre o software no libre.

- **MIT**: La licencia MIT es una de tantas licencias de software que ha empleado el Instituto Tecnológico de Massachusetts. Esta licencia es muy parecida a la licencia BSD en cuanto a efectos y permite su uso en programas privativos.

- **Módulo**: Un módulo es una porción de un programa de computadora. De las varias tareas que debe realizar un programa para cumplir con su función u objetivos, un módulo realizará, comúnmente, una de dichas tareas (o varias, en algún caso).

- **MVC**: Model-view-controller: es un patrón arquitectónico para implementar interfaces de usuario. Divide el software en tres componentes. El modelo es el encargado de gestionar los datos y la lógica de negocio. La vista se encarga de la representación gráfica de la información. Por último, el controlador es el encargado de poner en comunicación los otros componentes.

- **OGG**: OGG es un formato contenedor open source diseñado para proporcionar una difusión de flujo eficiente en el ámbito de la multimedia.
- **ORM**: Object-Relational Mapping: El mapeo objeto-relacional es una herramienta de programación que establece una correspondencia entre clases y objetos de un lenguaje de programación orientado a objetos con tablas y filas de una base de datos relacional siguiendo el patrón arquitectónico active record.
- **Plataforma**: Una plataforma es un sistema que sirve como base para hacer funcionar determinados módulos de hardware o de software con los que es compatible.
- **Plug-in**: Un plug-in o complemento, es una aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy específica. Esta aplicación adicional es ejecutada por la aplicación principal e interactúan por medio de la API.
- **Radio comunitaria**: Una radio comunitaria es una estación de transmisión con fines no lucrativos que no se somete a la lógica del dinero, está caracterizada por la participación y la defensa de los intereses de la comunidad.
- **ReMC**: La Red de Medios Comunitarios es un espacio que aglutina, coordina y defiende los fines de una diversidad de medios, iniciativas y prácticas de comunicación ciudadanas englobadas dentro del denominado Tercer Sector de la Comunicación (siendo el Primero el público institucional, y el Segundo el privado comercial).
- **RSS**: Se trata de un formato XML para sindicar o compartir contenido en la web. Se utiliza para difundir información actualizada frecuentemente a usuarios que se han suscrito a la fuente de contenidos.
- **Script**: Un script es un programa usualmente simple, que por lo regular se almacena en un archivo de texto plano. El uso habitual de los scripts es realizar diversas tareas como combinar componentes, interactuar con el sistema operativo o con el usuario.
- **WSGI**: Web Server Gateway Interface: es una interfaz estándar de Python para la comunicación entre servidores y aplicaciones web.

Apéndice E

Acrónimos

- *AJAX*: Asynchronous JavaScript And XML.
- *AMARC*: Asociación Mundial de Radios Comunitarias.
- *API*: Application programming interface.
- *ASCII*: American Standard Code for Information Interchange.
- *BD*: Base de datos.
- *BSD*: Berkeley Software Distribution.
- *CMS*: Content Management System.
- *CRUD*: Create Retrieve Update Delete.
- *CSS*: Cascading Style Sheets.
- *DAO*: Data Access Object.
- *DOM*: Document Object Model.
- *DRY*: Don't Repeat Yourself.
- *FTP*: File Transfer Protocol.
- *GNU*: GNU's Not Unix.
- *GPL*: General Public License.

- *HTML*: Hypertext Markup Language.
- *HTTP*: Hypertext Transfer Protocol
- *IDE*: Integrated development environment.
- *JS*: JavaScript.
- *JSON*: JavaScript Object Notation.
- *MIT*: Massachusetts Institute of Technology.
- *MVC*: Model View Controller.
- *LGPL*: Lesser General Public License.
- *ORM*: Object Relational Mapping.
- *PDF*: Portable Document Format
- *PHP*: PHP Hypertext Pre-processor
- *ReMC*: Red de Medios Comunitarios.
- *RSS*: Really Simple Syndication.
- *SQL*: Structured Query Language.
- *UML*: Unified Modeling Language.
- *WAV*: Waveform Audio File Format.
- *WWW*: World Wide Web.
- *XML*: eXtensible Markup Language
- *XP*: Extreme programming.

Bibliografía

[1] **Documentación Bootstrap**: http://getbootstrap.com/

[2] **Documentación CSS**: http://www.w3schools.com/cssref/default.asp

[3] **Apuntes Diseño de aplicaciones web**: Fernando Bellas, *Programación Avanzada*, Universidad de A Coruña, 2013.

[4] **Documentación Django**: https://www.djangoproject.com/

[5] **Apuntes Eclipse**: Javier Parapar, *Herramientas de desarrollo*, Universidad de A Coruña, 2014.

[6] **Documentación FullCalendar**: http://arshaw.com/fullcalendar/docs/

[7] **Apuntes Git**: Javier Parapar, *Herramientas de desarrollo*, Universidad de A Coruña, 2014.

[8] **Libro Git**: Scott Chacon, *Pro Git*, Apress, 1 edition, 2009.

[9] **Documentación HTML**: http://www.w3schools.com/tags/default.asp

[10] **Documentación Javascript**: http://www.w3schools.com/jsref/default.asp

[11] **Libro Javascript**: Douglas Crockford, *JavaScript: The Good Parts*, O'Reilly Media, 2008

[12] **Libro Django**: Daniel Greenfeld and Audrey Roy, *Two Scoops of Django: Best Practices For Django 1.6*, Two Scoops Press, 2014.

[13] **Libro Django**: Adrian Holovaty and Jacob KaplanMoss, *The Definitive Guide to Django: Web Development Done Right, Second Edition*, Apress, Berkely, CA, USA, 2nd edition, 2009.

[14] **Documentación de jQuery**: http://api.jquery.com/

[15] **Documentación de LaTeX** : http://latex-project.org/guides/

[16] **Documentación de MySQL**: http://dev.mysql.com/doc/

[17] **Apuntes Patrones de Arquitectura**: Laura Castro, *Arquitectura del Software*, Universidad de A Coruña, 2013.

[18] **Documentación de Python**: https://docs.python.org/2/

[19] **Libro Python**: Mark Summerfield, *Python 3*, Anaya Multimedia, 2009.

[20] **Apuntes Redmine**: Javier Parapar, *Herramientas de desarrollo*, Universidad de A Coruña, 2014.

[21] **Apuntes XP**: Pedro Cabalar, *Metodologías de Desarrollo*, Universidad de A Coruña, 2014.

[22] **Libro XP**: Kent Beck and Cynthia Andres, *Extreme Programming Explained: Embrace Change, 2nd Edition (The XP Series)*, Addison-Wesley, 2nd edition, 2004.