PROYECTO FIN DE CARRERA

Universidad Carlos III de Madrid

Ingeniería Técnica de Telecomunicación: Telemática.

Análisis, diseño e implementación de una aplicación web para gestión de información policial municipal basada en Plone.

Autora: Fátima Sánchez Cabo.

Tutora: Florina Almenárez Mendoza.

25 de Junio, 2.009

Agradecimientos

Escribir estas líneas supone una gran alegría para mí, porque por fin se pone punto y final a una etapa de mi vida, que me ha dado buenos y malos momentos y que me ha dado la posibilidad de hacer amigos que sé que son para toda la vida.

Estos amigos son los que los malos momentos los hacen más llevaderos, por eso ahora me gustaría dar las gracias a Isa y a Sara en especial, por tantas risas, confidencias y apoyo en estos años, niñas sois geniales, y a Nacho, Noelia, Jose, Emilio, Javi, David, Ángel, Alejandro, Oscar y Álvaro, por acompañarme también en este camino y hacerlo más alegre.

También me gustaría dar las gracias a mi tutora Florina, por la ayuda que me ha prestado, y la dedicación y paciencia que ha tenido conmigo.

Por supuesto, quiero dar las gracias también, a mis amigas de toda la vida, Pili, Alicia, Laura y Maria, porque han sufrido mis momentos estresantes sin una queja y porque sé que van a celebrar este momento conmigo como si fuera suyo.

Finalmente llego a las personas más importantes para mí, por un lado esta mi niño, Javi, ha sido una fuente inagotable de apoyo, comprensión, cariño y fuerza, nunca ha dudado de mis posibilidades aún en momentos donde yo no las tenía todas conmigo, y por eso sé que este momento es doblemente alegre para él, gracias por todo.

Y en último lugar quiero dar las gracias a mi familia, a mis hermanos, a mi prima Vicky, que es como otra hermana más, gracias chicos por estar conmigo y hacerme la vida más feliz, y en especial a mis padres, que gracias a su esfuerzo han hecho todo esto posible. Gracias por confiar en mí, por apoyarme, por nunca perder la esperanza, por enseñarme todo lo que sé y por consolarme cuando lo he pasado mal. Os quiero un montón y espero celebrar este momento con vosotros lo más pronto posible. Sin vosotros esto no hubiera ocurrido, GRACIAS.

RESUMEN

Todos los sectores que se encuentran dentro de las administraciones públicas tienden a automatizar el trabajo que realizan diariamente, debido a la gran cantidad de gestión que estos requieren. Esto requiere que todos los datos e información que generan deben de ser gestionados y administrados de forma segura. La automatización de este proceso supone menos papeleos y menos errores humanos.

El cuerpo de policía de un municipio es uno de estos sectores, y para esta automatización, la mayoría de ellos en España, utilizan una solución comercial, que consiste en una aplicación de escritorio. Recientemente, además de la aplicación de escritorio, existe una versión web, la cual supone una ventaja respecto a la de escritorio, en cuanto a los procesos de instalación y actualización, a la posibilidad de usarlas independientemente del puesto de trabajo, puesto que lo único necesario es disponer de una conexión a Internet, etc. No obstante, el gran desembolso de dinero al que tienen que hacer frente los municipios y si a esto le sumamos la existencia de numerosas tecnologías para el desarrollo web gratuitas de “software libre”, nos encontramos con la posibilidad de realizar una aplicación web, que sea igual de eficiente que la comercial, pero con un coste mínimo, ya que nos ahorramos el pago de licencias, soporte, etc.

Así es como surgió la idea de este proyecto. A partir de aquí tuvimos que, primero, aprender la forma de trabajar de la policía, conocer el de tipo de información que generan, los trámites que realizan, etc.; segundo, analizar y seleccionar las tecnologías más apropiadas para materializar nuestra idea, llegando a la conclusión de que un gestor de contenido web, un sistema de gestión de bases de datos y un servidor web seguro, serían las idóneas; por último, diseñar e implementar un sitio web para la gestión de información básica policial generada en un municipio que estuviese dotado de servicios de seguridad. A través de este sitio se debería permitir tanto la automatización del trabajo diario realizado por los agentes de policía como de la organización de aspectos internos de la plantilla.

El sitio web se diseña en su totalidad a partir de herramientas gratuitas de software libre, eligiendo un gestor de contenidos Web denominado Plone, por su características de gestión y seguridad.

ABSTRACT

Every sector in the civil service tends to automate the daily work, because of the management they require. This means that all data and information generated must be managed in a safe way. This automation involves less paperwork and less human mistakes.

The police department is one of these sectors, and most of them, in order to carry out this automation, use a commercial solution consisting of a desk application. Lately, Apart from the desk application, there is a web application, which provides an advantage over the desk one, in terms of installation and update, since they can be used independently from the working place, the only requirement is to have an Internet connection. If we add to this the existence of the numerous technologies for web development, freeware, we find the possibility to execute a web application, as efficient as the commercial one, but with a minimum cost, given that we save ourselves the cost of licences, medium, etc.

That is how the idea of this thesis arose. From this moment, we had to, firstly, learn the police way of working, the kind of information they generate, their procedures, etc, secondly we thought about the best technologies to carry out our idea, coming to the conclusion that a content management system, a database management system and a web server, would be the most suitable; finally, design and implement a website for managing police basic information generated in a municipality with security services. Through this site both the automation of the daily work done by police officers, and the organization of staff internal aspects should be allowed.

This website has been totally designed with freeware tools, we have chosen a content management system called Plone, because of its management and security features.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1
INTRODUCCIÓN

1 INTRODUCCIÓN

1.1 Motivación

Hoy en día, en todas las administraciones públicas se utilizan diferentes tipos de aplicaciones diseñadas por encargo, para la gestión y administración de la información, que se genera en el desempeño de sus funciones diarias.

Un sector dentro de las administraciones públicas es la policía municipal. Este sector genera diariamente información relevante acerca de los ciudadanos en forma de denuncias, actas, inspecciones, atestados, etc. Esta información debe de ser gestionada y administrada de forma segura, y este es un trabajo que los agentes han de realizar diariamente, por lo que cuanto más automatizada esté esta labor, menos tiempo se perderá en papeleos y menos errores humanos se cometerán.

En la actualidad, la mayoría de los municipios de España, utilizan una aplicación de escritorio, desarrollada con software comercial, para esta labor. Esta aplicación, no obstante, conlleva el problema de ser una solución propietario, con el consiguiente desembolso de dinero que esto supone en el pago de licencias, mantenimiento, actualizaciones, etc.

Las aplicaciones de tipo escritorio necesitan una instalación, suelen depender de la plataforma, las actualizaciones hacia nuevas versiones son más incómodas, y además para su uso, como hemos comentado anteriormente, se suele requerir un coste significativo.

Por otro lado, para el uso de aplicaciones Web, solo se necesita tener un buen navegador web, se pueden utilizar desde cualquier ordenador con una conexión a Internet, sin depender del sistema operativo que este utilice, la actualización del software es sencilla y sin riesgos de incompatibilidades, ya que existe solo una versión en el servidor, lo que implica que no haya que distribuirla entre los demás computadores, siendo un proceso rápido, limpio y transparente al usuario, y además, al funcionar en un navegador, se requiere un conocimiento básico de informática por parte del usuario.

Si juntamos las características anteriores, a la existencia de numerosas tecnologías de "software libre" para el desarrollo Web, y al ahorro que el uso de éstas supone, en desarrollo y mantenimiento, sin perder efectividad, para las administraciones públicas, nos lleva a pensar que nuestra propuesta debe ser una aplicación de este tipo.

Por todos estos motivos, surgió la idea de crear una alternativa a la anterior solución comercial, desarrollándola en su totalidad con "software libre", para aquellos municipios no muy extensos, cuya información policial es igualmente relevante, y que con presupuestos menores no pueden hacer frente a un coste elevado, y menos en los tiempos de crisis que corren. Para ello hemos contado con dos agentes de la policía municipal, pertenecientes a diferentes municipios del sur de Madrid, que nos han asesorado en cuanto a la sistemática de trabajo y gestión de información de una comisaría, además de servirnos como usuarios finales para probar la aplicación.

1.2 Planteamiento del problema

Como se ha comentado en el apartado anterior, lo que se pretende realizar en este proyecto, es un sistema de gestión policial web, en la que los agentes de policía de un municipio, puedan de forma segura, automatizar las tareas que su labor requiere día a día. Todo ello desarrollado y mantenido con el menor coste posible, de modo que cubra los siguientes aspectos:

El tipo de información policial más común que se genera en un municipio, y sobre que aspectos internos de la plantilla son más susceptibles de ser automatizados. Esta información se obtuvo gracias a contactos personales que forman parte de cuerpos de la policía municipal de diferentes municipios. De esta forma conocimos los tipos de denuncias más usuales, en cuanto a infracciones en la conducción, ocupación de la vía, etc, como funcionaban los precintos de establecimientos, que información se debía de recoger en un atestado por accidente laboral, doméstico, etc. Y en cuanto a la organización interna, las vacaciones, turnos y el uniforme son los aspectos más generales en los que centrar nuestra atención.

Teniendo en cuenta esto y las opciones disponibles en el mercado para encargarse de esta labor, nos dimos cuenta de que no existe ninguna solución no comercial en la actualidad, por lo tanto buscando el mínimo coste, nos informamos sobre las tecnologías gratuitas que podían facilitarnos esta automatización, y que eran idóneas para realizar un sitio web, en el que diferentes usuarios pudieran insertar, modificar y eliminar datos. Además el acceso web, nos libraba de estar supeditados a realizar esta gestión, solo a través de los puestos de trabajo de la comisaría, facilitándonos el acceso desde cualquier lugar o momento. Todo esto debía de hacerse de forma intuitiva y fácil para el usuario, por lo que se querían encontrar herramientas que permitieses construir formularios, a semejanza de los formularios en papel y las libretas que utiliza la policía en la calle. Con lo cual pensamos en utilizar un gestor de contenido, CMS "Content Management System", herramienta propicia para esta labor.

Otro tema era el almacenamiento de los datos. Buscamos un gestor de bases de datos, que pueda interactuar con el gestor de contenido sin problemas. La base de datos se plantea que sea externa, para dar la posibilidad de ser accedida por otros sistemas en futuras ampliaciones, como por ejemplo otras bases de datos municipales o aplicaciones del cuerpo de la policía nacional.

Finalmente, nos planteamos el tema de la seguridad, aspecto muy importante, al manejarse información personal y de carácter policial, y encima haciéndose a través de Internet. Por lo tanto se quería un acceso seguro a la base de datos que almacenase la información, que se conseguiría restringiendo el acceso a la aplicación a usuarios previamente identificados y que según sus características, podrían navegar solo por las partes de la aplicación para las que estuvieran autorizados. Y además utilizar un protocolo adecuado, para crear una aplicación segura en la transmisión de datos en Internet.

Por lo tanto, planeamos construir un nuevo sistema de gestión, a partir de un gestor de contenido y diversas tecnologías auxiliares, pero utilizando siempre software libre, buscando un rendimiento y eficiencia óptimos, al mínimo coste.

1.3 Objetivos

La finalidad de este proyecto de fin de carrera es, diseñar e implementar una aplicación Web basada en Plone, para la gestión de la información policial básica generada en un municipio, traspasando el trabajo que el agente realiza en la calle, a una forma de almacenamiento segura como es una base de datos relacional, y encargándose de la organización de aspectos internos de la plantilla.

Concretamente, los objetivos que se desean cubrir con este proyecto son:

- Analizar los requisitos funcionales y no funcionales de la aplicación con el fin de realizar el diseño óptimo, práctico y seguro, que cubra las necesidades básicas de un cuerpo de policía municipal.
- Estudiar y evaluar los diferentes gestores de contenido de software libre y multiplataforma, que abaraten considerablemente los costes de producción, licencias y mantenimiento sin perder efectividad de una aplicación web, en el mercado actual.
- Conocer el funcionamiento de un gestor de contenido.
- Crear una herramienta de trabajo diario, segura, práctica e intuitiva, orientada a un colectivo.
- Evaluar la aplicación, utilizando un usuario real perteneciente al colectivo al que va dirigido el sistema.

1.4 Contenido de la memoria

La memoria consta de siete capítulos junto con tres apéndices y la bibliografía. En ella se explica el diseño, implementación y funcionalidad del proyecto fin de carrera realizado. Los capítulos son:

El capítulo 2, contiene el **“Estado del arte”.** Dentro de él se realiza una presentación formal de los sistemas de gestión de contenido, así como sus características, ventajas y desventajas y los diferentes tipos que hay. Seguidamente se presentaran ejemplos de los más usuales actualmente, señalando de cada uno de ellos, sus características más

relevantes, lo que nos permite seleccionar uno de entre todos ellos. Y para finalizar se describe, el sistema de gestión de contenido en el que está basado este proyecto de forma minuciosa, Plone, el servidor de aplicaciones utilizado, Zope, y el lenguaje de programación usado, Python y TALES.

El capítulo 3, se encarga de **"la descripción general del sistema"**, se explica el funcionamiento de la aplicación y se especifican los requisitos tanto funcionales como no funcionales que ha de cumplir la misma, los cuales se han hecho en función de solicitudes explícitas de un agente policial real.

El capítulo 4, explica el sistema desarrollado **"SMALLPOL"**, en él se describe como ha sido propiamente la implementación, tanto de la aplicación Web, como de la base de datos externa.

El capítulo 5, describe las **"pruebas"** que se le han realizado a la aplicación para probar su correcto funcionamiento, basadas en diferentes aspectos.

El capítulo 6 es la **"historia del proyecto".** En este capítulo se relata fase a fase, la evolución del proyecto, desde que se inició su desarrollo hasta lo que ha llegado a ser en la actualidad.

El capítulo 7 trata sobre **"conclusiones y trabajo futuros".** En él se expresan las conclusiones alcanzadas una vez finalizado el proyecto. Además, se plantean posibles trabajos futuros como continuación a la línea de este proyecto.

En el **"Apéndice A"**, se muestra un presupuesto orientativo del coste que supondría la realización de este proyecto.

El **"Apéndice B"**, describe como han sido las instalaciones de las tecnologías que se han utilizado en este proyecto.

El **"Apéndice C"**, describe el manual de desarrollo para el Administrador.

Por último se muestra un glosario de términos, para aclarar la nomenclatura utilizada a lo largo de la memoria, así como la bibliografía utilizada para todo el desarrollo del proyecto.

CAPÍTULO 2

ESTADO DEL ARTE

2 ESTADO DEL ARTE

2.1 Introducción

Para este proyecto hemos utilizado un Gestor de Contenido que permite la creación de portales Web y que está disponible con licencia de "software libre", pero para poder comprender el avance que suponen este tipo de sistemas, en lo que concierne al desarrollo y mantenimiento de un sitio Web, debemos conocer en que consisten, la evolución de las herramientas utilizadas, los diferentes tipos que existen en el mercado actualmente y cuales son de cara al usuario, sus ventajas y desventajas.

Una vez conocido esto, nos centraremos en conocer las herramientas y lenguajes que en este proyecto en concreto, se han utilizado para el desarrollo e implantación y que constituyen la base principal del mismo.

2.2 Sistemas de gestión de contenidos

Realizar un Web puede ser un trabajo complicado y muy laborioso si no se dispone de las herramientas adecuadas. En el pasado las herramientas eran básicamente editores que permitían generar una página, que evolucionaron para incorporar el control de la estructura de la Web y otras funcionalidades, pero en general estaban enfocadas más a la creación que al mantenimiento. En los últimos años se ha desarrollado el concepto de sistema de gestión de contenidos, CMS "Content Management System". Se trata de herramientas que permiten crear y mantener un Web con facilidad, encargándose de los trabajos más tediosos que hasta ahora ocupaban el tiempo de los administradores de las Webs.

Teniendo en cuenta el ahorro de tiempo que supone la utilización de estas herramientas, y el coste de desarrollarlas, sería lógico esperar que su precio fuera muy elevado. Eso es cierto para algunos productos comerciales, pero existen potentes herramientas de gestión de contenidos de acceso libre, disponibles con licencias de "software libre", que funcionan perfectamente y son muy competentes en el mercado. Esta cualidad a las

administraciones públicas o ayuntamientos, les resulta especialmente interesante y deseable, puesto que la reducción de costes es uno de sus objetivos.

Los gestores de contenidos proporcionan un entorno que posibilita la actualización, mantenimiento y ampliación de la Web con la colaboración de múltiples usuarios. En cualquier entorno virtual ésta es una característica importante, que además puede ayudar a crear una comunidad cohesionada que participe más de forma conjunta.

En este apartado se describen las características más relevantes, los diferentes tipos que existen, los criterios más importantes a la hora de seleccionar un gestor de contenidos y los requerimientos en función de los objetivos que se quieran alcanzar. Por eso, se hace un breve repaso de las herramientas de código abierto que permiten construir sistemas gestores de contenido generales y se hace una particularización de aquellas más orientadas hacia la construcción de portales Web.

2.2.1 Funcionalidad de los Sistemas de Gestión de Contenidos

Los sistemas de gestión de contenidos, CMS “Content Management System” [1] son un software que se utiliza principalmente para facilitar la gestión de Webs, ya sea en Internet o en una intranet, y por eso también son conocidos como gestores de contenido Web, WCM “Web Content Management”. Hay que tener en cuenta, sin embargo, que la aplicación de los gestores de contenido no se limita sólo a las Webs.

Se propone una división de la funcionalidad de los sistemas de gestión de contenidos en cuatro categorías: creación de contenido, gestión de contenido, publicación y presentación.

- Creación de contenido:

  Un gestor de contenido aporta herramientas para que los creadores sin conocimientos técnicos en páginas Web puedan concentrarse en el contenido. Lo más habitual es proporcionar un editor de texto WYSIWYG “What You See Is What You Get”, lo que ves es lo que obtienes, en el que el usuario ve el resultado final mientras escribe, al estilo de los editores comerciales, pero con un rango de formatos de texto limitado. Esta limitación tiene sentido, ya que el objetivo es que el

creador pueda poner énfasis en algunos puntos, pero sin modificar mucho el estilo general del sitio Web.

Hay otras herramientas como la edición de los documentos en XML, utilización de aplicaciones ofimáticas con las que se integra el CMS, importación de documentos existentes y editores que permiten añadir marcas, habitualmente HTML, para indicar el formato y la estructura de un documento.

Un gestor de contenido puede incorporar una o varias de estas herramientas, pero siempre tendría que proporcionar un editor WYSIWYG por su facilidad de uso y la comodidad de acceso desde cualquier ordenador con un navegador y acceso a Internet. Un editor de este tipo que es proporcionado como una opción en Plone, es FCKeditor.

Para la creación del sitio propiamente dicho, los gestores de contenido aportan herramientas para definir la estructura, el formato de las páginas, el aspecto visual, uso de patrones, y un sistema modular que permite incluir funciones no previstas originalmente.

- Gestión de contenido:

Los documentos creados se depositan en una base de datos central donde también se guardan el resto de datos de la Web, cómo son los datos relativos a los documentos (versiones hechas, autor, fecha de publicación y caducidad, etc.), datos y preferencias de los usuarios, la estructura de la Web, etc.

La estructura de la Web se puede configurar con una herramienta que, habitualmente, presenta una visión jerárquica del sitio y permite modificaciones. Mediante esta estructura se puede asignar un grupo a cada área, con responsables, editores, autores y usuarios con diferentes permisos. Eso es imprescindible para facilitar el ciclo de trabajo “workflow” con un circuito de edición que va desde el autor hasta el responsable final de la publicación. El gestor de contenido permite la comunicación entre los miembros del grupo y hace un seguimiento del estado de cada paso del ciclo de trabajo.

- Publicación :

Una página aprobada se publica automáticamente cuando llega la fecha de publicación, y cuando caduca se archiva para futuras referencias. En su publicación se aplica el patrón definido para toda la Web o para la sección concreta donde está situada, de forma que el resultado final es un sitio Web con un aspecto consistente en todas sus páginas. Esta separación entre contenido y forma permite que se pueda modificar el aspecto visual de un sitio Web sin afectar a los documentos ya creados y libera a los autores de preocuparse por el diseño final de sus páginas.

- Presentación:

Un gestor de contenido puede gestionar automáticamente la accesibilidad del Web, con soporte de normas internacionales de accesibilidad como WAI "Web Accessibility Initiative", y adaptarse a las preferencias o necesidades de cada usuario.

También puede proporcionar compatibilidad con los diferentes navegadores disponibles en todas las plataformas (Windows, Linux, Mac, Palm, etc.) y su capacidad de internacionalización le permite adaptarse al idioma, sistema de medidas y cultura del visitante.

El sistema se encarga de gestionar muchos otros aspectos como son los menús de navegación o la jerarquía de la página actual dentro del Web, añadiendo enlaces de forma automática. También gestiona todos los módulos, internos o externos, que incorpore al sistema. Así por ejemplo, con un módulo de noticias se presentarían las novedades aparecidas en otro Web, con un módulo de publicidad se mostraría un anuncio o mensaje animado, y con un módulo de foro se podría mostrar, en la página principal, el título de los últimos mensajes recibidos. Todo eso con los enlaces correspondientes y, evidentemente, siguiendo el patrón que los diseñadores hayan creado.

2.2.2 ¿Por qué se necesita un gestor de contenido?

Hasta ahora se han mostrado bastantes motivos para ver la utilidad de un sistema que gestione un entorno Web, pero se podría pensar que no es necesario para un Web relativamente pequeño o cuando no se necesitan tantas funcionalidades. Eso sólo podría

ser cierto para un Web con unas pocas páginas estáticas para el que no se prevea un crecimiento futuro ni muchas actualizaciones, lo que no es muy realista. En cualquier otro caso, la flexibilidad y escalabilidad que permiten estos sistemas, justifican su utilización en prácticamente cualquier tipo de Web.
Muchos usuarios particulares utilizan gestores de contenido gratuitos para elaborar y gestionar sus Webs personales, obteniendo Webs dinámicos llenos de funcionalidades. El resultado que obtienen es superior al de algunas empresas que se limitan a tener páginas estáticas que no aportan ningún valor añadido.

Éstos son algunos de los puntos más importantes que hacen útil y necesaria la utilización de un gestor de contenido:

- Inclusión de nuevas funcionalidades en el Web. Esta operación puede implicar la revisión de multitud de páginas y la generación del código que aporta las funcionalidades. Con un gestor de contenido eso puede ser tan simple como incluir un módulo realizado por terceros, sin que eso suponga muchos cambios en la Web. El sistema puede crecer y adaptarse a las necesidades futuras.

- Mantenimiento de gran cantidad de páginas. En una Web con muchas páginas hace falta un sistema para distribuir los trabajos de creación, edición y mantenimiento con permisos de acceso a las diferentes áreas. También se tienen que gestionar los metadatos de cada documento, las versiones, la publicación y caducidad de páginas y los enlaces rotos, entre otros aspectos.

- Reutilización de objetos o componentes. Un gestor de contenido permite la recuperación y reutilización de páginas, documentos, y en general de cualquier objeto publicado o almacenado.

- Páginas interactivas. Las páginas estáticas llegan al usuario exactamente como están almacenadas en el servidor Web. En cambio, las páginas dinámicas no existen en el servidor tal como se reciben en los navegadores, sino que se generan según las peticiones de los usuarios. De esta manera cuando por ejemplo se utiliza un buscador, el sistema genera una página con los resultados que no existían antes de la petición. Para conseguir esta interacción, los gestores

de contenido conectan con una base de datos que hace de repositorio central de todos los datos de la Web.

- Cambios del aspecto de la Web. Si no hay una buena separación entre contenido y presentación, un cambio de diseño puede acarrear la revisión de muchas páginas para su adaptación. Los gestores de contenido facilitan los cambios con la utilización, por ejemplo, del estándar CSS "Cascading Style Sheets" u hojas de estilo en cascada [17], con lo que se consigue la independencia de presentación y contenido.

- Consistencia de la Web. La consistencia en un Web no quiere decir que todas las páginas sean iguales, sino que hay un orden (visual) en vez de caos. Un usuario nota enseguida cuándo una página no es igual que el resto de las de la misma Web por su aspecto, la disposición de los objetos o por los cambios en la forma de navegar. Estas diferencias provocan sensación de desorden y dan a entender que el sitio Web no lo han diseñado profesionales. Los gestores de contenido pueden aplicar un mismo estilo en todas las páginas con el mencionado CSS, y aplicar una misma estructura mediante patrones de páginas.

- Control de acceso. Controlar el acceso a un Web no consiste simplemente en permitir la entrada a la Web, sino que también conlleva gestionar los diferentes permisos a cada área del Web aplicados a grupos o individuos.

2.2.3 Gestores de contenido comerciales y de "software libre"

Se puede hacer una primera división de los gestores de contenido según el tipo de licencia que tengan. Por una parte están los gestores de contenido comercializados por empresas que consideran el código fuente un activo más que tienen que mantener en propiedad, y que no permiten que terceros tengan acceso, como pueden ser ADSM Portal, Content-SORT, Jarimba, Oracle Portal, etc. Por la otra tenemos los de código fuente abierto "software libre", desarrollados por individuos, grupos o empresas que permiten el acceso libre y la modificación del código fuente, como por ejemplo, Magnolia, Drupal, Joomla, Plone, Dragonfly, etc.

La disponibilidad del código fuente posibilita que se hagan personalizaciones del producto, correcciones de errores y desarrollo de nuevas funciones. Este hecho es una garantía de que el producto podrá evolucionar incluso después de la desaparición del grupo o empresa creadora.
Algunas empresas también dan acceso al código, pero sólo con la adquisición de una licencia especial o después de su desaparición. Generalmente las modificaciones sólo pueden hacerlas los mismos desarrolladores, y siempre según sus prioridades.

Los gestores de contenido de código abierto o "software libre", son mucho más flexibles en este sentido, pero se podría considerar que la herramienta comercial será más estable y coherente al estar desarrollada por un mismo grupo. En la práctica esta ventaja no es tan grande, ya que los gestores de contenido de código abierto también están coordinados por un único grupo o por empresas, de forma similar a los comerciales.

Utilizar una herramienta de gestión de contenidos de código abierto tiene otra ventaja que hace decidirse a la mayoría de usuarios: su coste. Habitualmente todo el software de código abierto es de acceso libre, es decir, sin ningún coste en licencias. Sólo en casos aislados se hacen distinciones entre empresas y entidades sin ánimo de lucro o particulares. En comparación, los productos comerciales pueden llegar a tener un coste que sólo una gran empresa puede asumir.

En cuanto al soporte, los gestores de contenido comerciales acostumbran a dar soporte profesional, con un coste elevado en muchos casos, mientras que los de código abierto se basan más en las comunidades de usuarios que comparten información y solución a los problemas. Las formas de soporte se pueden mezclar, y así encontramos gestores de contenido de código abierto con empresas que ofrecen servicios de valor añadido y con activas comunidades de usuarios. En el caso comercial también sucede, pero el coste de las licencias hace que el gran público se decante por otras opciones y por lo tanto las comunidades de soporte son más pequeñas.

Un problema que acostumbra a tener el software de código abierto es la documentación, generalmente escasa, bastante técnica o mal redactada. Este problema se agrava en el

caso de los módulos desarrollados por terceros, que no siempre incorporan las instrucciones de su funcionamiento de forma completa y entendible.

En el mercado hay gestores de contenido de calidad tanto comerciales como de código abierto. Muchos gestores de contenido de código abierto están poco elaborados (aunque en plena evolución), pero también lo encontramos entre los comerciales. En definitiva, un buen gestor de contenido de código abierto es mucho más económico que su homólogo comercial, con la ventaja de disponer de todo el código fuente y de una extensa comunidad de usuarios.

Por todos estos motivos, y como apuesta por la filosofía del software libre, en este proyecto se ha utilizado uno de estas características.

2.2.4 Tipos de gestores de contenido

Hay multitud de diferentes gestores de contenido. Los podemos agrupar según el tipo de sitio que permiten gestionar y a su vez, en si son de acceso libre o por el contrario son comerciales. A continuación se muestran los más representativos, centrándonos principalmente en los de código libre para sitios Web:

- Genéricos: Ofrecen la plataforma necesaria para desarrollar e implementar aplicaciones que den solución a necesidades específicas. Pueden servir para construir soluciones de gestión de contenidos, para soluciones de comercio electrónico, blogs, portales, etc.
  -Ejemplos de código libre: Drupal, Joomla, Plone, Apache lenya, etc.

- Foros: sitio que permite la discusión en línea donde los usuarios pueden reunirse y discutir temas en los que están interesados.
  -Ejemplos de código libre: phpBB, SMF, MyBB, etc.

- Blogs: Publicación de noticias o artículos en orden cronológico con espacio para comentarios y discusión.
  -Ejemplos de código libre: WordPress, Drupal, etc.

- Wikis: Sitio Web dónde todos los usuarios pueden colaborar en los artículos, aportando información o rescribiéndola. También permite espacio para discusiones. Indicado para material que irá evolucionando con el tiempo.
  -Ejemplos de código libre: MediaWiki, TikiWiki.

- eCommerce: Son Sitios Web para comercio electrónico.
  -Ejemplos de código libre: osCommerce, PrestaShop, etc.

- Portal: Sitio Web con contenido y funcionalidad diversa que sirve como fuente de información o como soporte a una comunidad.
  -Ejemplos de código libre: PHP-Nuke, Postnuke, Joomla, Drupal, e-107, Plone, Dragonfly CMS, etc.
  -Ejemplos comerciales: Magnolia CMS, Content-SORT, Oracle Portal, etc.

- Galería: Permite administrar y generar automáticamente un portal o sitio Web que muestra contenido audiovisual, normalmente imágenes.
  -Ejemplos de código libre: Gallery, FileBrowser, etc.

- e-Learning: Sirve para la enseñanza de conocimientos. Los usuarios son los profesores y estudiantes, tienen aulas virtuales donde se pone a disposición el material del curso, etc. La publicación de un contenido por un profesor es la puesta a disposición de los estudiantes, en un aula virtual, de ese contenido.
  -Ejemplos de código libre: Moodle, Dokeos, etc.

- Publicaciones digitales: son plataformas especialmente diseñadas teniendo en cuenta las necesidades de las publicaciones digitales, tales como periódicos, revistas, etc.
  -Ejemplos de código libre: ePrints, Thinkindot CMS, etc.

- Difusión de Contenido Multimedia (streaming): especialmente para plataformas que necesitan integrar video y sonido, tales como sitios televisiones, radios, periódico, etc.

2.2.5 Criterios para la selección de un gestor de contenido

Antes de empezar el proceso de selección de un gestor de contenido concreto, hay que tener claros los objetivos de la Web, teniendo en cuenta al público destinatario, y estableciendo una serie de requerimientos que tendría que poder satisfacer el gestor de contenido.

Los siguientes aspectos que se muestran, se han recopilado teniendo en cuenta las funciones principales de los gestores de contenido y los requerimientos básicos de una Web.

- Código abierto. Por los motivos mencionados anteriormente, el gestor de contenido tendría que ser de código fuente abierto "software libre".

- Arquitectura técnica. Tiene que ser fiable y permitir la escalabilidad del sistema para adecuarse a futuras necesidades con módulos. También tiene que haber una separación de los conceptos de contenido, presentación y estructura que permita la modificación de uno de ellos sin afectar a los otros. Es recomendable, pues, que se utilicen hojas de estilo, CSS, y patrones de páginas.

- Grado de desarrollo. Madurez de la aplicación y disponibilidad de módulos que le añaden funcionalidades.

- Soporte. La herramienta tiene que tener soporte tanto por parte de los creadores como por otros desarrolladores. De esta manera se puede asegurar de que en el futuro habrá mejoras de la herramienta y que se podrá encontrar respuesta a los posibles problemas.

- Posición en el mercado y opiniones. Una herramienta poco conocida puede ser muy buena, pero hay que asegurarse de que tiene un cierto futuro. También son importantes las opiniones de los usuarios y de los expertos.

- Usabilidad. La herramienta tiene que ser fácil de utilizar y aprender. Los usuarios no siempre serán técnicos, por lo tanto hace falta asegurar que podrán

utilizar la herramienta sin muchos esfuerzos y sacarle el máximo rendimiento.

- Accesibilidad. Para asegurar la accesibilidad de una Web, el gestor de contenido tendría que cumplir un estándar de accesibilidad. El más extendido es WAI "Web Accessibility Initiative" del WWWC "World Wide Web Consortium".

- Velocidad de descarga. Teniendo en cuenta que no todos los usuarios disponen de líneas de alta velocidad, las páginas se tendrían que cargar rápidamente o dar la opción.

- Funcionalidades. No se espera que todas las herramientas ofrezcan todas las funcionalidades, ni que éstas sean las únicas que tendrá finalmente la Web. Entre otras:
  - Editor de texto WYSIWYG a través del navegador.
  - Herramienta de búsqueda.
  - Comunicación entre los usuarios (foros, correo electrónico, chat).
  - Noticias.
  - Artículos.
  - Ciclo de trabajo "workflow" con diferentes perfiles de usuarios y grupos de trabajo.
  - Fechas de publicación y caducidad.
  - Webs personales.
  - Carga y descarga de documentos y material multimedia.
  - Avisos de actualización de páginas o mensajes en los foros, y envío automático de avisos por correo electrónico.
  - Envío de páginas por correo electrónico.
  - Páginas en versión imprimible.
  - Personalización según el usuario.
  - Disponibilidad o posibilidad de traducción
  - Soporte de múltiples formatos (HTML, Word, Excel, Acrobat, etc.).
  - Soporte de múltiples navegadores (Internet Explorer, Netscape, etc.).
  - Soporte de sindicación (RSS, NewsML, etc.).
  - Estadísticas de uso e informes.

- Control de páginas caducadas y enlaces rotos.

Para finalizar, vamos a realizar un breve comparación entre tres de los ejemplos mencionados en apartados anteriores sobre CMS de "software libre". Los tres ejemplos son Plone, y otros dos gestores de contenido muy en alza actualmente en el mercado, como son Joomla [4] y Drupal [3]. De esta forma, se ha realizado la selección del gestor de contenido a utilizar en este proyecto, ayudándonos para dicha decisión, en el estudio realizado por Michelle Murrain, Laura Quinn y Maggie Starvish [2]. La comparación puede verse en la Tabla 1.

Plone vs Drupal vs Joomla:

CARACTERÍSTICAS	Joomla	Drupal	Plone
Facilidad de Hosting e Instalación			
Facilidad para crear un sitio simple			
Facilidad para aprender a configurar un sitio más complejo			
Facilidad en el uso de la administración de contenidos			
Flexibilidad gráfica			
Flexibilidad estructural			
Uso de roles y workflows			
Comunidad/Funcionalidad Web 2.0			
Ampliación e integración			
Seguridad y escalabilidad			
Mantenimiento del sitio			
Soporte/Fuerza de la comunidad			

Excelente Muy Bueno Bueno

Tabla 1: Comparación de tres CMS- Joomla vs. Drupal vs. Plone.

A continuación comentaremos cada uno de los aspectos comparados:

- Facilidad de Hosting e Instalación: Tanto Joomla como Drupal, y sobre todo Drupal tienen una facilidad mayor a la hora de realizar su instalación y Hosting. Plone requiere un entorno de alojamiento especial, para el que se necesitan más conocimientos técnicos a la hora de la instalación que para Joomla o Drupal.

  Aunque en el momento en que queramos un entorno de alojamiento más sofisticado, con mayor soporte de tráfico o necesidades avanzadas, las diferencias de instalación entre los tres CMS se minimizan.
- Facilidad para crear un sitio simple: Joomla ofrece un gran número de temas, apariencias, pero la creación de la estructura del sitio y la navegación no es tan intuitiva como podría ser, incluso para un experto en web. En Plone es fácil configurar la navegación por el sitio, y es el único que ofrece un calendario de eventos fuera de la caja, pero la instalación de un nuevo tema es técnicamente complicado. La navegación y configuración de un tema en Drupal es sencilla, pero a menos que se tengan conocimientos de HTML, se necesita instalar un WYSIWYG ("lo que ves es lo que obtienes"). Se necesita por tanto, añadir un módulo que actúa como editor para que se pueda editar texto o imágenes en el sitio.
- Facilidad para aprender a configurar un sitio más complejo: En Joomla se requiere un poco de esfuerzo para comprender su terminología y estructura, pero una vez hecho, es relativamente sencillo crear un sitio bastante avanzado. Drupal va más allá en complejidad, hay un gran número de opciones, pantallas, etc, por lo que la flexibilidad del sistema te hace pensar la mejor manera de hacer las cosas, antes de hacerlas. Plone es el más complicado de aprender, consta de un complejo grupo de opciones y alternativas, y para configuraciones avanzadas, también debes de conocer el funcionamiento de la aplicación que se encuentra por debajo, que es Zope.
- Facilidad en el uso de la administración de contenidos: Plone destaca en ese aspecto. Aprender a administrar el contenido es relativamente fácil, además presta soporte para pegar texto directamente de Microsoft Word. Drupal es más simple para editar páginas sencillas, pero una vez que has instalado un módulo para permitir la edición, sin conocimientos de HTML. Joomla es el que tiene la

interfaz más pulida y amigable, pero requiere un poco de aprendizaje para añadir nuevas páginas o secciones para la navegación.

- Flexibilidad gráfica: Estos tres sistemas permiten crear apariencias personalizadas, que controlan el diseño, el tipo de letra, los colores, etc, de sus páginas. Pueden realizar un control extremo sobre su apariencia, soportando casi cualquier diseño gráfico.
- Flexibilidad estructural: Plone y Drupal son fuertes en este ámbito. Ambos permiten crear tipos de contenido, estructuras, y determinar que contenido va en una determinada página de un determinado lugar del sitio. Joomla por el contrario, ofrece alguna flexibilidad pero no es tan potente como los anteriores en cuanto a la creación de nuevos tipos o localización dentro del sitio. Además Joomla solo soporta tres niveles de jerarquía: secciones, categorías y artículos.
- Uso de roles y workflows: Plone es el más potente en este área. Permite el mayor nivel de control sobre roles de usuario, permisos de cada usuario, y una detallada configuración del flujo de los contenidos a través del sistema. El administrador del sitio puede establecer roles personalizados a los que se les asignan los permisos que se deseen. Hay además módulos que permiten asignar permisos según el contenido o la categoría. Pero Drupal no tiene esta poderosa configuración de serie, necesita añadir módulos o plug-ins. Joomla por su parte, tiene un número relativamente pequeño de tipos de usuarios definidos, no se pueden definir diferentes roles de usuarios, y los usuarios solo tienen acceso a su propio contenido.
- Comunidad/Funcionalidad Web 2.0: Drupal brilla en este ámbito. Fue diseñado desde el principio para ser una plataforma de comunidad, ofreciendo perfiles, blogs y comentarios de serie, con mejoras disponibles a través de módulos añadibles. Joomla y Plone ofrecen menos características de comunidad en este sentido, pero gracias a módulos añadibles ofrecen cierto soporte.
- Ampliación e integración: Plone y Drupal ofrecen flexibilidad en cuanto al soporte de diferentes tipos de contenido y estructuras en el sitio, por lo tanto pueden soportar más fácilmente inusuales complementos que se les añadan, que Joomla, el cual, no soporta esta flexibilidad. Por otro lado todos soportan la posibilidad de conectarse con bases de datos constituidas y que no las tengan integradas de serie.

- Seguridad y escalabilidad: Plone es el mejor en este sector. Tiene muy pocos informes de vulnerabilidades de seguridad y es inmune a los ataques de inyección SQL. Tanto Drupal como Joomla tienes mas informes, pero esos son corregidos sin demora publicando actualizaciones del sistema.
- Mantenimiento del sitio: Plone es el más complejo de los sistemas en cuanto a la actualización, no publica actualizaciones muy a menudo. Joomla ha publicado una única actualización en el último año. Drupal es que publica más frecuentemente actualizaciones.
- Soporte/Fuerza de la comunidad: Los tres sistemas cuentan con una red importante de desarrolladores, diseñadores y consultores que proveen un soporte pagado. Están disponibles varios buenos libros que son adecuados para el comienzo tanto de los usuarios como de los desarrolladores. La ayuda en todos los sistemas esta disponible ahora y en el futuro previsiblemente.

En líneas generales tanto Drupal como Plone podían haber sido la opción válida para este proyecto. Aunque Plone cuenta con el inconveniente de que la puesta en funcionamiento y la creación de un sitio complejo para principiantes, resulta más complicado que con Drupal, a nosotros esta característica no nos resulta relevante puesto que ya contamos con algunos conocimientos sobre este sistema, tanto a nivel de usuario como de administrador. De forma, que este inconveniente queda solventado en nuestro caso.

Si unimos lo anterior, a que Plone es el sistema de los tres que mayor nivel de seguridad ofrece, solventado problemas como la utilización de técnicas de inyección de SQL, ataque que es neutralizado de forma automática, gracias a los conectores que utiliza para conectarse a bases de datos de SQL, ataques de denegación de servicio (DoS), a los que se hace prácticamente impenetrable si se configura detrás de un servidor como Apache, etc. Y a que gracias a la gran flexibilidad para la definición de roles y de asignación de permisos y “workflows”, se evitan muchas vulnerabilidades de seguridad, puesto que las capacidades de cada usuario están muy controladas, delimitando el acceso de los usuarios finales a partes concretas y siempre fuera de la configuración de seguridad principal.

Por todo se elige Plone, porque es idóneo para aplicaciones complejas, es potente, está altamente probado y consta del menor número de vulnerabilidades, ofreciendo el mayor nivel de seguridad, característica en la que en un sistema policial hay que poner especial énfasis.

2.3 Plone

Los Gestores de Contenido incluyen de todo, desde software para foros como PHP-BB, hasta programas para blogs como Wordpress o sistemas complejos como Typo3. Muchas de estas herramientas están basadas en PHP. Como la mayoría de los servicios de hospedaje Web soportan actualmente PHP, las aplicaciones CMS basadas en PHP se están expandiendo. Las herramientas PHP, sin embargo, rápidamente llegan al límite si se requiere un alto nivel de seguridad, una gestión potente de acceso de los usuarios y los grupos de trabajo y una gestión de los procesos dirigida por el flujo de trabajo. En los últimos años, el CMS elegido para desarrollar sitios Web más sofisticados ha sido Plone, que está basado en el servidor de aplicaciones Web Zope.

En este proyecto se utiliza la versión 3.0. Plone 3.0 [5] hereda la reputación de sus antecesores de alta seguridad y capacidades avanzadas de flujo de trabajo. Esta última versión también viene con nuevas características, como la edición en línea, comprobación de enlaces y referencias e indexación de texto completo de los documentos Word y PDF.

Zope es un sistema profesional y además es la competencia basada en Python de los sistemas de desarrollo potentes del mundo Java, como JBoss, Websphere o Weblogic. Con su arquitectura basada en componentes, Zope proporciona un pilar sólido para Plone, haciéndolo realmente interesante para las aplicaciones de negocio críticas. Al igual que Zope, Plone también está basado en Python, un lenguaje de programación orientado a objetos reconocido por su elegancia y simplicidad. El código Python es fácilmente legible y útil en proyectos en los que los grupos de trabajo son importantes. Python también posee una de las comunidades de desarrollo de código abierto más grandes y activa.

Plone puede ejecutarse en la mayoría de los sistemas operativos, incluyendo Windows, Linux y Mac OS X, existiendo instaladores que se encargarán de instalar todo de forma automática. Para instalaciones profesionales, probablemente se prefiera recompilar todos los componentes, es decir, Python, Zope y Plone, desde el código fuente. Plone 3.0 requiere la versión 2.4.4 de Python y la 2.10.4 de Zope.

No requiere una base de datos relacional, ya que el sistema utiliza la base de datos de objetos integrada de Zope, ZODB. ZODB es compatible con ACID; soporta opciones para deshacer, replicar y funcionalidades de copias de seguridad, pudiendo utilizar ZEOs "Zope Enterprise Objects" como un servidor de base de datos central, para servir los datos a múltiples instancias en paralelo de Zope para balancear la carga. Esto hace que Plone sea altamente escalable, pero si se necesita utilizar una base de datos SQL, como es el caso de este proyecto, se pueden usar adaptadores de objetos relacionales para vincularlos con Plone.

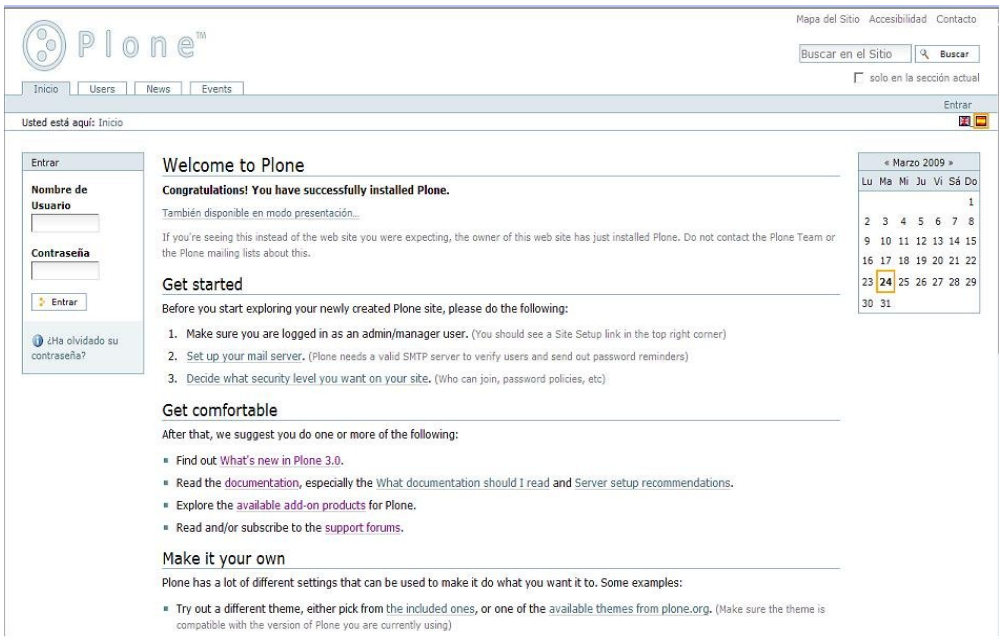

Figura 1: La GUI del CMS Plone tras su instalación. Seleccionando la opción Get Started se accede a las funciones críticas del sistema de gestión.

2.3.1 Características principales

Edición Simple

Plone posee una GUI bastante intuitiva. La edición está integrada con la maquetación normal del sitio, en vez de requerir una interfaz de edición independiente. Con esto, los usuarios pueden ver inmediatamente los efectos de los cambios realizados, en lugar de tener que pelearse con las herramientas de navegación de un sistema de edición independiente.

Si se edita un artículo, éste se muestra simplemente con un marco adicional que contiene los menús de edición requeridos. Suponiendo que se tengan los permisos, se puede hacer clic para modificar los elementos aislados en una página, como el título, la descripción y el cuerpo del texto, para editarlos sin afectar al resto de la página. Este método es realmente útil si se necesitan realizar algunos cambios editoriales rápidos. Este efecto se consigue haciendo uso de Ajax, que Plone lo soporta gracias KSS "Kinetic Style Sheets".

Plone almacena la información en objetos denominados "article types" (AT). Para cada tipo de artículo existe un esquema, que los usuarios pueden mostrar como un formulario en el que se pueden introducir los datos. El sistema es lo suficientemente flexible como para manejar diferentes tipos de información; algunos de ellos se representan en la tabla 2.

TIPO DE ARTÍCULO	DESCRIPCIÓN
Folder	Utilizado para estructurar el sitio Web.
Page	Comprende un título, una descripción y un cuerpo de texto
News	Posee una estructura similar, pero también incluye una figura, que se muestra en la página de resumen de la noticia.
Image	Para almacenar, escalar y mostrar formatos gráficos.
File	Para almacenar archivos Word, PDF y de otros formatos, que los usuarios podrán descargar.
Link	Para almacenar enlaces con comentarios.
Date	Para fechas, eventos, etc.
Collection	Carpeta virtual que puede mostrar contenido en base a patrones de búsqueda complejos.

Tabla 2: Algunos de los tipos de artículos estándar de Plone.

En todos ellos, el contenido se clasifica por medio de un sistema de metadatos que es compatible con los esquemas Dublin Core. Los metadatos permiten enlazar diferentes contenidos en el sitio Web o encontrar el contenido por medio de búsquedas con objetivos. Por ejemplo, las colecciones pueden listar los elementos haciendo referencia a sus metadatos.

Extensiones para los Tipos

Las extensiones de Plone proporcionan tipos de artículos adicionales, por ejemplo, encuestas o contenido multimedia, o bien añaden funciones adicionales a los tipos de artículos ya existentes. Los desarrolladores pueden utilizar la infraestructura arquetipo para definir sus propios tipos de artículos individuales; en la mayoría de los casos, hay disponible interfaces Web que simplificarán esta tarea.

Además de la gestión de contenidos basada en el navegador, Plone también soporta clientes Webdav y FTP. Una carpeta Plone puede mostrarse en una máquina cliente como un directorio del sistema de ficheros por medio de Webdav. De este modo, se pueden almacenar documentos de Office directamente en Plone ya sean de Word o de OpenOffice.

Opciones de Entrada

Los usuarios pueden elegir campos, menús desplegables o botones de tipo radio para introducir sus entradas. Plone también posee múltiples opciones para crear el cuerpo del texto con títulos, listas con viñetas, imágenes integradas y enlaces. Soporta lenguajes de etiquetado como Restructured Text, Markdown y Textile, que se convierte a HTML tras guardarlo. Estos formatos de entrada son muy valorados, porque ofrecen una solución simple, rápida y eficaz de introducir comandos de formato por medio del teclado.

La mayoría de los usuarios preferirán el editor de textos Kupu, que posee un menú que recuerda al de la mayoría de los menús de los programas de procesamiento de textos. Este editor se puede observar en la Figura 2:

Figura 2: El editor Kupu integrado permite a los usuarios sin conocimientos de HTML introducir y editar el contenido.

Además de los diversos estilos que trae por defecto, Kupu contiene herramientas para introducir tablas, enlaces, imágenes y vídeos. Plone elimina antes las etiquetas HTML potencialmente peligrosas antes de guardar. Kupu se ejecuta en los navegadores Internet Explorer y Mozilla, pero no es compatible con Konqueror, Safari ni Opera.

Gestión de Acceso

Gracias a su esquema de gestión de acceso, Plone soporta la colaboración entre grandes grupos de usuarios en un entorno Web seguro. La edición colaborativa de la información por un lado y la confianza por el otro son requerimientos que demandan una máxima flexibilidad por parte del CMS en un sitio Web grande. Para manejar esto, Plone posee una gestión de acceso basada en roles que permite dar a los usuarios individuales o a los grupos de usuarios privilegios de acceso de forma granular, ya sean globalmente para el sitio Web completo o localmente para una sección específica del sitio.

De forma sencilla, los roles son una capa de abstracción que contiene una colección de permisos específica y granular que Zope y Plone 3.0 soportan bajo la superficie. Como manejar una gran colección de permisos individuales es difícil, los roles proporcionan una clase de contenedor intuitivo. Los administradores pueden activar y desactivar los permisos en los roles en cualquier momento y de este modo cambiar la configuración. Plone 3.0 define los siguientes roles por defecto:

- Miembro es un usuario registrado.
- Lector puede leer el contenido.
- Contribuidor puede añadir contenido.
- Revisor puede editar el contenido.

- Editor puede publicar el contenido.
- Gerente puede modificar la estructura del sitio.
- Propietario es el usuario que creó originalmente el contenido.

Los roles se pueden asignar de forma individual a los usuarios o a los grupos de forma global o de forma local para una zona específica del sitio.

Flujos de Trabajo

Relacionado de forma directa con la gestión del acceso está la capacidad de asignar los flujos de trabajo. En Plone, los flujos de trabajo definen el alcance de un proceso editorial organizado. El contenido puede pasar por diferentes estados, por ejemplo, privado o público. Simplificando, el contenido privado sólo puede ser visto por sus creadores, mientras que el público es accesible por todos los visitantes del sitio Web. Los usuarios con los permisos apropiados pueden modificar los estados o transiciones.
Un flujo de trabajo consiste en una colección de estados y transiciones. Plone 3.0 viene con cuatro flujos de trabajo diferentes preconfigurados:

- Flujo de trabajo de Estado Único: El contenido se publica inmediatamente y está visible para todos los usuarios.
- Flujo de trabajo de Publicación Simple: Los propietarios pueden publicar el contenido ellos mismos.
- Flujo de trabajo de Intranet: El contenido sólo será visible para los usuarios registrados. Los editores pueden publicar internamente; si fuera necesario, el contenido puede hacerse accesible a los visitantes anónimos del sitio Web.
- Flujo de trabajo de la Comunidad: El contenido se hace accesible al público por los editores.

Estos flujos de trabajo son cuanto se necesita para obtener tanto sitios Web privados como grandes intranets empresariales. En un sitio Web se pueden desplegar varios flujos de trabajo en paralelo; es decir, se puede definir un flujo de trabajo global para el sitio Web completo, pero puede asignarse fácilmente a tipos de artículos individuales (por ejemplo, noticias, imágenes o fechas) o simplemente asignarlos localmente a secciones específicas del sitio Web.

El administrador del sitio posee bastante grado de libertad para modelar las necesidades de los usuarios. Por ejemplo, puede instalarse una zona que sólo sea accesible a un grupo de trabajo interno y asignar un flujo de trabajo de estado único para permitir a los usuarios acceder de forma inmediata a la información. Otra zona del sitio podría estar gobernada como una intranet de múltiples estados o como un flujo de trabajo de tipo comunidad, para asegurarse de que el contenido se aprueba antes de publicarse.

Versiones y Eventos

Plone 3.0 posee un sistema de control de versiones. Para editar un artículo existente, el usuario primero crea una copia de trabajo y edita el contenido, mientras que la versión original permanece visible al público en general. Tras completar la edición, el autor puede publicar la nueva versión, comparar las dos versiones e incluso volver a una versión anterior. Los artículos se bloquean para impedir que otros usuarios realicen cambios mientras permanecen abiertos para su edición.

Una de las principales características nuevas de Plone 3.0 es el control de eventos, que automatiza varios eventos críticos. Los administradores pueden ligar eventos como crear, modificar y borrar los artículos, o transiciones de los flujos de trabajo con otras acciones del CMS. Incluyendo la realización de copias o el envío de notificaciones por correo electrónico. Las reglas que gobiernan cómo tienen que suceder las acciones también pueden incluir condiciones. Se pueden especificar las condiciones en un menú y asignarlas a artículos y carpetas individuales.

Gestión de Enlaces

El contenido movido o borrado puede producir enlaces huérfanos. Plone 3.0 resuelve este problema por medio de una comprobación automática integrada. Cuando se mueve el contenido, Plone envía peticiones del navegador al nuevo destino. Si alguien borra un artículo que está referenciado por otras páginas, el usuario verá un mensaje de aviso y podrá cancelar la operación o abrir los documentos que contienen la referencia para su edición.

Las funciones de búsqueda ayudan a los usuarios a encontrar la información. Zope almacena el contenido y los metadatos en índices separados, en los que se pueden realizar búsquedas de forma independiente. Los usuarios pueden iniciar una búsqueda

rápida en todos los catálogos, y si poseen un navegador compatible con Javascript, los resultados se mostraran mientras el usuario teclea las palabras clave.

Si la búsqueda no devuelve los resultados esperados, la búsqueda avanzada permitirá a los usuarios definir criterios de búsqueda más complejos y restringir las coincidencias por tiempo, estado, palabras clave, autor o tipo de artículo. En ambos casos, está soportado el uso del comodín * y de los operadores lógicos *AND* y *OR*. Si los paquetes wvware o xpdf/pdftotext están instalados en el servidor, Zope también podrá buscar dentro de los ficheros Word y PDF.

Mapas de Sitio Google

Como Plone 3.0 soporta el estándar para los mapas de sitios utilizados por la mayoría de los buscadores como Google, Yahoo o MSN para encontrar páginas nuevas de forma más rápida y eficiente, la cooperación entre los sitios Plone y los motores de búsqueda es muy eficiente.

ZPT “Zope Page Templates” permite a los usuarios crear páginas HTML modulares en Plone. Zope lleva utilizando desde hace bastantes años plantillas de páginas para crear sitios Web dinámicos y esta característica está bien documentada. Si desean diseñarse plantillas nuevas, en Internet existen multitud de herramientas y HOWTOs. Una página estándar está formada por varias plantillas. Para personalizarlas, pueden modificarse o simplemente cambiar la forma en que se disponen.

Las plantillas son las responsables de mostrar el contenido; la lógica de la aplicación, como la búsqueda de contenidos específicos, reside en scripts externos, de este modo se asegura una separación estricta del contenido, la lógica de la aplicación y la representación.

Plone es compatible con XHTML y está basado en CSS. Cumple con los estándares internacionales W3C WAI-AA/WCAG 1.0 y los requisitos de la ley US (Sección 508) relativa a la accesibilidad de los discapacitados. Los diseñadores Web pueden utilizar las hojas de estilo en cascada para modificar la apariencia de cada elemento individual de la página Web, por muy básico que sea. Lo que significa que se puede modificar radicalmente la apariencia predefinida por Plone simplemente modificando las hojas de estilo.

Una hoja de estilo especial permite a los usuarios imprimir las páginas de Plone sin los molestos elementos de navegación. La versión 3.0 también posee una hoja de

estilo S5 que representa el contenido en un navegador para su presentación por medio de un proyector de vídeo. La GUI de Plone se ha traducido a más de 50 idiomas y la extensión Linguaplone permite a los editores mantener el contenido en varios idiomas. Plone crea una copia para cada traducción; la copia se enlaza con el original.

Identificación Única

Plone posee su propio mecanismo de autenticación. Para implementar un método de identificación única, se le puede añadir una extensión para LDAP u OpenID; se admiten otros mecanismos de autenticación centrales pero requieren trabajo manual extra.

Plone 3.0 ofrece a los usuarios registrados una página de bienvenida personalizada o cuadro de mandos. Desde aquí se puede acceder a la información del sitio Web de forma individual y mostrar, por ejemplo, las últimas noticias o cualquier contenido en una vista personalizada.

A pesar de las funciones nuevas, Plone es más rápido que nunca. Las pruebas con el banco de pruebas de Apache mostraron mejores valores para Plone 3.0 comparados con la versión v2.5.3. Además, las páginas HTML generadas por Plone 3.0 tienen un tamaño menor que las creadas con la versión anterior.

Más Rápido con Caché

Las peticiones de páginas dinámicas que ejecutan scripts y generan una página HTML final a partir de una colección de plantillas devoran recursos. Para contrarrestar este efecto, Plone posee varias funciones caché que reducen el tiempo requerido por las rutinas más complejas.

Aunque Zope posee su propio servidor Web, Plone normalmente se instala con Apache o con un Proxy caché como Squid o Barniz. ZEOs permite a los administradores distribuir la base de datos y repartir la carga entre los servidores Zope para escalar un sitio Web Plone sin tener que modificar su estructura.

La complejidad y la potencia de Plone no lo excluyen para su uso en aplicaciones simples. En particular, la seguridad que el servidor de aplicaciones Zope ofrece hace que Plone sea interesante para sitios pequeños, especialmente cuando se considera que proporciona un CMS profesional con sólo presionar un botón. Por supuesto, Plone es ideal para grandes sitios Web empresariales e institucionales.

Plone se licencia bajo GPL (GNU Public License). La marca Plone y el código del programa pertenecen a la Fundación Plone, lo que dará tranquilidad a aquellos usuarios que están pensando en invertir tiempo en él.

En conclusión, Plone 3.0 posee numerosas características nuevas que hacen que el trabajo con un CMS sea mucho más sencillo para los usuarios. La gestión de acceso para los usuarios y los grupos son uno de sus puntos fuerte, y la nueva versión presenta mejoras significativas. También facilita el camino a los administradores para implementar aplicaciones complejas, y al estar basado en Zope hace que sea una apuesta segura en cuestión de seguridad.

2.3.2 Arquetipos

Como hemos comentado anteriormente, se puede utilizar la infraestructura arquetipo [6] para definir tus propios tipos de contenido individuales. Utilizando esta opción, los formularios se generan de forma automática, se automatizan las transformaciones de contenido, se pueden definir campos personalizados, etc.

La definición de los campos del formulario, se hace dentro de un “schema”, el cual contendrá los campos del formulario, validadores, atributos, etc. Un ejemplo de un “schema” sencillo podemos verlo en la figura 3.

```
from Products.Archetypes.config import PKG_NAME
from AccessControl import ClassSecurityInfo
from Products.CMFCore.permissions import ModifyPortalContent
from Products.Archetypes.public import MetadataSchema
from Products.Archetypes.public import RichWidget
from Products.Archetypes.public import ReferenceField
from Products.ATReferenceBrowserWidget.ATReferenceBrowserWidget import
ReferenceBrowserWidget

produccion = BaseSchema.copy() + Schema((
    StringField('anio',
               required=0,
               widget=StringWidget(label="Año"),
    ),
    ImageField('imagen',
        required = 0,
        searchable = 1,
        widget = ImageWidget(label = "Imagen"),
     ),
    StringField('descripcion',
            widget = StringWidget(label="Descripcion"),
             searchable = 1,
             required = 1,
                ),
    TextField('cuerpo',
            widget = RichWidget(label="Cuerpo", rows=25),
             searchable = 1,
             required = 0,
                ),
    FileField('fichero',
            widget = FileWidget(label="Archivo de música"),
             searchable = 1,
             required = 0,
                ),
    ))
class Produccion(BaseContent):
    """ discográfica """
    security = ClassSecurityInfo()
    archetype_name = "Producción musical"
    meta_type = "Produccion"
    portal_type = "Produccion"
    global_allow = 1
    immediate_view = 'produccion_view'
    default_view = 'produccion_view'
    at_rename_after_creation = True
    schema = produccion
registerType(Produccion, PKG_NAME)
```

Figura 3: Ejemplo del “schema” que representa un arquetipo.

Posteriormente habría que crear un producto que instalase ese nuevo tipo de contenido, con la creación de una serie de ficheros de inicialización, configuración, test, etc. Es decir, con una estructura como la que se muestra en la figura 4.

```
- __init__.py
- configure.zcml
- config.py
- interfaces.py
- content
    - __init__.py
    - message.py
- profiles
    - default
- browser
    - __init__.py
    - configure.zcml
    - instantmessage.pt
- tests
    - __init__.py
    - base.py
    - test_setup.py
```

Figura 4: Estructura de ficheros que conforman un producto.

Por lo tanto para formularios simples, esta técnica reduce la curva de aprendizaje. Pero por el contrario, para formularios extensos, que además van relacionados con otros, y cuyos datos deben de almacenarse y recuperarse de una base de datos externa, no de la ZOBD, como es el caso de este proyecto, la curva de aprendizaje aumenta considerablemente. Además al tratarse de formularios, en algunos casos, con gran cantidad de campos, los arquetipos no ofrecen mucha flexibilidad en la colocación de los campos dentro del formulario. Y si a esto le sumamos que es necesario crear un producto que instale este nuevo tipo de contenido, la complejidad aumenta todavía más, puesto que la creación del producto no es algo trivial.

Por todo esto, en este proyecto no se optó por la utilización de esta técnica.

2.3.3 Lenguaje de TAL/TALES/METAL

TAL, “The Template Attribute Language”, es un lenguaje de plantillas cuyo objetivo es la generación de páginas HTML y XML. Se centra principalmente en simplificar la colaboración entre programadores y diseñadores, para que utilizando herramientas comunes se creen plantillas en HTML y XML correctas. Fue creado para Zope pero también es usado en proyectos basados en Python, como Plone.

Las plantillas TAL, se utilizan a menudo en páginas de resultado, como es nuestro caso, para reemplazar el contenido de una variable por el de un atributo especial.

Estas plantillas se suelen utilizar en conjunción con las TALES, “TAL Expression Syntax”, que se utilizan por ejemplo en implementaciones basadas en Python, para

poder utilizar expresiones en Python, dentro de las páginas en HTML y por extensión XHTML.

Algunas de estas expresiones son:

- <tal:define=....> Crea variables locales.
- <tal:condition=...>Actúa como un “if”, y decide si se muestra o no el contenido de la etiqueta.
- <tal:repeat=...> Actúa como un “for” y crea un bucle en el que se va repitiéndola etiqueta dentro de una secuencia, se utiliza por ejemplo, para mostrar datos en una tabla.

Y si además añadimos el uso de METAL “Macro Expansion TAL”, es posible la reutilización de código entre plantillas. Algunas de estas expresiones son:

- <html metal:define-macro=...> Crea un nuevo marco.
- <html metal:define-slot=...> Crea un espacio dentro de un marco.

2.3.4 Seguridad en Plone

Plone utiliza un modelo de seguridad sofisticado, basado en roles, lo que da seguridad a sus contenidos. Este modelo es el modelo básico de seguridad de Zope, pero además, Plone, le añade el concepto de usuarios y grupos, los cuales son formas convenientes de manejar roles y por tanto, permisos, para un número de usuarios simultáneo.

Plone pone a disposición de los usuarios, roles y grupos con mucha flexibilidad. También es posible manejar localmente la seguridad y no solamente al nivel del conjunto del sistema. Es decir un usuario puede ser administrador de la zona "/financiero" al tiempo que sólo es miembro restringido de "/soporte".

Si a este control sobre los contenidos, le añadimos una navegación segura con HTTPS, utilizando otro servidor como frontal que soporte SSL, estaremos creando un sitio Web muy robusto y fiable frente a ataques de seguridad externos.

2.3.5 Arquitectura general

A continuación se muestra la relación de elementos que conforman la arquitectura de desarrollo de este proyecto. Esta relación la mostramos de forma esquemática en la figura 5, acompañada de una leyenda explicativa. La figura pretende mostrar como se corresponden uno a uno los elementos que componen las tecnologías, para formar un todo, que posibilita el desarrollo completo del sistema de gestión Web.

Estas tecnologías que complementan a Plone, se describen en profundidad en los apartados posteriores.

Para situarnos inicialmente, se puede ver como Python es el núcleo de todo el sistema, puesto que las diferentes tecnologías están escritas en este lenguaje, y como hay una capa que se corresponde con los componentes de Zope (servidor de aplicaciones) y otra que envuelve a todo, que se corresponde con Plone y los diferentes componentes que interactúan con Zope, como la base de datos externa y el adaptador necesario para interactuar con ella.

Plone 3.0→ -CMS.
-Productos.
-Interfaz de Usuario.

Zope 2.10.4→ -Servidor Web.
-Servidor Aplicaciones.
-Motor de búsqueda.
-Base de Datos de Objetos (ZOBD).

Python 2.4.4→Lenguaje de Programación.

Plug-ins→ Productos a través de los cuales se extiende Zope.

Psycopg1→ Adaptador para la conexión Postgre-Plone.

PostgreSQL 8.3→ Gestor de la Base de Datos externa, Postgre 8.3.

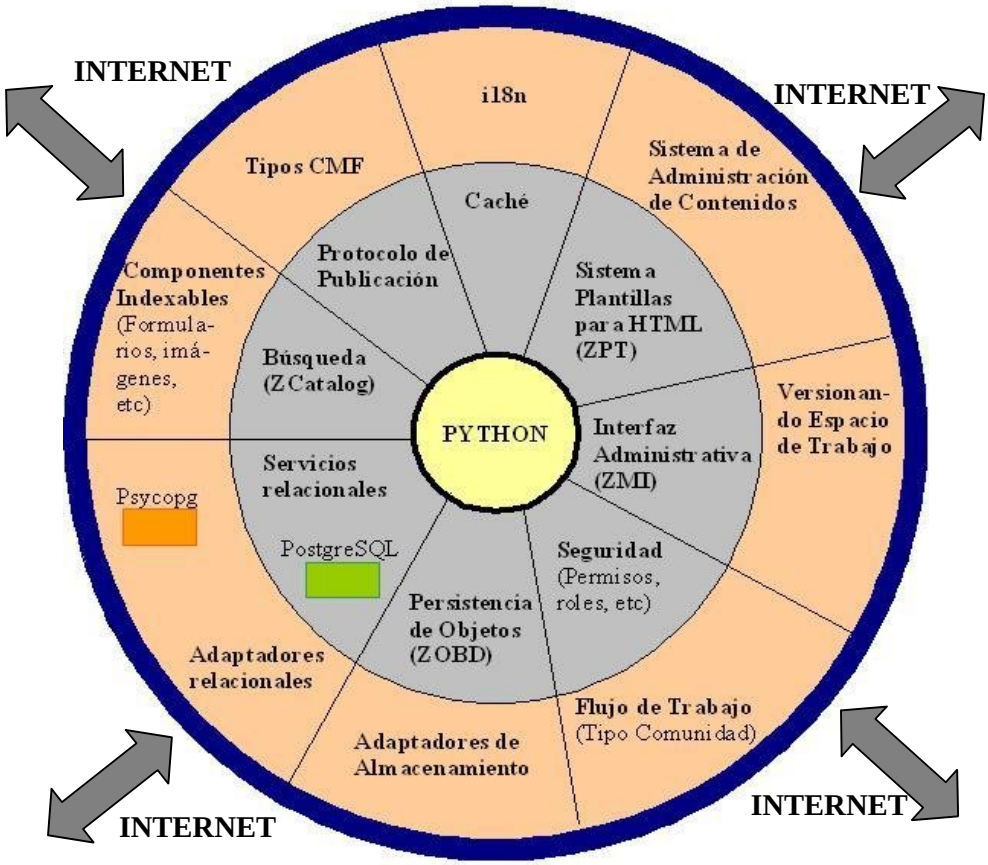

Figura 5: Arquitectura Python-Zope-Plone.

2.4 Sistema de gestión de Bases de datos

Como se ha comentado anteriormente, a la hora de plantearnos como desarrollar este proyecto, se necesita almacenar toda la información que se va recopilando a través de la aplicación de una forma segura y organizada, para su posterior consulta. Por esto necesitamos un gestor de bases de datos, SGDB "Database Management System".
Un SGDB es un programa que permiten almacenar y posteriormente acceder a los datos de forma rápida y estructurada, sirviendo de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan.
Los sistemas de gestión de bases de datos, que utilizan los gestores de contenido que se han sopesado para este proyecto, son MySQL y PostgreSQL. MySQL se publica bajo una licencia Dual, y PostgreSQL bajo una BSD, por lo tanto su uso puede ser libre en ambos casos. Los dos sistemas utilizan el lenguaje estándar de programación SQL.

PostgreSQL[8] es un sistema de gestión de base de datos relacional orientada a objetos, ORDBMS, "Object-relational Database Management System", ya que incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional.
Más claramente un ORDBMS consiste en lo siguiente:

- Es un sistema, que simplemente pone una interfaz orientada a objetos sobre una base de datos relacional RDBMS, "Relational Database Management System". Cuando las aplicaciones se comunican con este tipo de base de datos, lo habitual es que la comunicación se haga como si los datos hubieran sido almacenados como objetos. Sin embargo, el sistema convierte la información de los objetos, a tablas de datos con filas y columnas y maneja los datos de la misma forma que una base de datos relacional. Del mismo modo, cuando los datos se recuperan, deben de ser reensamblados y pasar, de simples datos a objetos complejos.

  Debido al trabajo adicional que realiza la base de datos para esta conversión de los datos, la velocidad de ejecución de la base de datos se degrada considerablemente.

  Por otro lado el principal beneficio de este tipo de base de datos radica en el hecho de que el software para realizar la conversión se proporciona, por lo que no es necesario que los programadores escriban el código para ello y además el

acceso a la base de datos es fácil a partir de un lenguaje orientado a objetos, como puede ser en nuestro caso Python.

.

MySQL [10] es un sistema de gestión de bases de datos relacional propiamente dicho. Este gestor de bases de datos es, probablemente, el gestor más usado en el mundo del software libre y se caracteriza principalmente por su velocidad de ejecución.

Seguidamente se muestra una breve comparación entre MySQL y PostgreSQL.

MySQL vs. PostgreSQL:

Lo mejor de PostgreSQL:

- Posee una gran escalabilidad. Es capaz de ajustarse al número de CPUs y a la cantidad de memoria que posee el sistema de forma óptima, haciéndole capaz de soportar una mayor cantidad de peticiones simultáneas de manera correcta.
- Implementa el uso de rollback's, subconsultas y transacciones, haciendo su funcionamiento mucho más eficaz, y ofreciendo soluciones en campos en las que MySQL no podría.
- Tiene la capacidad de comprobar la integridad referencial, así como también la de almacenar procedimientos en la propia base de datos, equiparándolo con los gestores de bases de datos de alto nivel, como puede ser Oracle.

Por contra, los mayores inconvenientes que se pueden encontrar a este gestor son:

- Consume gran cantidad de recursos.
- Tiene un límite por fila, aunque se puede aumentar, con una disminución considerable del rendimiento.
- Es de 2 a 3 veces más lento que MySQL.

Lo mejor de MySQL:

- Su velocidad a la hora de realizar las operaciones, lo que le hace uno de los gestores que ofrecen mayor rendimiento.
- Su bajo consumo lo hacen apto para ser ejecutado en una máquina con escasos recursos sin ningún problema.

- Las utilidades de administración de este gestor son envidiables para muchos de los gestores comerciales existentes, debido a su gran facilidad de configuración e instalación.
- Tiene una probabilidad muy reducida de corromper los datos, incluso en los casos en los que los errores no se produzcan en el propio gestor, sino en el sistema en el que está.

Los mayores inconvenientes son:

- Carece de soporte para transacciones, rollback's y subconsultas con el motor que viene por defecto.
- El hecho de que no maneje la integridad referencial, hace de este gestor una solución pobre para muchos campos de aplicación, sobre todo para aquellos programadores que provienen de otros gestores que sí que poseen esta característica.
- No es viable para su uso con grandes bases de datos, a las que se acceda continuamente, ya que no implementa una buena escalabilidad.

En nuestra aplicación particular, quizá le hemos dado más importancia a la potencia y escalabilidad, que a la velocidad, y en esto PostgreSQL es mejor. La información policial generada en un municipio, puede llegar a ser enorme y más si miramos hacia el crecimiento de población que se va produciendo cada año, esto, junto con que se van a producir accesos bastante a menudo, y que para seleccionar cierto tipo de información es mucho más práctico utilizar subconsultas, es lo que nos hace decantarnos por PostgreSQL en este caso.

2.5 Zope

Zope [11] es un servidor de aplicaciones web de código abierto, escrito principalmente en el lenguaje de programación Python.

Zope puede ayudarle a crear aplicaciones web dinámicas, tales como sitios de intranet y portales rápidamente. Zope viene con todo lo necesario, incluido el apoyo de una comunidad, búsquedas, y noticias.

Para las funciones de edición de contenidos, así como personalizaciones básicas, puede ser usado mediante un navegador web. La programación avanzada así como el desarrollo de nuevas funcionalidades requiere la edición de componentes.
Un sitio web de Zope está compuesto de objetos en lugar de archivos, como es normal en la mayoría de servidores web. Las ventajas de usar objetos en lugar de archivos son:

- Combinan el comportamiento y los datos en una forma más natural que los archivos de texto plano.
- Alientan el uso de componentes estándares que se ocupan de una parte particular de las que forman una aplicación Web, permitiendo flexibilidad y buena descomposición.
- Posibilitan procesos automáticos de gestión de información.

2.5.1 Características principales

- Lo más característico de Zope es su base de datos orientada a objetos, llamada ZODB "Zope Object Database". Esta base de datos almacena objetos ordenados en un sistema similar a un sistema de ficheros, pero cada objeto tiene propiedades, métodos u otros objetos, estos objetos pueden ser datos personalizados, plantillas dinámicas HTML, scripts, un motor de búsqueda, código, etc. Esta aproximación es muy diferente de las base de datos relacionales habituales. Sin embargo, Zope dispone de múltiples conectores para las diferentes bases de datos relacionales y ofrece sistemas básicos de conexión y consulta abstrayéndolos como objetos.

- Zope cuenta con un modelo de desarrollo a través de la web, muy consistente, lo que te permite poder actualizar tu sitio Web desde cualquier parte del mundo. Esta flexibilidad es gracias a un hermético modelo de seguridad.
  Existen numerosos productos (plug-in de componentes de Zope) disponibles para descargar y ampliar el conjunto básico de herramientas de creación de sitios con Zope. Estos productos incluyen nuevos objetos de contenido, base de datos relacionales, conectores de bases de datos externas, herramientas avanzadas de gestión de contenidos y aplicaciones completas de comercio electrónico, contenidos y gestión de documentos, seguimiento de errores, etc.

- Zope incluye su propio HTTP, FTP, WebDAV y las capacidades de XML-RPC, pero también puede ser utilizado con Apache y otros servidores web.

- Zope fue originalmente escrito por la Zope Corporation. Algunas partes de Zope a eran código abierto en 1996, y todo el servidor de aplicaciones desde 1998. La Zope Corporation todavía dirige la visión primaria y el desarrollo de Zope, siendo cada versión mantenida y desarrollada por más y más miembros de la comunidad, o bien mediante el envío de parches para errores o mediante comprobaciones de las nuevas características por parte de los desarrolladores con acceso. La sección de desarrollo del sitio de Zope contiene los proyectos en curso y las propuestas de Zope en el pasado y el futuro.

- Dentro de los usuarios de Zope se encuentran Viacom, Red Hat, la NASA, la Marina de los EE.UU, ishopehere.com, Storm Linux, etc. Existe una larga lista de proveedores de soluciones de Zope, desarrolladores de todo el mundo y listas de correo disponibles para la ayuda en el desarrollo, despliegue o cualquier otro tema, de aplicaciones Zope.

Actualmente existen dos ramas principales, Zope2 y Zope3. Este último es una reimplementación del servidor Zope, donde se ha tratado de volcar toda la experiencia adquirida en Zope2. Zope3 no trae compatibilidad hacia atrás, por lo que los componentes hechos para Zope2 no funcionan. Aún se está en un proceso de adaptación hacia este nuevo Zope, para lo cual está usando un componente llamado “five”, con el cual desde Zope2 pueden tener la facilidad de Zope3.

2.6 Python

Python [12] es un lenguaje de programación creado a principios de los años 90, por Guido van Rossum en los Países Bajos, como sucesor de un lenguaje llamado ABC. Guido sigue siendo el principal autor de Python, aunque incluye multitud de contribuciones de otros.

Se compara habitualmente con TCL, Perl, Scheme, Java y Ruby. Python es considerado como la "oposición leal" a Perl, lenguaje con el cual mantiene una rivalidad amistosa, pero con una sintaxis muy limpia y que favorece un código legible.

Permite dividir el programa en módulos reutilizables desde otros programas Python. Viene con una gran colección de módulos estándar que se pueden utilizar como base de los programas o como ejemplos para aprender Python. También hay módulos incluidos que proporcionan E/S de ficheros, llamadas al sistema, sockets y hasta interfaces a GUI (interfaz gráfica con el usuario) como Tk, GTK, etc.

Es un lenguaje de programación multiparadigma. Esto significa que permite varios estilos: programación orientada a objetos, programación estructurada y programación funcional. Otros muchos paradigmas más están soportados mediante el uso de extensiones. Python usa tipo de dato dinámico y “reference counting” para el manejo de memoria. Una característica importante de Python es la resolución dinámica de nombres, lo que enlaza un método y un nombre de variable durante la ejecución del programa (también llamado ligadura dinámica de métodos).

2.6.1 Características principales

- Lenguaje Interpretado: Un lenguaje interpretado es aquel que se ejecuta utilizando un programa intermedio llamado intérprete, en lugar de compilar el código a lenguaje máquina que pueda comprender y ejecutar directamente una computadora. La ventaja de los lenguajes compilados es que su ejecución es más rápida. Sin embargo los lenguajes interpretados son más flexibles y más portables. Python tiene, no obstante, muchas de las características de los lenguajes compilados, por lo que se podría decir que es semi-interpretado. En Python, como en Java y muchos otros lenguajes, el código fuente se traduce a un pseudo código máquina intermedio llamado bytecode la primera vez que se ejecuta, generando archivos .pyc o .pyo (bytecode optimizado), que son los que se ejecutarán en sucesivas ocasiones.

- Tipado dinámico: Se refiere a que no es necesario declarar el tipo de dato que va a contener una determinada variable, sino que su tipo se determinará en tiempo de ejecución según el tipo del valor al que se asigne, y el tipo de esta variable puede cambiar si se le asigna un valor de otro tipo.
- Fuertemente tipado: No se permite tratar a una variable como si fuera de un tipo distinto al que tiene, es necesario convertir de forma explícita dicha variable al nuevo tipo previamente. Por ejemplo, si tenemos una variable que contiene un texto (variable de tipo cadena o string) no podremos tratarla como un número (sumar la cadena "9" y el número 8). En otros lenguajes el tipo de la variable cambiaría para adaptarse al comportamiento esperado, aunque esto es más propenso a errores.

- Multiplataforma: El intérprete de Python está disponible en multitud de plataformas (UNIX, Solaris, Linux, DOS, Windows, OS/2, Mac OS, etc.) por lo que si no utilizamos librerías específicas de cada plataforma nuestro programa podrá correr en todos estos sistemas sin grandes cambios.

- Orientado a objetos: La orientación a objetos es un paradigma de programación en el que los conceptos del mundo real relevantes para nuestro problema se trasladan a clases y objetos en nuestro programa. La ejecución del programa consiste en una serie de interacciones entre los objetos. Python también permite la programación imperativa, programación funcional y programación orientada a aspectos.

- Modular: Python permite dividir el programa en módulos reutilizables desde otros programas Python. Python tiene una gran biblioteca estándar, con módulos incluidos que proporcionan E/S de ficheros, llamadas al sistema, sockets y hasta interfaces a GUI "Graphical User Interface" como Tk, GTK, entre otros. Esto viene de la filosofía "baterías incluidas" ("batteries included"). Si hay áreas que son lentas se pueden reemplazar por plugins en C o C++, siguiendo la API para extender o empotrar Python en una aplicación. Debido a la gran variedad de herramientas incluidas en la biblioteca estándar combinada con la habilidad de usar lenguajes de bajo nivel como C y C++, los cuales son capaces de interactuar

con otras bibliotecas, Python es un lenguaje que combina su clara sintaxis con el inmenso poder de lenguajes menos elegantes.

- Documentado: Python cuenta con una entusiasta comunidad de desarrolladores y usuarios que mantienen un wiki, acogen conferencias internacionales y locales, contribuye a los repositorios de código en línea, etc.
  Cuenta además con una documentación completa, integrada tanto en el lenguaje como en páginas web a parte, y con tutoriales en línea para la producción rápida, ya sea para un programador avanzado como para un principiante.

- Código abierto: Python se desarrolla como un proyecto con licencia de código abierto que hace que sea de libre uso y distribución, incluso para su uso comercial. La licencia de Python esta administrada por la Python Software Foundation, ésta es compatible con la licencia GPL a partir de la versión 2.1.1, e incompatible en ciertas versiones anteriores.

- Modo interactivo: El intérprete de Python estándar incluye un *modo interactivo*, en el cual se escriben las instrucciones en una especie de shell; las expresiones pueden ser introducidas una a una, pudiendo verse el resultado de su evaluación inmediatamente. Esto resulta útil tanto para las personas que se están familiarizando con el lenguaje como también para los programadores más avanzados: se pueden probar porciones de código en el modo interactivo antes de integrarlo como parte de un programa.

- Python está escrito en portable ANSI C la mayor parte.

2.6.2 Elementos del lenguaje

- Python fue diseñado para ser leído con facilidad. Entre otras cosas se utilizan palabras en inglés donde otros lenguajes utilizarían símbolos (por ejemplo, los operadores lógicos `||` y `&&` en Python [13] se escriben or y and, respectivamente).

- En vez de delimitar los bloques de código mediante el uso de llaves (`{}`), Python utiliza la indentación. Esto hace que la misma sea obligatoria, ayudando a la claridad y consistencia del código escrito.
- Los comentarios se inician con el símbolo `#`, y se extienden hasta el final de la línea.
- Las variables se escriben de forma dinámica. El signo igual (`=`) se usa para asignar valores a las variables.
- Los tipos de datos se pueden resumir en la tabla 3:

Tipo	Clase	Notas	Ejemplo
`str`	String	Inmutable	`'Integrador'`
`unicode`	String	Versión Unicode de `str`	`u'Integrador'`
`list`	Secuencia	Mutable, puede contener diversos tipos	`[4.0, 'string', True]`
`tuple`	Secuencia	Inmutable	`(4.0, 'string', True)`
`set`	Conjunto	Mutable, sin orden, no contiene duplicados	`set([4.0, 'string', True])`
`frozenset`	Conjunto	Inmutable, sin orden, no contiene duplicados	`frozenset([4.0, 'string', True])`
`dict`	Mapping	Grupo de pares claves, valor	`{'key1': 1.0, 'key2': False}`
`int`	Número entero	Precisión fija	`42`
`long`	Número entero	Precisión arbitraria	`42L` ó `456966786151987643L`
`float`	Número	Coma flotante	`3.1415927`
`bool`	Booleano	Valor booleano verdadero o falso	`True o False`

Tabla 3: Tipos de datos básicos de Python.

-Mutable: si su contenido (o dicho valor) puede cambiarse en tiempo de ejecución.
-Inmutable: si su contenido no puede cambiarse en tiempo de ejecución.

- LISTAS Y TUPLAS: Para declarar una *lista*, basta usar los corchetes `[ ]`, mientras que para declarar una *tupla* se deben usar los paréntesis `()`.

  Tanto las *listas* como las *tuplas* pueden contener elementos de diferentes tipos. No obstante las listas suelen usarse para elementos del mismo tipo en cantidad

variable mientras que las tuplas se reservan para elementos distintos en cantidad fija.

Para acceder a los elementos de una *lista* o *tupla*, se utiliza un índice entero. Se pueden utilizar índices negativos para acceder elementos a partir del final.

Las *listas* se caracterizan por ser mutables, es decir, se puede cambiar su contenido, mientras que no es posible modificar el contenido de una *tupla* ya creada, puesto que es inmutable.

- Los diccionarios se declaran entre llaves (`{}`), la que puede contener pares de valores separados por dos puntos (:).
- Los conjuntos se declaran mediante la instrucción `set`. Los conjuntos no tienen orden ni permiten tener elementos repetidos.
- Las funciones se definen con la palabra clave `def`, seguida del nombre de la función y sus parámetros. Otra forma de escribir funciones, aunque menos utilizada, es con la palabra clave `lambda` (que aparece en lenguajes funcionales como Lisp).
- Existen muchas propiedades que se pueden agregar al lenguaje importando módulos. Los módulos se agregan a los códigos escribiendo "import" seguido del nombre del módulo que queramos usar.
- Las clases, al ser objetos, son instancias de una metaclase. Python además soporta herencia múltiple y polimorfismo.

Un ejemplo de código Python se puede ver en la figura 6.

```
def initial_color(s, colordb):
    # function called on every color
    def scan_color(s, colordb=colordb):
        try:
            r, g, b = colordb.find_byname(s)
        except ColorDB.BadColor:
            try:
                r, g, b = ColorDB.rrggbb_to_triplet(s)
            except ColorDB.BadColor:
                return None, None, None
        return r, g, b
    #
    # First try the passed in color
    r, g, b = scan_color(s)
    if r is None:
        # try the same color with '#' prepended, since some shells require
        # this to be escaped, which is a pain
        r, g, b = scan_color('#' + s)
    if r is None:
        print 'Bad initial color, using gray50:', s
```

Figura 6: Ejemplo de código en Python.

Por lo tanto, Python tiene una sintaxis simple, clara y sencilla; el tipado dinámico, el gestor de memoria, la gran cantidad de librerías disponibles y la potencia del lenguaje, entre otros, hacen que desarrollar una aplicación en Python sea sencillo y muy rápido.
Su diseño y sintaxis es tan fácil de leer y de comprender, acercándose considerablemente al lenguaje natural, cumpliendo así su principal objetivo, la legibilidad, que los programas elaborados en Python se leen prácticamente como pseudocódigo.
Python no es adecuado sin embargo para la programación de bajo nivel o para aplicaciones en las que el rendimiento sea crítico.

Algunos casos de éxito en el uso de Python son Google, Yahoo, la NASA, Industrias Light & Magic, y todas las distribuciones Linux, en las que Python cada vez representa un tanto por ciento mayor de los programas disponibles.

2.6.3 PHP vs. Python

Para finalizar este apartado sobre Python, sólo comentar que la inmensa mayoría de los gestores de contenido que hemos nombrado en apartados anteriores, están escritos en PHP. Plone por el contrario está escrito en Python, así que a continuación se muestra una comparativa entre estos dos lenguajes:
Python y PHP tienes una serie de características en común, como pueden ser:

- Son lenguajes interpretados de alto nivel y con un tipo de dato dinámico.
- Son desarrollados en código abierto (“software libre”).
- Son mantenidos por una gran comunidad de usuarios.
- Son fáciles de aprender (en comparación con Java e incluso Perl)
- Son fáciles de extender en C, C++ y Java.
- Etc

Algunas características de PHP que Python no tiene:

- La sintaxis de C y Perl, con llaves y signos de dólar, etc.
- Las construcciones 'switch' y 'do ... while'.
- Incremento y decremento.
- Sentencias del tipo:(... ? ... : ...)

- Un lenguaje muy informal, donde todas las variables son “set” (NULL), y un sistema de tipos un poco débil, que no debe de confundirse con tipos dinámicos.
- Referencias ('$a= & $b', significa que cuando cambia $b, $a cambia también).
- Modificadores “private”, “protected” y “public” para propiedades y métodos.
- Etc.

Algunas características de Python que PHP no tiene:

- Es un lenguaje de propósito general, no solo para Web.
- La tabulación se utiliza para marcas los bloques de la estructura en vez de las llaves.
- Tiene un pequeño núcleo.
- Es muy claro, conciso y tienen una sintaxis ortogonal.
- Verdaderamente orientado a objetos.
- Sentencia “del” para todo tipo de datos.
- Incorpora lambdas y otras construcciones de la programación funcional.
- Estructuras de manejo de excepciones.
- Integración de SWIG “Simplified Wrapper and Interface Generator”.
- Varios depuradores e IDEs “Integrated Development Environment”.
- Diferenciación entre arrays (listas) y arrays asociativos (diccionarios).
- Perfecto para aplicaciones escalables, mediante la importación de módulos.
- Documentación amplia y en general de mayor calidad que la de PHP.

Comparados para desarrollos Web.

A diferencia de PHP, que tiene todas las funcionalidades para el desarrollo Web integradas, directamente en el núcleo, Python tienes estas funcionalidades dispuestas en módulos. Las capacidades básicas de CGI “Common Gateway Interface”, vienen en el módulo CGI, que se encuentra en la biblioteca estándar de Python. También hay una amplia gama de módulos disponibles para Python, algunos son complementarios, y otros rivalizan entre sí. Como resultado, Python proporciona una base más flexible para el desarrollo Web.

CAPÍTULO 3

DESCRIPCIÓN GENERAL DEL SISTEMA

3 DESCRIPCIÓN GENERAL DEL SISTEMA

En este capítulo se va a describir el sistema que se ha utilizado para el desarrollo de la aplicación web, describiendo tanto la arquitectura de desarrollo necesaria como el diseño concreto y la funcionalidad que se desempeña con este Proyecto de Fin de Carrera.

3.1 Diseño de la aplicación

Se pretende proveer una funcionalidad básica para el trabajo diario del cuerpo de policía, que puede ir aceptando extensiones en líneas futuras a medida que el Municipio lo requiera, ya sea por aumento de la población, por la necesidad de cubrir un aspecto no controlado en la versión inicial, etc. Esta funcionalidad básica se pone a disposición del usuario vía Web y es desarrollada en su totalidad con "software libre", abaratando los costes tanto de producción como de mantenimiento, al no ser necesarias licencias.

De esta forma se ha querido dar otra opción, frente a soluciones similares en el mercado, que utilizan licencias y son desarrolladas con "software comercial" como por ejemplo, .NET, que utilizan otro tipo de "software libre", como por ejemplo, PHP, MySQL, y otros gestores de contenido como Drupal, Joomla, etc.

El diseño de esta aplicación web, cumple todos los principios de un modelo cliente/servidor. Modelo que puede observarse en la figura 7.

La aplicación estará alojada en un servidor web accesible a todos los usuarios a través de Internet. Los usuarios podrán conectarse a la aplicación mediante distintos dispositivos hardware, con la ayuda de un navegador web, usando el protocolo HTTPS. Toda la información que va a ser generada por el uso de la aplicación va a ser almacenada en una base de datos relacional. El acceso a la información de dicha base de datos se hará únicamente a través de la aplicación.

A continuación se muestra un esquema con los elementos básicos de un modelo cliente/servidor.

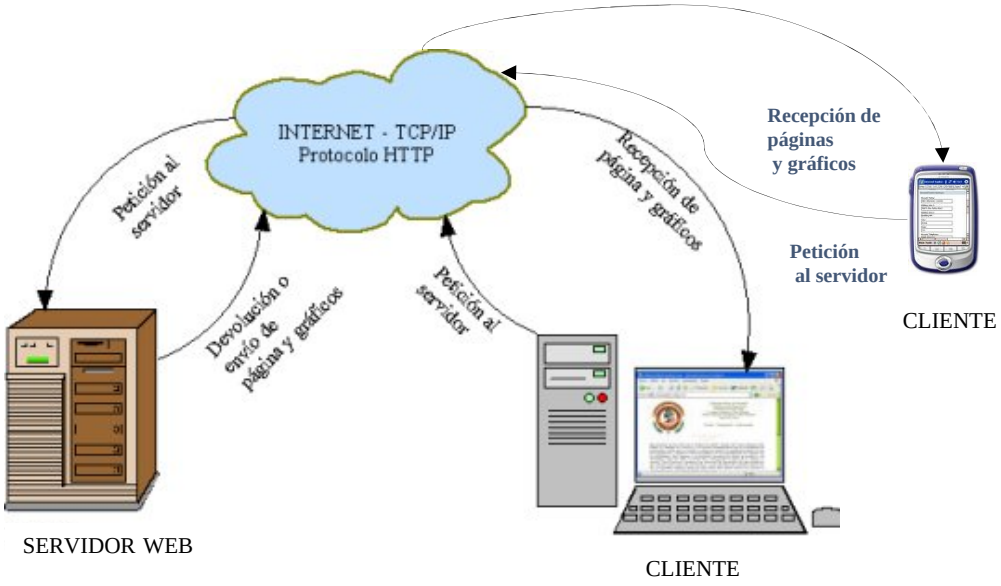

Figura 7 : Modelo Cliente/Servidor.

Teniendo en cuenta este modelo, para nuestro proyecto identificaremos el servidor de aplicaciones Web con el servidor de aplicaciones Zope, sobre el que correrá Plone como gestor de contenido, y como clientes, a los agentes del cuerpo de Policía del Municipio en cuestión.

3.1.1 Arquitectura del sistema

Por lo tanto para nuestro diseño utilizamos una serie de elementos que se combinan entre sí y que dan lugar a la arquitectura que conforma nuestro sistema de gestión web. Concretamente los elementos son:

- **Un servidor de aplicaciones Web: Zope 2.10.4**, está orientado a objetos y escrito en Python.

- **Una base de datos orientada a objetos: ZOBD**, "Zope Object Database". En Zope los objetos usualmente persisten en una base de datos orientada a objetos, que hace posible un alto nivel de transparencia.

En nuestro diseño, en esta base se almacenan todos los formularios creados, las imágenes, las páginas que muestran las opciones de cada sección, los scripts de validación de los formularios, los métodos que facilitan el almacenamiento de los datos introducidos en los formularios, etc.

- **Un sistema de Gestión de Bases de Datos: PostgreSQL,** gestiona la base de datos relacional **Postgre 8.3**. Esta base de datos será externa y en ella se almacena toda la información que se genera al rellenar los formularios por los agentes, así como información necesaria para la organización de los agentes y su trabajo diario.

- **Un Gestor de Contenido: Plone 3.0,** está basado en Zope y programado en Python, es un sistema de publicación de documentos y una herramienta de trabajo para colaborar entre entidades distintas. En este proyecto se utiliza como sistema, para construir una aplicación web centrada en contenido.

- **Un adaptador para la Base de Datos Externa: Psycopg 1.1.21** [9]**,** nos permite conectar la base de datos Postgre con Plone, a través de Python.

- **Un lenguaje de programación: Python 2.4.4,** es un lenguaje de propósito general, interpretado, orientado a objetos y multiplataforma, que facilita una programación simple, versátil y de rápido desarrollo.

- **Una plataforma a través de la cual interactuar con el usuario: la Web**, WWW "World Wide Web", conjunto de protocolos que permite, de forma sencilla, la consulta remota de archivos de hipertexto y utiliza Internet como medio de transmisión.

La interacción de todos estos elementos dentro de un modelo cliente/servidor se muestra en la figura 8. Se puede observar detalladamente tanto la arquitectura lógica de Zope, como la asociación de Plone y Postgre con los componentes de Zope y cuales son los protocolos utilizados en la comunicación entre ellos y los usuarios finales.

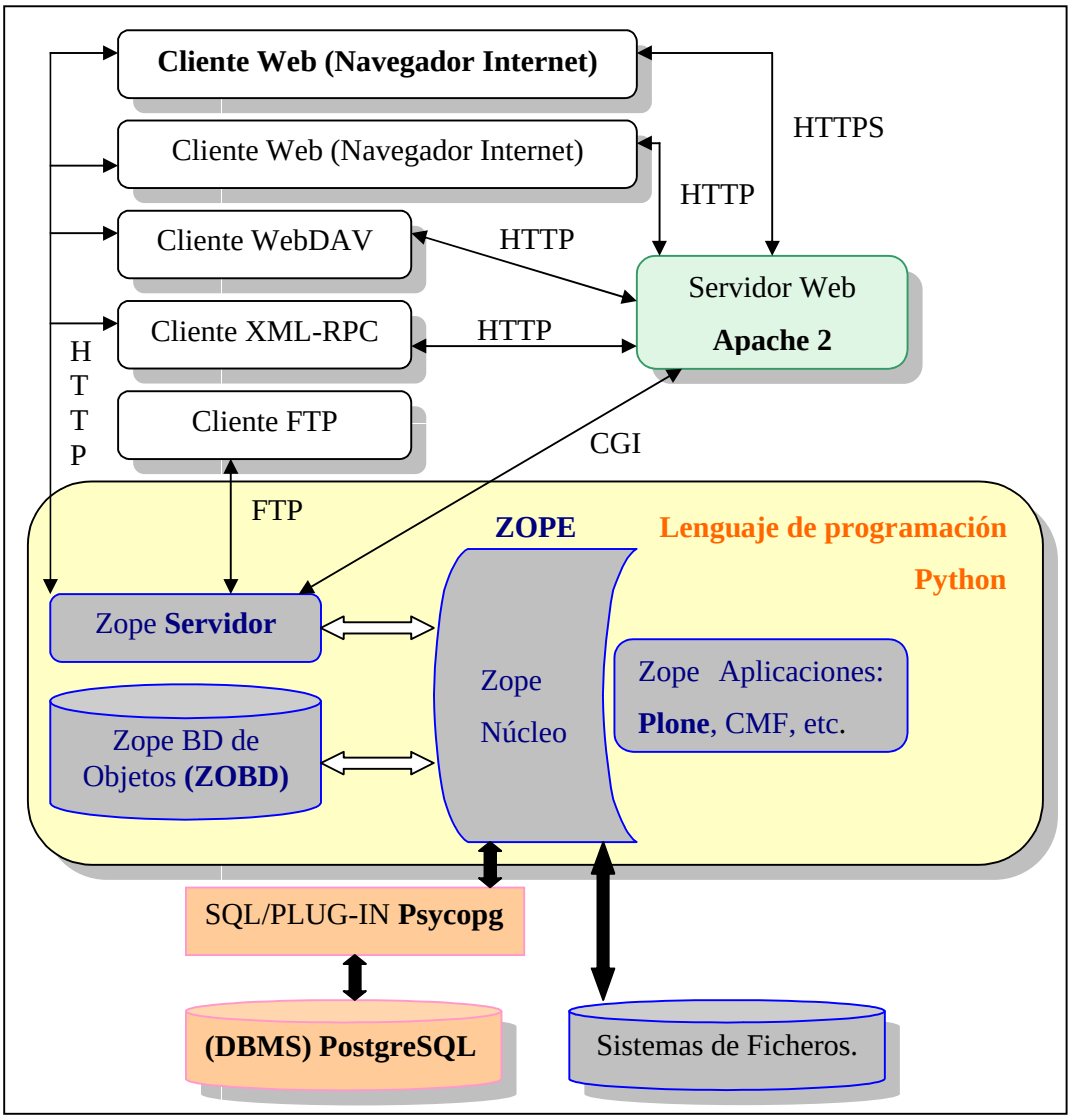

Figura 8: Esquema de la conexión de Zope/Plone/Python, formando un modelo cliente/servidor.

3.2 Funcionalidad

La aplicación desarrollada en este proyecto funciona prácticamente como una Intranet, puesto que cualquier persona puede entrar a la página de Inicio de la aplicación, pero sólo los usuarios registrados, pueden acceder a las secciones que la componen y realizar las acciones que esta facilita.

Lo que se pretende es permitir a los agentes de policía de un Municipio gestionar los aspectos más comunes de la organización interna de la plantilla, así como el almacenamiento y administración de la información de seguridad más relevante y usual, que se produce a lo largo de la jornada laboral en un Municipio.

En esta aplicación se han contemplado los tipos de accidente, los tipos de actas, los atestados, más comunes en la vida cotidiana de un Municipio, etc. Y la misma filosofía, se ha aplicado en cuanto a los aspectos a tratar de la administración interna del cuerpo, siendo estos, los días de vacaciones, los turnos y el uniforme.

Por lo tanto la funcionalidad de la aplicación paso a paso es la siguiente:

3.2.1 Conexión con la aplicación

A través de un navegador Web nos conectamos con la aplicación, visualizando la página principal de la interfaz de usuario, un ejemplo de cómo sería la conexión se muestra en la figura 9. En ella nos referimos a la arquitectura Python-Zope-Plone que se describe en la figura 5 del capítulo anterior.

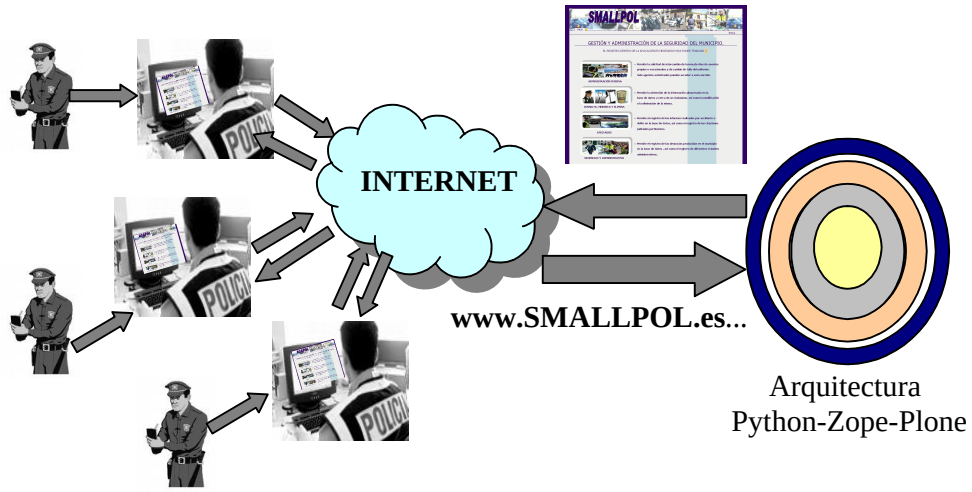

Figura 9: Conexión con la aplicación SMALLPOL.

Esta conexión se hace de forma segura utilizando HTTPS, de esta manera, la comunicación a través de Internet es fiable al utilizarse el protocolo de seguridad SSL. Esto ha de ser así puesto que se trata de una aplicación en la que se manejarán datos personales de los ciudadanos del Municipio.

3.2.2 Funcionalidad general de la aplicación

A continuación, se muestra un diagrama de flujo con la funcionalidad general del sistema de gestión policial desarrollado. En él, se observan las distintas fases por las que el usuario ha de ir pasando y las opciones que puede seleccionar según el tipo de información a gestionar. De esta manera nos hacemos una visión general del sistema y nos es más fácil identificar cada una de las partes que se describen en apartados posteriores.

Identificaremos cada una de las “fases” por las que vamos pasando y explicando en los apartados posteriores con un recuadro de la forma: 

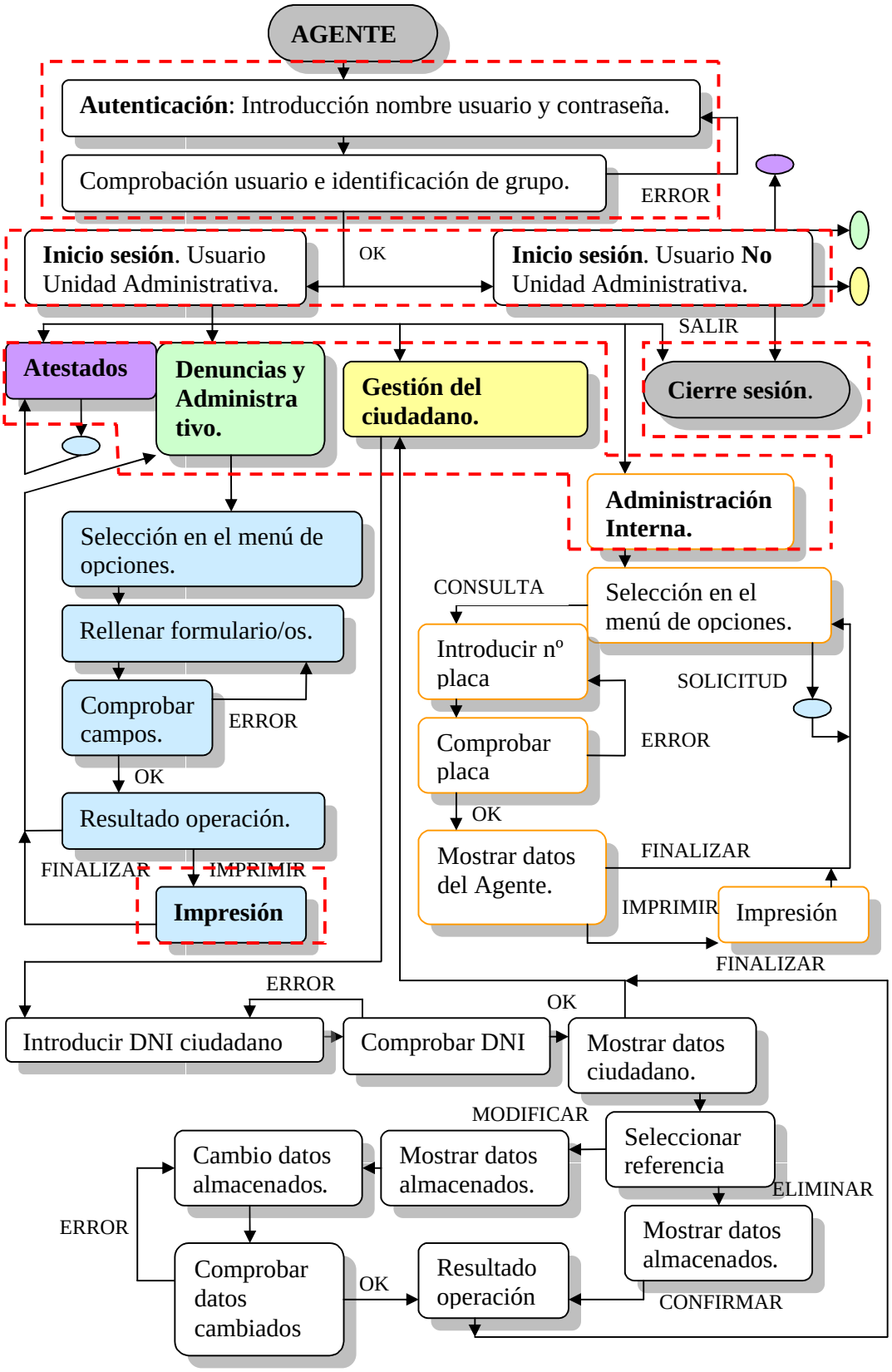

Figura 10: Diagrama de flujo del funcionamiento general de la aplicación.

3.2.3 Identificación de usuarios

Los agentes de policía del Municipio son los usuarios previamente registrados en el sistema por el Administrador de la misma, con un nombre de usuario y una contraseña. El nombre de usuario está formado por las letras "ag" seguido del número de su placa.
En la figura 11, se muestra como deben de identificarse los usuarios para poder acceder a la aplicación.

Figura 11: Control de acceso a la aplicación.

Los agentes se encuentran repartidos en diferentes unidades, que son:

- La unidad de Tráfico.
- La unidad de Seguridad Ciudadana.
- La unidad de Polígonos.
- La unidad Administrativa.

Según la unidad a la que pertenecen, los usuarios forman parte de uno de los siguientes dos grupos:

- noAdministrativos: Los pertenecientes a cualquier unidad menos la Administrativa.
- Administrativos: Los pertenecientes a la unidad Administrativa.

Y a su vez cada grupo tiene un rol asociado, que le permite tener dentro del sistema, diferentes accesos.

- Grupo "Administrativos" → Rol asociado "administrativo".
- Grupo "noAdministrativos" → Rol asociado "member".

Una vez que se comprueba su identidad, pueden acceder a la aplicación dentro de una sesión particular del agente en cuestión; su nombre y apellidos aparecerán en pantalla como prueba de una autenticación exitosa, además de un mensaje explicito informándole sobre la nueva situación, como se muestra en la figura 12.

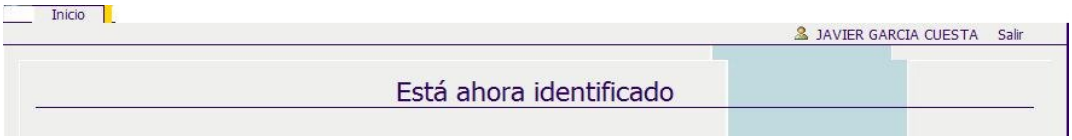

Figura 12: Agente que ha iniciado la sesión.

3.2.4 Secciones de la aplicación

La aplicación consta de cuatro secciones, que se describen a continuación, mostrando de cada una de ellas una visión general y destacando, una de las funcionalidades de las que constan y que servirá de ejemplo, teniendo en cuenta que el resto de funcionalidades cuentan con una ejecución similar. Esta demostración la haremos mediante diagramas de flujo.

3.2.4.1 Administración Interna

A través de ella se gestionan los turnos, las vacaciones y las tallas de los uniformes de los agentes. Concretamente, se puede solicitar el intercambio de turnos entre agentes, el cambio de talla de una prenda del uniforme o solicitar días de vacaciones o asuntos propios. Solo agentes de la unidad Administrativa tienen acceso, teniendo así que realizar ellos las solicitudes o los cambios oportunos para el resto de agentes de las otras unidades. Además pueden consultar el turno asignado para cada mes, la talla de cada prenda del uniforme, o el número de días que tiene todavía disponibles, un agente.

Por lo tanto las opciones del menú principal de la sección son:

- Solicitar vacaciones.
- Solicitar intercambio de turnos.
- Solicitar cambio de talla del uniforme.
- Consultar Turnos asignados actualmente.
- Consultar días disponibles actualmente.
- Consultar el tallage asignados actualmente.

La figura 13, se va a utilizar como un ejemplo de funcionalidad dentro de la sección. Vamos a ver concretamente el caso en el que un Agente que pertenece a la unidad

Administrativa y que ha iniciado sesión, necesita solicitar un período y un día de asuntos propios para otro agente de la plantilla.

El diagrama de flujo de esta función es el siguiente:

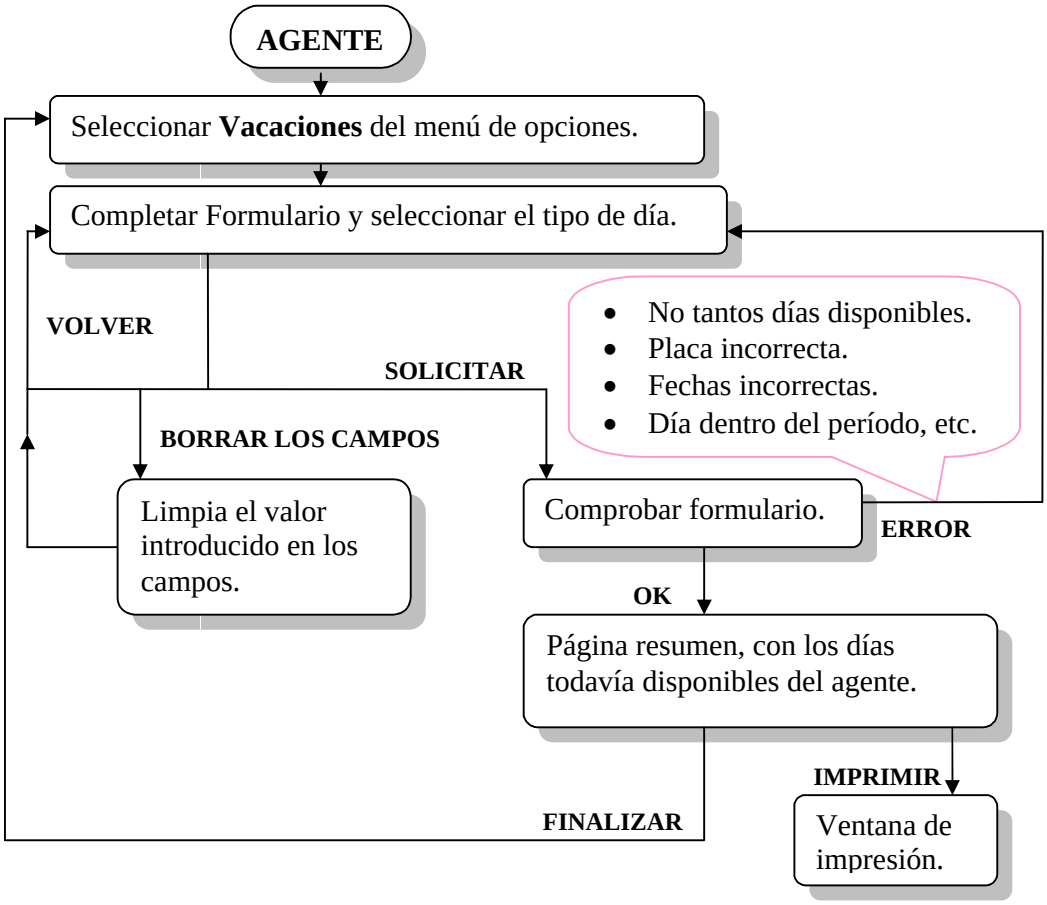

Figura 13: Diagrama de flujo de la función que se encarga de la solicitud de días de asuntos propios o vacacionales de los agentes.

3.2.4.2 Denuncias y Administrativo

A través de ella se registran una serie de actas y las denuncias que se han producido en el Municipio. Tanto las actas como las denuncias son de diferentes tipos, los cuales se muestran en el menú principal de la sección, y son los siguientes:

Las actas se levantan para dar constancia de los siguientes hechos:

- Precinto/desprecinto de un establecimiento.
- Precinto/desprecinto de un vehículo.

- Inspección a establecimientos (Salubridad e higiene, drogas, seguridad, cumplimiento de horarios, etc).
- Medición de ruidos.
- Daños en bienes públicos.

Las denuncias a ciudadanos se realizan por cometer infracciones en:

- Conducción.
- Tráfico.
- Ocupación indebida de la vía pública.
- Infracción Ley Orgánica 1/1992 (Botellón).

La figura 14, se va a utilizar como un ejemplo de funcionalidad dentro de la sección. Vamos a ver concretamente el caso en el que un Agente que puede pertenecer a cualquier unidad y que ha iniciado sesión, genera una Denuncia por cometer infracciones en la conducción de un vehículo. Calcula los puntos a sustraer al conductor y el importe que suponen las infracciones.

El diagrama de flujo de esta función es el siguiente:

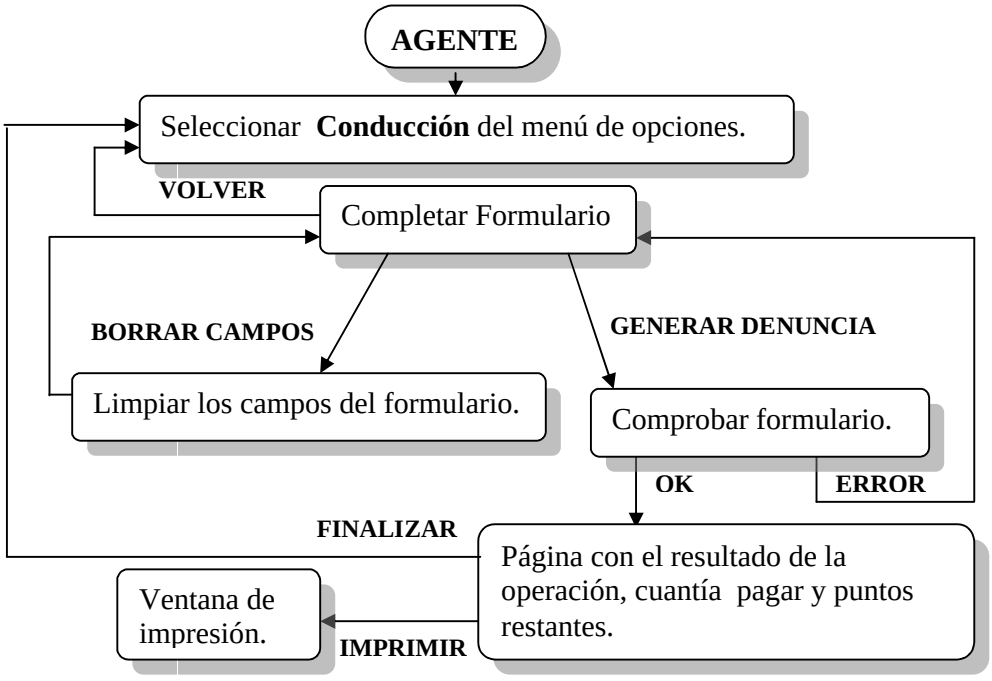

Figura 14: Diagrama de flujo de la función que se encarga de generar una denuncia por cometer infracción es en la conducción.

3.2.4.3 Atestados

A través de ella se registran los informes de accidentes de diversos tipos producidos en el Municipio, así como se provee la descarga de las actas de información para las personas denunciadas o denunciantes. Las diferentes opciones que se muestran en el menú principal de la sección son:

Las actas se levantan para dar constancia de los siguientes hechos:

- Hurto, Delito con citación judicial
- Información

Informes de accidentes producidos en diferentes entornos y por diferentes causas:

- Laboral
- Doméstico
- Circulación

La figura 15, se va a utilizar como un ejemplo de funcionalidad dentro de la sección.
Vamos a ver concretamente el caso en el que un Agente que puede pertenecer a cualquier unidad y que ha iniciado sesión, genera un Acta de delito con su correspondiente citación para el denunciado. La función identificará si se trata de un hurto o de un delito según los datos insertados en el formulario, como son el importe de lo sustraído o la utilización de la violencia, en caso de tratarse de un delito se pasará automáticamente a generar la citación correspondiente.

El diagrama de flujo de esta función es el siguiente:

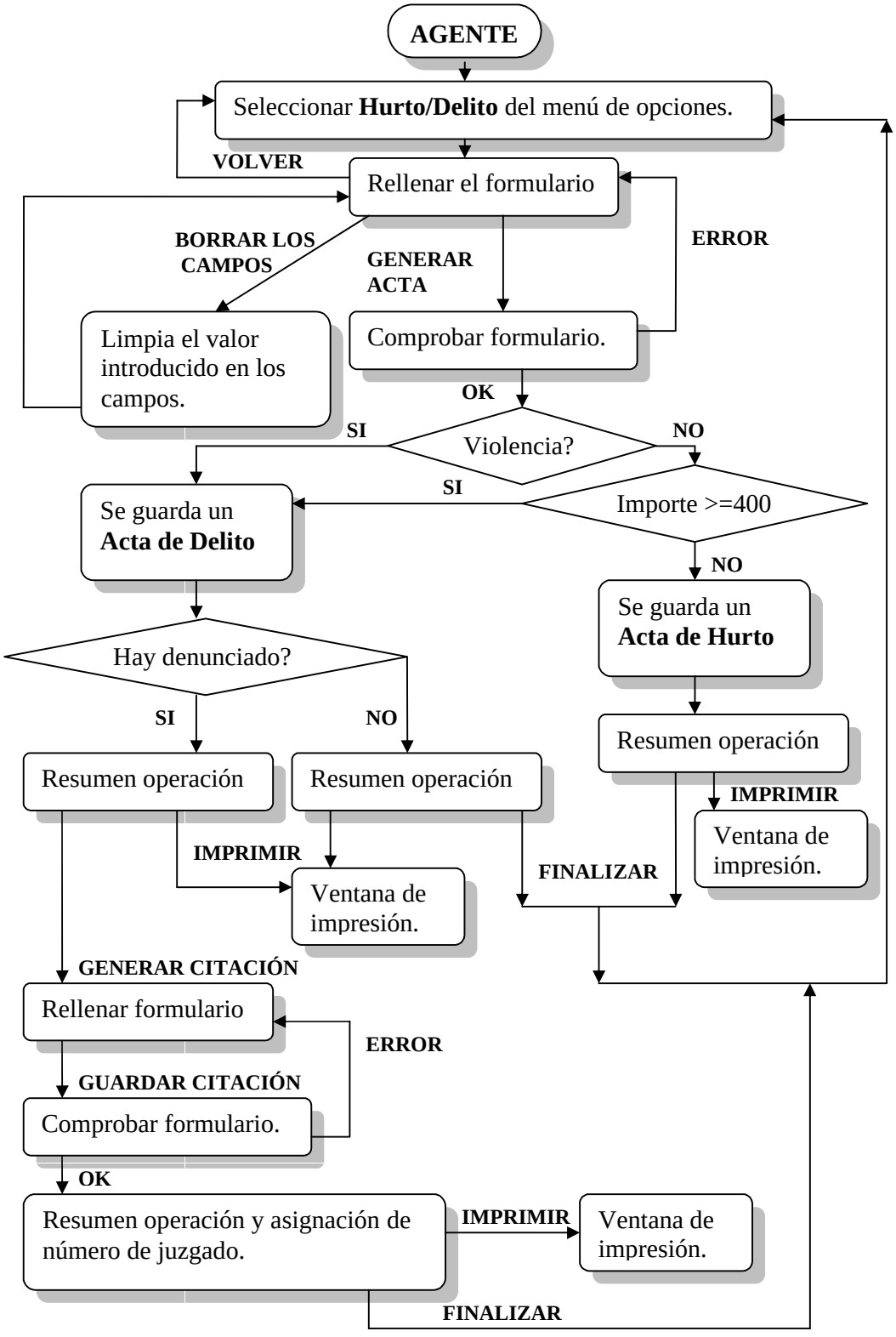

Figura 15: Diagrama de flujo de la función que se encarga generar un acta por hurto o delito, con su correspondiente citación por juicio rápido si es necesario.

3.2.4.4 Gestión del ciudadano

A través de ella se consulta la información registrada a través de la aplicación sobre un ciudadano, sólo para su consulta o para la modificación o eliminación de la misma. Esta información es seleccionada, mediante la introducción del DNI del ciudadano en cuestión.

Por lo tanto una vez se obtiene la información almacenada sobre el ciudadano, se puede elegir modificarla o eliminarla. La información que se obtiene puede ser porque esa persona se encuentre denunciada por algún motivo, o esté implicado de alguna manera en un accidente, etc.

La figura 16, se va a utilizar como un ejemplo de funcionalidad dentro de la sección.

Vamos a ver concretamente el caso en el que un Agente que puede pertenecer a cualquier unidad y que ha iniciado sesión, **elimina** información asociada a un ciudadano. Esta información en cuestión, es la que recoge a este ciudadano como implicado o testigo en un accidente Doméstico.

Si se trata del único participante en el accidente, se elimina la información sobre él y el acta del accidente, si hay más participantes se elimina la del individuo en cuestión, y se resta uno al número de participantes que se refleja en el acta del accidente.

El diagrama de flujo de esta función es el siguiente:

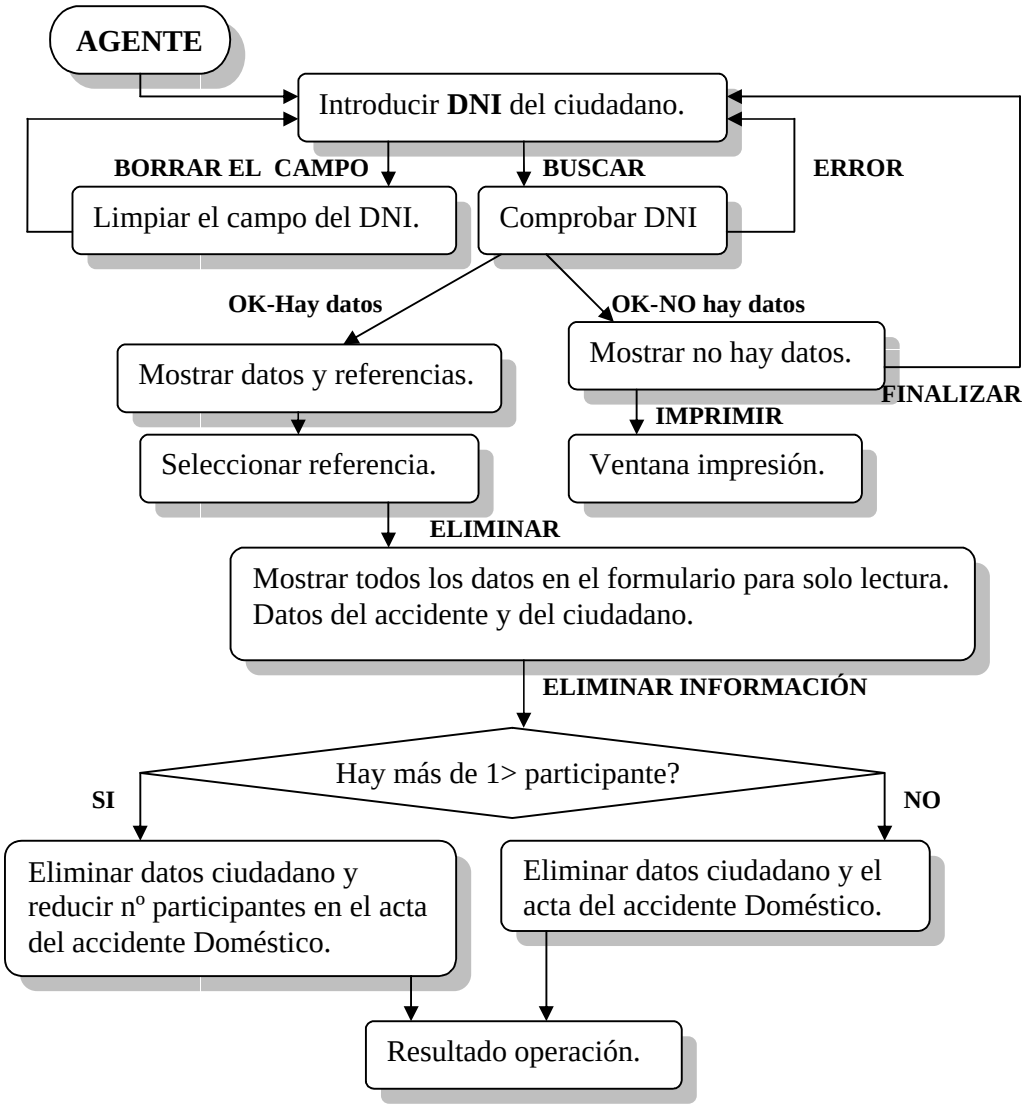

Figura 16: Diagrama de flujo de la función que se encarga de eliminar información sobre un ciudadano, participante en un accidente Doméstico.

Por lo tanto una vez que tenemos identificadas las cuatro secciones en que se divide la aplicación, ya podemos continuar con las fases que se están utilizando, para explicar la funcionalidad del sistema más claramente.

3.2.5 Inicio sesión

En esta fase, una vez que los usuarios han sido identificados satisfactoriamente, se tiene en cuenta el grupo y el rol que tienen asignado, puesto que si pertenecen al grupo "Administrativos" tendrán acceso a las cuatro secciones que componen la aplicación y por el contrario si pertenecen al grupo "noAdministrativos" sólo tendrán acceso a tres de las cuatro; la sección de Administración Interna no aparecerá como una opción, ya que únicamente es accesible por el personal de la unidad Administrativa.

3.2.6 Funcionalidad

Una vez identificado y clasificado el usuario según su grupo, éste puede, seleccionar la sección que le interese; para cada una de ellas visualizará una página con las diferentes opciones de las que se dispone.

Una vez elegida la que se desea realizar, aparecerá un formulario con ayuda en cada uno de sus campos, que se deberá rellenar y aceptar.

Si no ha habido errores o incoherencias al completar el formulario o formularios, se visualizará una página avisando de su éxito en la operación realizada. En caso contrario, los errores se mostrarán sobre el campo en cuestión, resaltándolo con color rosa.

Estos datos introducidos en los formularios, se almacenarán en una base de datos relacional, a la que nos conectamos a través del Gestor de Contenidos que soporta la aplicación. Se insertan en la tabla o tablas correspondientes, almacenándose siempre con la referencia del agente que los ha introducido.

3.2.7 Impresión de reportes

Si el usuario lo desea, puede imprimir la página que se ha obtenido como resultado de la operación seleccionada. Podrá imprimir por ejemplo los datos que se tienen almacenados en la base de datos sobre un ciudadano, la tabla con los turnos asignados a un agente, el mensaje de aviso cuando una modificación se ha realizado con éxito, etc.

Para ello seleccionará la impresora que desee, en la ventana que se abre al pinchar el enlace "IMPRIMIR" que se encuentra en las páginas de resultado.

3.2.8 Cierre de sesión

Una vez que el agente a traspasado los datos que ha recogido a lo largo de la jornada laboral, en papel, como por ejemplo, en las libretas de multas de tráfico, etc, a la base de datos para que su almacenamiento sea seguro, o si se trata de un agente de la unidad Administrativa, a realizado la solicitud de días de vacaciones de un agente del cuerpo, etc, el agente sale de su sesión, volviendo así a la página de Inicio. Al salir el nombre del usuario dejará de aparecer y se mostrará un mensaje para informar del nuevo estado del usuario, como puede verse en la figura 17.

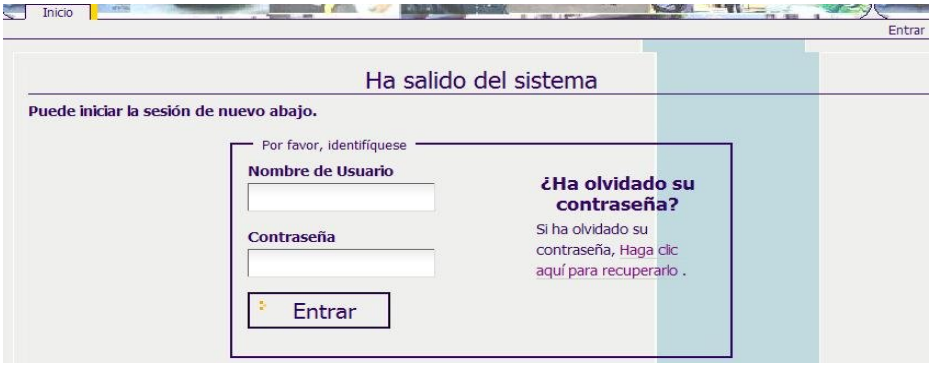

Figura 17: El agente ha salido de la sesión.

3.3 Especificación de requisitos

En este apartado se recogen los requisitos de software establecidos tanto funcionales como no funcionales para este proyecto.

Primeramente debemos identificar a tres perfiles de usuario, el Administrador, el agente que pertenece a la unidad Administrativa y el agente que pertenece a cualquiera de las otras tres unidades. Los agentes independientemente de la unidad a la que pertenezcan son los usuarios principales del sistema.

- El Administrador:

  Es el perfil encargado de realizar tareas de mantenimiento, actualización y administración del propio sistema incluida la base de datos externa. En líneas generales sus funciones son:

- Gestionar usuarios: altas, bajas y modificaciones de los usuarios de la aplicación, entendiéndose como tales, a los usuarios que pertenezcan a cualquier unidad. En este apartado también entra la gestión de los roles de los usuarios, puesto que el número puede variar y los permisos que estos atribuyen también, facilitando así la escalabilidad del sistema.
- Gestionar secciones: añadir, modificar o eliminar funcionalidades al sistema. Dentro de cada sección pueden variar los aspectos controlados, modificándose de esta manera la interfaz gráfica, el número de formularios, los scripts de validación, las tablas que forman la base de datos externa, etc.
- Realizar el mantenimiento tanto de la aplicación web como de la base de datos, realizando las copias de seguridad pertinentes, asegurándose de la integridad de los datos, eliminando información desactualizada, controlando que el servidor se encuentre permanentemente activo, etc.

- El agente de la unidad Administrativa:

  El administrador determina los permisos de acceso de este perfil a las funcionalidades del sistema. Estos permisos le permiten el acceso a todas las secciones de las que consta la aplicación. A grandes rasgos, sus funcionalidades son las siguientes:

  - Gestionar de forma interna la plantilla policial: solicitud de intercambio de turno, solicitud de días de asuntos propios o de vacaciones y solicitud de cambio de talla en las prendas del uniforme para si mismo o para otro agente.
  - Consultar, modificar o eliminar información: Obtener los datos almacenados sobre un ciudadano en concreto en la base de datos externa, para su consulta, modificación o eliminación.
  - Insertar Denuncias y Actas administrativas en la base de datos externa.
  - Insertar Atestados por accidentes e informes.

- El agente que no pertenece a la unidad Administrativa:

  El administrador determina los permisos de acceso de este perfil a las funcionalidades del sistema. Estos permisos le permiten el acceso a todas las

secciones excepto a la de Administración Interna. Por lo tanto sus funciones son las mismas que el tipo anterior eliminado, “Gestionar de forma interna la plantilla policial”.

3.3.1 Requisitos funcionales

Los requisitos funcionales para cualquier agente son los siguientes:

1. **Ser registrado en la aplicación y en la base de datos externa**: Debe de ser dado de alta en la aplicación con un nombre de usuario y una contraseña. Debe ser insertado en la base de datos, representado por su número de placa, su nombre y apellidos y la unidad a la que pertenece.
2. **Ser asignado dentro de un grupo de usuarios y con un rol específico**: Según la unidad a la que pertenezca pertenecerán a un grupo o a otro y el rol asignado será diferente.
3. **Autenticarse en la aplicación para iniciar una sesión y poder acceder a alguna sección**: Debe introducir nombre de usuario y contraseña.
4. **Modificar su contraseña**: Se rellena un formulario con la contraseña actual y dos veces la nueva contraseña. La nueva contraseña será actualizada instantáneamente en la aplicación.
5. **Crear Acta de precinto de un establecimiento**: Se rellena un formulario y los datos serán almacenados en la base de datos externa.
6. **Crear Acta de desprecinto de un establecimiento:** Se rellena un formulario y los datos serán almacenados en la base de datos externa. Se guarda con una referencia al acta de precinto del mismo establecimiento de forma automática.
7. **Crear Acta de precinto de un vehículo**: Se rellena un formulario y los datos serán almacenados en la base de datos externa.
8. **Crear Acta de desprecinto de un vehículo:** Se rellena un formulario y los datos serán almacenados en la base de datos externa. Se guarda con una referencia al acta de precinto del mismo vehículo de forma automática.
9. **Crear Acta de Inspección de Seguridad, Salubridad e Higiene en Establecimientos de venta de productos alimenticios:** Se rellena un formulario inicial con los datos de establecimiento que serán guardados en la base de datos externa y a continuación otro, con los datos específicos de este

tipo de inspección, esta información se guarda en la base de datos con una referencia a los datos del establecimiento.

10. **Crear Acta de Inspección de Seguridad para espectáculos públicos en Establecimientos:** Se rellena un formulario inicial con los datos de establecimiento que serán guardados en la base de datos externa y a continuación otro, con los datos específicos de este tipo de inspección, esta información se guarda en la base de datos con una referencia a los datos del establecimiento.
11. **Crear Acta de Inspección de Drogodependencias y trastornos adictivos en Establecimientos:** Se rellena un formulario inicial con los datos de establecimiento que serán guardados en la base de datos externa y a continuación otro, con los datos específicos de este tipo de inspección, esta información se guarda en la base de datos con una referencia a los datos del establecimiento.
12. **Crear Acta de Medición de Ruidos:** Se rellena un formulario y los datos serán almacenados en la base de datos externa.
13. **Crear Acta de una Denuncia por Daños Públicos:** Se rellena un formulario y los datos serán almacenados en la base de datos externa.
14. **Crear una Denuncia por Infracciones en la Conducción:** Se rellena un formulario y los datos serán almacenados en la base de datos externa. Por un lado los datos del vehículo y el motivo de la denuncia, y por otro los datos personales del denunciado con una referencia a la denuncia en cuestión.
15. **Crear una Denuncia por el Incumplimiento de normas de Tráfico:** Se rellena un formulario y los datos serán almacenados en la base de datos externa. Por un lado los datos del vehículo y el motivo de la denuncia, y por otro los datos personales del denunciado con una referencia a la denuncia en cuestión.
16. **Crear una Denuncia por Ocupación indebida de la vía pública:** Se rellena un formulario y los datos serán almacenados en la base de datos externa. Por un lado los datos de la ocupación (mercancía, basura, lugar, etc), y por otro los datos personales del denunciado con una referencia a la denuncia en cuestión.
17. **Crear una Denuncia por Ocupación indebida de la vía pública:** Se rellena un formulario y los datos serán almacenados en la base de datos externa. Por un

lado los datos de la ocupación (mercancía, basura, lugar, etc), y por otro los datos personales del denunciado con una referencia a la denuncia en cuestión.

18. **Crear una Denuncia por Infringir la Ley orgánica 1/1992 (botellón) sobre protección de la seguridad ciudadana:** Se rellena un formulario y los datos serán almacenados en la base de datos externa. Por un lado los datos de las sustancias intervenidas, y por otro los datos personales del denunciado con una referencia a la denuncia en cuestión.
19. **Crear Acta de un Hurto o de un Delito:** Se rellena un formulario y los datos serán almacenados en la base de datos externa, en diferentes tablas, dependiendo de si los hechos son clasificados como un delito o como un hurto.
20. **Crear Citación de Juicio rápido por Delito:** Se rellena un formulario y los datos serán almacenados en la base de datos externa. La citación se guardara con una referencia al Delito asociado.
21. **Obtener Actas de Información para denunciados o denunciantes**: Puede imprimirse el acta que necesite.
22. **Crear Acta de Accidente Laboral:** Se rellena un formulario con los datos del accidente y sus causas, todo será almacenado en la base de datos externa. Seguidamente debe rellenar un formulario por cada implicado en dicho accidente, cuyos datos personales, serán almacenados en otra tabla con una referencia al accidente en cuestión.
23. **Crear Acta de Accidente Doméstico:** Se rellena un formulario con los datos del accidente y sus causas, todo será almacenado en la base de datos externa. Seguidamente debe rellenar un formulario por cada implicado en dicho accidente, cuyos datos personales, serán almacenados en otra tabla con una referencia al accidente en cuestión.
24. **Crear Acta de Accidente de Circulación:** Se rellena un formulario con los datos del accidente y sus causas, todo será almacenado en la base de datos externa. Seguidamente debe rellenar un formulario por cada implicado en dicho accidente, cuyos datos personales, serán almacenados en otra tabla con una referencia al accidente en cuestión.
25. **Consultar la Información contenida en la base de datos externa sobre un ciudadano concreto**: Se introduce el DNI del interesado y se muestra un

resumen de los accidentes, actas, delitos, precintos, etc, en los que el ciudadano figure.

26. **Modificar la Información contenida en la base de datos externa sobre un ciudadano concreto**: Se introduce el DNI del interesado y se muestra un resumen de los accidentes, actas, delitos, precintos, etc, en los que el ciudadano figure. Se selecciona la referencia asignada al registro que queremos modificar y se modifican los valores que se desee, exceptuando dicha referencia.
27. **Eliminar la Información contenida en la base de datos externa sobre un ciudadano concreto**: Se introduce el DNI del interesado y se muestra un resumen de los accidentes, actas, delitos, precintos, etc, en los que el ciudadano figure. Se selecciona la referencia asignada al registro que queremos eliminar y se elimina toda la información relacionada con ese registro.
28. **Imprimir la página de resumen después de realizar cualquier función de las que ofrece la aplicación**.

Sólo si el agente pertenece a la unidad Administrativa, además de los anteriores tiene los siguientes requisitos funcionales:

29. **Consultar los Turnos asignados a un agente en el año actual o el siguiente**: Se introduce el número de placa del agente en cuestión y el año que se desea seleccionar. Se muestra una tabla con los turnos asignados mes a mes.
30. **Consultar las Tallas asignados a cada prenda del uniforme de un agente**: Se introduce el número de placa del agente en cuestión y se obtiene una tabla con las tallas asignadas a cada prenda en ese momento.
31. **Consultar el número de días disponibles para vacaciones o asuntos propios de un agente**: Se introduce el número de placa del agente en cuestión y se obtiene una tabla con el número de días de los que aún dispone.
32. **Intercambiar los Turnos entre dos agentes para un mes en concreto:** Se introduce el número de placa del agente en cuestión y el año que se desea seleccionar. Se muestra una tabla con los turnos asignados mes a mes.
33. **Cambiar la Talla de una prenda del uniforme de un agente:** Se introduce el número de placa del agente en cuestión y se selecciona la prenda y la talla que se desea. Se muestra una tabla con las nuevas tallas asignadas si es posible.

34. **Solicitar un Período o un Día de vacaciones o de asuntos propios para un agente:** Se introduce el número de placa del agente en cuestión y el período y/o día que se desea seleccionar, junto con el tipo de días. Se muestra una tabla con el número de días restantes de los que dispone después de habérsele concedido la petición si es posible.

Los requisitos funcionales de cara al Administrador son los siguientes:

1. **Actualizar, modificar y mantener la base de datos externa:** Actualizar datos específicos para la gestión de la aplicación así como ampliar la base de datos si es necesario.
2. **Registrar a los usuarios dentro de la aplicación y la base de datos externa.**
3. **Asignar a cada usuario dentro del grupo correspondiente y con el rol adecuado.**
4. **Modificar los permisos de cada grupo o tipo de rol.**
5. **Dar de baja a un usuario de la aplicación y la base de datos externa.**
6. **Añadir funcionalidad a la aplicación:** Añadir más opciones que se puedan controlar y gestionar desde la aplicación. Esto conlleva la creación de formularios, scripts, etc.

3.3.2 Requisitos no funcionales

Los requisitos no funcionales pueden referirse a diferentes aspectos como pueden ser, la interfaz gráfica, los recursos, el rendimiento, etc. Es decir, aseguran que se disponga de un sistema manejable y gestionable que ofrezca la funcionalidad requerida de manera fiable, ininterrumpida o con el tiempo mínimo de interrupción, incluso ante situaciones inusuales.

Los requisitos no funcionales de cara al Agente son los siguientes:

Operación: Especifican como va a realizar el sistema las tareas para las que ha sido construido.

1. Para que un agente puede acceder a la aplicación, el Administrador debe de darle de alta en la aplicación y asignarle dentro de un grupo y rol.
2. El agente podrá cambiar su contraseña una vez autenticado, introduciendo la nueva dos veces.

3. La entrada/salida de datos a/de la aplicación se hará a través de formularios web (cuadros de texto, combos de selección, checkbox, botones, etc).No existe otra manera de comunicación con el sistema por parte del Agente.
4. Para acceder a la aplicación el agente debe autenticarse, y según al grupo al que pertenezca o el rol que tenga asignado, podrá acceder a diferentes secciones.

Los requisitos no funcionales de cara al Administrador son los siguientes:

Rendimiento:

1. La aplicación y el servidor web deben asegurar un óptimo rendimiento independientemente del número de sesiones abiertas a la vez (número de usuarios conectados simultáneamente a la aplicación).
2. El tiempo de espera del resultado de una consulta a la base de datos externa ha de ser en un tiempo mínimo.

Interfaz: Especifican hardware y/o software con el que el sistema o componentes del sistema deben interactuar o comunicarse.

1. La aplicación estará conectada a un sistema gestor de bases de datos desde donde leerá y guardará datos.
2. La base de datos se ha de implementar en PostgreSQL, pudiendo estar en el mismo o distinto servidor que el servidor Web.
3. Plone cumple con varios estándares, cumple cuidadosamente los estándares para usabilidad y accesibilidad. Sus páginas cumplen con la "sección 508"de EE.UU., así como las pautas de accesibilidad del W3C para utilizar estándares basados en las mejores prácticas como XHTML y CSS. Por este motivo puede verse de forma correcta en la gran mayoría de los navegadores, habiéndose probado su correcto funcionamiento en las versiones de Internet Explorer igual o superior a la 6.0 y en las versiones de Mozilla Firefox igual o superior a la 2.0.0.5.

Operación: Especifican como va a realizar el sistema las tareas para las que ha sido construido.

1. Todos los datos del sistema (usuarios, denuncias, actas, etc) serán almacenados en una base de datos.
2. Para cambiar la contraseña de un agente, hay que introducir la nueva dos veces.

Recursos:

1. No existe un límite de filas en las tablas de la base de datos externa, ni en los datos introducidos por un agente en concreto.
2. Existe un límite en el número de conexiones simultáneas que acepta la base de datos externa. Este límite debe de ser de uno, por encima del número de agentes que conforman la plantilla del Municipio.

Comprobación: Especifican las limitaciones que afectan a cómo el software debe verificar los datos de entrada y salida.

1. Verificar que la entrada de datos en campos numéricos (teléfonos, cantidades de dinero, etc) es correcta. No permitir dígitos que no sean numéricos.
2. Verificar la validez de los datos introducidos tales como: DNI, NIF, matriculas, números de serie, etc.
3. Verificar la validez de campos que solo aceptan caracteres, como los nombres de individuos, de provincias, de localidades, etc.
4. Cálculo de los puntos a restar en función de las infracciones de tráfico y circulación cometidas.
5. Cierta información introducida en los formularios se pasa a mayúsculas para ser almacenada en la base de datos de forma transparente al usuario.

Aceptación: Especifican las limitaciones de cómo el software debe de ser validado, es decir, cómo se debe comprobar que el software cumple con los requisitos establecidos.

1. Realizar pruebas de caja negra, para comprobar que según los datos que el agente introduzca en cada formulario el resultado que se obtiene en el resumen de la operación es el esperado.
2. Realizar pruebas de caja blanca, para comprobar que las funciones que hemos definido para la comprobación de campos, funcionan correctamente en todos los casos posibles.

Documentación:

1. El análisis, diseño e implementación serán convenientemente documentadas para facilitar la posterior modificación.

Seguridad: Especifican los requisitos para asegurar el sistema contra amenazas de confidencialidad, integridad y disponibilidad.

1. El Administrador podrá modificar la contraseña de cada agente tantas veces como considere necesario por motivos de seguridad.
2. Para acceder al sistema es necesario introducir un nombre de usuario y contraseña.
3. El acceso a la base de datos debe de estar protegido por una contraseña.
4. Las contraseñas deben de tener como mínimo 5 caracteres.
5. El nombre de usuario será su número de placa precedido de las letras “ag” y sólo podrá ser modificado por el Administrador.
6. El sistema debe de cumplir con las normas establecidas en la Ley de protección de datos de carácter personal [18].
7. Un agente sólo debe acceder a las funcionalidades para las que su grupo y rol le den permisos.

Calidad: Especifican los atributos del software que aseguran que será adecuado para su propósito.

1. El sistema debe de cumplir todos los requisitos especificados en este documento.

Mantenimiento: Especifican la facilidad que tendrá el software para reparar los defectos o adaptarlo a nuevos requisitos.

1. El diseño del sistema debe de contemplar una arquitectura escalable y adaptable a nuevas necesidades.
2. Definir secciones diferentes para cada una de las funcionalidades de la aplicación. De este modo se facilita la escalabilidad.
3. Generar las pestañas y menús de la aplicación, a los que los agentes pueden acceder según los permisos que posean, de forma dinámica, como resultado de la adhesión de una nueva sección o funcionalidad a la aplicación.

CAPÍTULO 4

IMPLEMENTACIÓN DEL SISTEMA DE GESTIÓN POLICIAL *SMALLPOL*

4 SMALLPOL

En este capítulo se explica detalladamente la fase de implementación del proyecto, tanto la parte del gestor de contenido como la base de datos externa. Se tratará de ofrecer una visión completa de lo más representativo, explicando los diferentes elementos que se han utilizado para la implementación del sistema.

Estos elementos son una serie de páginas estáticas y dinámicas, para cuya creación y funcionamiento se han utilizado los lenguajes HTML, XHTML, TAL, JavaScript y Python 2.4.4.
Además de las páginas, se utiliza una serie de scripts de validación también implementados con Python 2.4.4, y un conjunto de funciones desarrolladas en SQL que nos permiten interactuar con la base de datos externa.
Por último, existen unos ficheros de CSS para dar a la interfaz el estilo que se ha considerado más oportuno.

Todos estos elementos se organizan de la manera siguiente en el directorio principal del sitio Plone implementado:

- La aplicación se ha dividido en cuatro secciones, que como se ha comentado anteriormente son Administración Interna, Atestados, Denuncias y Administrativo y Gestión del ciudadano. En la figura 18, podemos observar las carpetas que guardan todas las páginas estáticas, con las opciones que se pueden seleccionar dentro de una sección y las páginas dinámicas, que constituyen los formularios que han de rellenarse para cumplir la funcionalidad de cada sección.
  Además de algunas de las imágenes utilizadas en la aplicación, que se encontrarán fuera de estas carpetas.

Figura 18: Las cuatro secciones principales en las que se divide la aplicación.

También contamos con otro apartado que recopila todas las consultas a la base de datos externa que son necesarias, como se muestra en la figura 19.

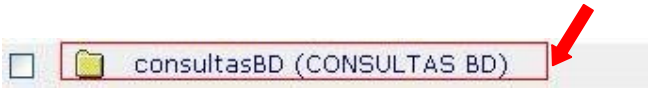

Figura 19: Sección donde se almacenan todas las consultas a la base de datos externa.

Y por último, ya no en el directorio principal sino en un subdirectorio, se encuentra el apartado donde se almacena tanto los scripts de validación creados para este proyecto, como los ficheros de CSS. La figura 20 representa esta carpeta.

Figura 20: Sección "custom" almacena scripts de python y ficheros de CSS.

4.1 Páginas estáticas y dinámicas

Nos referimos con página estática, a aquella en la que solo se muestra un menú de opciones donde cada opción es un enlace, o un párrafo como resultado de una operación, etc, pero siempre son textos que no varían y que no son completados por datos recuperados de una consulta.
Y cuando hablamos de página dinámica, nos referimos a aquella cuyo texto varía, o bien porque según haya sido la página anterior, mostrará unos textos u otros, o bien porque el texto que muestra se completa con datos obtenidos de una consulta a la base de datos.

A continuación nos detendremos en los subdirectorios que se corresponden con cada una de las secciones de la aplicación, para ver su implementación más detalladamente.
Al mostrar el contenido de los directorios utilizaremos unas flechas explicativas tales como:

- Página estática con el menú de opciones de la sección, o el resultado.
- Páginas dinámicas CPT "Controller Page Template", que representan cada formulario o página con el resumen o resultado de la operación realizada.

Administración Interna:

Permite la solicitud de intercambio de turnos, de días de asuntos propios o vacacionales y de cambio de talla del uniforme. Los contenidos del directorio se pueden ver en la figura 21:

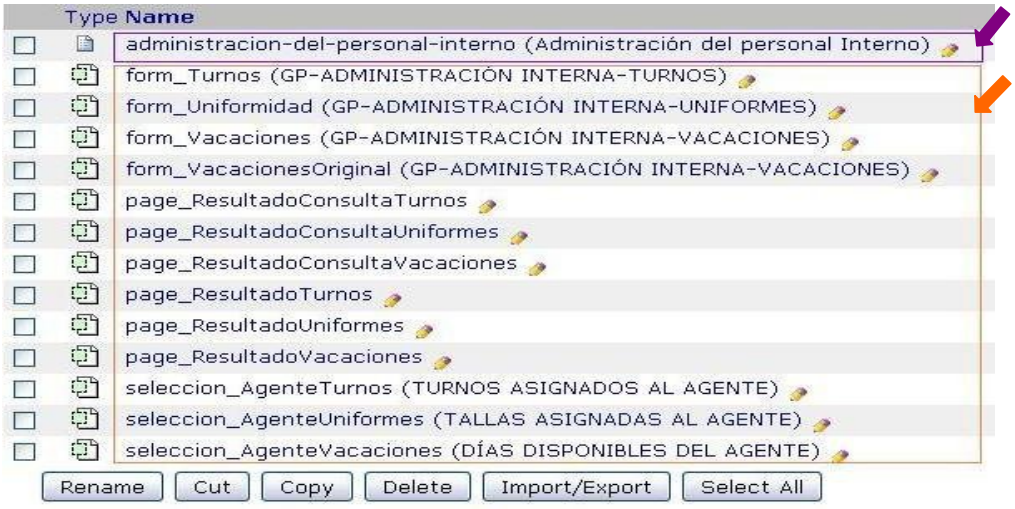

Figura 21: Subdirectorio de Administración Interna.

Atestados:

Permite el registro de los informes realizados por accidente o delito en la base de datos, así como el registro de las citaciones judiciales pertinentes. Los contenidos del directorio se pueden ver en la figura 22:

Figura 22: Subdirectorio de Atestados.

Denuncias y Administrativos:

Permite el registro de las denuncias producidas en el municipio en la base de datos, así como el registro de diferentes trámites administrativos. Los contenidos del directorio se pueden ver en la figura 23:

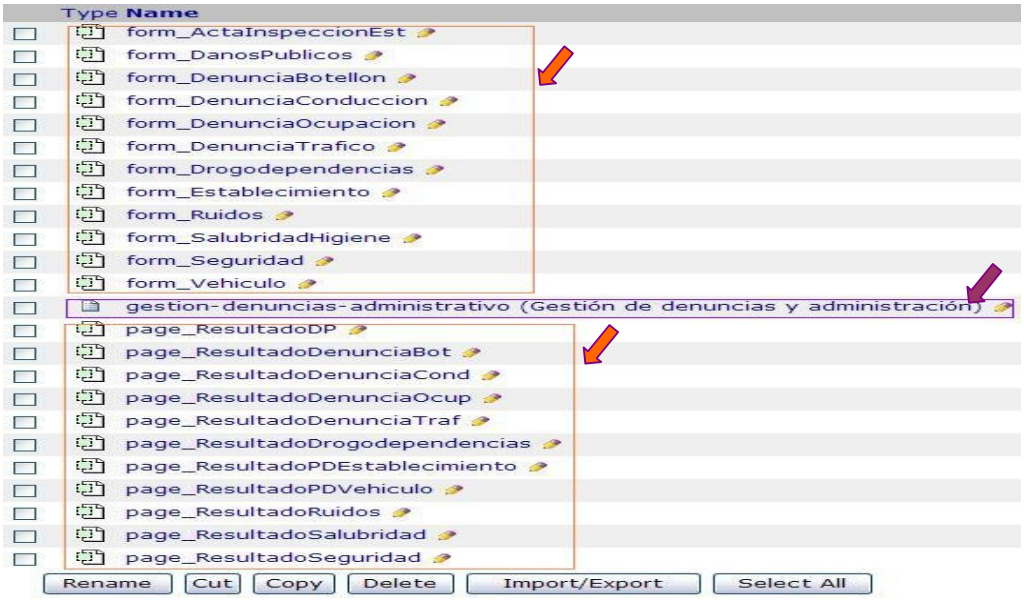

Figura 23: Subdirectorio de Denuncia y Trámites Administrativos.

Gestión del ciudadano:

Permite la obtención de la información almacenada en la base de datos a cerca de un ciudadano, a partir de su DNI, así como la modificación o la eliminación de la misma.

Los contenidos del directorio se pueden ver en la figura 24:

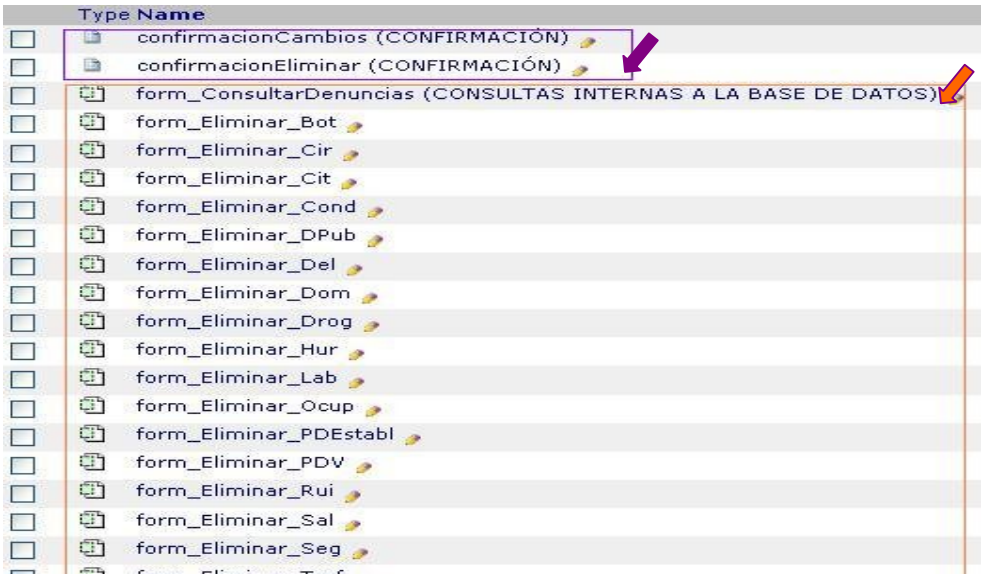

Figura 24: Subdirectorio de Gestión del ciudadano.

A continuación vamos a mostrar una serie de ejemplos de páginas estáticas y dinámicas, señalando que lenguajes se han utilizado en combinación con HTML para crearlas.

- **Página estática en HTML:**

Un ejemplo de página estática en HTML [15], puede ser cualquiera de los menús principales de las secciones que componen la aplicación. Están escritos con HTML únicamente. En la figura 25, podemos ver uno de esos menús.

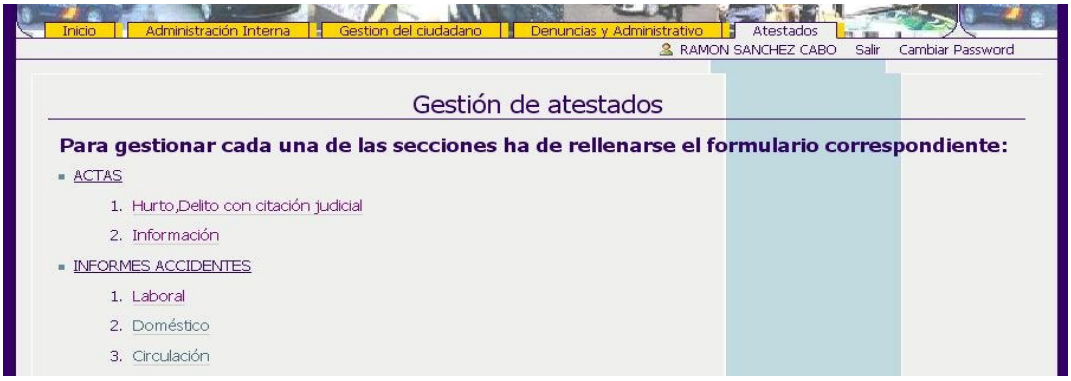

Figura 25: Página estática con el menú principal de la sección Atestados.

- **Página dinámica en XHTML , TALES y Python:**

En los siguientes ejemplos se ha combinado XHTML con el lenguaje de TALES [14] y Python. Este tipo de combinación se utiliza en la mayor parte de los formularios de la aplicación. Las expresiones de estos lenguajes, se utilizan para recoger y mostrar los datos obtenidos de una consulta a la base de datos, para indicar el nombre del error que se puede dar en un campo de un formulario o para seleccionar que texto es mostrado según el valor de los datos obtenidos, como suele hacerse en las páginas de resultados.

La información obtenida a través de una consulta se puede mostrar en los desplegables, para poder ser seleccionada, como es el caso que se muestra en la figura 26:

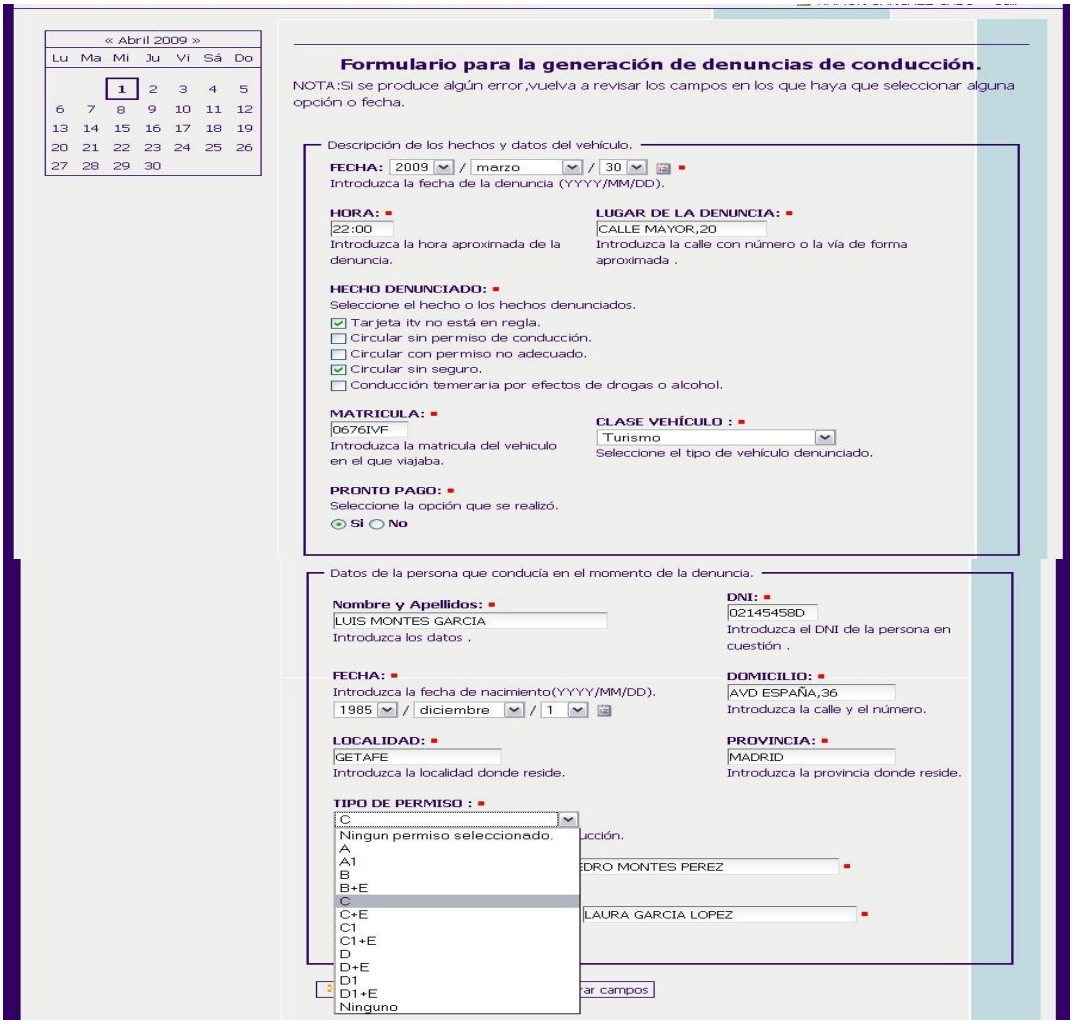

Figura 26: Formulario para la generación de la Denuncia por infracciones en la Conducción.

O simplemente para mostrarla organizada dentro de una tabla, como es el caso que se puede observar en la figura 27:

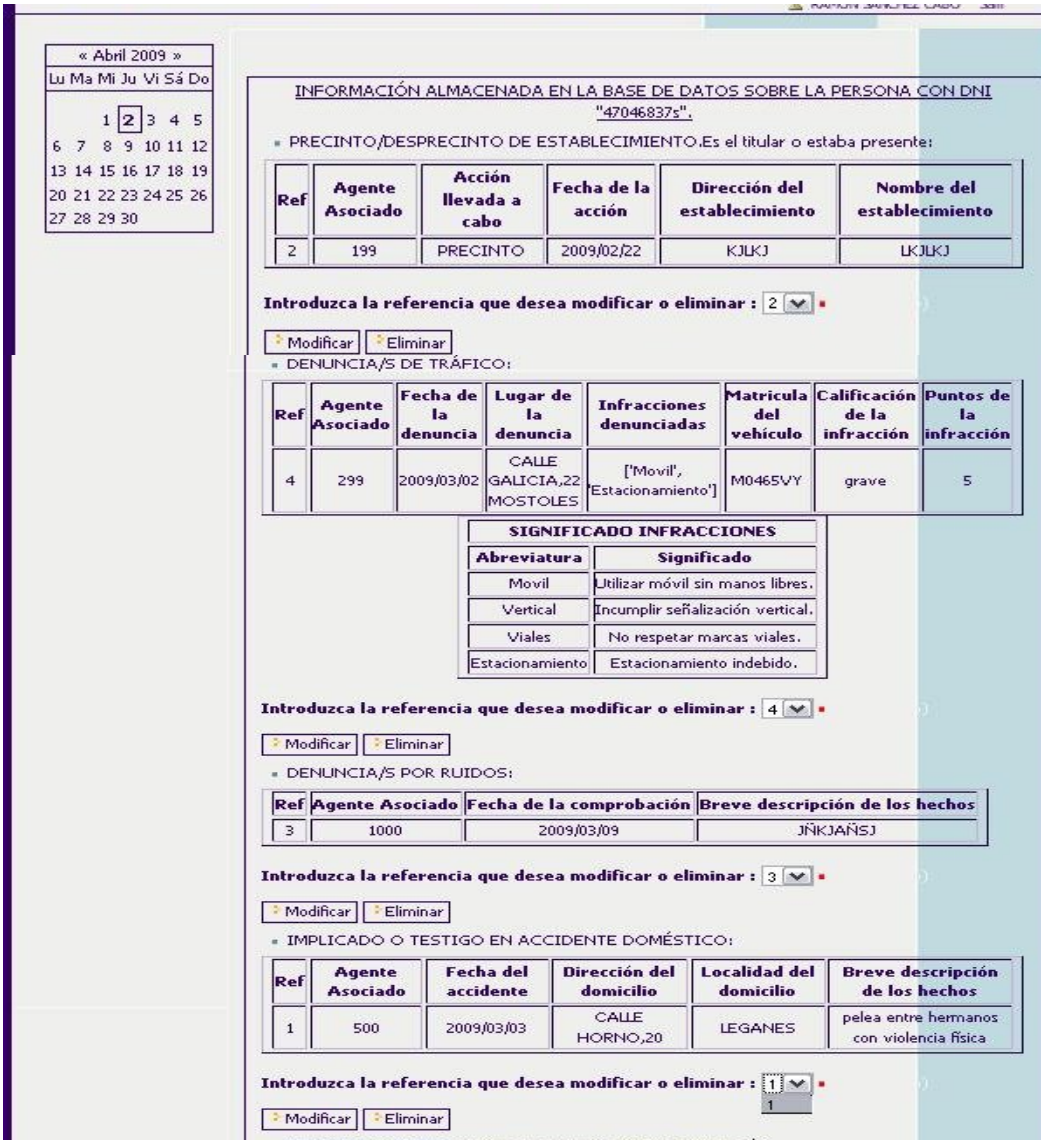

Figura 27: Muestra parte de la información que contiene la base de datos sobre el individuo en cuestión.

También tenemos un ejemplo del código combinado de estos tres lenguajes, en él, se puede ver la declaración del error que se puede dar en ese campo, utilizando tanto lenguaje de TALES como Python.

```
<div id="etiqueta" class="field" tal:define="error_placa
errors/placa|nothing;" tal:attributes="class python:test(error_placa,
'field error', 'field')">
<label i18n: translate="placa_feedback_comments">Nº AGENTE QUE REALIZA
EL ACTA :</label>
<span class="fieldRequired" title="Required" i18n:attributes="title"
i18n:translate="label_required">(Required)</span><br/>
<div tal: condition="error_placa">
<tal: block i18n: translate="" content="error_placa">Error</tal:block>
</div>
```

- **Página dinámica en XHTML , TALES, Python y JavaScript:**

Se ha utilizado JavaScript para el control de los calendarios que aparecen en multitud de formularios de la aplicación. De esta manera al pinchar sobre la imagen del calendario que hay a la derecha de los campos correspondientes, aparecerá un calendario emergente para que el usuario pueda seleccionar el día que desee pinchando sobre él simplemente. Este calendario lo podemos ver en la figura 28:

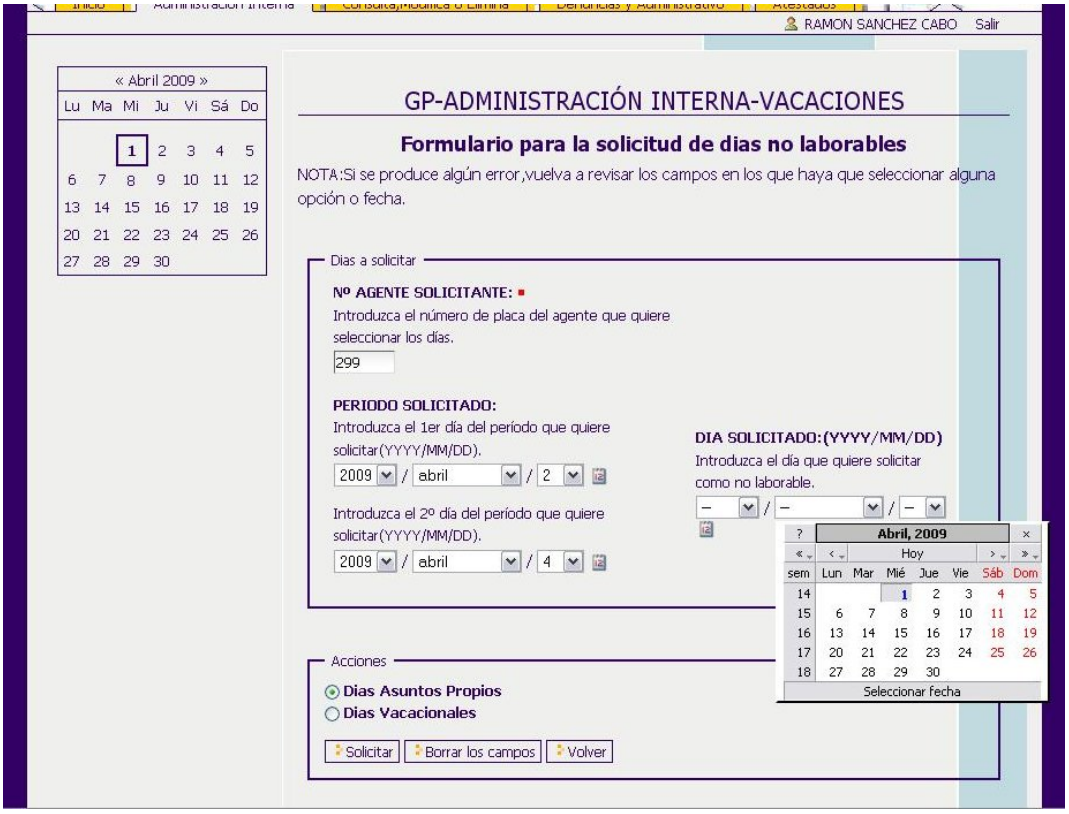

Figura 28: Formulario de solicitud para días no laborables a rellenar.

El código para la llamada de este script se muestra a continuación:

```
<metal:javascriptslot fill-slot="javascript_head_slot">
        <style type="text/css" media="all"><!-- @import
url(../jscalendar/calendar-system.css); --></style>
        <script type="text/javascript"
            src="../jscalendar/calendar_stripped.js"></script>
        <script type="text/javascript" charset="iso-8859-1"
            src="../jscalendar/calendar-es.js"></script>
</metal:javascriptslot>
```

4.2 Sentencias SQL

En este apartado centraremos nuestra atención en mostrar el subdirectorio que guarda las consultas SQL [16] para la base de datos relacional externa.

Algunos de los contenidos del subdirectorio se pueden observar en la figura 29:

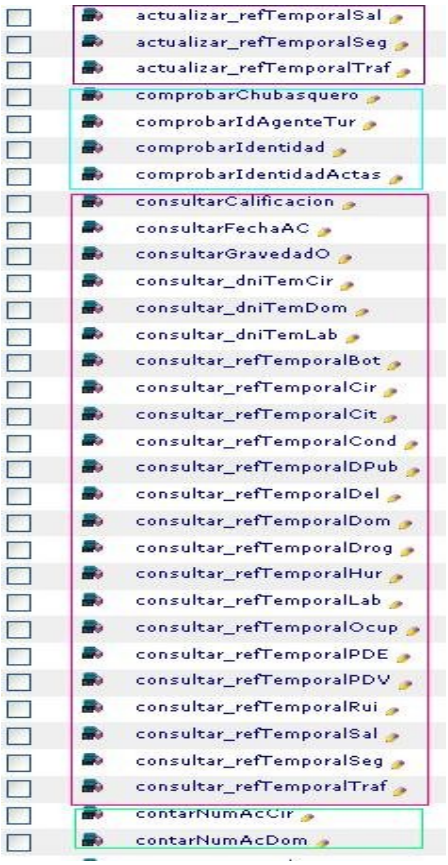

Figura 29: Muestra de algunos de los diferentes tipos de consultas utilizados.

Las consultas utilizadas son de diversos tipos, de actualización, de consulta, de eliminación, de inserción, etc. Un ejemplo es el siguiente:

```
select referencia,fecha,hora,placa,deposito from
"tb_PDVehiculo" where accion='PRECINTO' and matricula=<dtml-
sqlvar matriculaOK type="string">;
```

En este caso se trata de una consulta de selección con la que se quiere obtener una cierta información sobre el Acta de Precinto de un Vehículo.

4.3 Scripts de validación en Python

Se describe el subdirectorio donde se guardan los scripts validadores. Mostraremos las funciones más representativas que se utilizan dentro de los scripts. Algunos de los contenidos de este subdirectorio pueden verse en la figura 30:

Figura 30: Muestra de algunos de los scripts de Python (Controller Validator)

En total hay como unos 60 scripts de validación. Estos scripts se encargan de controlar los valores introducidos en los campos y de realizar las operaciones que sean oportunas para cumplir la funcionalidad de la opción, asignándose el correspondiente a cada CPT.

A cada CPT también se les asigna una acción “Actions” que les indica la página a la que deben dirigirse si hay algún error o si por el contrario el proceso finaliza sin fallos.
Dentro de estos scripts se definen una serie de funciones. A continuación se exponen algunos ejemplos de ellas:

Función “obtenerPlaca”:

Obtiene el número de placa del agente autenticado en la sesión actual. Cada registro de datos en la base de datos se acompaña del número de placa del agente que lo realiza.
En este caso no se le pasa ningún parámetro de entrada, y lo que devuelve es una variable con el número de placa en cuestión.
Además se utiliza un número de placa para el Administrador de la aplicación, para que cuando éste realice alguna prueba de funcionamiento, no aparezca un error porque el usuario de la sesión no tiene asignado un número de placa. El código puede verse en la figura 31:

```
def obtenerPlaca():
    #Obtenemos el usuario registrado en la aplicación
    miembro = context.portal_membership.getAuthenticatedMember()

    #Obtenemos su nombre de usuario
    usuario = miembro.getId()
    if usuario!='admin':
           placa=usuario.strip('ag')
           placa=int(placa)
    else:
           placa=500
    return placa
```

Figura 31: Código de la Función “obtenerPlaca”.

Función “comprobarHora”:

Se encarga de comprobar que la hora introducida sea correcta en formato y con valores adecuados, comprobando también que la hora sea adecuada según la fecha en que se produjo, es decir, si se está utilizando la aplicación para almacenar en la base de datos los datos el mismo día de la denuncia, la hora de la denuncia no puede ser mayor a la

del momento de uso de la aplicación. Este ejemplo en concreto, se refiere a la hora en la que se produce la denuncia de Conducción y puede verse en la figura 32.

Los parámetros que se le pasan, se identifican de la siguiente manera:

- String con la hora de la denuncia→ "h".
- String con la fecha de la denuncia→ "f"
- String con el nombre del campo del formulario donde se introduce la hora→ "campo".
- Variable con el valor del error, 1 si hay error 0 si no lo hay→ "errorHora".

```
def comprobarHora(f,h,campo):
    #variables
    errorHora=0
    hh = h[0:2]
    mm = h[3:5]
    try:
         hh = int(hh)
         mm = int(mm)

         if hh==24:
             errorHora=1
             state.setError(campo, 'Para referirse a las 24:00 horas
                            use 00:00. ')
         elif hh < 0 or hh > 23:
             errorHora=1
             state.setError(campo, 'Hora fuera de rango')
         if mm <0 or mm >59:
             errorHora=1
             state.setError(campo, 'Minutos fuera de rango')
    except ValueError:
             errorHora=1
             state.setError(campo, 'Debe introducir la hora
                            de forma correcta')
    if errorHora==0 and f:
        fechaDia=DateTime(int(f[0:4]),int(f[5:7]),int(f[8:]))

        if fechaDia.isCurrentDay(): #En este caso la hora tiene
                                    #que ser menor a la hora  actual.
           if hh>DateTime().hour():
                errorHora=1
                state.setError(campo, 'La hora no debe ser superior
                               a la actual.')
           elif hh==DateTime().hour():
                if mm>DateTime().minute():
                   errorHora=1
                   state.setError(campo, 'La hora no debe ser superior
                                  a la actual.')
    return errorHora
```

Figura 32: Código de la Función "comprobarHora".

Función "calcularImportePuntos":

Se trata del cálculo de puntos y dinero que supone cometer una infracción en la conducción, teniendo en cuenta que se hace uso del pronto pago o no.

Los parámetros que se le pasan, se identifican de la siguiente manera:

- Array de strings con las infracciones que se han cometido→ "hecho".
- String con el tipo de vehículo que conduce el infractor→ "clases".
- String con el tipo de permiso que posee el infractor→ "permiso".
- Variable de tipo String que indica si se ha seleccionado el pronto pago o no→ "pago".
- El número de puntos e importe correspondiente→ "resultado".

El código de la función se muestra en la figura 33:

```
def calcularImportePuntos(hecho,clases,permiso,pago):
      #variables
      resultado=[0,0]
      i=0
      puntos=0
      importe=0
      numI=len(hecho)
      if numI>5:#No es array solo hay 1 hecho.
          numI=1
      if numI==5:#Puede ser array o el hecho 'NoItv'
          if hecho[0]=='N':#Es solo el hecho 'NoItv'
             numI=1
      while i<numI :
         if hecho[i]=='NoItv' or hecho=='NoItv' :
            if pago=='SiPago':
                 importe=importe+105
            else:
                 importe=importe+150
         elif hecho[i]=='NoPermiso' or hecho=='NoPermiso' :
            if pago=='SiPago':
                 importe=importe+315
            else:
                 importe=importe+450
         elif hecho[i]=='NoAdecuado' or hecho=='NoAdecuado' :
            if pago=='SiPago' and permiso!='A' and permiso!='A1':
                 importe=importe+315
                 puntos=puntos+4
            if pago=='NoPago' and permiso!='A' and permiso!='A1':
                 importe=importe+450
                 puntos=puntos+4
            if pago=='SiPago' and permiso=='A' and clases!='Ciclomotor' :
                 importe=importe+160
                 puntos=puntos+4
            if pago=='NoPago' and permiso=='A' and clases!='Ciclomotor' :
                 importe=importe+225
                 puntos=puntos+4
            if pago=='NoPago' and permiso=='A1' and clases!='Motocicleta'
            and clases!='Ciclomotor':
                 importe=importe+225
                 puntos=puntos+4
            if pago=='SiPago' and permiso=='A1' and clases!='Motocicleta'
            and clases!='Ciclomotor':
                 importe=importe+160
                 puntos=puntos+4
         elif hecho[i]=='NoSeguro' or hecho=='NoSeguro' :
            importe=importe+610
         elif hecho[i]=='Temeraria' or hecho=='Temeraria' :
            importe=importe+520
            puntos=puntos+5

         i=i+1
      resultado=[importe,puntos]
      return resultado
```

Figura 33: Código de la Función “calcularImportePuntos”.

4.4 Hojas de estilo (CSS)

Las hojas de estilo se encuentran en el mismo subdirectorio que los scripts de validación, el cual se muestra en la figura 34:

Figura 34: Muestra de los ficheros que se encargan del estilo de la aplicación.

Para la personalización de la apariencia de la aplicación, en este caso, se han modificado los valores de algunas propiedades que vienen por defecto en Plone, así como cambios en algunas declaraciones de los CSS que también vienen con Plone. Algún ejemplo se muestra en la figura 35:

```
#visual-portal-wrapper {
  margin: 0 auto 0 auto; width: 880px; background-color:#EFEFEE;
background-image: url(../FONDOCUERPO.JPG );
}
body { background-repeat: no-repeat; background-position: center;
background-color: #330066;
}
.documentContent {
font-size: 110%;
padding: 1em 1em 2em 1em !important;
background-image: url(../FONDOCUERPO.JPG );
}
h1,h2,h3{
text-align: center;
}
.documentFirstHeading {
font-size: 20px bold;
}
.cabecera{
font-size: 15px;
text-align: center;
text-style: bold;
}
#portal-top {
    margin-top: 20px;
    background-image: url(../CABECERA );
}
```

Figura 35: Una muestra de la hoja de estilo "ploneCustom.css".

Con esto se finaliza la muestra de ejemplos para poder hacerse una idea de cómo se ha implementado la aplicación web, en cuanto a lo que se refiere al gestor de contenido como tal.

4.5 Base de Datos externa con PostgreSQL

En este apartado se define la estructura física de la base de datos a través de un modelo relacional. La base de datos cuenta en total con 43 tablas, y pueden dividirse para su descripción en dos grupos. Uno sería el grupo de tablas cuyos datos son fijos, y que se utilizan para funcionar internamente, como pueden ser las tablas que almacenan los datos que aparecen en los listbox de los formularios, etc. Y por otro lado tendríamos las tablas en las que se guardan los datos dinámicos y variables, son tablas que van aumentando o disminuyendo el número de filas, a medida que el agente va interactuando con la aplicación.

Por lo tanto una muestra de las tablas que componen la base de datos, sería la que se muestra en la figura 36:

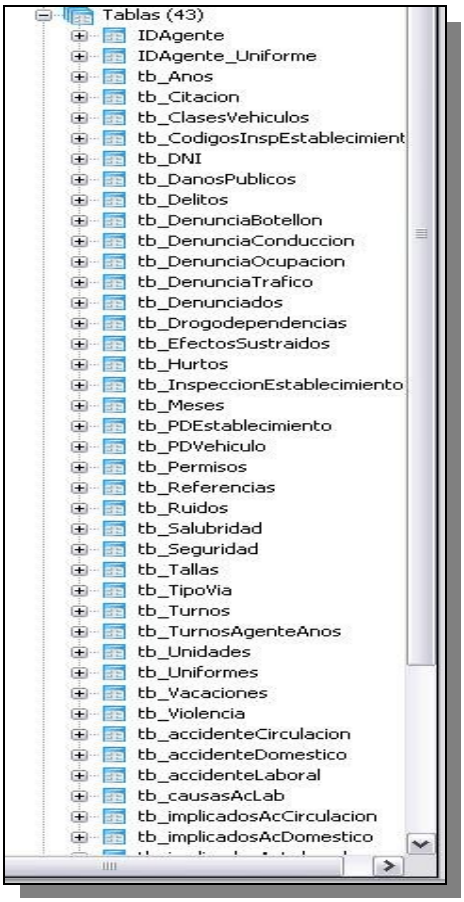

Figura 36: Muestra la relación de tablas que forman la Base de Datos externa.

Seguidamente se muestran la estructura, que forman las tablas que son utilizadas para cada una de las secciones que componen la aplicación, por separado, junto con las relaciones entre ellas y las claves primarias definidas.

En este diagrama se pueden ver los dos tipos de tablas de la base de datos, que son:

- Las tablas "estáticas": Estas tablas no modifican sus valores durante la ejecución de la aplicación, pero si pueden variar, tanto en el valor de los datos, como en la definición de la tabla, si se amplían las funcionalidades de la aplicación en versiones futuras. Dentro de este grupo también se incluyen dos tablas, cuyos valores no cambian aunque se produzcan ampliaciones de funcionalidad en la aplicación web.
- Las tablas "dinámicas": Estas tablas van variando sus datos y el tamaño de las mismas, a medida que el usuario va gestionando información a través de la aplicación web. El número de estas tablas variará según se modifiquen las funciones de la aplicación.

Con lo cual los diagramas entidad-relación por secciones son:

Administración Interna:

En este diagrama que refleja la figura 37, existen tablas no relacionadas, que son tablas estáticas, encargadas de almacenar los datos que se muestran en algunos de los desplegables de los formularios de esta sección.

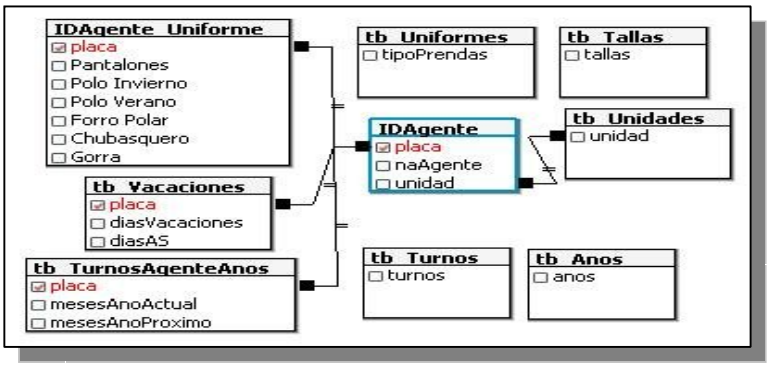

Figura 37: Diagrama entidad-relación de las tablas utilizadas en la sección de Administración Interna.

Atestados:

En este diagrama que refleja la figura 38, aparecen también algunas tablas estáticas pero en este caso si están relacionadas. Son las encargadas de almacenar los diferentes tipos de violencia que se pueden usar, o los diferentes tipos de permisos que existen, etc.

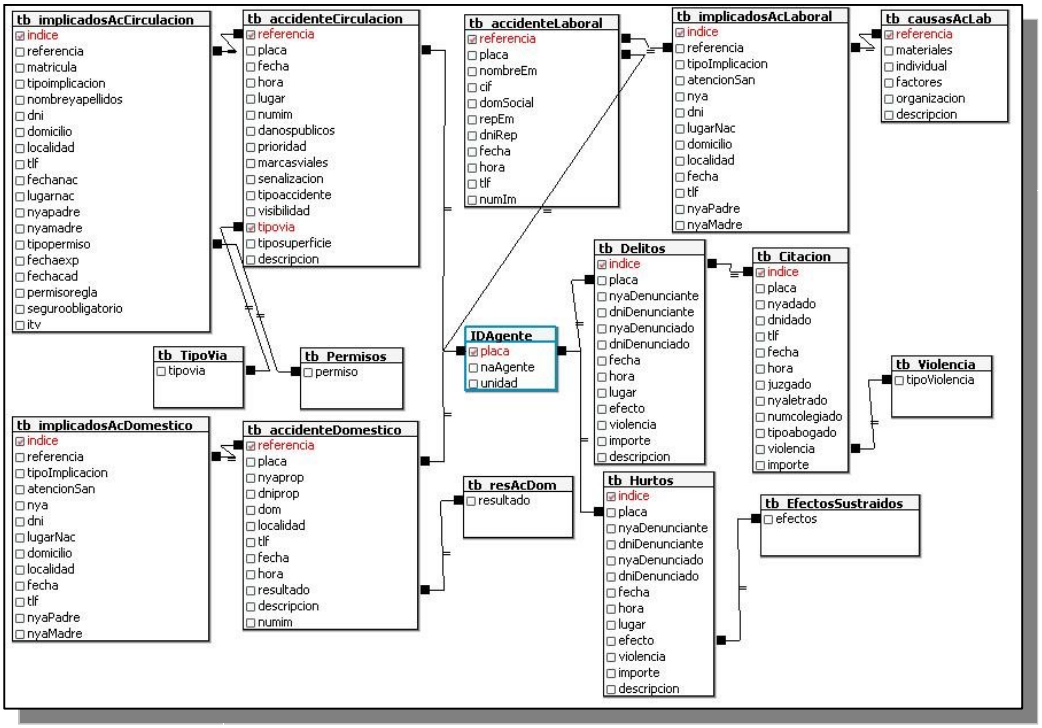

Figura 38: Diagrama entidad-relación de las tablas utilizadas en la sección de Atestados.

Denuncias y Administrativo:

En este diagrama que refleja la figura 39, aparecen también algunas tablas estáticas pero en este caso como en el anterior si están relacionadas. Son las encargadas de almacenar los diferentes tipos de vehículos, o las diferentes marcas de vehículos que existen.

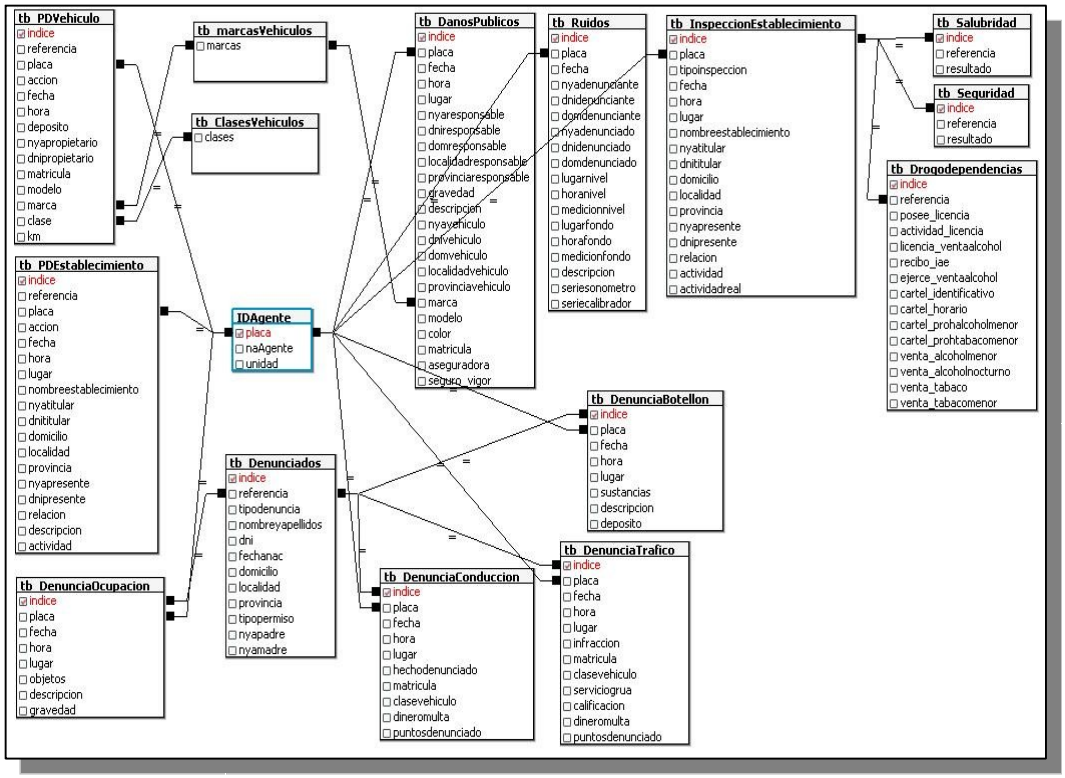

Figura 39: Diagrama entidad-relación de las tablas utilizadas en la sección de Denuncias y Administrativo.

<u>Gestión del ciudadano:</u>

En este diagrama que refleja la figura 40, aparecen las mismas tablas estáticas y dinámicas que aparecen en las figuras 38 y 39, puesto que se realizan consultas, tanto de selección como de actualización y eliminación, sobre esas mismas tablas. Además de estas tablas, aparecen las que se encargan de guardar las referencias de los registros que se van a actualizar o eliminar.

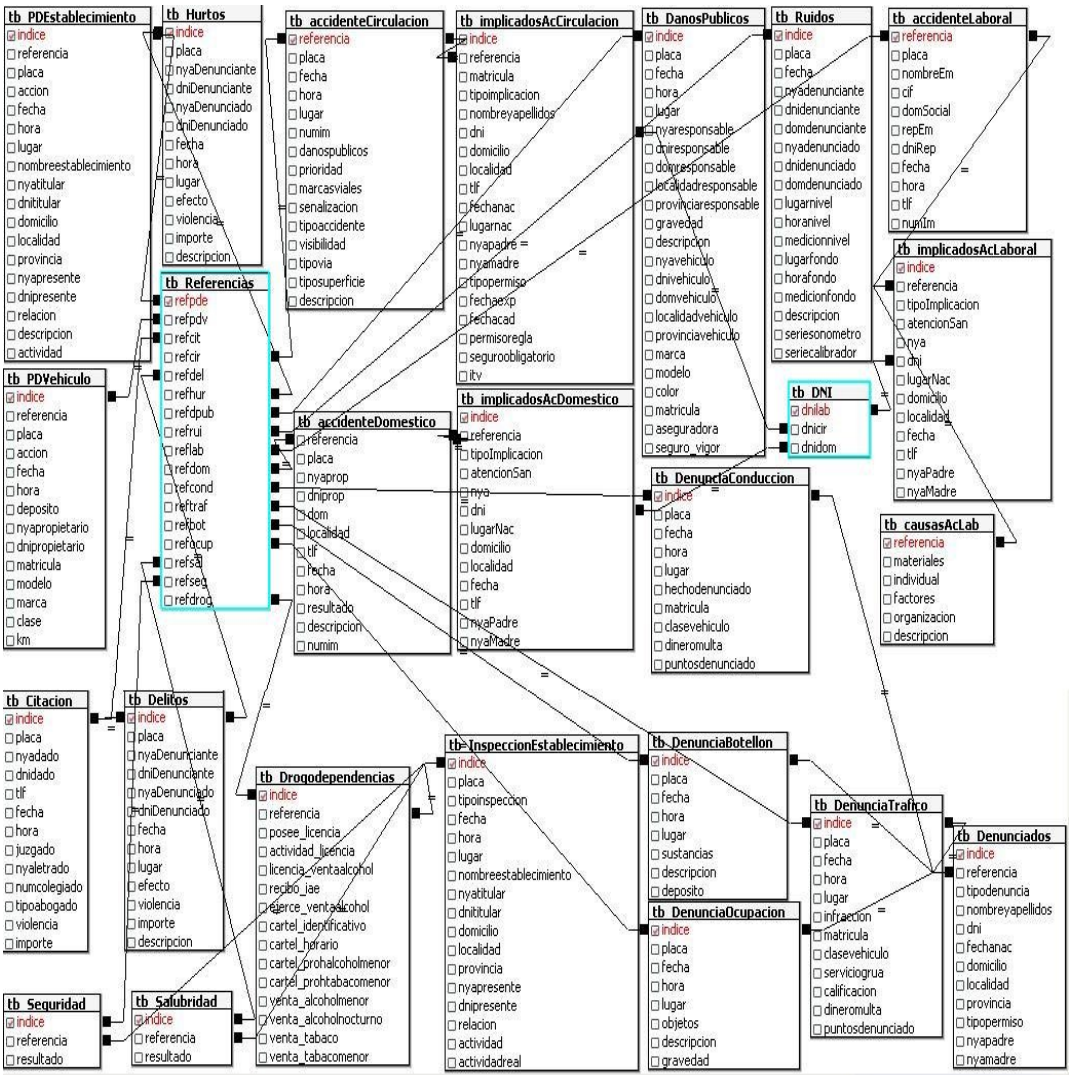

Figura 40: Diagrama entidad-relación de las tablas utilizadas en la sección de Gestión del ciudadano.

Para finalizar este apartado, vamos a mostrar algunos ejemplos tanto de tablas estáticas como dinámicas, para ver como es su contenido en el caso de las estáticas, y su declaración en ambos casos.

- Ejemplos tablas estáticas:

La tabla “tb_Violencia” que contiene los diferentes tipos de violencia que pueden ser usados en un delito. La declaración y el contenido se muestran en la figura 41:

```
CREATE TABLE "tb_Violencia"
(
  "tipoViolencia" character varying NOT NULL,
  CONSTRAINT "tb_Violencia_pkey" PRIMARY KEY ("tipoViolencia")
)
WITH (OIDS=FALSE);
ALTER TABLE "tb_Violencia" OWNER TO postgres;
```

	tipoViolencia [PK] character varying
1	Arma Blanca
2	Arma de fuego
3	Física
4	Otros
*	

Figura 41: Declaración y contenido de la tabla "tb_Violencia".

La tabla "tb_Permisos" que contiene todos los tipos de permisos de conducción actualmente en vigor. Vease la figura 42:

```
CREATE TABLE "tb_Permisos"
(
  permiso character varying NOT NULL,
  CONSTRAINT "tb_Permisos_pkey" PRIMARY KEY (permiso)
)
WITH (OIDS=FALSE);
ALTER TABLE "tb_Permisos" OWNER TO postgres;
```

	permiso [PK] character varying
1	A
2	A1
3	B1
4	B+E
5	C
6	C+E
7	C1
8	C1+E
9	D
10	D+E
11	D1
12	D1+E
13	Ninguno
*	

Figura 42: Declaración y contenido de la tabla "tb_Permisos".

- Ejemplos tablas dinámicas:

La tabla "tb_inspeccionEstablecimiento" y la tabla "tb_Drogodependencias"se utilizan de forma conjunta para almacenar los datos generados al realizar un Acta de Inspección de Drogodependencias y trastornos adictivos en Establecimientos. La declaración de las tablas puede verse en las figuras 43 y 44 respectivamente.

En la primera tabla se guarda la información para identificar al establecimiento, los responsables y el tipo de inspección que se le realiza. Y en la segunda se guardan los resultados, a los diferentes aspectos que se controlan en la inspección. La relación entre ellas se hace a través del campo "índice" de la primera y el campo "referencia" de la segunda. Este mismo razonamiento se lleva a cabo de igual forma con los otros tipos de inspecciones.

```
CREATE TABLE "tb_InspeccionEstablecimiento"
(
  indice oid NOT NULL,
  placa numeric NOT NULL,
  tipoinspeccion character varying(15) NOT NULL,
  fecha date NOT NULL,
  hora time without time zone NOT NULL,
  lugar character varying NOT NULL,
  nombreestablecimiento character varying NOT NULL,
  nyatitular character varying NOT NULL,
  dnititular character varying NOT NULL,
  domicilio character varying NOT NULL,
  localidad character varying NOT NULL,
  provincia character varying NOT NULL,
  nyapresente character varying NOT NULL,
  dnipresente character varying NOT NULL,
  relacion character varying NOT NULL,
  actividad character varying NOT NULL,
  actividadreal character varying NOT NULL,
  CONSTRAINT "tb_InspeccionEstablecimiento_pkey" PRIMARY KEY (indice)
)
WITH (OIDS=FALSE);
ALTER TABLE "tb_InspeccionEstablecimiento" OWNER TO postgres;
```

Figura 43: Declaración de la tabla "tb_inspeccionEstablecimiento".

```
CREATE TABLE "tb_Drogodependencias"
(
  indice oid NOT NULL,
  referencia numeric NOT NULL,
  posee_licencia character varying NOT NULL,
  actividad_licencia character varying NOT NULL,
  licencia_ventaalcohol character varying NOT NULL,
  recibo_iae character varying NOT NULL,
  ejerce_ventaalcohol character varying NOT NULL,
  cartel_identificativo character varying NOT NULL,
  cartel_horario character varying NOT NULL,
  cartel_prohalcoholmenor character varying NOT NULL,
  cartel_prohtabacomenor character varying NOT NULL,
  venta_alcoholmenor character varying NOT NULL,
  venta_alcoholnocturno character varying NOT NULL,
  venta_tabaco character varying NOT NULL,
  venta_tabacomenor character varying NOT NULL,
  CONSTRAINT "tb_Drogodependencias_pkey" PRIMARY KEY (indice)
)
WITH (OIDS=FALSE);
ALTER TABLE "tb_Drogodependencias" OWNER TO postgres;
```

Figura 44: Declaración de la tabla "tb_Drogodependencias".

4.6 Acceso seguro

Plone es muy seguro por si solo a nivel de aplicación, pero si le añadimos unas precauciones extra, puede llegar a ser un entorno también muy seguro, a nivel de comunicaciones. Por lo tanto para implementar esta seguridad extra, se necesita instalar un servidor web seguro como frontal [7], ese servidor se trata de un Apache 2.2.11.

Con lo cual se procedió a la instalación, configuración e integración de este servidor con Plone/Zope para dar seguridad a la comunicación, debido que la información que se gestiona es sensible y los usuarios del sistema pueden acceder a él de forma remota.

Y se generó un certificado de seguridad a partir de una CA "Certification Authority" para nuestro sitio.

Con estos elementos se procede a establecer como única forma de acceso a nuestro sitio Plone, HTTPS, garantizando así, que toda la información esté cifrada.
Para ello, una vez instalado el servidor Apache adecuado a nuestro entorno e instalado el certificado de seguridad, se han escrito una serie de reglas en el fichero de configuración de Apache y en el que se define el "VirtualHost" para HTTPS, para redirigir los accesos a nuestro sitio Plone, hacia un acceso seguro a través de HTTPS.

En estas reglas se añaden dos claves especiales (VirtualHostBase y VirtualHostRoot) que permiten configurar el "VirtualHost" y la carpeta base interna a la instancia Zope.

- La clave "VirtualHostBase" se utiliza para iniciar el "Virtual Hosting", estableciendo la máquina y el puerto.
- Y la clave "VirtualHostRoot" indica la raíz del "VirtualHost". Todo lo que va después de "VirtualHostRoot" será visible por el navegador.

Por lo tanto la regla para el fichero httpd.conf, en la que se redirige hacia nuestro servidor Apache, es:

```
RewriteEngine on
RewriteRule ^/SMALLPOL(.*)$
http://debussy.gast.it.uc3m.es:8080/VirtualHostBase/http/debussy.gast.
it.uc3m.es:8880/VirtualHostRoot/SMALLPOL/$1 [L,P]
```

La regla para el fichero httpd_ssl.conf para redirigir hacia el acceso seguro es:

```
RewriteEngine On
RewriteRule ^/SMALLPOL(.*)$
http://debussy.gast.it.uc3m.es:8080/VirtualHostBase/https/debussy.gast
.it.uc3m.es:8443/VirtualHostRoot/SMALLPOL/$1 [L,P]
```

Para que estas reglas que suponen un cambio de URLs, sean entendibles por Zope, en la raíz de nuestra instancia de Zope, debemos añadir un objeto "VirtualHostMonster". Este objeto se encarga de traducir la información que le pasa Apache a través de las reglas y así hacer que Zope reproduzca correctamente las páginas, los vínculos, etc, en el navegador, completando así la implementación del acceso seguro a nuestro sistema de gestión.

CAPÍTULO 5

PRUEBAS

5 PRUEBAS

En este capítulo se describen los diferentes tipos de pruebas realizados durante la fase de desarrollo de la aplicación, así como las pruebas de usuario realizadas a posteriori, para comprobar el correcto funcionamiento de SMALLPOL.

Es importante recordar que las pruebas no pueden garantizar al 100% que el software esté libre de fallos, pero si deben intentar cubrir el mayor número de situaciones de error posible.

Se relacionará en la medida de lo posible, la prueba con el requisito funcional o no funcional que se corresponda, indicando un código para identificar el tipo de requisito y a continuación el número del requisito en cuestión.

5.1 Interfaz de usuario

Para la realización de esta batería de pruebas se ha utilizado el perfil de Agente que pertenece a la unidad Administrativa, y el que pertenece a cualquier otra unidad.

5.1.1 Acceso a la aplicación

A través del navegador nos conectamos a la aplicación, obteniendo la página principal de la misma. En ella se comprueba que no hay posibilidad alguna de acceder a las secciones que la componen, sin autenticarse previamente, funcionando por lo tanto, como una intranet.

5.1.2 Proceso de Autenticación

Se trata de verificar que funciona correctamente la validación de los usuarios para poder acceder a las secciones de la aplicación.

Requisito	Requisito-Funcional-3
Datos de entrada	Nombre de usuario registrado en el sistema y la contraseña asociada a dicho usuario. Pulsar Entrar.
Resultado esperado	No debe producirse ningún error. El usuario accede al sistema. Se muestra una pantalla informándole de que está identificado.
Resultado	CORRECTO.

Requisito	Requisito-Funcional-3
Datos de entrada	Nombre de usuario registrado en el sistema y una contraseña aleatoria. Pulsar Entrar.
Resultado esperado	Se informa al usuario que el nombre de usuario o la contraseña son incorrectos y que estos dos elementos son sensibles a mayúsculas o minúsculas.
Resultado	CORRECTO.

Requisito	Requisito-Funcional-3
Datos de entrada	Nombre de usuario no registrado en el sistema y una contraseña. Pulsar Entrar.
Resultado esperado	Se informa al usuario que el nombre de usuario o la contraseña son incorrectos y que estos dos elementos son sensibles a mayúsculas o minúsculas.
Resultado	CORRECTO.

Una vez autenticado el agente, comprobamos que el agente de la unidad Administrativa puede visualizar las cuatro pestañas correspondientes a las cuatro secciones que componen la aplicación, mientras que el agente de cualquier otra unidad, solo puede visualizar tres de las cuatro pestañas que conforman la aplicación.

Además una vez autenticados, podemos verificar si la modificación de la contraseña de un usuario, por parte del usuario, funciona correctamente.

Requisito	Requisito-Funcional-4
Datos de entrada	Introducir un usuario registrado en el sistema. Introducir la contraseña asociada a dicho usuario. Introducir la nueva contraseña. Repetir la nueva contraseña. Pulsar aceptar.
Resultado esperado	La contraseña se actualiza correctamente.
Resultado	CORRECTO.

Requisito	Requisito-Funcional-4
Datos de entrada	Introducir un usuario registrado en el sistema. Introducir la contraseña asociada a dicho usuario. Introducir la nueva contraseña. Repetir la nueva contraseña de forma errónea Pulsar aceptar.
Resultado esperado	La contraseña no se actualiza correctamente. Se informa al usuario con un mensaje.
Resultado	CORRECTO.

Requisito	Requisito-Funcional-4
Datos de entrada	Introducir un usuario registrado en el sistema. Introducir una contraseña aleatoria para dicho usuario. Introducir la nueva contraseña. Repetir la nueva contraseña. Pulsar aceptar.
Resultado esperado	La contraseña no se actualiza correctamente. Se informa al usuario con un mensaje.
Resultado	CORRECTO.

5.1.3 Navegación entre secciones

Una vez autenticado con éxito, se inicia la sesión y se comprueba que arriba a la derecha aparece el nombre completo del agente en cuestión. La siguiente serie de pruebas se realizan para cada una de las cuatro secciones que componen la aplicación.

Para todas las secciones a excepción de la sección de “Gestión del ciudadano”:

Prueba	Seleccionar las diferentes secciones, pinchando sobre la pestaña correspondiente.
Resultado esperado	Se muestra la página con el menú de opciones que ofrece la sección. Cada opción es un enlace a otra página.
Resultado	CORRECTO.

Para la sección “Gestión del ciudadano”:

Requisito	Requisito-Funcional-25
Prueba	Seleccionar sección, pinchando sobre la pestaña “Gestión del ciudadano”.
Resultado esperado	Se muestra un formulario en el que introducir el DNI de la persona sobre la que se quiere consultar, modificar o eliminar la información almacenada en la base de datos externa.
Resultado	CORRECTO.

Prueba	Comprobar que se visualizan correctamente los botones de “Modificar” o “Eliminar” y el desplegable con las referencias, debajo de cada una de las tablas con la información que se tiene almacenada en la base de datos de un ciudadano.
Resultado esperado	Se muestra la página con la/las tablas con los datos, junto con los dos botones debajo de cada una de ellas y el desplegable con la lista de referencias que se muestran en las tablas.
Resultado	CORRECTO.

Para todas las secciones:

Prueba	Seleccionar cada uno de los enlaces correspondientes a las distintas opciones del menú de la sección.
Resultado esperado	Se muestra el formulario correspondiente a la opción seleccionada, con un título identificado y los campos requeridos con un texto de ayuda adicional.
Resultado	CORRECTO.

Prueba	Comprobar el funcionamiento de los botones de cada formulario.
Resultado esperado	Se muestran en la parte inferior de cada formulario, y son de diferentes tipos según en el formulario que nos encontremos. Aparecerá el botón para limpiar los datos introducidos en los campos, para volver a la página anterior, para avanzar al siguiente formulario o para guardar la información en la base de datos.
Resultado	CORRECTO.

Prueba	Comprobar que las páginas que muestran el resultado de la operación llevada a cabo son correctas.
Resultado esperado	Se muestra el resultado tras realizar la función seleccionada. Esto conlleva mostrar un resumen de la información guardada en la base de datos, o un aviso de que la función se ha o no realizado con éxito.
Resultado	CORRECTO.

Prueba	Seleccionar la opción Finalizar de las páginas que muestran el resultado de operación.
Resultado esperado	Se navega hacia la página que muestra el menú de opciones de la sección en cuestión.
Resultado	CORRECTO.

Prueba	Seleccionar la pestaña “Inicio” desde cualquier pantalla.
Resultado esperado	Se navega a la página de inicio de la aplicación, mostrando la portada con la descripción de la misma.
Resultado	CORRECTO.

5.1.4 Impresión

Una vez que nos encontramos en las páginas de resultado con datos que resumen la operación, podemos imprimir esa página si lo deseamos.

Requisito	Requisito-Funcional-28
Prueba	Seleccionar la opción de IMPRIMIR de las páginas que muestran el resultado de la operación.
Resultado esperado	Abre una ventana nueva para seleccionar la impresora y sus propiedades, facilitando la impresión del resultado de la operación.
Resultado	CORRECTO.

5.1.5 Proceso de Desconexión

Se trata de verificar que funciona correctamente la desconexión de la aplicación mediante el cierre de sesión, independientemente del tipo de agente que inició la sesión.

Prueba	Desde cualquier pantalla de la aplicación, pulsar la opción Salir.
Resultado esperado	La aplicación navega a la pantalla de login y le informa de que ha salido del sistema.
Resultado	CORRECTO.

5.2 Inserción y almacenamiento de datos

Se ha probado ampliamente la correcta inserción de los datos dentro de la base de datos externa, puesto que es una de las partes más importantes dentro de la aplicación. Para ello se han probado diferentes combinaciones, en los datos introducidos a través de los campos de los formularios, para comprobar que si los datos eran incorrectos o incoherentes, la aplicación muestra mensajes de error en los campos en cuestión, explicando cual es la manera correcta de inserción.

Por lo tanto para la batería de pruebas hemos utilizado, diferentes usuarios autenticados en diferentes máquinas, y en ocasiones, en diferentes navegadores para la inserción de datos a través de la aplicación. Las pruebas se han realizado sobre las funciones posibles de cada una de las secciones.

5.2.1 Inserción

Requisito	Requisito-Comprobación-3
Prueba	Verificar que funcionan correctamente las validaciones de todos los campos de los formularios a rellenar. • Introducir caracteres no permitidos en los campos de solo texto, como los que hacen referencia a nombre y apellidos (%, =, ! , etc).
Resultado esperado	La aplicación informa en cada caso con el mensaje de error correspondiente y resaltándolo en otro color.
Resultado	CORRECTO.

Prueba	Verificar el comportamiento de los campos obligatorios, como por ejemplo, el nombre y los apellidos de los padres para menores de edad, el titular de un vehículo, la fecha de un suceso, etc.
Resultado esperado	La aplicación informa en cada caso con el mensaje de aviso correspondiente y resaltándolo en otro color.
Resultado	CORRECTO.

Requisito	Requisito-Comprobación-2
Prueba	Verificar que funcionan correctamente las validaciones de todos los campos de los formularios a rellenar. • Introducir caracteres no permitidos o que incumplen el formato de campos específicos como por ejemplo, la matricula, el número de teléfono, el DNI, el CIF, mediciones del nivel de ruido, etc.
Resultado esperado	La aplicación informa en cada caso con el mensaje de error correspondiente y resaltándolo en otro color.
Resultado	CORRECTO.

Prueba	Verificar que funcionan correctamente las validaciones sobre campos para fechas. Comprobar que existen y que son lógicas dependiendo de la opción en la que nos encontremos.
Resultado esperado	La aplicación informa en cada caso con el mensaje de error correspondiente y resaltándolo en otro color (Por ejemplo, la fecha de caducidad no puede ser posterior a la de expedición, la fecha de una denuncia para almacenar, no puede ser posterior a la actual, etc).
Resultado	CORRECTO.

Prueba	Verificar que funcionan correctamente las validaciones sobre campos para horas. Comprobar que existen y que son lógicas dependiendo de la opción en la que nos encontremos.
Resultado esperado	La aplicación informa en cada caso con el mensaje de error correspondiente y resaltándolo en otro color (Por ejemplo, la hora de una denuncia para almacenar, no puede ser posterior a la actual, para referirnos a las doce de la noche utilizar “00:00” en vez de “24:00”, etc).
Resultado	CORRECTO.

5.2.2 Almacenamiento

Se comprueba que la información que se ha insertado a través de los formularios, y que se encuentra almacenada en la base de datos es correcta y coherente según lo insertado. Para ello se han comprobado todas las funcionalidades de la aplicación, indicados en una serie de requisitos funcionales.

Requisitos	Req-Funcionales-5 al 24 y Req-Funcionales- 29 al 34.
Prueba	Verificar que funciona correctamente la inserción de un nuevo registro, del tipo que sea, en la base de datos.
Resultado esperado	Se navega a la página del resultado, en la que se muestra un resumen de la operación realizada y si se ha realizo con éxito o no.
Resultado	CORRECTO.

Prueba	Verificar que los datos insertados, referentes a cualquier sección de la aplicación, producen otros valores que son almacenados también y que son coherentes con los introducidos.
Resultado esperado	Se navega a la página del resultado, en la que se muestra un resumen de la operación realizada y se comprueba que los datos mostrados son los correctos (Por ejemplo, según el tipo de la infracción cometida, los puntos que supone o el importe de la multa, etc).
Resultado	CORRECTO.

Prueba	Verificar como Administrador, que los datos insertados, se almacenan correctamente en las tablas, comprobando que si se almacenan en varias tablas las referencias entre los registros coincidan.
Resultado esperado	Se visualizan los datos de las tablas y se comprueba que las referencias coinciden (Por ejemplo, los datos de un accidente se almacenan con un nº de referencia, y los implicados o testigos de él, tienen que estar registrados con esa misma referencia, el titular de una citación judicial se almacena con la misma referencia que el delito por el que es denunciado, etc).
Resultado	CORRECTO.

Prueba	Verificar que funciona correctamente la búsqueda de información interna sobre un agente registrado.
Resultado esperado	Se muestra toda la información, que se encuentra almacenada en la base de datos externa sobre este agente. Si el agente no existe en la base de datos, se mostrará un mensaje de aviso.
Resultado	CORRECTO.

Prueba	Verificar que funciona correctamente la búsqueda de información acerca de un ciudadano.
Resultado esperado	Se muestra toda la información, distribuida en tablas, que se encuentra almacenada en la base de datos externa sobre esta persona. O un mensaje indicando que no se encuentra información, si es este el caso.
Resultado	CORRECTO.

5.3 Modificación

Se trata de comprobar que los datos que contiene la referencia seleccionada, se modifican correctamente y son los mostrados en el formulario completo a modificar.

Requisito	Requisito-Funcional-26
Prueba	Verificar que funciona correctamente la modificación de registros sobre un ciudadano, en la base de datos externa. Teniendo en cuenta que la modificación de información puede acarrear la modificación de registros de diferentes tablas.
Resultado esperado	Se navega a la página del resultado, en la que se indica que la operación se ha realizo con éxito.
Resultado	CORRECTO.

5.4 Eliminación

Se trata de comprobar que los datos que contiene la referencia seleccionada, se eliminan correctamente y son los mostrados en el formulario completo a eliminar.

Requisito	Requisito-Funcional-27
Prueba	Verificar que funciona correctamente la eliminación de registros sobre un ciudadano, en la base de datos externa. Teniendo en cuenta que la eliminación de información puede acarrear la eliminación de registros de diferentes tablas.
Resultado esperado	Se navega a la página del resultado, en la que se indica que la operación se ha realizo con éxito.
Resultado	CORRECTO.

Tanto en la inserción, modificación y eliminación de registros en la base de datos externa, se comprueba que esas operaciones se han realizado correctamente, visualizando directamente los nuevos datos en las tablas correspondientes, utilizando la interfaz gráfica "pgAdmin", que se ha mencionado en capítulos anteriores, y de la que se dispone siendo el Administrador.

Para comprobarlo como usuario de la aplicación, habría que realizar de nuevo una búsqueda a través del DNI de uno de los ciudadanos implicados, en la sección "Gestión del ciudadano", o en el caso de la información interna sobre un agente, realizando una

búsqueda mediante su número de placa, en las opciones de consulta de la sección Administración Interna.

5.5 Concurrencia

Se ha comprobado que tanto la inserción de información, como la modificación o eliminación de la misma sigue funcionando adecuadamente, aún cuando son varios usuarios simultáneos los que están autenticados en la aplicación, e intentan acceder a la vez a un registro.

Estos usuarios simultáneos se conectan a la aplicación desde máquinas con diferentes características y navegadores, comprobando que el tiempo de respuesta es minimamente superior a cuando el número de usuarios es inferior, y que en el caso de coincidir, en la modificación o eliminación del mismo registro, se muestran mensajes de espera o de aviso de esta situación.

Prueba	Verificar que funciona correctamente la inserción de un nuevo registro, del tipo que sea en la base de datos, coincidiendo simultáneamente con otro usuario.
Resultado esperado	Al pulsar Aceptar, o similar, aparecerá un mensaje indicándole que intente el registro de nuevo en unos segundos. Espere y vuelve a pulsar Aceptar, el registro se almacena de forma correcta.
Resultado	CORRECTO.

Prueba	Verificar que funciona correctamente la modificación de registros sobre un ciudadano, en la base de datos, coincidiendo simultáneamente con otro usuario.
Resultado esperado	Al pulsar Guardar, aparecerá un mensaje indicándole que existe otro usuario intentando modificar el mismo registro en ese momento.
Resultado	CORRECTO.

Prueba	Verificar que funciona correctamente la eliminación de registros sobre un ciudadano, en la base de datos, coincidiendo simultáneamente con otro usuario.
Resultado esperado	Al pulsar Eliminar, aparecerá un mensaje indicándole que existe otro usuario intentando eliminar el mismo registro en ese momento.
Resultado	CORRECTO.

5.6 Usabilidad

Se ha realizado una prueba en la que se demuestra la utilidad de la funcionalidad de este sistema de gestión policial. Esta prueba ha consistido en el acceso a la aplicación web, a través del protocolo HTTPS, que proporciona un acceso seguro para proteger los datos personales que sean manipulados por la aplicación, desde el navegador web Mozilla/Firefox, por parte de un usuario, y a través del navegador web Internet Explorer por otro.

Estos dos usuarios son dos agentes reales de la policía municipal, y han podido comprobar que la aplicación realiza una gestión de información básica de forma efectiva, y que se hace de manera sencilla y rápida, gracias a una interfaz clara e intuitiva. Esta interfaz cuenta con formularios con campos desplegables para una rápida selección de la opción adecuada, calendarios emergentes que permiten en un solo “clic”, seleccionar la fecha con día, mes y año, ayuda en cada uno de los campos a rellenar, para evitar dudas sobre los datos a introducir, etc. De esta forma, completar el formulario en cuestión, resulta mucho más rápido y eficaz.

Además esta interfaz resulta amigable desde el inicio, con secciones bien diferenciadas, que cuentan con un menú de opciones bien definido y con un resumen, sobre que aspectos se pueden gestionar desde dicha sección.

CAPÍTULO 6

HISTORIA DEL PROYECTO

6 HISTORIA DEL PROYECTO

En este capítulo se describen las diferentes fases en las que se ha desarrollado este proyecto de fin de carrera y la duración aproximada de cada una de ellas.

6.1 Fases del proyecto

1. Selección del entorno.

Inicialmente se realizó un estudio y comparación de los diferentes gestores de contenido de software libre que había en el mercado, así como de los diferentes gestores de bases de datos que podían interactuar con cada uno de ellos. Llegando a la elección de uno de ellos.

La duración fue de una semana.

2. Instalación de los entornos de desarrollo.

En esta segunda fase se realizó la instalación del entorno de desarrollo que se iba a utilizar, Plone 3.0 y PostgreSQL 8.3, herramientas de "software libre" y fáciles de manejar. Junto con la instalación del servidor Apache 2.2.11 para proporcionar un acceso seguro al sistema.

La elección de las versiones de las dos primeras tecnologías utilizadas para el entorno, se debe a que éstas son las últimas versiones, en el momento en que se inicia este proyecto. Además estas tecnologías son multiplataforma, por lo cual, aunque inicialmente la instalación se realizó sobre Windows de forma local, finalmente se decidió realizarla sobre Linux en un servidor del laboratorio del grupo de Aplicaciones y Servicios Telemáticos (GAST). De esta forma se facilita el acceso múltiple y desde cualquier lugar con acceso a Internet, acercándose así, al entorno real que tendría una aplicación de este tipo, si fuera comercializada. Y debido a que el acceso se hará a través de Internet y que los datos a manejar son sensibles, se ha de proporcionar una comunicación segura, y de ahí la necesidad de un frontal como Apache, que proporcione esa seguridad.

La duración fue de unas tres semanas.

3. Familiarización con el gestor de contenido Plone.

Teniendo en cuenta que teníamos unos conocimientos básicos como usuario, no como desarrollador, de la versión 2.5 de Plone; la siguiente tarea que se tuvo que realizar, fue aprender las nuevas características que presentaba la versión 3.0 de cara al usuario, así como el aprender a manejar este gestor para desarrollar tu propio sitio Plone. Esta tarea se realizó consultando varios tutoriales y foros especializados.

La duración fue de cuatro semanas.

4. Familiarización con los lenguajes Python y TALES.

De igual forma que en el punto anterior, se tuvo que aprender estos dos lenguajes, utilizando, tutoriales y libros. Para el lenguaje TALES, bastó con conocer las expresiones más comunes, así como saber utilizarlas en casos de características diferentes. En el caso de Python, se necesitó un conocimiento más amplio, puesto que la parte de software de la aplicación en su mayoría está realizada con este lenguaje. Este aprendizaje se hizo de forma continuada durante el desarrollo de la aplicación, a medida que se presentaban nuevos objetivos en la funcionalidad del sistema, y fue menos costoso gracias al conocimiento previo de Java, lenguaje con el que Python guarda ligeras similitudes.

La duración fue de dos semanas.

5. Familiarización con el gestor de bases de datos PostgreSQL.

El gestor de bases de datos PostgreSQL era completamente desconocido, por lo que se tuvo que realizar un aprendizaje completo de sus características, limitaciones, forma de uso, etc. Gracias a la herramienta de gestión gráfica de la que dispone, “pgAdmin”, y al conocimiento de SQL del que se disponía, el desarrollo de la base de datos fue mucho más intuitivo y llevadero.

Además se aprendió como hacer la interacción entre la aplicación y la base de datos, para lo que se necesitaba un adaptador, “Psycopg1”, cuya instalación fue uno de los problemas más complicados que se tuvieron que resolver.

Todo este conocimiento se adquirió gracias a tutoriales, foros especializados y libros.

La duración fue de tres semanas.

6. Documentación con agentes de policía local.

En esta fase se realizó una labor de documentación, búsqueda y aprendizaje de la forma de proceder y actuar, en lo que se refiere al papeleo y administración, de la policía local. Para ello se consultó a dos agentes reales y se obtuvieron muestras reales de denuncias, actas, formularios, recopilatorios de sanciones, etc.

Para esta fase se desarrollo una consulta inicial para definir los aspectos en los que se centraría la aplicación, y posteriormente en paralelo con la implementación de la aplicación se continúo con el asesoramiento por parte de los agentes.

La duración fue de unas cuatro semanas.

7. Diseño de la aplicación y la base de datos.

Una vez familiarizado con el entorno y el lenguaje en que se desarrollaría la aplicación, se procedió al diseño de la misma.

Aquí se engloban, la definición de los requisitos y las funcionalidades de la aplicación, la estructuración en secciones, el reparto de permisos para controlar el acceso a las secciones, la determinación de las tablas que se necesitarían y como debían relacionarse entre sí, etc. De igual forma se evaluó las distintas formas de implementación, teniendo en cuenta las posibilidades que ofrece Plone y las diferencias que había con la versión anterior. Para esta decisión se consultaron tutoriales y foros especializados.

La duración fue de unas dos semanas.

8. Implementación de la aplicación Web SMALLPOL.

En esta fase se realizó la implementación de la aplicación Web usando el gestor de contenido. Se crearon las páginas en HTML/XHTML y TALES, las plantillas para los formularios, los scripts de validación, la conexión con la base de datos, las consultas en SQL necesarias, y se realizó la integración entre todas ellas para que la funcionalidad requerida se cumpliera.

La duración fue de unas siete semanas.

9. Implementación de la base de datos.

Se procedió a la declaración de las tablas necesarias y sus relaciones, así como a la inserción de datos auxiliares dentro de algunas de ellas.

La fase anterior y ésta, se desarrollaron de forma conjunta de forma que se iba avanzando en el cumplimiento de funcionalidades.
La duración fue de unas tres semanas.

10. Personalización de la interfaz gráfica de la aplicación Web.

Una vez desarrollada la aplicación Web completa, se pasó a modificar las hojas de estilo (CSS) y propiedades visuales que trae por defecto un sitio Plone, para crear una interfaz seria, atractiva, representativa e intuitiva para el usuario.
La duración fue de una semana.

11. Configuración del acceso seguro a la aplicación.

Una vez desarrollada la aplicación completamente se configuró en el servidor, un acceso seguro a través del protocolo SSL, de esta forma los usuarios se conectarían a través de HTTPS para garantizar la seguridad de los datos personales de los ciudadanos.
La duración fue de una semana.

12. Pruebas.

Esta fase en realidad se fue desarrollando a medida que se iba implementando cada funcionalidad. Al finalizar toda la implementación, se realizó una batería de pruebas por parte de un agente real, para corregir los errores encontrados, teniendo en cuenta su opinión profesional.
La duración fue de unas cinco semanas.

13. Documentación.

Esta última fase se dedicó a la documentación de esta memoria.
La duración fue de unas cinco semanas.

En la figura 45 se muestra el diagrama con la duración del proyecto por fases.

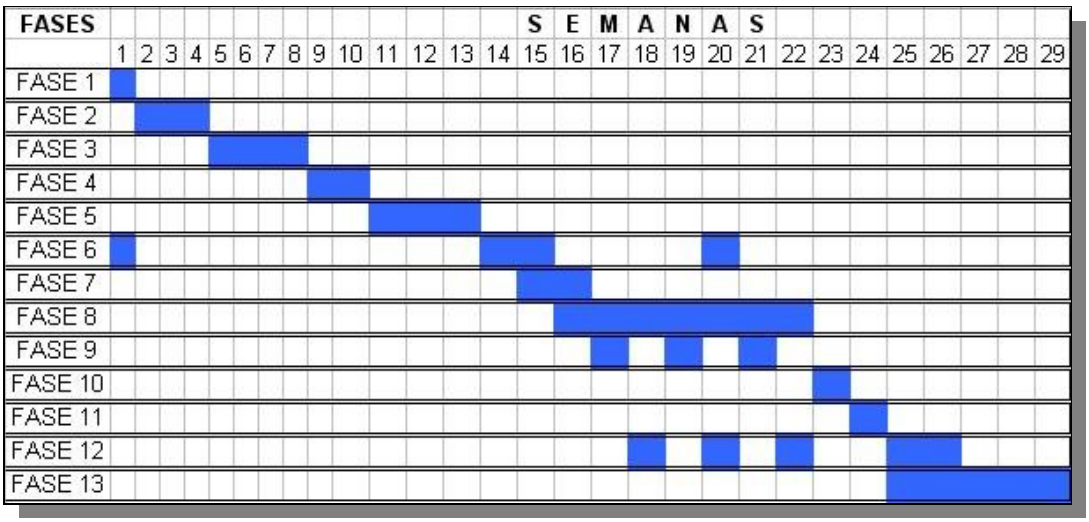

Figura 45: Duración del proyecto.

6.2 Problemas encontrados

Las dificultades que hemos encontrado para el desarrollo de este proyecto, se pueden dividir en dos grupos. Por un lado, las dificultades encontradas a la hora de la instalación y puesta en marcha, del gestor de base de datos PostgreSQL y su correspondiente adaptador para Plone, Psycopg, sobre una máquina Debian/Linux. Y por otro, conseguir el acceso seguro para nuestro sitio Plone a través de HTTPS.

Instalación de componentes del sistema:

- La dificultad para la instalación de PostgreSQL y el adaptador Psycopg, radicaba en que las instrucciones de instalación estaban dadas para un administrador de la máquina en la que se hiciera, pero este no era nuestro caso, ya que se trataba de un servidor del GAST (Grupo de aplicaciones y servicios telemáticos), y para la utilización de rutas por defecto. Por lo cual, a la hora de la instalación de cada uno de ellos, fue necesario utilizar algunas opciones del comando “configure”, como fueron la opción “--prefix =” para indicarle la nueva ruta que apuntaba al directorio, dentro de nuestra cuenta en el servidor, en el que queríamos que se situaran los ficheros de instalación. Y además la opción “--with-pgport =” para indicarle que el puerto en el que corriese el servidor no

fuese el de por defecto, sino el que nosotros eligiésemos, para nuestro entorno personalizado.

Configuración de acceso seguro:

- Las dificultades para conseguir el acceso a través de Plone radicaron, en conseguir establecer una serie de reglas sobre Apache, que facilitaran la correcta redirección a nuestro sitio Plone utilizando SSL sobre HTTP. La comprensión y la correcta escritura de las reglas, para que las páginas fueran cargadas de forma completa y correcta cuando se accedía a ellas a través de HTTPS.

  Por lo tanto se estableció correctamente cual era la ruta adecuada para la raíz de nuestro sitio Plone con la directiva “VirtualHostRoot”, definida en las reglas de reescritura de Apache.

CAPÍTULO 7

CONCLUSIONES Y LINEAS FUTURAS

7 CONCLUSIONES Y LINEAS DE TRABAJO FUTURAS

Una vez finalizado el proyecto, puedo concluir que se han cumplido todos los objetivos que se marcaron meses atrás, habiendo implementado una aplicación Web que facilita la gestión y administración de la documentación generada durante la jornada laboral, así como, el control de información interna, del cuerpo de policía de un Municipio. Y todo ello teniendo como objetivo, la reducción de costes en desarrollo y mantenimiento, puesto que en tiempos de crisis, las administraciones públicas son uno de los organismos que más sufren. A lo largo del proyecto han ido apareciendo dificultades que se han solventado satisfactoriamente.

Teniendo en cuenta esta reducción de costes, ha sido necesario consultar las diferentes tecnologías que se están utilizando actualmente en el desarrollo de aplicaciones Web, distribuidas bajo una licencia pública, como Drupal, Joomla, Plone, PostgreSQL, MySQL, etc. He aprendido el funcionamiento de un gestor de contenido como Plone y las capacidades que éste puede tener, he adquirido conocimientos de programación en Python y de la utilización del lenguaje de TALES en conjunción con HTML/XHTML y he aprendido a utilizar un gestor de bases de datos como PostgreSQL.
Además he reforzado mis conocimientos sobre HTML, hojas de estilo, JavaScript y SQL.

La implementación final de la aplicación cumple fielmente con la arquitectura cliente/servidor, de este modo todos los accesos desde las páginas hasta la base de datos externa, se hacen a través de llamadas al servicio web.

Uno de los objetivos iniciales del proyecto era el aseguramiento de la calidad y utilidad del software desarrollado. Para ello, el sistema ha sido sometido a un plan de pruebas definido para este fin. Los resultados han sido satisfactorios y por ello se puede concluir que el software cumple con las funcionalidades básicas para las que fue diseñado.

Con este trabajo he intentado poner en práctica los conocimientos adquiridos durante la carrera y mi corta experiencia laboral. Tareas relacionadas con diferentes temas como por ejemplo, la gestión de proyectos, desarrollo de documentación, análisis de

requisitos, diseño de la aplicación, implementación del código fuente, especificación del plan de pruebas, etc.

En conclusión, pienso que las aplicaciones Web son y seguirán siendo en los próximos años, una de las mejores y más baratas, si se tienen en cuenta tecnologías como las que se ha usado en este proyecto, alternativas a la hora de implementar cierto tipo de sistemas, pensados para llegar a grandes masas de usuarios y ser accesibles desde una amplia variedad de soportes, plataformas, dispositivos, etc sin importar el lugar en el que te encuentres. Por este motivo creo que hay que seguir formándose y actualizándose en tecnologías de "software libre" involucradas con el mundo Web.

7.1 Líneas de trabajo futuras

El sistema desarrollado tiene infinidad de posibilidades o líneas de trabajo futuras en las que se podría seguir trabajando. Básicamente se pueden englobar en las siguientes:

7.1.1 Desarrollo de nuevas secciones o ampliación de funciones, dentro de las ya existentes, para la versión para PC

La arquitectura del sistema está preparada para soportar un crecimiento en las funcionalidades de la aplicación, sin que ello afecte significativamente a su rendimiento, asegurando así la escalabilidad del sistema. El número de posibles secciones que se pueden ir agregando al sistema, la profundización o el aumento en las funcionalidades de las ya existentes, es indefinido, dependiendo únicamente de las necesidades concretas del cuerpo de policía del Municipio en cuestión y de las características de éste.

Por ejemplo, se le podría añadir una sección orientada al ciudadano, en vez de al agente exclusivamente:

- Nueva sección para el ciudadano.

  Nuevo enfoque, donde el acceso no estuviera restringido y donde el ciudadano previamente identificado, pudiera consultar el número de puntos que posee, solicitar un vado permanente para su garaje, etc.

O se podría ampliar las funcionalidades de una sección ya existe:

- Ampliación funciones de la sección de Administración Interna.

  Una profundización en los aspectos controlados en esta sección, como por ejemplo la posibilidad de realizar partes de horas, control del patrulla y la emisora asignada al agente, control de las citaciones a las que debe presentarse el agente, etc.

7.1.2 Adecuación para la versión en PDA

Por otro lado, como hemos comentado anteriormente, una de las posibles líneas de trabajo puede ser la migración progresiva de toda la funcionalidad que ofrece esta aplicación Web para PC, a una PDA, teniendo en cuenta las limitaciones que presenta un dispositivo móvil. Solo se necesitaría una optimización de la tecnología ya utilizada para la versión para PC.

7.1.3 Gestión de logs

Mejorar la seguridad en el manejo de la aplicación, realizando un sistema de logs que nos permita controlar los eventos que se produzcan en la aplicación, especialmente orientados a conocer quien ha sido, el agente que ha realizado los registros, modificaciones o eliminaciones de los datos de un ciudadano, en la base de datos externa.

APÉNDICES

APÉNDICE A: Presupuesto

El presupuesto para la realización de este proyecto está formado por los honorarios del Ingeniero Técnico de Telecomunicaciones que realiza la aplicación, y el coste de los materiales y el software que éste necesita para desarrollarla.

En cuanto al ritmo de trabajo de los 7 meses y una semana que han transcurrido, no ha sido constante, pero se puede hacer un cálculo aproximado para computarlo con vistas al presupuesto total de proyecto. De media se ha trabajo unas 6 horas al día, descontando festivos, vacaciones y fines de semana. Teniendo en cuenta el baremo que el Colegio Oficial de Ingenieros Técnicos de Telecomunicaciones [19], propone para la realización de un proyecto de estas características por un Ingeniero Técnico de Telecomunicaciones, los honorarios son de 65 euros por hora.
Por lo tanto los costes pueden verse en la tala 4:

PRESONAL:

Concepto	Nº de horas	Honorarios	Importe
Ingeniero Técnico de Telecomunicaciones	870	65 euros/hora	**56.550**

Tabla 4: Costes de personal.

Con respecto al software utilizado, el proyecto se ha desarrollado en su totalidad con herramientas de "software libre" (Zope, Plone, Python, PostgreSQL…), sin coste adicional alguno. Y en cuanto al material empleado, se ha utilizado un ordenador de sobremesa con licencia del sistema operativo Microsoft Windows, y otro PC (linux) en el que se han instalado las herramientas para actuar como servidor Web y SQL.
A esto hay que añadirle un coste de material informático del tipo, ADSL, papel, electricidad, un dominio, Hosting, etc.
Este coste de material puede observarse en la tabla 5:

MATERIAL:

Concepto	Cantidad	Precio	Importe Total (euros)
PC de sobremesa	1	800	800
PC Servidor/Hosting (anual)	1	90	90
Dominio (anual)	1	25	25
			915

Tabla 5: Costes de material.

Por lo tanto el presupuesto estimado para la realización de este proyecto quedará como se muestra en la tabla 6:

Concepto	Importe total (euros)
Costes del personal	56.550
Costes del material	915
	57.465

Tabla 6: Costes totales.

APÉNDICE B: Manual de instalación

El sistema desarrollado en este proyecto consta de tres elementos principales:

- Servidor Web Zope y Aplicación Web (Gestor de Contenido)
- Servidor Apache.
- Base de datos externa.

Todos estos elementos han sido instalados sobre una máquina Debian/Linux. A continuación, se va a detallar en este apartado, el proceso de instalación de cada uno de ellos.

- **SERVIDOR Y APLICACIÓN WEB:**

El servidor web utilizado es Zope 2.10.4, y la aplicación web, se desarrolla utilizando el gestor de contenido Plone 3.0, ambos construidos con python 2.4.4.
La instalación de estos elementos se ha hecho de forma conjunta, gracias al paquete Plone-3.0-UnifiedInstaller-r2(2).tar. Este paquete para Linux/BSD/OS X/UNIX/Solaris, permite compilar e instalar, Python, Zope, Plone y una serie de dependencias de una sola vez.

En este caso se tiene una instalación no siendo *root* y con una única instancia de Zope.
Además el Python que se instala con este paquete, no interfiere con ningún otro Python que haya sido instalado en el servidor del grupo GAST que utilizamos como soporte.
Por lo tanto se descomprime el paquete anterior con el comando:

```
"fatima@chopin:~$ tar xzvf  Plone-3.0-UnifiedInstaller-r2 (2).tar"
```

Ahora tendremos el paquete descomprimido en el subdirectorio Plone-3.0 y situados dentro de él, se inicia la instalación con el comando:

```
"fatima@chopin:~/Plone-3.0/ ./install.sh standalone"
```

Una vez realizada la instalación sin problemas, debemos iniciar Plone:

```
"fatima@chopin:~/Plone-3.0/zinstance/bin$  ./zopectl start"
```

Utilizando un navegador cualquiera, nos metemos en la interfaz de gestión de nuestra instancia Zope para poder crear nuestra aplicación, creando un nuevo sitio Plone desde ahí. Se introduce en el navegador:

```
http://nombredelservidor:8080/manage
```

Se accede introduciendo el nombre de usuario y la contraseña que se han generado por defecto en la instalación, y que se encuentran en el fichero adminPassword.txt de nuestra instancia.

Si no se quiere utilizar este usuario y contraseña, se puede elegir el que se quiera, introduciéndose a través del navegador en la sección de gestión de usuarios, de la siguiente manera:

```
http://localhost:8080/acl_users/users/manage_users
```

Una vez dentro de la interfaz de gestión de Zope como administrador, se selecciona en el desplegable de la derecha, la opción “Plone Site”, creando así un nuevo sitio Plone que se utiliza para desarrollar la aplicación SMALLPOL. La configuración y desarrollo por parte del Administrador que se realiza en el nuevo sitio Plone, se describe en el apartado siguiente. La interfaz de gestión de Zope puede verse en la figura 46:

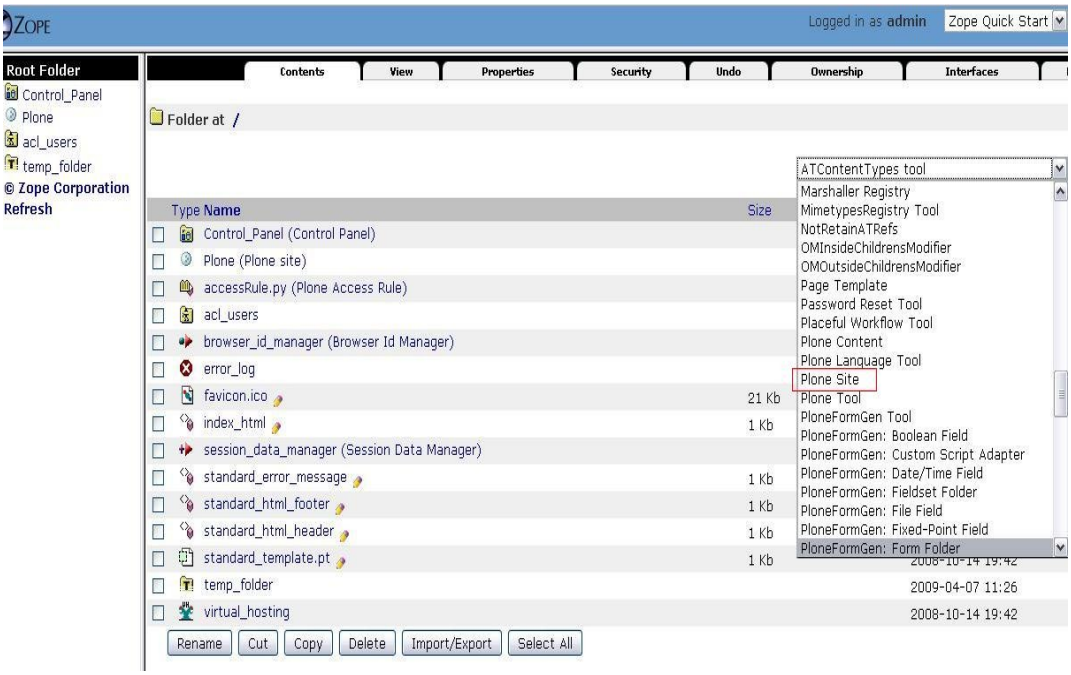

Figura 46: Interfaz de Gestión de Zope.

- **SERVIDOR APACHE:**

Se ha utilizado el servidor Apache 2.2.11. Para su instalación se ha usado el archivo httpd-2.2.11.tar.gz, el cual se ha descomprimido de la siguiente manera:

```
fatima@debussy:~$ tar xzvf httpd-2.2.11.tar.gz
```

Al descomprimirlo se crea el subdirectorio httpd-2.2.11 en el que nos meteremos para realizar la instalación y configuración del servidor posteriormente.

Pero antes debemos de instalar OpenSSL, que consiste en un robusto paquete de herramientas de administración y librerías relacionadas con la criptografía, que suministran funciones criptográficas a navegadores web para acceso seguro a sitios HTTPS. De esta forma podremos implementar el Secure Sockets Layer (SSL) en nuestro servidor Apache y crear los certificados de seguridad digitales que necesitamos. Para la instalación de este paquete se descomprime el fichero openssl-0.9.8k.tar.gz como se muestra a continuación y se obtiene el subdirectorio openssl-0.9.8k.

```
fatima@debussy:~$ tar xzvf openssl-0.9.8k.tar.gz
```

A continuación nos situamos dentro del subdirectorio que se ha creado y procedemos a la instalación, utilizando las opciones –prefix y –openssldir, para indicar el directorio donde queremos que se realice la instalación completa son sus librerías y ficheros binarios, en vez de en el directorio por defecto.

```
fatima@debussy:~/openssl-0.9.8k/ ./config --prefix=/home/apt/fatima
                            -- openssldir=/home/apt/fatima/openssl

fatima@debussy:~/openssl-0.9.8k/ make
fatima@debussy:~/openssl-0.9.8k/ make test
fatima@debussy:~/openssl-0.9.8k/ make install
```

Seguidamente establecemos el valor adecuado de la variable de entorno PKG_CONFIG_PATH, de la siguiente manera:

```
export PKG_CONFIG_PATH=/home/apt/fatima/lib/pkgconfig/openssl.pc
```

Y ahora si, se procede a la instalación del servidor propiamente dicho. Nos situamos dentro del fichero httpd-2.2.11, y se utiliza la opción “-- prefix” para cambiar la ruta de instalación por defecto, se indican también el puerto donde se quiere que corra el

servidor y el que se utiliza para las conexiones seguras, al tratarse de puertos diferentes a los de por defecto, ya que estos son usados por el servidor Apache de la Universidad, y se habilitan unas serie de modos que son necesarios para poder escribir las reglas de direccionamiento que se necesitarán.

```
fatima@debussy:~/httpd-2.2.11/ ./configure --
prefix=/home/apt/fatima/apache-2 --with-port=8880 --with-sslport=8443
--enable-rewrite --enable-speling --enable-ssl --enable-proxy --
enable-mods-shared=all --with-ssl=/home/apt/fatima/openssl-0.9.8k

fatima@debussy:~/httpd-2.2.11/ make
fatima@debussy:~/httpd-2.2.11/ make install
```

Las reglas se escribirán tanto en el fichero de configuración de Apache (httpd.conf), como en el que se define el "virtual host" para HTTPS (httpd_ssl.conf). Además en estos ficheros se deberán adecuar las diferentes directivas o includes y añadir los certificados de seguridad generados, para que se correspondan con el entorno de este proyecto.

Seguidamente nos situamos en la raíz de nuestra instancia de Zope, y añadimos el objeto "VirtualHostMonster", como se ve en la figura 47, dándole simplemente el nombre que queramos, siempre que no este repetido por ningún objeto de la aplicación. Gracias a este objeto nuestro sitio Plone entenderá el cambio de URLs que se produce al añadir las reglas anteriores.

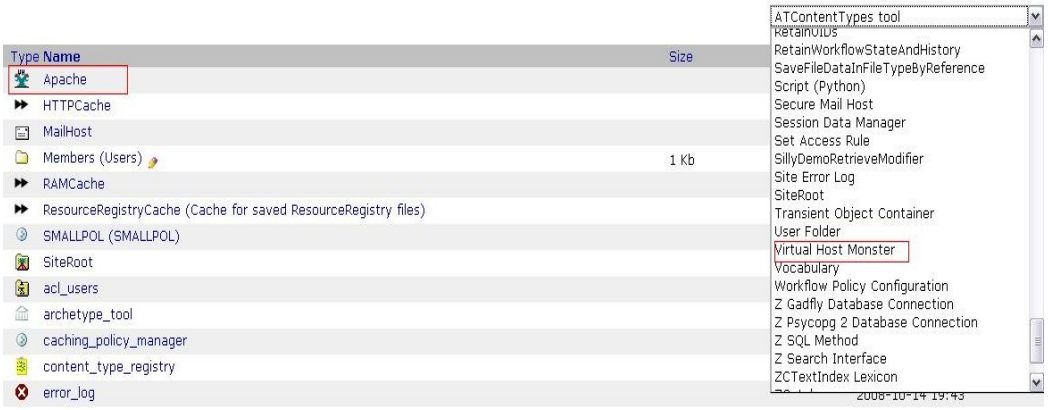

Figura 47: Añadir el objeto VirtualHostMonster a la raíz de Zope.

Finalmente se arranca el servidor, de la siguiente forma:

```
fatima@debussy:~/apache-2/bin/./apachectl start
```

- **BASE DE DATOS EXTERNA (PostgreSQL):**

Se ha utilizado el sistema de gestión de bases de datos relacional PostgreSQL 8.3.5. Para su instalación se ha utilizado el paquete postresql-8.3.5.tar.gz. Con lo cual, descomprimimos el archivo con:

```
"fatima@chopin:~$ tar xzvf  postresql-8.3.5.tar.gz"
```

Al descomprimirlo se crea el subdirectorio postresql-8.3.5 con todos los ficheros fuente. Además se ha de crear un fichero para la compilación y configuración de lo contenido en el subdirectorio anterior, este directorio se llama "bd2". La ruta a este directorio se hace utilizando la opción de "-- prefix" cambiando así la ruta de instalación por defecto. A continuación se realizan una serie de pasos para la instalación y configuración del gestor. Una vez situado dentro del subdirectorio postresql-8.3.5, se hace lo siguiente:

```
./configure --prefix=/home/apt/fatima/bd2 --without-readline --with-
pgport=5600
make
make install
mkdir /home/apt/fatima/bd2 data2
```

Una vez instalado el gestor se ha de crear la base de datos. Además el puerto en el que PostgreSQL estará disponible por defecto hay que cambiarlo, ya que en el servidor del GAST, existe otra instalación de PostgreSQL, que escucha en este puerto.

```
/home/apt/fatima/bd2/bin~$ ./initdb -D /home/apt/fatima/bd2/data2 –
with-port= 5600
/home/apt/fatima/bd2/bin~$ ./postgres –p 5600 -D
/home/apt/fatima/bd2/data2 >logfile 2>&1 &
```

Ya está arrancado el proceso servidor SQL en el puerto que se quiere, por lo tanto solo resta crear la base de datos:

```
/home/apt/fatima/bd2/bin~$ createdb –p 5600 nombreBaseDeDatos
```

Y una vez creada, se accede a ella a través de un terminal interactivo, para proceder a la declaración de tablas, inserción de datos, ejecutar comandos SQL, etc.

```
/home/apt/fatima/bd2/bin~$ psql –U fatima –p 5600 –h /tmp
nombreBaseDeDatos
```

Este terminal interactivo puede verse en la figura 48:

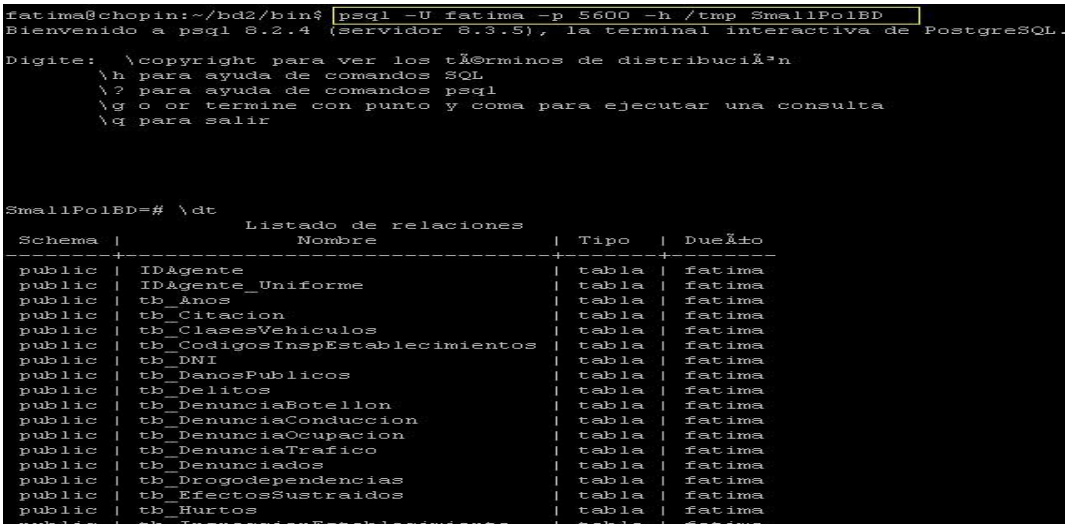

Figura 48: Terminal para la gestión de la base de datos externa.

También se puede gestionar la base de datos de forma más intuitiva a través de una herramienta gráfica como es “pgAdmin”, que viene dentro del paquete postresql-8.3.5.tar.gz. Como se usa esta herramienta, se describe en el apartado siguiente de este capítulo. La interfaz gráfica de “pgAdmin” se muestra en la figura 49:

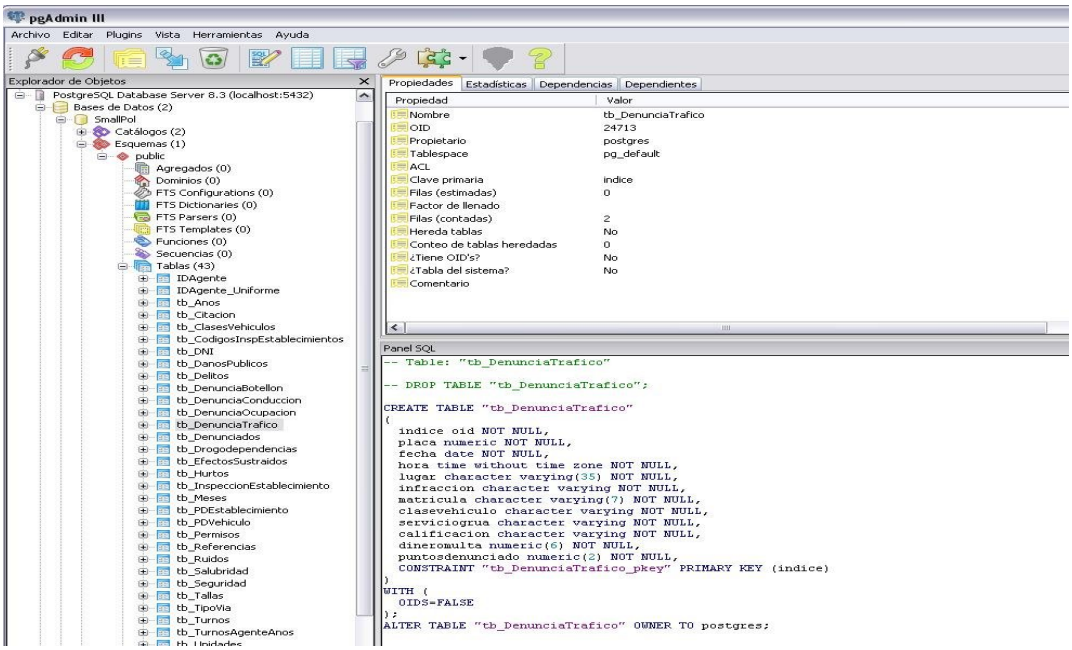

Figura 49: Interfaz gráfica de pgAdmin.

Por último falta por describir la instalación del adaptador necesario, que permite conectar la base de datos externa con el gestor de contenido a través de python. Este

adaptador esta contenido en el fichero psycopg-1.1.21.tar.gz y por tratarse de un producto de Zope, debe de guardarse en la carpeta de Productos de nuestra instancia. Con lo cual descomprimimos el fichero anterior con:

```
"fatima@chopin:~$ tar xzvf  psycopg-1.1.21.tar.gz"
```

Tras descomprimir, se genera un directorio con lo descomprimido llamado Psycopg-1.1.21. Una vez dentro de este directorio, se pasa a la configuración a través del comando "configure" y una serie de prefijos para establecer las rutas que se adecuan a nuestro entorno de instalación.

Para nuestro caso concreto son:

```
fatima@chopin:~/psycopg-1.1.21$    ./configure
--with-mxdatetime-includes=/home/apt/fatima/Plone-
3.0/lib/python/egenix-mx-base-3.1.1/mx/DateTime/mxDateTime
--with-postgres-includes=/home/apt/fatima/bd2/include
--with-postgres-libraries=/home/apt/fatima/bd2/lib
--with-python=/home/apt/fatima/Plone-3.0/Python-2.4.4/bin/python
--with-zope=/home/apt/fatima/Plone-3.0/zinstance
```

Posteriormente se procede a la instalación:

```
fatima@chopin:~/psycopg-1.1.21$ make
fatima@chopin:~/psycopg-1.1.21$ make install
fatima@chopin:~/psycopg-1.1.21$ make install-zope
```

Para finalizar ha de trasladarse la carpeta *ZpsycopgDA* que se ha generado, a la carpeta de productos de nuestra instancia de Zope y el fichero *psycopgmodule.so* a la carpeta lib-dynload de nuestra instalación de Python. Una vez realizada la instalación completa, se comprueba que desde la interfaz de gestión del sitio Plone SMALLPOL, se puede seleccionar en el desplegable de la derecha el adaptador "Z Psycopg DB Connection", como se muestra en la figura 50.

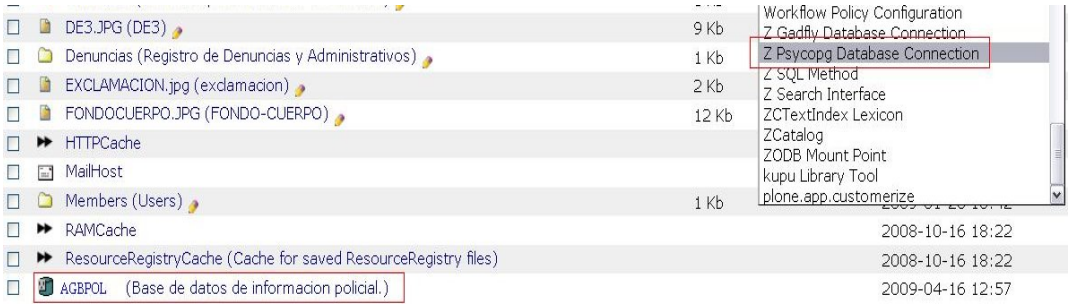

Figura 50: ZMI desde donde se selecciona el adaptador.

APÉNDICE C: Manual de desarrollo del administrador

En este apartado se explica cómo utilizar diferentes funcionalidades tanto en el gestor de contenido, como en la base de datos externa, para la creación, desarrollo y mantenimiento de la aplicación SMALLPOL, desde el punto de vista del Administrador.

Se empieza por la descripción de la base de datos externa.

- **BASE DE DATOS:**

Para la gestión de la base de datos se utiliza la herramienta gráfica "pgAdmin" que se ha nombrado en el apartado anterior.

Se supone que ya se encuentra arrancado el proceso servidor, así que ahora se debe de crear una base de datos, como se muestra en la figura 51.

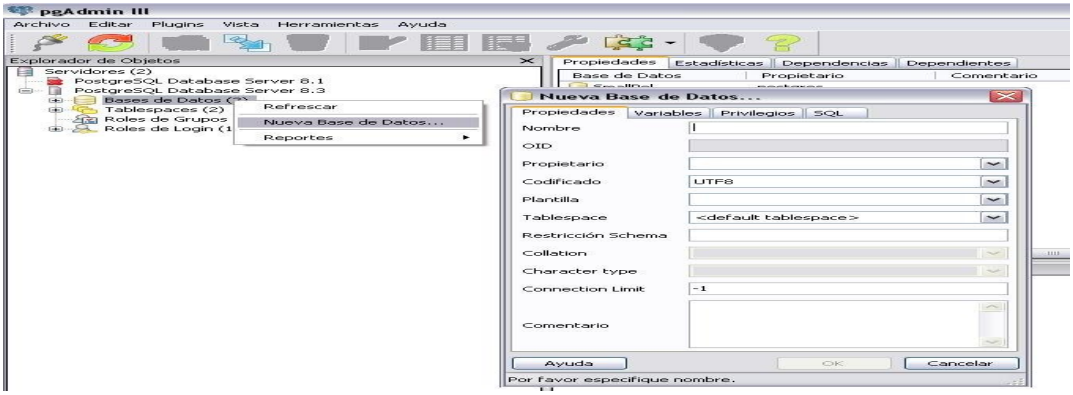

Figura 51: Creación base de datos.

Una vez creada la base de datos, se le añaden las tablas, definiendo sus nombres, columnas y tipos de cada una de ellas, definición de claves, etc. Esta opción puede verse en la figura 52.

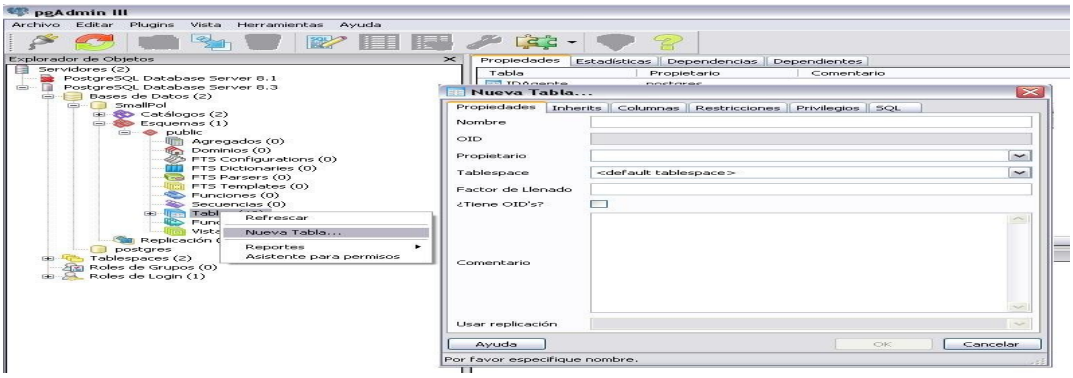

Figura 52: Creación de una tabla en la base de datos.

Con esta herramienta también se pueden hacer consultas de cualquier tipo para pruebas del Administrador, como por ejemplo la mostrada en la figura 53:

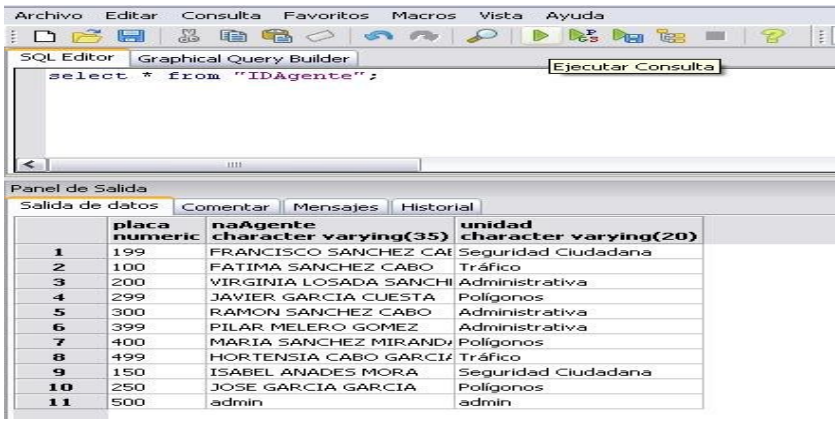

Figura 53: Realizar una consulta de prueba.

En cuanto a las copias de seguridad de la base de datos esta herramienta también es útil, aunque en ella se denomina un "resguardo". La manera de crearla se muestra en la figura 54.

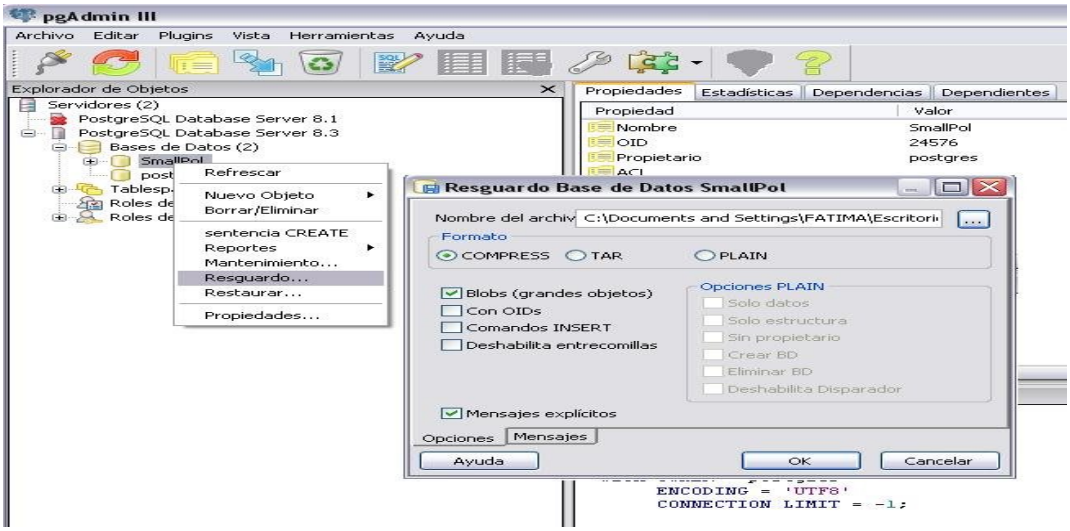

Figura 54: Realizar un resguardo de la base de datos (Backup).

- **<u>GESTOR DE CONTENIDO</u>:**

Una vez creado tu sitio Plone como se ha explicado en la figura 46 del apéndice B de esta memoria, se entra en la página principal del sitio, en la cual se informa de los pasos a seguir para la configuración del sitio. Esta página principal es como puede verse en la figura 55.

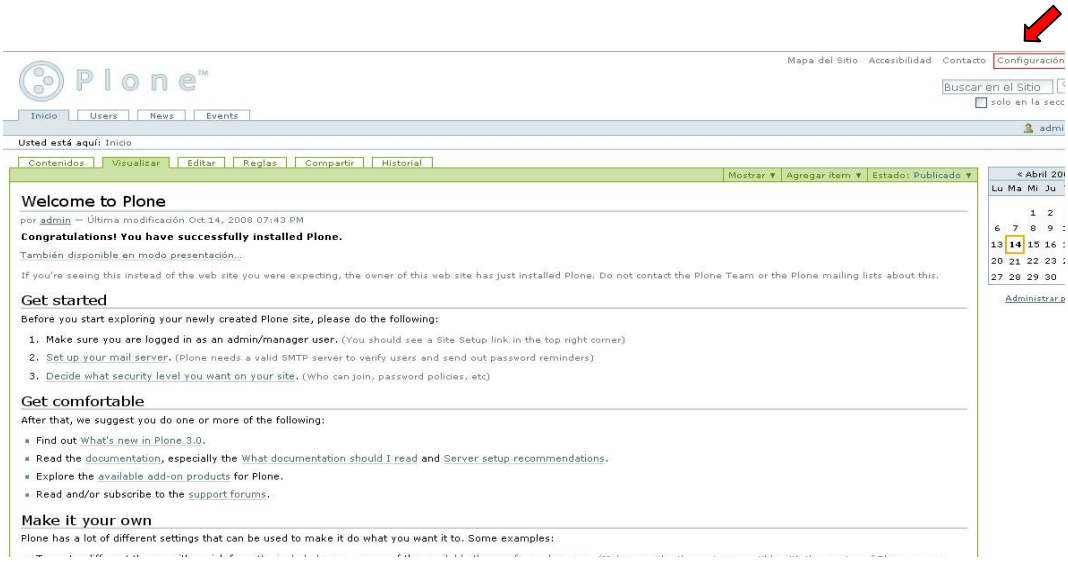

Figura 55: Página principal del sitio Plone creado.

Se selecciona la opción de configuración y se accede a la interfaz de gestión, en ella se seleccionan los diferentes aspecto que se quieran modificar, como por ejemplo, reglas de contenido, el idioma, el tipo de flujo de trabajo para la publicación de contenido, la inserción de nuevos usuarios o grupos, etc. Algunas de estas opciones se muestran en la figura 56.

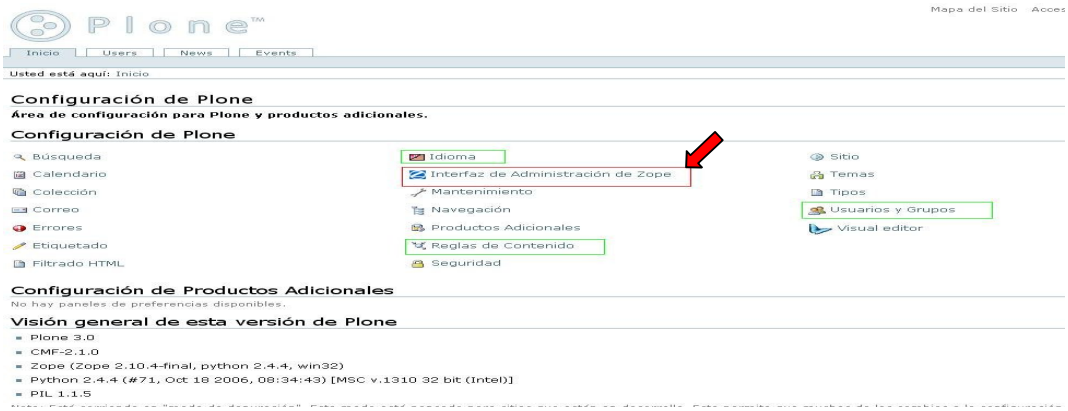

Figura 56: Página de Configuración.

Seguidamente se selecciona la Interfaz de Administración de Zope (ZMI), desde aquí se crean las capetas en las que se almacenan todas las páginas, formularios, imágenes, etc, que conforman cada una de las secciones en que se ha dividido la aplicación. También desde aquí se establece la conexión con la base de datos y se crea la carpeta con las consultas SQL. Esta interfaz constituye el entorno principal para el desarrollo de la aplicación y se muestra en la figura 57.

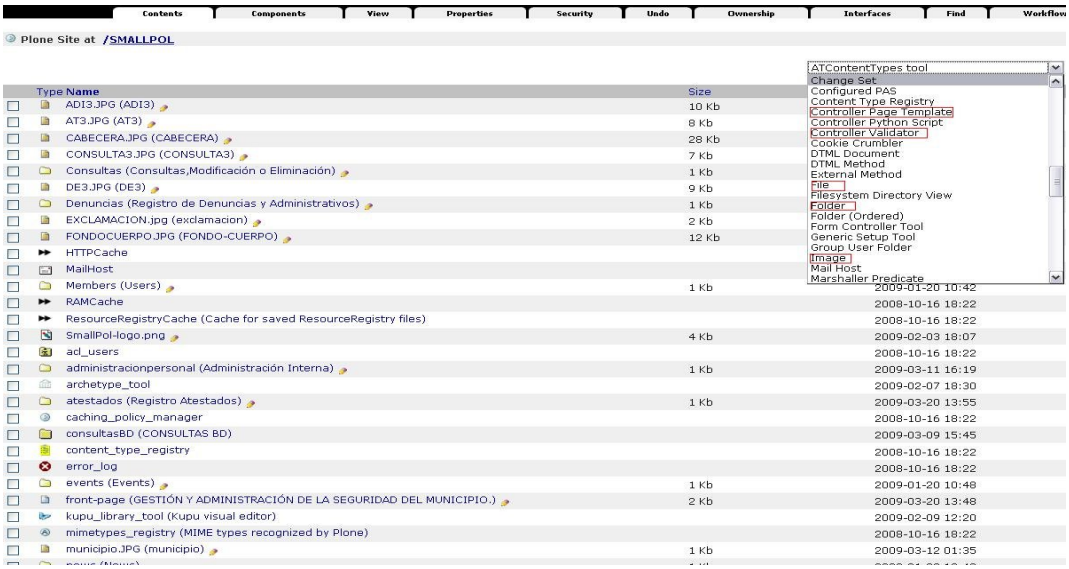

Figura 57: Página de Configuración.

Como se observa en la figura 57, en el desplegable de la derecha se seleccionan los elementos que se quieran añadir al nuevo sitio Plone.

A continuación se describen los pasos básicos que realiza el Administrador en esta aplicación, para la creación de una página estática con un menú, que enlaza con un formulario al que se le asignan unos validadores y que finaliza con otra página dependiendo del resultado obtenido.

Se abren dos pestañas en un navegador Web, en una se selecciona la ZMI de nuestro sitio Plone, llamémosle "***Pestaña1***" (Figura 56) y en la otra, nos introduciremos en el sitio Plone, identificados como Administrador, llamémosle "***Pestaña2***", como muestra la figura 58.

Figura 58: Pestaña2, interfaz de la aplicación autenticado como Administrador.

Los pasos básicos son:

1. Situados en la *Pestaña2* (Figura 58), se seleccionan los contenidos del sitio, aquí se selecciona la carpeta que representa una sección, nos situamos dentro de ella y se le añade un nuevo ítem, que se trata de una página (AT Document), en la que se escribe el menú que se desee con los enlaces a los formularios correspondientes. La selección se muestra en la figura 59.

Figura 59: Añadir una página estática.

2. Situados en la *Pestaña1*, se selecciona la sección que se quiere configurar y se crea un formulario, seleccionando del desplegable de la derecha, la opción "Controller Page Template", se le asigna un nombre y se rellena con el código XHTML y TALES que sea necesario. La selección se muestra en la figura 60.

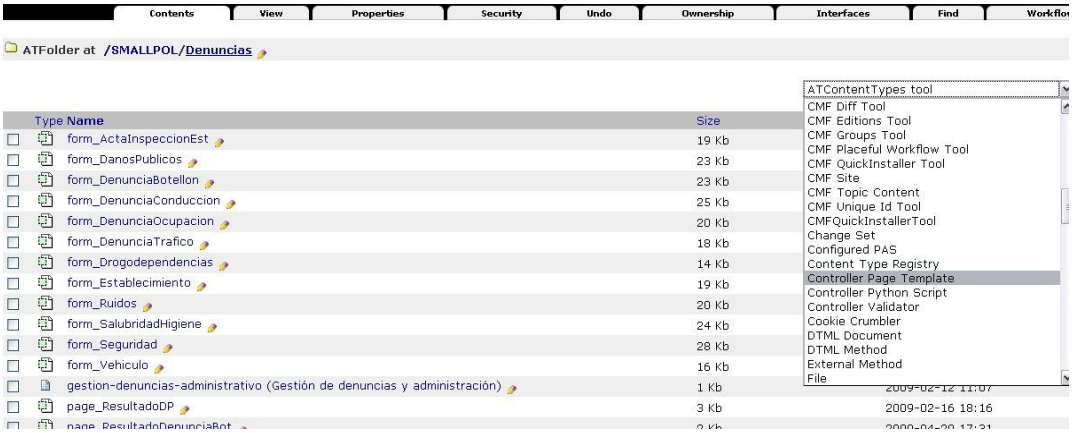

Figura 60: Añadir un formulario (Controller Page Template)

3. Situado en la *Pestaña1*, nos metemos en la sección *portal_skins*, dentro de la carpeta *custom*, en esta carpeta se han de crear todos los scripts de python que se quieran utilizar en la aplicación. En este caso se selecciona del desplegable de la derecha la opción "Controller Validator", se le asigna un nombre y se rellena

con el código de python necesario para el control de un formulario en cuestión. La selección se muestra en la figura 61.

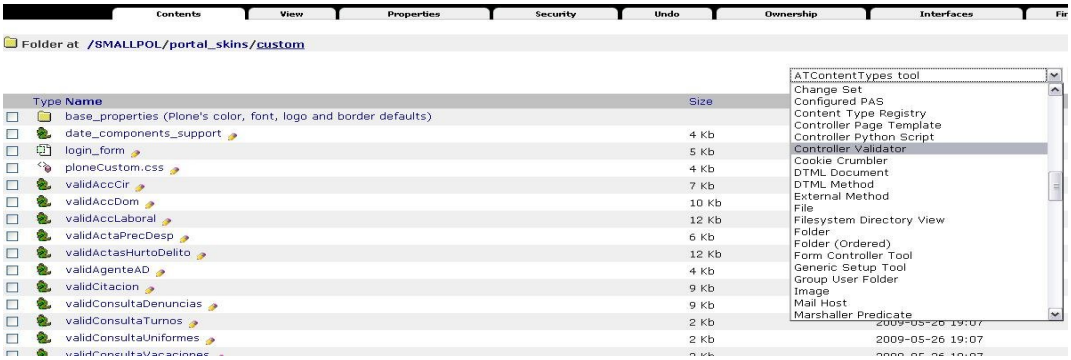

Figura 61: Añadir un script de python (Controller Validator).

4. Una vez creado el/los formularios y el script de Python, se han de relacionar, para que se controlen los valores de los campos y según el resultado se vaya a una página o a otra.

   Esta asignación se hace en el propio formulario, seleccionando la pestaña "Validator", y la pestaña "Actions".

   La asignación de "acciones" se muestra en la figura 62.

Figura 62: Asignación del destino según se cometan errores o haya éxito.

La asignación de "validadores" se muestra en la figura 63.

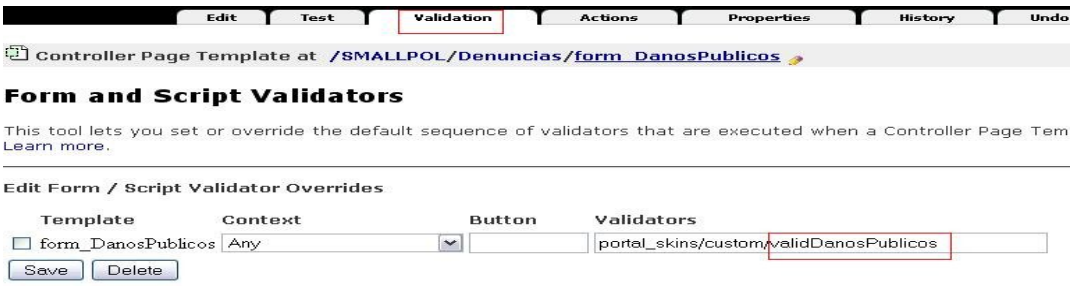

Figura 63: Asignación del validador al formulario.

Cada uno de los elementos creados en el sitio Plone cuenta con una pestaña “Security” a través de la cual se seleccionan los permisos que se le quieren asignar según el tipo de usuario. Algunos de esos permisos se muestran en la figura 64.

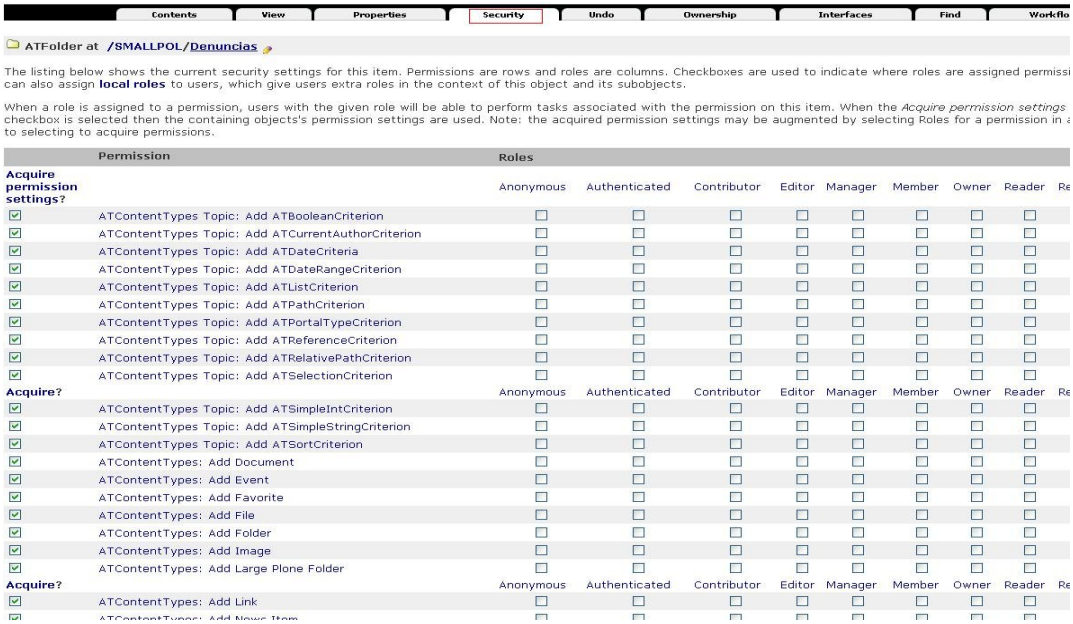

Figura 64: Asignación de los permisos para una sección.

Por otro lado tenemos la conexión con la base de datos, que se realiza situado en la “Pestaña1” y seleccionando en el desplegable de la derecha el adaptador necesario. Las características de la conexión se muestran en la figura 65.

Figura 65: Parámetros para la conexión con la base de datos.

Para finalizar este breve manual, faltaría relatar como personalizar el aspecto del sitio Plone creado. Para ello situados en la “Pestaña1”, nos metemos en la sección *portal_skins* dentro de la carpeta *plone_styles*, en ella se encuentran las hojas de estilo y las propiedades que controlan el aspecto por defecto de un sitio Plone. Modificando

estos ficheros dentro de una copia que se realiza dentro de la carpeta *custom*, de esta misma sección, se puede modificar el aspecto por completo del nuevo sitio Plone. La figura 66 muestra los dos ficheros a manipular.

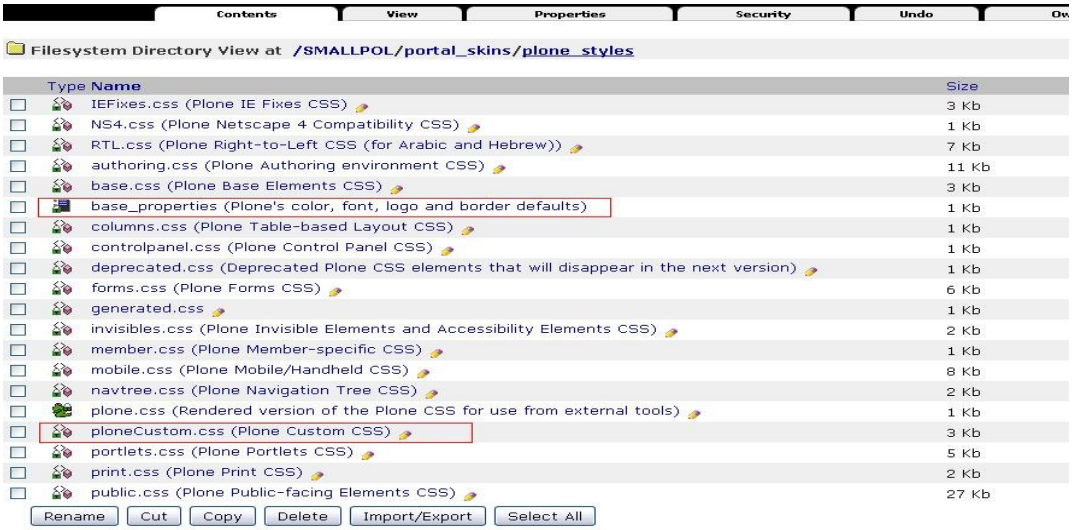

Figura 66: Hojas de estilo para modificar.

Uno de esos ficheros con el estilo de los caracteres, el color del resaltado de los errores, etc, se puede ver en la figura 67.

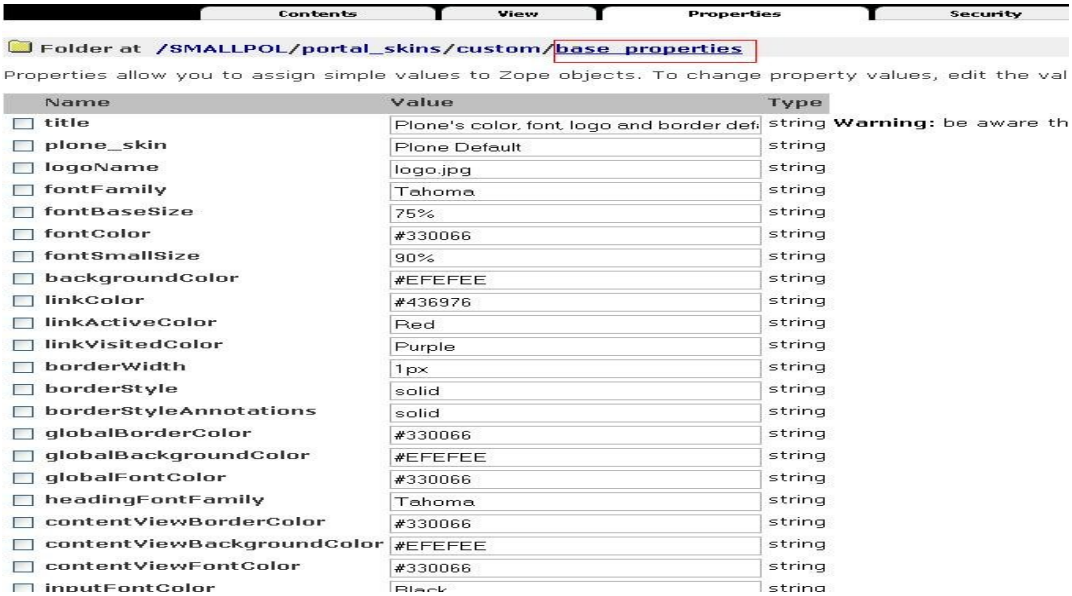

Figura 67: Ejemplo de propiedades visuales.

Una vez creado los elementos necesarios, el administrador debe salir de aplicación y refrescar la pantalla, para que los cambios tengan efecto. En el caso de que se hayan realizado cambios en el aspecto, el Administrador debe salir de aplicación y el servidor Web debe ser reiniciado para que el aspecto se actualice.

GLOSARIO DE TÉRMINOS

Software Libre: Es la denominación del software que brinda libertad a los usuarios sobre su producto adquirido y por tanto, una vez obtenido, puede ser usado, copiado, estudiado, modificado y redistribuido libremente.

CMS: Sistema de gestión de contenidos "Content Management System" es un programa que permite crear una estructura de soporte (framework) para la creación y administración de contenidos, principalmente en páginas Web, por parte de los participantes.

GPL: La Licencia Pública General de GNU "General Public License", es una licencia creada por la Free Software Foundation, y está orientada principalmente a proteger la libre distribución, modificación y uso de 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.

SSL: Es el protocolo de Capa de Conexión Segura. SSL proporciona autenticación y privacidad de la información entre extremos sobre Internet mediante el uso de criptografía. Habitualmente, sólo el servidor es autenticado (es decir, se garantiza su identidad) mientras que el cliente se mantiene sin autenticar; la autenticación mutua requiere un despliegue de infraestructura de claves públicas (o PKI) para los clientes. Este protocolo permite a las aplicaciones cliente-servidor comunicarse de una forma diseñada para prevenir escuchas, la falsificación de la identidad del remitente y mantener la integridad del mensaje.

Licencia Dual de MySQL: Por un lado se ofrece bajo la GNU GPL para cualquier uso compatible con esta licencia, pero para aquellas empresas que quieran incorporarlo en productos privativos deben comprar a la empresa una licencia específica que les permita este uso.

Licencia BSD: Pertenece al grupo de licencias de software Libre. 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.

SHELL: Es un intérprete de comandos, un programa informático que actúa como interfaz de usuario para comunicar al usuario con el sistema operativo mediante una ventana que espera órdenes escritas por el usuario en el teclado (por ejemplo, PRINT CARTA.TXT), los interpreta y los entrega al sistema operativo para su ejecución. La respuesta del sistema operativo se muestra al usuario en la misma ventana. A continuación, el programa *shell* queda esperando más instrucciones. Se interactúa con la información de la manera más sencilla posible, sin gráficas, sólo el texto crudo.

BATERIAS INCLUIDAS: Se refiere a que Python viene en su distribución estándar con una librería de módulos extensibles incluida.

WAI: La Iniciativa para la Accesibilidad Web "Web Accessibility Initiative" es una rama del "World Wide Web Consortium"(*W3C*) que vela por la accesibilidad de la Web. Publica las Guías de Accesibilidad al Contenido Web. La idea general del WAI es crear una serie de reglas claras.

Se incide especialmente en que las capacidades tecnológicas punteras (entiéndase por ejemplo animaciones con Adobe Flash, JavaScript, AJAX) se usen con la moderación o consideración suficiente para llegar al máximo conjunto posible de usuarios con una funcionalidad suficiente, sin desvirtuar el concepto de acceso frente al de avance tecnológico de moda, y prestando especial cuidado de ofrecer información alternativa.

TALES: "Template Attribute Language Expression Syntax" Son expresiones dentro del lenguaje TAL "Template Attribute Language". Este se utiliza para la creación de páginas en HTML y XML. TAL fue creado por Zope pero también es usado en proyectos basados en python.

CSS: Las hojas de estilo en cascada "Cascading Style Sheets" son un lenguaje formal 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 formular la especificación de las hojas de estilo que servirán de estándar para los agentes de usuario o navegadores.

CGI: Interfaz de entrada común “Common Gateway Interface” es una importante tecnología de la World Wide Web que permite a un cliente (explorador web) solicitar datos de un programa ejecutado en un servidor web. CGI especifica un estándar para transferir datos entre el cliente y el programa. Las aplicaciones que se ejecutan en el servidor reciben el nombre de CGIs.

GUI: La interfaz gráfica de usuario “Graphical User Interface” es un tipo de interfaz de usuario que utiliza un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz. Habitualmente las acciones se realizan mediante manipulación directa para facilitar la interacción del usuario con la computadora.

HOWTO: es un documento informal, generalmente corto, que describe *cómo* cumplir con una cierta tarea. Manual para productos adicionales.

LDAP: Protocolo Ligero de Acceso a Directorios, “Lightweight Directory Access Protocol”, es un protocolo de acceso unificado a un conjunto de información sobre una red. Habitualmente, almacena la información de login (usuario y contraseña) y es utilizado para autenticarse aunque es posible almacenar otra información

OPENID: Sistema de identificación digital descentralizado, con el que un usuario puede identificarse en una página web a través de una URL (o un XRI en la versión actual) y puede ser verificado por cualquier servidor que soporte el protocolo. En los sitios que soporten OpenID, los usuarios no tienen que crearse una nueva cuenta de usuario para obtener acceso.

XML-RPC: es un protocolo de llamada a procedimiento remoto que usa XML para codificar los datos y HTTP como protocolo de transmisión de mensajes.

BIBLIOGRAFÍA

[1] Gestores de contenido:

- Introducción a los Sistemas de Gestión de Contenido de código abierto (2004): http://mosaic.uoc.edu/articulos/cms1204.html
- Tipos de Sistemas de Gestión de Contenido (2008): http://luisitob.wordpress.com/2008/07/22/tipos-de-cms/

[2] Comparación de Sistemas de Gestión de Contenido de código abierto (2009): http://idealware.org/comparing_os_cms/idealware_comparing_os_cms_report.pdf

[3] Drupal:

- Página oficial de Drupal: www.drupal.org

[4] Joomla:

- Página oficial de Joomla (2005-2009): www.joomla.org

[5] Plone 3.0:

- Página oficial de Plone (2000-2009): http://plone.org/

[6] Manual para el desarrollo e instalación de arquetipos en Plone: http://plone.org/documentation/manual/archetypes-developer-manual/a-semi-realistic-example/introduction

[7] Utilizar Plone detrás de Apache con SSL.

- Seguridad en Plone (2009): http://plone.org/documentation/how-to/securing-plone
- Ejemplo de VirtualHost para Apache2 (2009): http://plone.org/documentation/tutorial/plone-apache/virtualhost
- Información sobre como utilizar Zope y Apache de manera conjunta (2008): http://wiki.zope.org/zope2/ZopeAndApache

[8] PostgreSQL 8.3:

- Página oficial de PostgreSQL (1996-2009): http://www.postgresql.org

[9] Descarga e información sobre Psycopg: http://initd.org

[10] MySQL:

- Página oficial de MySQL (1995-2009): www.mysql.com

[11] Zope 2.10:

- Página oficial de Zope: http://www.zope.org/
- Zope, el servidor de aplicaciones libre (2003): www.deli.deusto.es/Resources/Documents/transparencias_curso_zope.pdf

[12] Python 2.4:

- Página oficial de Python (1990-2009): http://www.python.org
- Descripción general del lenguaje de programación Python: http://maracay.velug.org.ve/descargas/PonenciaPython.pdf
- Qué es Python y por qué me debería de interesar (2008): http://jotarp.org/var/lenguajePython/index.html

[13] Guido van Rossum "Guía de aprendizaje de Python."

FredL.Drake, Jr., editor.

Python Software Foundation. Release 2.4.la0-11Septiembre, 2005.

http://pyspanishdoc.sourceforge.net/tut/tut.html

[14] Guía de aprendizaje de lenguaje TAL/TALES y METAL (2005):

http://www.owlfish.com/software/simpleTAL/tal-guide.html

[15] Formularios en HTML:

- Guía de aprendizaje de HTML: http://html.conclase.net/w3c/html401-es/index/list.html

[16] Consultas en SQL:

- Manual de SQL: http://www.solorecursos.com/sql.htm

[17] Hojas de estilo (CSS):

- Manual de CSS (2009): http://www.webestilo.com/css

[18] Ley de protección de datos:

- Disposiciones generales de la Ley de Protección de Datos de Carácter Personal, número 298 del BOE. (Diciembre de 1999):

http://www.boe.es/boe/dias/1999/12/14/pdfs/A43088-43099.pdf

[19] Colegio oficial de Ingenieros Técnicos de Telecomunicación (2009): http://www.coitt.es/

[20] Ayuda sobre PostgreSQL:

- Foro especializado en PostgreSQL: pgsql-es-ayuda@postgresql.org

[21] Ayuda sobre Plone:

- Foro especializado en Plone: plone-conosur@lists.plone.org

[22] Información generalizada:

http://es.wikipedia.org

http://www.slideshare.net