VNIVERSIDAD NACIONAL AVTNOMA DE MEXICO

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

"MIGRACIÓN Y NUEVAS CARACTERÍSTICAS DEL SISTEMA DE VOTACIÓN ELECTRÓNICA DEL INSTITUTO DE MATEMÁTICAS DE LA UNAM"

T E S I S

QUE PARA OBTENER EL GRADO DE:

MAESTRO EN CIENCIAS
(COMPUTACIÓN)

P R E S E N T A:

IVÁN CHRISTIAN CERVANTES CORONADO

DIRECTOR DE TESIS: DR. SERGIO RAJSBAUM GORODEZKY

México, D.F. **2009.**

Agradecimientos

A mi familia. Quienes me apoyaron para sacar adelante este proyecto y de quienes estoy orgulloso de formar parte.

A mis amigos. En especial a Edith que me ha brindado su apoyo y cariño a lo largo de estos últimos años.

Resumen

La modernización de procesos en varios ámbitos de la vida moderna ha sido una realidad, gracias al avance en la tecnología en computación que se tiene hoy en día, la cual teóricamente tiene el potencial de modernizar el proceso de votación en elecciones de cualquier tipo. Con lo anterior en mente, el Instituto de Matemáticas de la UNAM, desarrollo un sistema de votaciones, el cuál he **rediseñado**, **migrado** y **añadido nuevas características**, lo cuál da como resultado un nuevo sistema con mejoras significativas.

La tesis está organizada de la siguiente forma: el **primer capítulo** es la introducción a este trabajo, se define el contexto de la tesis donde hablamos del *CMS* ***Plone*** que es la plataforma de desarrollo del sistema de la tesis, también menciono las ventajas y características importantes de *Plone*, se exponen trabajos relacionados como son el sistema de votaciones que viene por default en *joomla* que es un *CMS* como lo es *Plone*, los casos de éxito en el uso del sistema de votaciones se mencionan y por último se dan algunos de los sitios más importantes en cuanto a información sobre votaciones electrónicas se refiere. En el **segundo capítulo** define que son las votaciones electrónicas, los objetivos de seguridad que deberíamos tener en los sistemas de cómputo, se enlistan las características necesarias de un sistema de votación electrónica, estas listas fueron creadas por dos personajes muy importantes en la seguridad informática, además el lector podrá obtener el bagaje necesario en protocolos de seguridad. En el **tercer capítulo**, tenemos el estado del arte de los sistema de votación electrónica, se explicará el protocolo de seguridad del sistema de votación electrónica *KOA*, definiendo sus propiedades, las cuales se deben de cumplir en el protocolo, veremos como es que se cumplen dichas propiedades dentro del sistema, como también explicare cada uno de sus componentes, se mostrará el plugin *DutchTallySystem* que es un demo de una parte del sistema *KOA*, y por último complicaciones; hay que mencionar que el sistema de votaciones del *IMATE* esta basado en parte por el sistema *KOA*. El sistema de votación electrónica del *IMATE* es explicado de forma detallada en este capítulo, describiré las características del sistema, como son las fases del proceso, los usuarios involucrados, y los estados de la votación, enseguida se describe el protocolo de votación donde se exponen el manejo de las boletas, electores, candidatos, como se vota, y más. Además se muestran los diagramas en *UML* del sistema. **Capítulo cuatro**, describo las tareas más comunes para poder migrar un producto *Plone* 2 a *Plone* 3, se deja claro lo que significa migración en nuestro contexto de *Plone* y por último las problemáticas. Este pasó es la base para el rediseño y las nuevas características. En el **quinto capítulo**, vemos el rediseño y refactorización del producto de selección de usuarios y de votaciones, como se utilizó el framework GenericSetup. Además algunas de las nuevas características entre las que se encuentran: guardar configuraciones, utilizar adaptadores y votar por N candidatos. En el **capítulo seis**, vemos las características especificas, que se implementaron para la integración del sistema de votación con el sistema *infoMatem*, dichas características son: utilizar el administrador de usuarios *FacultyStaff*, crear diferentes tipos de pruebas automáticas del sistema e internacionalizarlo. En el **capítulo siete** se encuentran las conclusiones generales y los trabajos a futuro. Además de los capítulos principales tenemos una serie de apéndices de mucha importancia para los usuarios del sistema, el **apéndice A** se enfoca a introducir al lector en las tareas primarias de un sitio *Plone*, desde la instalación hasta la administración básica, todo esto directo al punto. **Apéndice B**, es el manual de usuario del producto de selección de usuarios, vemos como instalarlo, las dependencias, y sus funciones, todo esto ejemplificado y con imágenes de cada paso. **En el apéndice C** tenemos el manual mas extenso que es del producto de votaciones, se explica con lujo de detalle cada unos de los posibles pasos y funcionalidades del producto de votaciones, vemos como instalarlo, dependencias, todo ejemplificado y con imágenes del sistema efectuando cada paso. Por último tenemos el manual de *GPG* que corresponde al **apéndice D**, está enfocado a enseñar como utilizar la herramienta *GnuPG* para las funciones que son necesarias para que el proceso de votación se cumpla con éxito, aunque también veremos Tips a la hora de utilizar está herramienta.

Antecedentes

En el Instituto de Matemáticas (*IMATE*) se realizan votaciones de forma regular con el objetivo de elegir miembros de comisiones de funcionarios internos y de representantes ante comisiones externas al Instituto, como la que se especifica en los artículos 51 fracción IV, 51 A, 52-A, 52-B, 52-C y 52-D del Estatuto General de la UNAM; 3º, 4º, 5º y 7º del Reglamento Interno del Consejo Técnico de la Investigación Científica (RCTIC), y en el Reglamento Interno de este Instituto, en donde se dispone que cada 3 años se debe realizar la elección de un representante de su personal académico ante el consejo técnico de la investigación científica, mediante voto universal, libre y secreto [23][29].

Este proceso se realizaba de forma manual con una urna física; en el 2007 este proceso fue modernizado al implementarlo como un sistema de cómputo, utilizando tecnologías como son: *Plone*, *Zope* y *Python* [23]. Me refiero al sistema de votaciones electrónicas del *IMATE*, el cual cumple con importantes propiedades de seguridad [23], esto se debe a que está basado en el protocolo de seguridad de otros sistemas de votaciones electrónicas importantes, como lo es *KOA* [23, 2]. Además es el sistema que da pie a este trabajo de tesis, así como lo es el *CMS Plone* [4]. El software de votaciones del *IMATE* ha sido utilizado de forma exitosa en: ELECCIÓN COMISIÓN DICTAMINADORA - Diciembre de 2007 [23], y ELECCIÓN DE CONSEJEROS INTERNOS - abril 2009 [30]. Para una descripción amplia sobre el protocolo y el sistema de votaciones del *IMATE* véase el capítulo 3.

La corriente que nos lleva a utilizar estas tecnologías se debe al sistema *infoMatem*[30], el cual permite realizar funciones como son: actualizar información curricular de los investigadores, compartir información del instituto, administra y realizar búsqueda de la información relativa a los investigadores. Además *infoMatem* puede administrar usuarios y procesos relacionados con la Secretaría Académica, Consejo Interno y la Secretaría Técnica [40]. Este sistema *infoMatem* del cual forma parte el sistema de votación está siendo utilizado por otras dependencias de la UNAM. Además de que se ha formado un grupo importante de usuarios de *Plone* en México del cual se tienen reuniones mensuales en el Instituto de Matemáticas de la UNAM y este grupo se encarga del *Plone Day* en México (de este grupo formo parte) [5]. Hay que mencionar las conferencias y platica que se dará sobre los productos del sistema *InfoMatem* en el simposio organizado por uno de los grupos con mayor reconocimiento a nivel mundial, me refiero al grupo WebLion de la universidad de Penn State [31], Pensilvania, EE.UU.. Por último decir que se ha realizado con éxito una muestra de integración de todos los sistemas de *infoMatem* del cual forma parte el sistema de votación de la tesis al director del Instituto de Matemáticas de la UNAM (Junio 2009) con lo cuál se ha cumplido con la prueba de aceptación.

Objetivo general de la tesis

El objetivo general es rediseñar el sistema de votación electrónica del *IMATE* en la versión 3 de *PLONE*.

Objetivos específicos de la tesis

Los objetivos específicos que se deben cumplir para el buen desarrollo de está tesis son los siguientes:

- Determinar las propiedades de seguridad que debe cumplir un protocolo de votación electrónica.
- Conocer los protocolos de votación electrónica de los sistemas de caso de estudio: *IMATE* y *KOA* [2, 30].
- Conocer la plataforma de *Plone* 3 como gestor de Contenido, *Zope* 3 como gestor de base de datos y *Python* como lenguaje de Programación [4, 33, 51].
- Rediseñar los productos de selección de usuarios y de votaciones.
- Migrar los productos de selección de usuarios y de votaciones a *Plone* 3.

- Integrar los productos de selección de usuarios y de votaciones con el sistema *infoMatem* [30].
- Analizar e implementar nuevas características que han surgido como requerimientos.
- Implementar una suite de pruebas de instalación y de creación de la instancia del producto.
- Internacionalizar el sistema de votación mediante los estándares utilizados por la comunidad de *Plone*.
- Generar documentación sobre el gestor de Contenido *Plone* [4].
- Generar manuales de los dos productos que forman el sistema de votación, y del software *GnuPG* [52].

Metodología

- Recopilar documentos relacionados con la votación electrónica.
- Redacción de la tesis.
- Analizar e instalar el sistema de votación electrónica *KOA*.
- Analizar e instalar el sistema de votación electrónica del *IMATE*.
- Definición de una estrategia para el rediseño del producto de votaciones y de selección de usuarios, así como en la migración, en las cuales se incluyó la apropiada definición de estándares de programación y aseguramiento de la calidad del código, mediante el aprendizaje autodidacta de la plataforma *Plone*, *Zope* y *Python*.
- Analizar cual es la mejor forma de implementar un requerimiento, mediante platicas con los usuarios de *Plone* en español, y mediante IRC en el canal oficial de *Plone*. En estos dos canales de comunicación se intercambian ideas con los expertos mundiales en *Plone* (incluyendo los que desarrollan *Plone*).
- Identificar las propiedades de seguridad que debe cumplir un sistema de votación electrónica.
- Aplicación de las técnicas de administración de proyectos tecnológicos requeridas para dividir el proceso en fases y darle un enfoque efectivo y eficiente. Los resultados han sido supervisados mediante revisiones periódicas del avance en el rediseño, migración y nuevos requerimientos del producto con el grupo de trabajo del sistema *infoMatem*.
- Revisión de la literatura sobre sistemas de votación electrónica, Plone, patrones de diseño, Python, Zope, y frameworks.
- Pruebas de confiabilidad.

Contribución y relevancia

La contribución de este trabajo es mejorar el sistema de votación electrónica del *IMATE* y mantenerlo como un sistema de uso real y confiable, todo esto mediante el rediseño del sistema. Específicamente se contribuyó con el rediseño, migración, integración con *infoMatem*, internacionalización, pruebas, interfaces, adaptadores, eficiencia, nuevos requerimientos, y los manuales de los productos de votacion, selección de usuarios y *GPG*. Su relevancia es continuar con el proyecto de un sistema de votación electrónica que sea configurable, confiable, seguro y que forme parte de otro proyecto que es *infoMatem*.

Índice general

Capítulo 1

INTRODUCCIÓN

La modernización de procesos en varios ámbitos de la vida moderna ha sido una realidad, gracias al avance en la tecnología en computación que se tiene hoy en día, la cual teóricamente tiene el potencial de modernizar el proceso de votación en elecciones de cualquier tipo.

Las motivaciones de está modernización son: las grandes ventajas para los administradores de la elección, las boletas pueden ser provistas en cualquier lenguaje, hacerlas accesibles a personas con discapacidades. Estas son sólo algunas de las motivaciones generadas por el hecho de estar en un contexto computacional. Otra motivación es que algunos sectores sociales combinan cada vez más el Internet con su vida cotidiana. Esto incluye el uso de redes sociales [5], portales, blogs y sitios Web.

Hay que recordar que pasar un proceso como el de votación al contexto computacional puede traer complicaciones fuertes, y es el caso como veremos en el desarrollo de la tesis, aunque hay que tomar en cuenta que el sistema de votación aquí tratado no se va a crear desde cero, se va a dar una continuidad al ya creado [23]. El sistema en cuestión está pensado para pequeñas y medianas organizaciones las cuales su padrón electoral no supere los 500 votantes.

La migración de un sistema ya establecido trae consigo complicaciones de importancia, ya que el usuario espera que la interfaz que se le ha provisto no cambie de forma sustancial, el administrador del sistema espera que los componentes tengan por lo menos la misma estructura, y todos quieren que el sistema contenga por lo menos la misma funcionalidad y propiedades de seguridad, dichas propiedades y sus requerimientos son moderados, por lo que para entornos pequeños y medianos es suficiente el esquema de seguridad aquí implementado, pero para sectores gubernamentales o federales se requiere un esquema más elaborado.

El sistema de votación migrado y mejorado en este trabajo de tesis se basa en los protocolos detallados en el estado del arte de la comunidad internacional y estándares. **El sistema de votación se divide en dos productos que son: selección de usuarios y votaciones**.

1.1. Sistema de gestión de contenido *Plone*

Esta sección está enfocada a describir que es un sistema de gestión de contenidos (*CMS* por sus siglas en inglés), en específico el *CMS Plone* [4, 3].

El gestor de contenidos describe el proceso de organizar contenido, específicamente en lidiar con documentos electrónicos. Esto puede incluir documentos de texto, como también imágenes, sonidos, correos electrónicos, bases de datos, o eventos. Como una regla, nos estamos refiriendo a todo tipo de información que pueda ser guardada dentro de la computadora [3].

Gracias a la prevalencia de las computadoras y las redes de datos, especialmente la Internet, la mayoría de la información está disponible digitalmente en la red. Un sistema de gestión de contenido le permite a grupos de personas administrar información que comparten. Un *CMS* es comúnmente usado en crear una intranet o en establecer una presencia en la Web. Para hacer esto, tenemos soluciones

altamente configurables o más generales. Por ejemplo, un *CMS* configurado puede estar especializado en organizar archivos e información de los usuarios, mientras que una aplicación *CMS* genérica puede ser utilizada en una variedad amplia de aplicaciones.

Plone es un *CMS* del tipo genérico. Esto se debe a que puede ser configurado para encajar las necesidades especiales de varias organizaciones, aunque también es una herramienta excepcional en diferentes áreas de aplicación.

1.1.1. Características principales

Veamos las características que un *CMS* le ofrece a sus usuarios:

Libertad e independencia

- Disponibilidad de los datos: El usuario tiene la libertad en cualquier momento de acceder a estadísticas, reportes o archivos, sin depender en la presencia de compañeros de trabajo o la disponibilidad de un grupo o departamento de trabajo.
- Los documentos son automáticamente procesados y disponibles para revisión en una variedad de formatos. Estos incluyen archivos PDF, formatos utilizados por aplicaciones como son OpenOffice.org, Word, Excel, JPEG, y muchos otros.
- Un *CMS* hace el contenido más accesible para usuarios con desventajas físicas. Una persona ciega, por ejemplo, puede hacer uso de un display Braille para acceder a los textos.
- Los documentos son fáciles de buscar en el sitio. Varios formatos de texto, eventos, imágenes y otro tipo de datos son todos automáticamente incluidos en los resultados de búsquedas. Búsquedas de texto completo incrementan la calidad de los resultados.
- No se requiere trabajar desde una ubicación especifica. Somos libres de acceder al *CMS* usando la computadora en nuestro lugar de trabajo, un teléfono celular, o una PDA.

Fiabilidad

- Con un *CMS*, los documentos están organizados de forma centralizada, con esto me refiero a que está en un sólo lugar. Esto nos permite evitar circulación de varios documentos que son únicos e incluso copias conflictivas, incrementando la fiabilidad del contenido ofrecido.
- Para que los datos puedan ser categorizados de forma sistemática, el contenido es guardado en un *CMS* junto con metadatos [3].
- Los workflows utilizados en *CMS* permiten automatizar la preparación y publicación del contenido de acuerdo con los lineamientos y principios de la organización.
- Un *CMS* hace posible eliminar contenido de partes públicas de un sitio web mientras lo retiene en los archivos privados del sistema.

Colaboración

- Con un *CMS*, muchas personas pueden trabajar juntas de forma simultánea, por ejemplo, en un sitio web de su empresa. Los recursos como imágenes utilizadas en ciertas páginas tal vez sean utilizadas por alguien más con un propósito diferente. Listas de contenidos muestran los archivos creados en ese momento o le recuerda a los usuarios sobre eventos.
- Al utilizar formatos estandarizados, un *CMS* puede integrar información proveniente de otro sistema. Esto puede consolidar noticias y eventos desde una agencia de noticias, por ejemplo, y preparar la información para presentarla al usuario.

- El contenido en un *CMS* puede ser desarrollado y publicado por una editorial pero también por usuarios de forma individual.
- A los usuarios se les puede asignar ciertas funciones con un *CMS* de acuerdo a las responsabilidades y áreas de trabajo. Pueden ser autores, responsables de la creación de contenido, o editores, quienes autorizan la revisión o publicación de la información. Un usuario debería ser un editor en su propia área de trabajo.
  Asignar funciones mejora la seguridad, facilita la organización y workflows, y facilita también la comunicación entre los responsables.

Seguridad

- A través de el uso de los lineamientos de seguridad en la administración de contenidos, un *CMS* asegura en múltiples niveles que los documentos pueden únicamente ser creados, editados, publicados, archivados, o vistos por usuarios a quienes se les autorizó hacerlo.
- Debido a que los workflows en un *CMS* pueden ser adaptados para atender las demandas especificas de las compañías, ellos pueden convertirse al workflow que ya tenía dicha compañía.
- Tener unos lineamientos de seguridad definidos en cada una de las áreas del *CMS* nos permite tener mayor flexibilidad y seguridad en el manejo confidencial de los datos.

Cada una de estas características son parte intrínseca del sistema de votación, del proceso de migración y de las nuevas características. En específico *Plone* nos otorga lo siguiente:

- Internacionalización
- Suite de pruebas
- Interfaces y adaptadores
- Archivos de configuración
- *GenericSetup*
- *Python eggs*

Para una introducción al manejo de *Plone* véase el **apéndice A** de este trabajo de tesis.

1.1.2. Las dos caras de *Plone*

Tenemos una cara que es ***Plone* como aplicación** que es utilizada para intranets, sitios web públicos, repositorios de documentos, y como un sistema basado en web. *Plone* compite de manera exitosa en el mercado de los *CMS*, y se escoge sobre sistemas como lo son RedDot *CMS* y MicrosoftSharedPoint [6].

Plone es desarrollado casi exclusivamente por voluntarios. Es open source, lo cual significa que puedes obtenerlo y usarlo libremente, y eres libre de modificar el código fuente. Cabe comentar que en mayo de 2009 se está llevando a cabo una votación entre la comunidad de *Plone* para relicenciar ciertas partes del framework [7].

La mayoría de los contribuyentes más importantes se ganan la vida por lo que se conoce como ***Plone* el framework**. Ellos son desarrolladores web que ofrecen consultorías y soporte, utilizando el API de *Plone* como una plataforma base, donde encima de ésta, realizan sus desarrollos.

Entonces tenemos bien definidas las dos caras de *Plone*, una en la que tenemos una aplicación que nos permite tener un sitio web y gestionar el contenido, y tenemos otra cara que es la del desarrollador de aplicaciones utilizando la API de *Plone*.

1.2. Trabajos relacionados

Veamos trabajos relacionados con está tesis, que como sabemos trata de *Plone* y su sistema de votaciones, algunos de los trabajos son, el *CMS Joomla* [8] y su sistema de votaciones, que a diferencia del nuestro, el de *Joomla* ya viene incluido de paquete (out-of-the-box), los productos de *Plone* para hacer ratings [55]. El *CMS Drupal* y su API de votaciones [61]. Otro trabajo relacionado es el sistema de votaciones de la UNAM [12].

1.2.1. Sistemas de votación en *CMS*

Los sistemas de votación que hay para *CMS* son de ranquear (*rating* o *poll*), los cuales no son estrictamente de votaciones electrónicas que automatizan un proceso electoral formal y con seguridad. Estos sistemas sólo sirven para calificar documentos con estrellas, dedo arriba/dedo abajo, etc.

1.2.1.1. *Joomla* y su sistema de votación integrado

Joomla! es un Sistema de Gestión de Contenidos (*CMS*) reconocido mundialmente, que le ayuda a construir sitios web y otras aplicaciones en línea potentes. Lo mejor de todo, es que *Joomla*! es una solución de código abierto y está disponible libremente para cualquiera que desee utilizarlo [9]. Esta programado con *PHP* [10] y utiliza una base de datos relacional *MySQL* [11].

Las aplicaciones en *Joomla* son programadas con *PHP*, *AJAX*, *HTML*, etc. El sistema de votaciones que incluye *Joomla* se encarga de calificar documentos, por medio de una política de cinco estrellas. Cualquier usuario puede entrar al sitio y votar por un documento eligiendo cuantas estrellas le damos. Podemos votar el número de veces que queramos.

El sistema de votacion de *Joomla* está diseñado con el patrón de diseño *MVC* [50], muy bien separados cada componente del patrón (Fig. 1.1, 1.2, 1.3), en su archivo respectivo, hay que tomar en cuenta que este sistema no está pensando para seguir un protocolo de seguridad ni nada por el estilo, la única seguridad es que checa una cierta cookie, para decirte que ya votaste el día de hoy, pero eso puede ser modificado con facilidad, además de este argumento se envía la calificación y un número aleatorio por documento, este número aleatorio identifica al documento votado.

```
function plgContentVote( &$row, &$params, $page=0 )
{
        $uri = & JFactory::getURI();

        $id     = $row->id;
        $html   = '';

        if (isset($row->rating_count) && $params->get( 'show_vote' ) && !$params->get( 'popup' ))
        {
                JPlugin::loadLanguage( 'plg_content_vote' );
                $html .= '<form method="post" action="' . $uri->toString( ) . '">';
                $img = '';

                // look for images in template if available
                $starImageOn    = JHTML::_('image.site',  'rating_star.png', '/images/M_images/' );
                $starImageOff   = JHTML::_('image.site',  'rating_star_blank.png', '/images/M_images/' );

                for ($i=0; $i < $row->rating; $i++) {
                        $img .= $starImageOn;
                }
                for ($i=$row->rating; $i < 5; $i++) {
                        $img .= $starImageOff;
                }
                $html .= '<span class="content_rating">';
                $html .= JText:: ( 'User Rating' ) .':'. $img .' / ';
                $html .= intval( $row->rating_count );
                $html .= "</span>\n<br />\n";

                if (!$params->get( 'intro_only' ))
                {
                        $html .= '<span class="content_vote">';
                        $html .= JText::_( 'Poor' );
                        $html .= '<input type="radio" alt="vote 1 star" name="user_rating" value="1" />';
                        $html .= '<input type="radio" alt="vote 2 star" name="user_rating" value="2" />';
                        $html .= '<input type="radio" alt="vote 3 star" name="user_rating" value="3" />';
                        $html .= '<input type="radio" alt="vote 4 star" name="user_rating" value="4" />';
                        $html .= '<input type="radio" alt="vote 5 star" name="user_rating" value="5" checked="checked" />';
                        $html .= JText::_( 'Best' );
                        $html .= ' <input class="button" type="submit" name="submit_vote" value="'. JText::_( 'Rate' ) .'" />';
                        $html .= '<input type="hidden" name="task" value="vote" />';
                        $html .= '<input type="hidden" name="option" value="com_content" />';
                        $html .= '<input type="hidden" name="cid" value="'. $id .'" />';
                        $html .= '<input type="hidden" name="url" value="'.  $uri->toString( ) .'" />';
                        $html .= '</span>';
                }
                $html .= '</form>';
        }
        return $html;
}
```

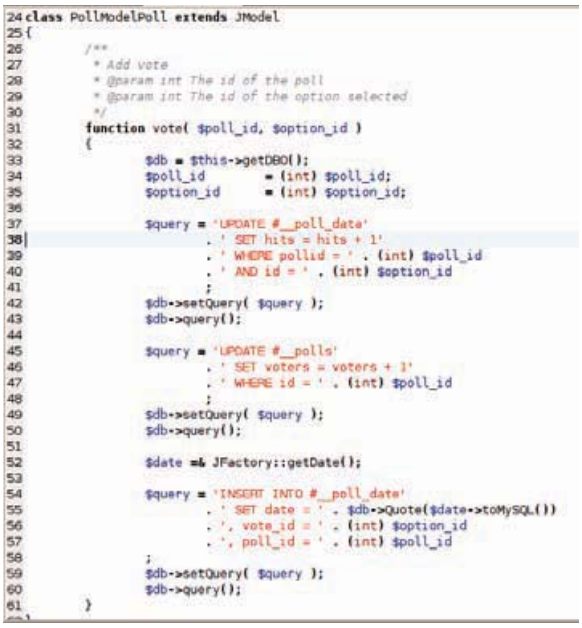
Figura 1.1: Vista

La vista se encarga de presentarnos la opción de votar por el documento mediante el mecanismo de las cinco estrellas. Envía los argumentos necesarios al controlador.

```
43          * Add a vote to an option
44          */
45         function vote()
46         {
47                 global $mainframe;
48
49                 // Check for request forgeries
50                 JRequest::checkToken() or jexit( 'Invalid Token' );
51                 $db                 =& JFactory::getDBO();
52                 $poll_id        = JRequest::getVar( 'id', 0, '', 'int' );
53                 $option_id      = JRequest::getVar( 'voteid', 0, 'post', 'int' );
54
55                 $poll =& JTable::getInstance('poll','Table');
56                 if (!$poll->load( $poll_id ) || $poll->published != 1) {
57                         JError::raiseWarning( 404, JText::_('ALERTNOTAUTH') );
58                         return;
59                 }
60                 $cookieName     = JUtility::getHash( $mainframe->getName() . 'poll' . $poll_id );
61                 // ToDo - may be adding those information to the session?
62                 $voted = JRequest::getVar( $cookieName, '0', 'COOKIE', 'INT');
63
64                 if ($voted || !$option_id )
65                 {
66                         if($voted) {
67                                 $msg = JText::_('You already voted for this poll today!');
68                         }
69
70                         if(!$option_id){
71                                 $msg = JText::_('WARNSELECT');
72                         }
73                 }
74                 else
75                 {
76                         setcookie( $cookieName, '1', time() + $poll->lag );
77
78                         require_once(JPATH_COMPONENT.DS.'models'.DS.'poll.php');
79                         $model = new PollModelPoll();
80                         $model->vote( $poll_id, $option_id );
81
82                         $msg = JText::_( 'Thanks for your vote!' );
83                 }
84                 // set Itemid id for links
85                 $menu =& JSite::getMenu();
86                 $items  = $menu->getItems('link', 'index.php?option=com_poll&view=poll');
87
88                 $itemid = isset($items[0]) ? '&Itemid='.$items[0]->id : '';
89
90                 $this->setRedirect( JRoute::_('index.php?option=com_poll&id='. $poll_id.':'.$poll->alias.$itemid, false), $msg );
91         }
```

Figura 1.2: Controlador

El controlador realiza la lógica de negocio, en está parte es donde se valida la cookie mencionada anteriormente, fuera de eso lo más importante es crear una instancia del modelo y mandar a guardar los datos a la base de datos relacional.



```
24 class PollModelPoll extends JModel
25 {
26         /**
27          * Add vote
28          * @param int The id of the poll
29          * @param int The id of the option selected
30          */
31         function vote( $poll_id, $option_id )
32         {
33                 $db = $this->getDBO();
34                 $poll_id        = (int) $poll_id;
35                 $option_id      = (int) $option_id;
36
37                 $query = 'UPDATE #__poll_data'
38                         . ' SET hits = hits + 1'
39                         . ' WHERE pollid = ' . (int) $poll_id
40                         . ' AND id = ' . (int) $option_id
41                         ;
42                 $db->setQuery( $query );
43                 $db->query();
44
45                 $query = 'UPDATE #__polls'
46                         . ' SET voters = voters + 1'
47                         . ' WHERE id = ' . (int) $poll_id
48                         ;
49                 $db->setQuery( $query );
50                 $db->query();
51
52                 $date =& JFactory::getDate();
53
54                 $query = 'INSERT INTO #__poll_date'
55                         . ' SET date = ' . $db->Quote($date->toMySQL())
56                         . ', vote_id = ' . (int) $option_id
57                         . ', poll_id = ' . (int) $poll_id
58                 ;
59                 $db->setQuery( $query );
60                 $db->query();
61         }
```

Figura 1.3: Modelo

Por último el modelo actualiza las tablas correspondientes con los datos pasados por el controlador,

como se puede notar en los querys no hay un mecanismo de seguridad en dicho componente.

1.2.1.2. Productos de *Plone* para ratings

Como hemos visto el sistema default de votaciones en *Joomla* sirve para hacer ratings de documentos más que ser una plataforma seria de procesos electorales. *Plone* también cuenta con una serie de productos que sirven para rankear contenido [55].

Un ejemplo concreto es *Content Ratings* [54], un paquete de *Zope* 3, que nos permite fácilmente asignarle rating al contenido del sitio *Plone*, inclusive un usuario no autenticado puede votar por un contenido. Provee una serie de interfaces, adaptadores y vistas para permitir que la aplicación de ratings sea adjuntada a cualquier objeto **IAnnotable** [54].

1.2.1.3. *Drupal* y su *Voting API*

Drupal es un sistema de gestión de contenido para sitios Web. Permite publicar artículos, imágenes, servicios añadidos como foros, encuestas, votaciones, blogs y administración de usuarios y permisos [60]. *Drupal* es un sistema dinámico: en lugar de almacenar sus contenidos en archivos estáticos en el sistema de archivos del servidor de forma fija, el contenido textual de las páginas y otras configuraciones son almacenados en una base de datos y se editan utilizando un entorno Web incluido en el producto.

Este *CMS* ha desarrollado una API de votaciones (Voting API), la cual ayuda a los desarrolladores que necesiten utilizar una API estandarizada y un esquema para almacenar, obtener, y tabular votos que se hacen sobre el contenido de *Drupal* [61]. También podemos hacer lo siguiente:

- Rankear cualquier contenido (comentarios, nodos, usuarios, etc)
- Votación multicriterio, por ejemplo calificar un usuario basado en sus vídeos, audio, etc.
- Tabulación automática de resultados, la cual soporta diferentes estilos, como porcentaje y '+1/-1'.

Sin embargo está API sigue siendo para hacer ratings sobre contenido del sitio *Drupal*.

1.2.2. Sistemas de votaciones electrónicas

Estos sistemas no son parte de un *CMS*, y además cumplen con ser sistemas para procesos electorales formales, como los descritos en el estado del arte en el capítulo 3. Como podremos notar más adelante, hay una diferencia abismal tanto en diseño como en seguridad. Por lo tanto el sistema de votaciones del *IMATE* sería el primero de su clase para un *CMS*.

1.2.2.1. SVE 3.5

El sistema de votaciones de la UNAM se encuentra en su versión 3.5 (datos sacados de la carta de auditoría del sistema de votaciones electrónicas [13]). El sistema se compone de programas en lenguaje *PHP*, funciones criptográficas en lenguaje *C*, ejecutables Linux y todo lo referente a la base de datos.

1.- Código del sistema

El código recibido está libre de rutinas maliciosas que pudieran permitir que se alteraran los resultados de las elecciones en forma automática [13].

Contiene las validaciones mínimas necesarias para evitar que el envío de datos espurios por parte de un atacante desde un programa navegador ordinario, como el Internet Explorer de Microsoft o el Fírefox de Mozilla, hagan fallar el sistema o alteren alguno de los resultados En las pruebas realizadas encontramos que los datos enviados por un programa malicioso que pudiera usar un atacante, no tienen efecto en los resultados de la elección.

El sistema permite la votación por parte de cualquier persona que esté inscrita en los padrones y se identifique con una clave de usuario y dé la contraseña preestablecida, con lo que en nuestra opinión se cumple con el requisito de voto universal.

Este mismo mecanismo apoya el concepto de voto directo siempre y cuando los votantes no divulguen sus contraseñas.

Dentro del código del sistema no encontramos ningún elemento que pudiera ser usado a fin de coaccionar el voto, como por ejemplo un recibo impreso con la selección del votante, con lo que, cuando menos desde el punto de vista técnico, se cumple con el concepto de voto libre. El sistema desacopla o desliga adecuadamente al votante de su selección, lo que cumple con el concepto de voto secreto.

Es importante señalar que éste es uno de los puntos más difíciles de lograr en un sistema de voto electrónico pues paradójicamente, en algún punto del sistema se tiene la necesidad de identificar sin ambigüedad al votante, asegurando que la selección recibida proviene de su voluntad, para luego depositar el voto en una urna electrónica en la que quede completamente desligada la selección con respecto al votante que la hizo.

2.- Seguridad lógica de los servidores.

Se revisó en los dos centros de cómputo la protección de los servidores a través de la red y se encontró que están correctamente protegidos con un sistema de dos capas: la primera es un cortafuegos (*firewall*) que sólo permite el acceso a los puertos o servicios necesarios para le elección y por la otra, el servidor mismo tiene cerrados todos los accesos no necesarios y filtrados a través de reglas de origen aquellos necesarios para la coordinación de los servidores [13].

La comunicación relevante con los servidores se realiza correctamente cifrada de acuerdo al protocolo *SSL*, lo que impide que aun cuando pudiera ser interceptada en la red, el atacante pudiera conocer el contenido de la misma. Suplantar al votante sería también muy difícil en la red.

3.- Seguridad física

Ninguna medida de seguridad lógica es sólida si hay acceso físico a los servidores de la elección, por lo que los centros de cómputo en los que se ubican deben tener medidas de seguridad física. Se realizaron visitas a los dos centros de cómputo con los siguientes resultados [13].

- Centro de cómputo comercial externo a la UNAM
    - Cuenta con las medidas de seguridad mínimas necesarias operando adecuadamente. En adición tiene otras medidas, que no son estrictamente necesarias, cuya implementación es más bien de forma o apariencia y que, por razones comerciales, son comunes en este tipo de instalaciones.
- Centro de cómputo de la UNAM
    - Es un centro fuera de las instalaciones de los campus universitarios, lo que mejora su situación en cuanto al posible bloqueo por parte de grupos radicales.
    - Cuenta con pocas medidas de seguridad pues se advierte que su objetivo es más bien operativo, lo que no se consideró importante dado que hay otro centro de cómputo con mejores medidas de seguridad física.
    - Se recomendó, dada la premura de las elecciones, que se pusiera una cámara de seguridad adicional y un guardia a la entrada de ese centro de cómputo durante la jornada electoral.

4.- Recomendaciones de mediano plazo

Como en toda obra humana, hay oportunidades de mejora, que en este caso llevarían a un sistema más eficiente, robusto y seguro computacionalmente hablando. Las recomendaciones en este sentido son las siguientes:

- El sistema actual tiene un nivel grande de complejidad interna que redunda en el uso de más recursos computacionales de los estrictamente necesarios y una mayor dificultad para absorber los cambios que por razones externas hay que hacer de vez en cuando a cualquier sistema. Una simplificación importante del flujo del sistema y de los programas mismos sería bienvenida aunque no sea estrictamente necesaria para garantizar que se cumpla con los objetivos de elecciones universales, libres, directas y secretas. Una versión nueva desarrollada desde cero y tomando en cuenta las observaciones derivadas tanto de está auditoría como de la hecha en el 2006 sería bienvenida [13].
- Mejorar los sistemas de rastreo de posibles ataques sobre el sistema, incluyendo que operen en tiempo real, a fin de poder identificar y atrapar en el acto a cualquier persona que intente subvertir el sistema de elecciones.
- Mejorar los sistemas de seguridad física en las instalaciones universitarias aun cuando se trate sólo del centro de cómputo de respaldo

Veamos las imágenes del manual en línea de votación del sistema sve 3.5 (Fig. 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 1.10, 1.11, 1.12)

Figura 1.4: Página Web - SVE

Figura 1.5: Aceptar aviso legal - SVE

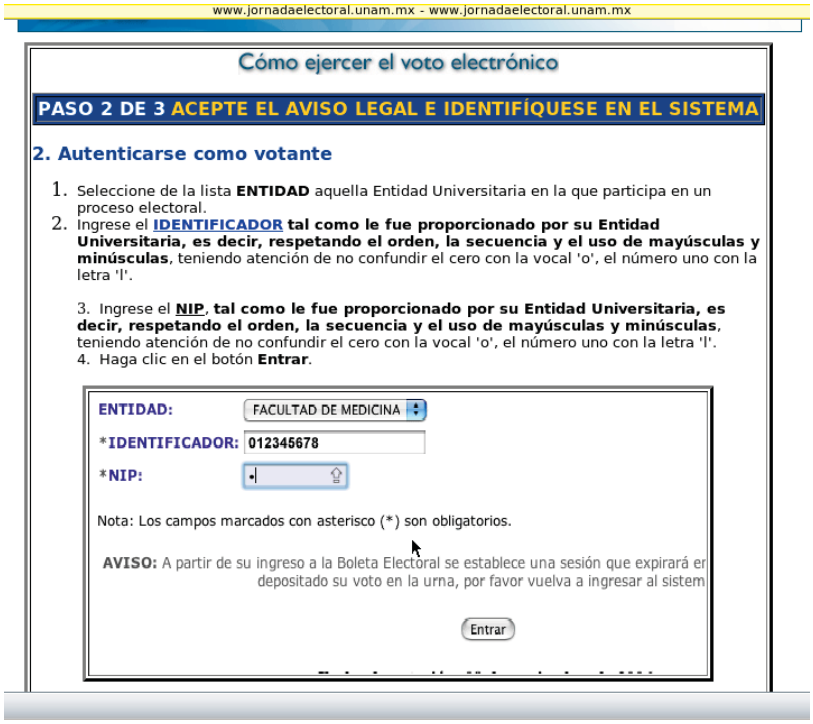


Figura 1.6: Autenticarse como votante - SVE

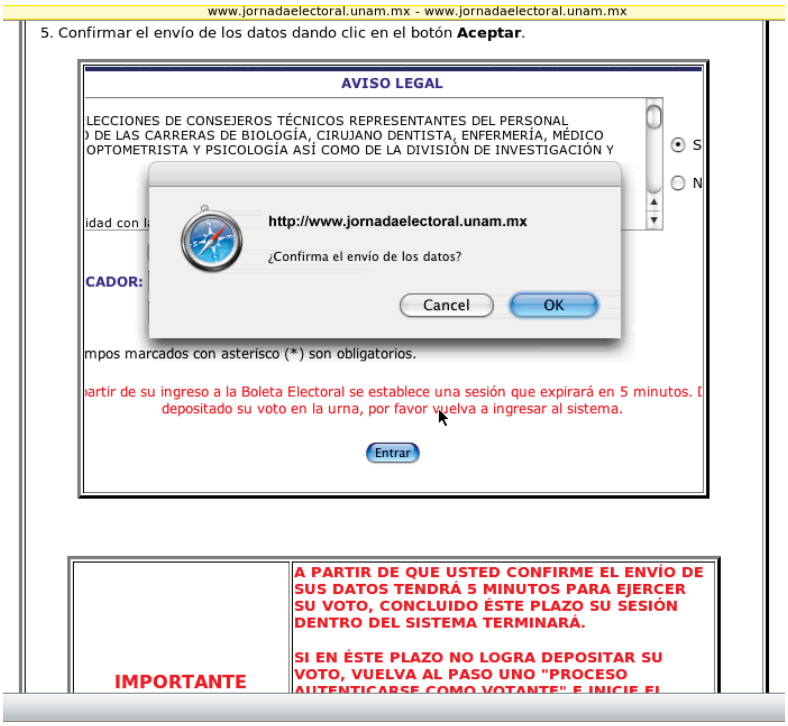


Figura 1.7: Aviso legal - SVE

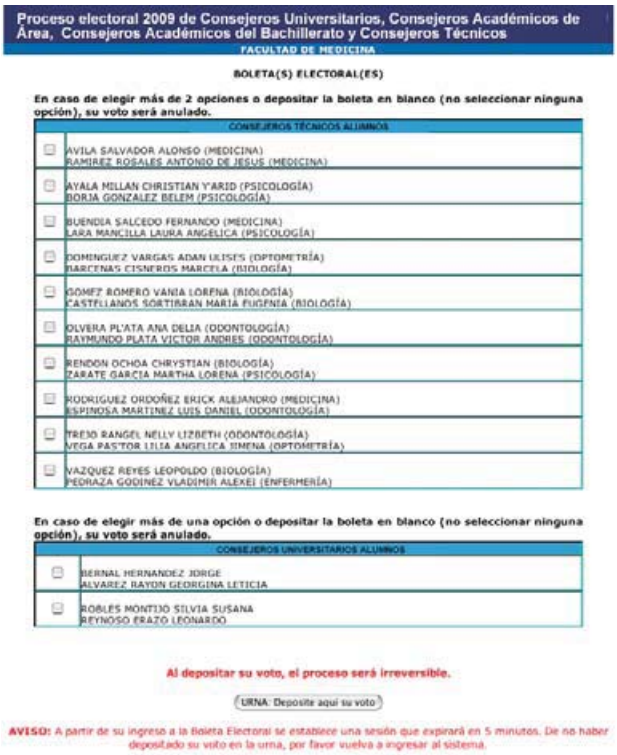


Figura 1.8: Boleta - SVE

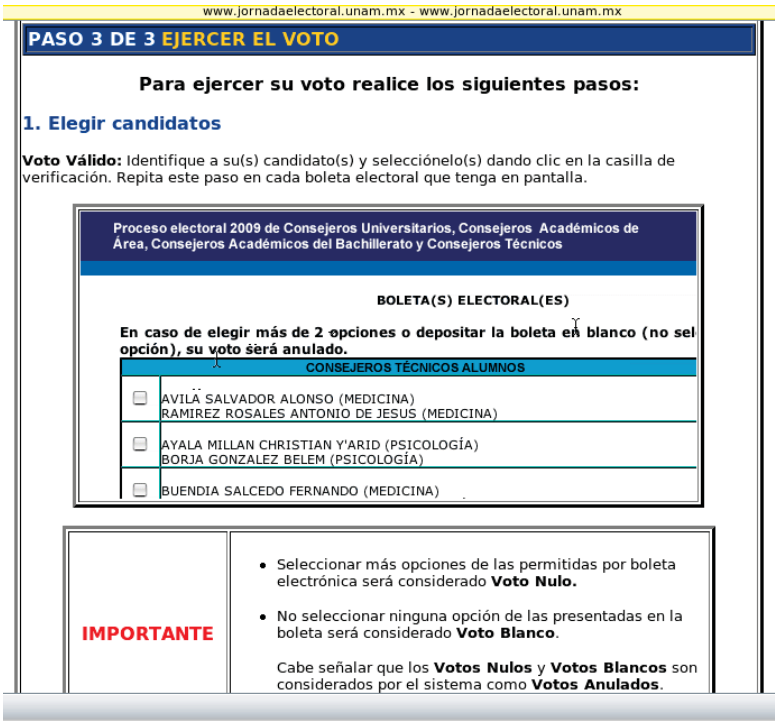


Figura 1.9: Elegir candidatos - SVE

Figura 1.10: Urna - SVE

Figura 1.11: Confirmar voto - SVE

Figura 1.12: Fin - SVE

1.3. Repositorios con información sobre votaciones electrónicas

En está sección veremos algunos repositorios importantes de información sobre votaciones electrónicas.

- **Proyecto ACE** [http://aceproject.org/]: La Red de conocimientos electorales ACE ofrece información completa y fidedigna sobre elecciones, promueve la vinculación entre especialistas involucrados en temas electorales y proporciona servicios para el desarrollo de capacidades institucionales [56].

ACE es una iniciativa conjunta de ocho instituciones asociadas que apoyan el Proyecto de Administración y Costo de Elecciones (ACE) bajo una perspectiva de largo plazo y son líderes en el campo de la provisión de asistencia técnica especializada en el campo electoral. El IFE es uno de los miembros de este proyecto [58].

- **notablesoftware** [http://www.notablesoftware.com/evote.html]: Es un sitio con mucha información sobre votaciones electrónicas, todo el material es de la Dra. Rebecca Mercuri, quien defendió su disertación doctoral "Electronic Vote Tabulation: Checks & Balances" en la Universidad de Pennsylvania [57].
- **Computational Social Choice** [http://staff.science.uva.nl/~ulle/COMSOC/]: En está página tenemos información sobre elección social computacional, el cual es un campo interdisciplinario de estudio entre la teoría de la elección social y ciencias de la computación. Le conciernen la aplicación de técnicas desarrolladas en ciencias de la computación, como son: el análisis de complejidad y diseño de algoritmos, para el estudio de mecanismos de elección social, como lo son algoritmos de procesos de votación o reparto equitativo. Es importante resaltar que se tienen congresos cada dos años [59].

1.4. Conclusiones

Se ha dado toda la información referente al tema de la tesis, sus objetivos, antecedentes, motivación y contribución; además hemos dado el *background* necesario para saber que es *Plone* y un *CMS*, lo cual es muy importante en la mayor parte de la tesis.

Los trabajos relacionados nos muestran que el sistema de votación del *IMATE* está muy avanzado en cuestiones de seguridad y configurabilidad, por lo menos en *CMS*. El sistema de la UNAM está pensado para un número mucho mayor de electores que el pensado para el sistema del *IMATE*. Por lo tanto en la UNAM contamos con dos sistemas de votación muy confiables cada uno diseñado para contextos bien definidos.

Por último damos unas referencias a repositorios que contienen información de primer nivel e interesante sobre las votaciones electrónicas.

Capítulo 2

SEGURIDAD EN VOTACIÓN ELECTRÓNICA Y PROTOCOLOS

La votación electrónica es cualquier forma de recolección de votos que involucre dispositivos electrónicos.

2.1. E-voting

El termino votación electrónica (o *e-voting*) es usado tanto para votar mediante el uso de máquinas de votación especialmente diseñadas en un entorno controlado y para votar vía Internet, la máquina será accedida en un entorno controlado o no controlado. Una máquina de votación de diseño especial consiste de hardware de computadora y software, y entonces los votantes manipulan una interfaz de usuario en una pantalla, teclado, botones u otros dispositivos, en lugar de emitir su voto mediante una boleta de papel. La tecnología de la Internet debería ser usada para e-voting en entornos controlados, y debería en dichos casos no ser accedida desde si misma, si no en entornos controlados, como una intranet.

La tecnología de *e-vote* puede incluir tarjetas perforadas, sistemas de escaneo óptico y kioscos de votación que incluyen a las máquinas de votación electrónica de registro directo (*DRE* por sus siglas en ingles). También se pueden transmitir las boletas y votos vía telefónica [2], redes de computadora privadas, o la Internet.

La tecnología de e-vote puede acelerar el conteo de boletas y puede proveer una mejor accesibilidad para votantes imposibilitados.

2.2. Objetivos de seguridad en cómputo

Algunos de los objetivos más importantes de la seguridad en el contexto de la seguridad en computación, son los siguientes: confidencialidad, integridad, y disponibilidad.

- La confidencialidad asegura que los *assets* relacionados con la computadora sean accedidos únicamente por las partes autorizadas. Con esto quiero decir que únicamente aquellos quienes deberían tener acceso para algo lo tengan. Por acceso me refiero no únicamente a leer, si no también a escribir, imprimir, o saber que un *asset* particular existe.

- Integridad significa que el asset puede ser modificado únicamente por partes autorizadas o únicamente por caminos autorizados. En este contexto, la modificación incluye escribir, modificar estado, eliminar, y crear.

- Disponibilidad significa que el asset es accesible por partes autorizadas a tiempos autorizados. Diciéndolo de otra forma, si una persona o sistema tiene acceso legítimo de un conjunto particular de objetos, ese acceso no debería de ser impedido.

La seguridad en computación se ocupa de estos tres objetivos. Uno de los retos de construir un sistema seguro es encontrar el balance correcto entre los objetivos, los cuales también están en conflicto. Por ejemplo, es sencillo preservar la confidencialidad de un objeto particular en un sistema seguro, simplemente no dejando que nadie lea tal objeto. De todas formas este sistema no es seguro, por que no tiene los requerimientos de disponibilidad para accesos apropiados. Esto es, debe haber un balance entre confidencialidad y disponibilidad.

2.2.1. Confidencialidad

Únicamente las personas autorizadas o sistemas pueden acceder a datos protegidos. Aunque asegurar confidencialidad puede ser difícil. Por ejemplo, ¿quien determina que personas o sistemas están autorizados para acceder al sistema? Definamos que quiere decir "acceder" a datos, ¿queremos decir que una parte no autorizada pueda acceder a un sólo bit? ¿A todo el conjunto? ¿Partes de datos?¿Puede alguien que está autorizado revelar los datos a partes no autorizadas?

Como vemos el problema no es muy complicado, la confidencialidad es la propiedad de seguridad que entendemos mejor debido a que su significado es más compacto que las otras dos. La entendemos bien por que podemos relacionar ejemplos de cómputo con aquellos de preservar la confidencialidad en el mundo real.

2.2.2. Integridad

La integridad es más difícil de definir. Basándome en lo escrito por [21, 15, 22], la integridad significa cosas diferentes en contextos diferentes. Cuando nosotros estudiamos la forma en la que algunas personas utilizan el término, encontramos muchos significados diferentes. Por ejemplo, si nosotros decimos que nosotros tenemos preservada la integridad de un ítem, nosotros tal vez queremos decir que el ítem es:

- preciso
- exacto
- no modificado
- modificado únicamente en caminos aceptables
- modificado únicamente por personas autorizadas
- modificado únicamente por procesos autorizados
- consistente
- internamente consistente
- utilizable

La integridad puede también significar dos o más de estas propiedades. Como lo dicen [21, 22] tres aspectos particulares de acciones de integridad, separación y protección de recursos, y detección y corrección de errores. La integridad puede ser forzada de la misma forma como la confidencialidad: por un control riguroso de quién o qué puede acceder a que recurso y en que formas. Algunas formas de integridad están bien representadas en el mundo real, y estas representaciones precisas pueden ser implementadas en un entorno computarizado. Pero no todas las representaciones de integridad son bien reflejadas por sus implementaciones en computadora.

2.2.3. Disponibilidad

La disponibilidad aplica tanto a datos y servicios, con esto quiero decir que afecta tanto a la información como al procesamiento de ella, y su complejidad es similar. Como con la noción de confidencialidad, diferentes personas esperan que la disponibilidad signifique diferentes cosas. Por ejemplo, un objeto o servicio se cree que es disponible si:

- Este es presentado en una forma utilizable.
- Este tiene capacidad suficiente para conocer las necesidades del servicio.
- Está haciendo un progreso claro, y si, en modo de espera, éste tiene un límite de tiempo de espera.
- El servicio es completado en un periodo aceptable de tiempo.

Nosotros podemos construir una descripción general de disponibilidad mediante la combinación de estos objetivos. Nosotros decimos que un ítem (recurso, objeto, unidad) de datos, servicio, o sistema está disponible si:

- Hay una oportuna respuesta a nuestra petición
- Los recursos son asignados equitativamente a fin de que algunas solicitudes no sean favorecidas en detrimento de los otros
- El servicio o sistema involucrado sigue una filosofía de tolerancia a fallos.
- El servicio o sistema puede ser usado de forma sencilla y en la forma que se supone debe ser usada.
- La concurrencia es controlada, esto es, accesos simultáneos, administración de *deadlocks*, y acceso exclusivo soportado.

Como podemos ver, las expectativas de disponibilidad son casi inalcanzables. La comunidad en seguridad está comenzando a entender que implica la disponibilidad, en especial en este trabajo de votación electrónica donde el cómputo es distribuido. Un simple control de acceso centralizado es fundamental para preservar la confidencialidad e integridad, pero no es claro que un simple punto de control de acceso pueda darnos disponibilidad. La mayoría de los éxitos en seguridad en el pasado se enfocaron en confidencialidad e integridad; pero la completa implementación de disponibilidad es el siguiente gran reto de la seguridad como lo afirma Schneier [17].

2.3. Objetivos de seguridad en e-voting

En está subsección se expondrán los objetivos de seguridad que se requieren para un sistema de votación electrónica. En específico los objetivos que mencionan Michael Ian Shamos [24] y Bruce Schneier [16].

2.3.1. Punto de vista: Shamos

Sugiere seis requerimientos del sistema. La función primaria de un sistema de votación electrónica es capturar la preferencia del elector de forma fiable y precisa. Las dos funciones son lógicamente separables pero deben ser realizadas por el mismo equipo. La captura involucra la interacción entre el elector humano y lo que se utilice para mostrar la boleta. Reportar (Escrutinio) se refiere a grabar, tabular, imprimir y auditar el total de votos. La captura es un problema de factor humano por lo que una solución mediocre puede resultar en confusión y perdida de confidencialidad en el proceso

electoral. El escrutinio es un problema para el cual una solución mediocre puede también resultar en confusión y pérdida de confidencialidad en el proceso electoral.

Los sistemas de votación electrónica son una fuente de preocupación debido a que realizan su trabajo en microcircuitos no accesible a examinación y no dejan un registro tangible de lo que han hecho. Por aquellos años de principio de los 1990 un día entero de votaciones podía producir no más de un cartucho pequeño con los resultados de la votación grabados en la memoria electrónica. Lo que va dentro de estas máquinas es un misterio para el público y si yo vote, ¿como sé que ellos lo programaron correctamente? ¿Pudo algún pirata informático (*hacker*) manipular los votos? ¿Quién aprovo está votación electrónica?

Después de estas ideas [24], pasemos a la lista en orden decreciente de importancia:

- Debe mantenerse el voto del elector como un secreto inviolable.
- Debe permitirse que cada elector sólo vote una vez, y sólo por los candidatos a los que está autorizado sufragar.
- No debe permitirse la modificación (tampering) mediante el sistema de votación, ni por el cambio de oro (algún otro recurso) por votos.
- Deben reportarse todos los votos de forma exacta.
- El sistema de votación debe mantenerse operable a lo largo de cada elección.
- Se debe mantener una auditoría para detectar pecados en contra de las encomiendas 2-4, pero la auditoría no debe violar la encomienda 1.

2.3.2. Punto de vista: Schneier

Un sistema de votación tiene cuatro características requeridas:

1. Exactitud (Accuracy): El objetivo de cualquier sistema de votación es establecer el intento de cada elector, y traducir estos intentos en un conteo final de votos. Excentar a un sistema de votación que falle en esto, no es deseable. Esta característica también incluye seguridad: Debe ser imposible cambiar el voto de otro elector, las boletas, destruir los votos, u otra cosa que afecte el conteo final.
2. Anonimato: Que las boletas sean secretas es fundamental para la democracia, y los sistemas de votación deben estar diseñados para facilitar el anonimato del elector.
3. Escalabilidad: Los sistemas de votación necesitan ser capaces de manejar elecciones sumamente largas. Cerca de 372 millones de personas votaron en mayo de 2004 en la India. La complejidad de la elección es otra característica. A diferencia de muchos países en donde las elecciones nacionales permiten sólo un voto por un candidato o partido, en los EE.UU. un elector se enfrenta con docenas de decisiones de elecciones individuales: nacionales, locales, y todo lo que hay de forma intermedia.
4. Velocidad: Los sistemas de votación deben producir resultados rápido. Esto es particularmente importante en EE.UU., donde la gente espera saber los resultados el mismo día de la elección antes de dormir (en cualquier país, diría yo).

2.4. Protocolos

2.4.1. ¿Que es un protocolo?

Un protocolo es una serie de pasos a seguir, que involucran dos o más partes, diseñados para realizar una tarea. Tenemos que poner énfasis en la definición ya que "una serie de pasos" significa que

el protocolo tiene una secuencia, que va del comienzo al final. Cada paso debe ser ejecutado en el turno que le toque, no antes no después, y no debe existir un paso que se ejecute antes de que termine el paso anterior. Lo que quiero decir con que se involucran dos o más partes es que al menos dos personas son requeridas para completar el protocolo, ya que una sola persona no hace un protocolo. Una persona sola puede realizar una serie de pasos para completar una tarea (ejemplo: Hacer un helado), pero eso no es un protocolo (si otra persona se comiera el helado entonces ya seria un protocolo). Por último, lo que quiero decir con diseñados para realizar una tarea significa que el protocolo debe conseguir algo. Algo que luce como un protocolo pero no completa una tarea no es un protocolo.. es una pérdida de tiempo [17].

Listo otras características de un protocolo:

- Cualquiera que este involucrado en el protocolo debe conocer el protocolo y todos los pasos para seguirlos con antelación
- Cualquiera que este involucrado en el protocolo debe estar de acuerdo en seguirlo.
- El protocolo no debe ser ambiguo; cada paso debe estar bien definido y no debe de haber oportunidad de un mal entendido.
- El protocolo debe ser completo; debe de haber una acción específica para cada posible situación.

2.4.2. ¿Para que sirven?

En la vida diaria hay protocolos informales para casi todo: ordenar cosas por teléfono, jugar dominó, votar en una elección. Nadie piensa mucho acerca de estos protocolos; ellos han evolucionado a través del tiempo, todos saben como usarlos, y funcionan razonablemente bien.

Cada vez más interacciones humanas se dan a través de una red de computadoras en vez de hacerlo de forma presencial (cara a cara). Las computadoras necesitan protocolos formales para hacer las mismas cosas que la gente hace sin pensar. Si por ejemplo alguien se muda a un estado diferente del al república y encuentra un kiosco de votación que luce completamente diferente de los que él solía utilizar, esa persona podría adaptarse rápidamente. Las computadoras no son ni cercanamente tan flexibles.

La mayoría de los protocolos de tipo cara a cara están basados en la presencia de las personas para asegurar equidad y seguridad. Por ejemplo, ¿enviarías a un extraño que va pasando a comprar dulces y dándole una bolsa llena de dinero?,¿enviarías tu boleta que tiene tu voto secreto al gobierno sin tener la certeza de anonimato?

Como vemos sería tonto asumir que la gente que utiliza las redes de computadora para ser más específico la Internet sean honestas, aunque hay gente que cree que las personas deberían de ser honestas y que los que diseñamos protocolos no deberíamos tener problema con la seguridad, pero esa idea si es que se aplica a algún país no es el caso de México, ya sea por connotaciones socio-políticas, económicas, etc. Eso es tema de otra tesis.

Hablando de diseñadores de protocolos; también es tonto asumir que los diseñadores de computadoras, sistemas, protocolos, etc. son honestos. Con que se tenga un sólo diseñador deshonesto tenemos para echar todo abajo. Todo esto suponiendo que no hay errores humanos no intencionados.

Mediante la formalización de protocolos, podemos examinar la manera en que las partes deshonestas pueden subvertirlos. Entonces se pueden desarrollar protocolos que no puedan ser trastocados mediante dicha subversión.

Los protocolos abstraen el proceso de completar una tarea del mecanismo por el cual la tarea es realizada. Ya que el protocolo es el mismo aun si es implementado en diferentes dispositivos. Esto nos permite examinar el protocolo sin meterse en detalles con la implementación.

2.4.3. Protocolo arbitrado

Un árbitro es una tercera parte confiable y desinteresada dentro del contexto de un protocolo para poder completarlo. No tiene un interés particular en el protocolo ni por ninguna de las personas involucradas. Con la palabra confiable quiero decir que toda la gente involucrada en el protocolo acepte lo que el árbitro dice que es verdad, los árbitros ayudan a completar protocolos entre dos grupos en los que no hay confianza.

Los protocolos arbitrados son divididos en dos subprotocolos de bajo nivel esto debido a un alto costo de contratación de árbitros. Uno es un subprotocolo no arbitrado, ejecutado cada vez que los participantes quieren completar el protocolo. El otro es un subprotocolo arbitrado, que se ejecuta únicamente en circunstancias excepcionales, como por ejemplo una disputa. Este tipo especial de árbitro es llamado un adjudicador.

Un adjudicador es también es una parte confiable y desinteresada dentro del contexto de un protocolo, no está involucrado directamente en cada protocolo. El adjudicador es llamado para determinar si un protocolo fue realizado correctamente. Un ejemplo son los jueces, este nunca ve el contrato entre dos personas hasta que uno de ellos manda a juicio al otro.

Este protocolo parecido a un contrato mediante firma puede ser formalizado como sigue:

Subprotocolo no arbitrado (se ejecuta cada vez)

1. Alicia y Bob negocian los términos del contrato
2. Alicia firma el contrato
3. Bob firma el contrato

Subprotocolo adjudicado (se ejecuta una sola vez en caso de disputa)

1. Alicia y Bob aparecen ante el juez
2. Alicia presenta su evidencia
3. Bob presenta su evidencia
4. El juez trabaja en la evidencia entregada

En un buen protocolo adjudicado, el adjudicador puede determinar si alguien está haciendo trampa y además la identidad del tramposo. En lugar de prevenir trampas los protocolos adjudicados detectan la trampa.

2.4.4. Protocolo *Self-Enforcing*

Un protocolo *self-enforcing* es el mejor tipo de protocolo. El protocolo por si mismo garantiza la propiedad de equidad. No se requiere un árbitro para llevar acabo el protocolo. No se necesita un adjudicador para resolver disputas. El protocolo es construido tal que no existan las disputas. Si alguna de las partes intenta hacer trampa, la otra parte inmediatamente detecta la trampa y el protocolo termina sin éxito. Lo que se acaba de describir nos sugiere que todos los protocolos deberían ser *self-enforcing*, pero desafortunadamente no hay un protocolo *self-enforcing* para cada situación.

2.5. Ataques contra protocolos

Los ataques a protocolos pueden ser dirigidos en contra de los algoritmos criptográficos usados en el protocolo (función *hash*), en contra de las técnicas criptográficas usadas para implementar los algoritmos y protocolos, o en contra de los protocolos mismos.

Las personas pueden probar varias formas para atacar un protocolo. Alguien que no este participando en el protocolo pude escuchar parte o todo el protocolo si lo desea. Este es llamado un ataque

pasivo, ya que el atacante no afecta el protocolo. Todo lo que puede hacer el atacante es observar el protocolo y tratar de ganar información. El tipo de ataque corresponde al ataque de texto cifrado.

Debido a que los ataques pasivos son difíciles de detectar, los protocolos intentan prevenir los ataques pasivos en vez de detectarlos.

Otro tipo de ataque seria que el atacante trate de alterar el protocolo para su propia ventaja. El atacante puede pretender ser alguien más, introducir nuevos mensajes en el protocolo, eliminar mensajes existentes, sustituir un mensaje por otro, retransmitir mensajes, interrumpir un canal de comunicaciones, o alterar información almacenada en una computadora. Estos son llamados ataques activos, por que ellos requieren intervención activa. La forma de estos ataques depende en la red.

Los atacantes pasivos intentan ganar información acerca de los actores dentro del protocolo. Ellos recolectan mensajes que se envían entre las partes involucradas e intentan hacer un criptoanálisis sobre ellos. Los ataques activos, por otro lado, pueden tener muchos más objetivos. El atacante puede estar interesado en obtener información, degradando el rendimiento, corrompiendo información existente, o ganando acceso no autorizado para los recursos.

Una vulnerabilidad es una debilidad en el sistema de seguridad, por ejemplo, en procedimientos, diseño, o implementación, que debe ser explotada para causar perdida o daño. Ejemplifiquemos lo dicho anteriormente con lo siguiente: un sistema puede ser vulnerable a manipulación de datos no autorizada por que el sistema no verifica la identidad de un usuario antes de permitirle el acceso a los datos.

Una amenaza a un sistema de cómputo es un conjunto de circunstancias que tienen el potencial para causar perdida o daño. Para poder ver la diferencia entre una amenaza y una vulnerabilidad consideremos lo siguiente. Un puente vehicular y debajo de las personas que diariamente pasan por ahí; ahora bien, tenemos que el puente tiene una gran fisura, la cual no es detectada por las razones que ustedes quieran, la vulnerabilidad es la fisura y la amenaza es los autos que pueden caer, por mencionar sólo una. Mientras la fisura siga intacta la amenaza hacia las personas no será realizada.

Hay muchas amenazas para un sistema de computadora, incluyendo las humanas y por computadora. Todos hemos vivido o escuchado de los resultados de errores humanos inadvertidos, fallas en el diseño del hardware, y fallas de software. Pero no se necesita ser tan perspicaz para saber que los desastres naturales también son amenazas; estas pueden tirar un sistema cuando el cuarto de las computadoras donde reside el sistema se inunda o el centro de cómputo se colapsa por un terremoto, por mencionar unos ejemplos.

Una persona que explota una vulnerabilidad perpreta un ataque en el sistema. Un ataque pude también ser lanzado por otro sistema, como cuando un sistema envía una desmesurada cantidad de mensajes a otro, haciendo que el sistema pierda su habilidad de ejecutar otros procesos. Como bien sabemos estos ataques son comunes, como los ataques de denegación de servicio llenando a los servidores con más mensajes de los que pueden manejar.

Como manejar estos problemas es una pregunta primaria en el ámbito de la seguridad, Pfleger define lo siguiente: utilizar un control como una medida de protección. Un control es una acción, dispositivo, procedimiento, o técnica que remueve o reduce una vulnerabilidad. Una amenaza es bloqueada por un control de una vulnerabilidad.

Para que podamos idear controles, se necesita que conozcamos lo más posible acerca de amenazas. Por algo, varios de los *hackers* reconocidos han recibido propuestas de trabajo de gobiernos o empresas para ocuparse de la seguridad. Podemos definir cualquier amenaza como parte de uno de los siguientes cuatro tipos (Fig. 2.1): intercepción, interrupción, modificación, y fabricación. Cada amenaza explota vulnerabilidades de los recursos importantes (*assets*) de los sistemas de cómputo.

- Una intercepción se refiere a que alguna parte no autorizada ha obtenido acceso a un asset. La parte externa puede ser una persona, un programa, o un sistema de cómputo. Ejemplos de estos tipos de falla son copias ilícitas de programas o archivos de datos, o escuchar mediante un *sniffer* para obtener datos en la red. Aunque una pérdida pueda ser detectada muy rápido, una intercepción silenciosa puede no dejar rastro, por lo cual la intercepción no puede ser fácilmente detectada.

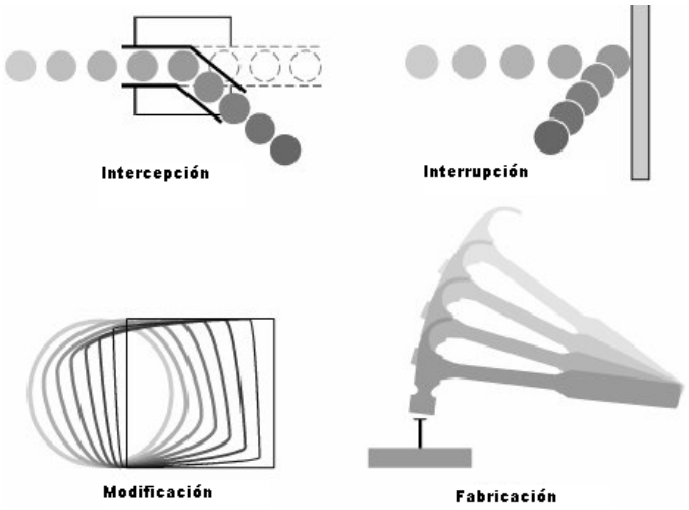


Figura 2.1: Amenazas

- En una interrupción, un *asset* del sistema se vuelve no disponible, pérdido, o no
- utilizable. Un ejemplo es la destrucción maliciosa de un dispositivo de hardware, la eliminación de un programa del dispositivo del almacenamiento o un archivo de datos, o el mal funcionamiento de un sistema de archivos de un sistema operativo tal que este no pueda encontrar un archivo de un disco en particular.
- Si una parte no autorizada accede y también modifica un asset, la amenaza es una modificación. Por ejemplo, alguien podría cambiar los valores en una base de datos, alterar un programa tal que este realice cómputo adicional, o modifique datos que son transmitidos electrónicamente. También es posible modificar el hardware. Algunos casos de modificación pueden ser detectados con medidas simples o algunas veces con mediadas más elaboradas, los cambios pueden ser casi imposibles de detectar.
- Finalmente, una parte no autorizada podría crear una fábrica de objetos falsificados en un sistema de cómputo. El intruso tal vez inserte transacciones espurias en un sistema de red de comunicación o añada registros en una base de datos existente. Algunas de estas adiciones pueden ser detectadas como falsificadas, pero si son hechas de forma profesional, ellas serán virtualmente no distinguibles de las reales.

Como podemos observar estas cuatro clases de amenazas de intercepción, modificación, interrupción y fabricación describen el tipo de problemas que nosotros podemos encontrar.

2.6. Conclusiones

Sentamos las bases de conocimiento referente a los protocolos de seguridad y los sistemas de votación electrónica. El protocolo de seguridad debería cumplir con los puntos mencionados en las secciones 2.3.1 y 2.3.2, un análisis más completo sobre el protocolo de seguridad del sistema tratado en está tesis puede encontrarse en [23] y en mi capitulo 3.

Capítulo 3

CASOS DE ESTUDIO: Sistema de votación *KOA* e *IMATE*

Se explicará el protocolo de seguridad del sistema de votación electrónica *KOA*, definiendo sus propiedades las cuales se deben de cumplir en el protocolo, veremos como es que se cumplen dichas propiedades dentro del sistema *KOA*. Se mostrará también el plugin *DutchTallySystem* el cual es un demo de una parte del proceso del sistema *KOA*, en especial vemos como se cargan los padrones y como se realiza el conteo. Está sección es muy importante, por qué el sistema de votaciones del *IMATE* esta basado en varias características del sistema *KOA* y de su protocolo de votaciones electrónicas.

La segunda parte de este capítulo esta dedicada al sistema de votaciones electrónicas del Instituto de Matemáticas de la *UNAM*, se va a detallar cada una de las fases del sistema, los actores que intervienen en el proceso electoral y el protocolo de votación también será abarcado. Este sistema es el que rediseñe para *Plone* 3. La primer versión de este sistema fue hecha por Alexander Zapata [23].

3.1. *KOA*

3.1.1. Definición de Propiedades de Seguridad de *KOA*

El sistema, junto con medidas organizacionales y de procedimiento, puede proveer un servicio de votación que cumpla con las siguientes propiedades:

Voto secreto: Sea imposible enlazar a un votante con un voto válido, con lo anterior aseguramos la confidencialidad del voto.

Unicidad: Cada uno de lo votantes válidos puede votar únicamente una sola vez y su voto será contado exactamente una vez en el resultado final.

Votantes válidos: Únicamente los votantes quienes tengan el derecho legal de votar deberían ser permitidos para votar.

Integridad: El resultado final de la boleta no puede ser influenciado de ninguna forma que no sea el de generar un voto válido.

Recuento: Conforme a las demandas constitucionales un recuento es posible (por ejemplo: la elección electoral en México en Julio del 2006).

Disponibilidad: Los votantes legales deberían poder votar con su boleta cuando lo quieran y donde decidan. Las sugerencias de la comisión Europea con respecto a la disponibilidad de los sitios Web del gobierno y de contenidos que tengan que ser tomados en consideración, y la resolución del Parlamento Europeo que concierne a la disponibilidad de sitios Web [20];

Transparencia para el votante: El votante debería entender y confiar en el proceso de votación.

3.1.2. Módulos del sistema

El sistema *KOA* está formado por varios componentes los cuales realizan funciones específicas e importantes dentro del proceso de votación.

Enseguida se muestra el diagrama y se define brevemente la función de cada módulo (Fig. 3.1).

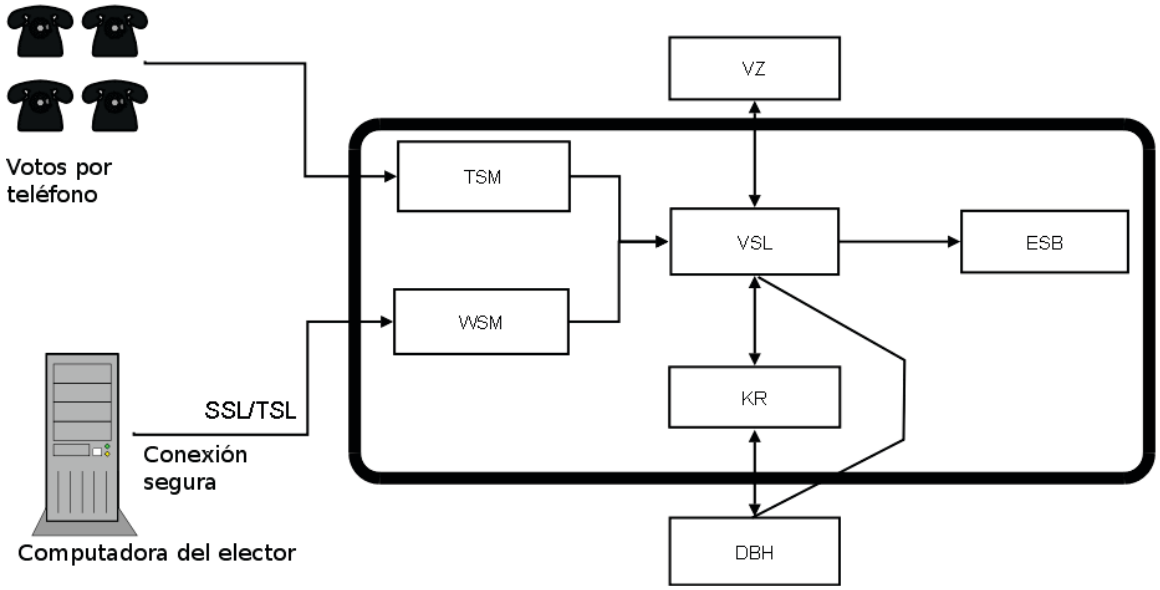


Figura 3.1: Componentes KOA

3.1.2.1. Componente VSL

Estación Virtual (VSL): Realiza la función de ser el control central del sistema *KOA*. Un VSL puede servir a múltiples círculos de votantes. Varias máquinas de votación pueden estar conectadas a un VSL, esto asegura que los diferentes módulos del sistema puedan realizarse como son: votar vía Internet, vía telefónica y posiblemente en otros sistemas, como es el caso del módulo para votación en dispositivos móviles.

3.1.2.2. Componente ESB

Caja de votación electrónica (ESB): Cada círculo de votación tiene un ESB; todos los votos generados son cifrados y almacenados en el ESB.

3.1.2.3. Componente KR

Registro de votantes (KR): Antes de comenzar las elecciones toda la información anónima de los votantes legales es almacenada en el KR. Además el KR registra si un votante ya genero su voto. sólo hay un KR, aún si el Experimento (una ejecución del sistema *KOA*) llegue a extenderse con múltiples estaciones virtuales.

3.1.2.4. Componente WSM

Máquina de votación electrónica (WSM). Este componente realiza el enlace entre la PC, el votante quien necesita votar vía Internet y la estación virtual.

3.1.2.5. Componente TSM

Máquina de votación telefónica (TSM). Este componente realiza el enlace entre el teléfono, el votante que desea votar por teléfono y la estación virtual.

3.1.2.6. Componente DBH

Base de datos (*datasource*) con la información necesaria para la elección, que no sea anónima.

3.1.3. Protocolo de *KOA*

3.1.3.1. Inicializar

El KR recibe del cliente una lista de votantes validados legalmente para ejercer su voto. El KR le reenvía la lista al cliente para propósitos de auditoría.

Esta lista de votantes incluye datos como son:

- Identificación anónima única
- Número de distrito al que pertenece
- Número de círculo de votantes (como padrón electoral)
- El sistema genera un código por cada votante de la lista que recibió del cliente. Este código del votante se genera mediante el algoritmo SHA1PRNG y cumpliendo la prueba-11 del número de seguro social Holandés (***RandomGenerator.java***).
- Un error en un dígito del código nunca resulta en un código de votante válido. El sistema le envía la lista con los códigos que generó al cliente.
- Además el votante escoge un password el cual se cifra mediante SHA-1.
- El KR recibe del cliente una lista de candidatos la cual incluye datos como son: nombre del candidato, iniciales, número de distrito y número de círculo de votantes (***Kandidaat.java***).
- El sistema genera por cada candidato una lista de códigos únicos y generados mediante el mismo mecanismo de los códigos de votante. El administrador del proceso electoral debe generar un par de llaves de tamaño 512 con RSA y cifrado MD5-DES (***GenerateKeyPair.java***).

3.1.3.2. Verificar votante válido

Aquí el sistema checa si el código del votante es parte del registro, y si el número de acceso coincide (password); si todos estos requerimientos son positivos el sistema checa si el votante ya emitió su voto o no (***KRSessionEJBBean.java***).

Para verificar el password, este es cifrado con la misma función (hash) que se mencionó antes y es comparado con el ya almacenado.

3.1.3.3. Verificar candidato válido

Este procedimiento es de ayuda para que una máquina de votación se asegure de que un código de candidato es válido para un votante específico.

Lo que se pide es que el código del candidato este contenido en la lista de candidatos que corresponde al círculo de votantes de dicho votante específico.

3.1.3.4. Votar

La pregunta consiste de un código de votante y código de acceso, junto con la elección del código de candidato del votante. El VSL checa la misma información que durante la verificación; si el votante está permitido para votar, la elección que eligió se cifra y se almacena en el ESB.

Los datos que son cifrados y almacenados son los siguientes:

- Código de candidato
- Nombre de candidato
- Iniciales del candidato
- Número de posición
- Número de distrito
- Número de lista de candidatos
- Número de circulo de votantes
- Modalidad mediante la cual se efectuó el voto (vía Web o vía telefónica)
- Y un número aleatorio llamado STEMNUMMER (***ESBSessionEJBBean.java***[1491]).
  - STEMNUMMER="148672708870144"
  - STEM="00000040A4A6FEFD1A178A57630364
  - 3A7BDA28705E2990E58977873A54E5000614E
  - 98636F84FDFBD22C81872AF4B27963F229F27
  - 47491EBE33D74A22DA4D74E52E53F41B0000
  - 002876BF1596933751A9AC5249745D941D100F
  - EAA559D9CACE8359917217723DF8F33F59E6DC4D17547D“

Entonces el estado del votante cambia, para evitar que vuelva a emitir voto. Un código de transacción es generado y es guardado en el KR y enviado al votante como confirmación de que su voto se emitió exitosamente (***StemprocesSession EJBBean.java***[748]).

Este código de transacción es un número aleatorio que es generado de la misma manera que un código de votante.

Los votos son cifrados cuando son almacenados usando la llave pública del administrador. Esto garantiza que los votos pueden abrirse únicamente por el administrador (llave privada). Datos aleatorios son añadidos a los votos cuando son cifrados, esto asegura que los votos con el mismo distrito y el mismo candidato tienen diferentes resultados en el cifrado.

3.1.3.5. Contar votos

El ESB recibe la llave primaria del administrador del proceso electoral (vía VSL). Esta llave es usada para descifrar los votos en un orden aleatorio, esto es para hacer el enlace de votos en cuanto a su orden no posible).

Los votos descifrados son almacenados en un archivo con formato de conteo especial. El módulo VSL usa este archivo para contar los votos y enviar el resultado de las elecciones al administrador.

Ejemplos de votos descifrados (***koa_ out/decrypt.txt***) son los siguientes:

- 772266165;van Zwëmmen;Z.S.G.Í;10;null;8;Partijvan de "sport";12
- 216504168;Azuur;W.F.;1;null;1;EuropeseKleurenpartij;12

- 761349042;Zonnegeel;O.M.;16;null;1;EuropeseKleurenpartij;12
- 684709442;Dennengroen;D.D.;25;null;1;EuropeseKleurenpartij;12
- 901329526;[Blanco];-;1;null;99;[Blanco];12

Dado que los votos son exportados en forma cifrada, un sistema externo puede descifrar y contar los votos. Entonces el hash de ese archivo puede ser comparado con el hash del archivo que genere el sistema y poder auditar con veracidad.

3.1.4. *DustchTallySystemEjemplo*

Ahora mostraré la ejecución del plugin *DustchTallySystem,* el cual es un demo de una parte del proceso del sistema *KOA*, en especial vemos como se cargan los padrones y como se realiza el conteo.

Restart: Esta opción nos permite comenzar una sesión de nueva cuenta (Fig. 3.2y 3.3). La información cargada en el directorio koa_out no es borrada. Esto ocurre únicamente con la opción clear. La función restart siempre está activa.

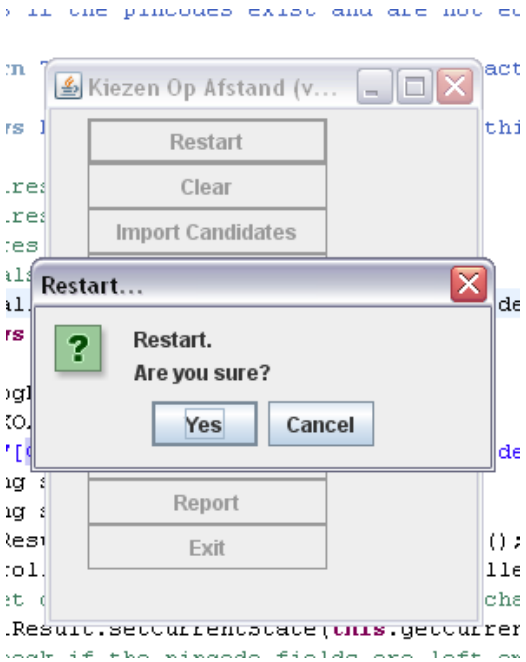


Figura 3.2: Restart

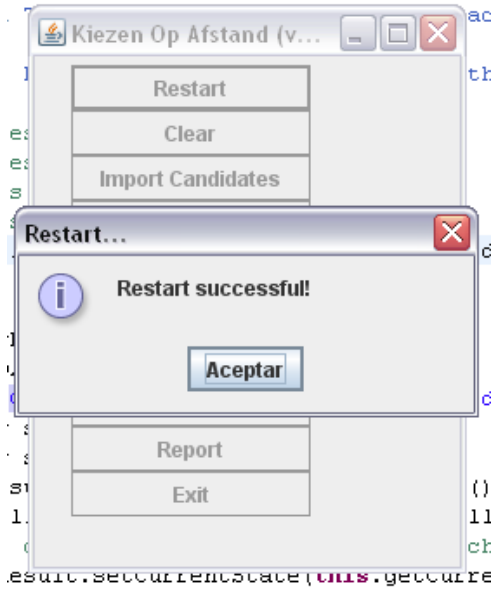


Figura 3.3: Confirmación de Restart

Clear: Esta función tiene acceso a la memoria interna asignada al programa y por lo tanto puede remover reportes creados que fueron erróneos. Para realmente eliminar los datos el usuario debe confirmar mediante un mensaje de dialogo. En el mensaje de dialogo aparecerán los reportes creados en el disco duro (Fig. 3.4 y 3.5).

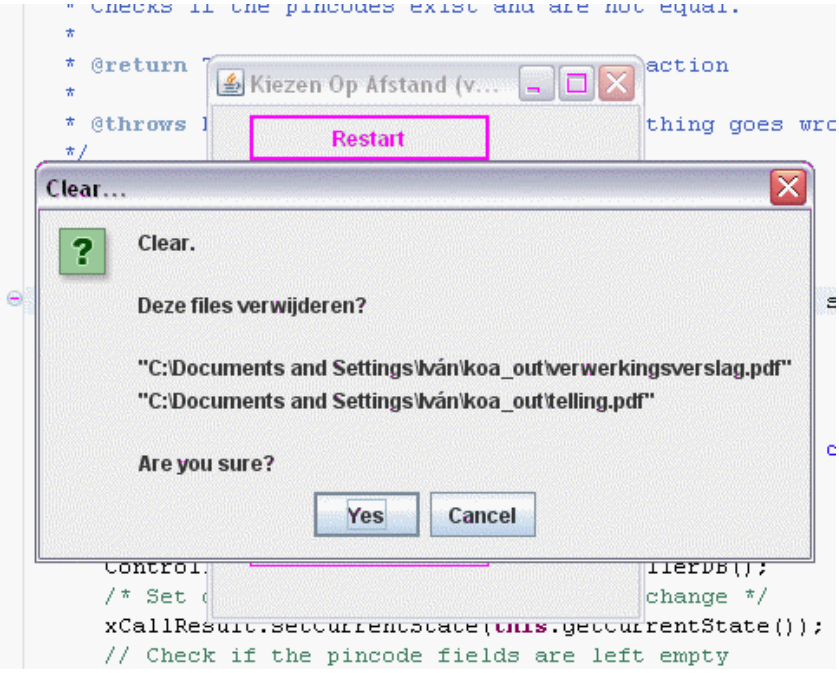


Figura 3.4: Clear

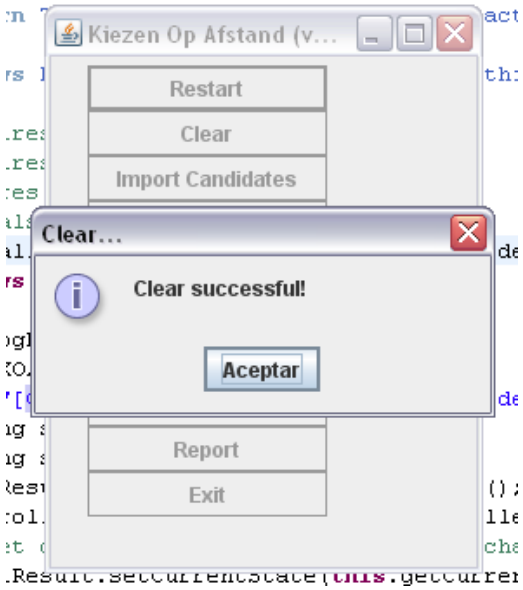


Figura 3.5: Confirmación de Clear

CandidateImport: Al ejecutar está función un explorador de archivos es mostrado al usuario, donde se filtran los archivos en formato XML (Fig. 3.6). Después de haber elegido el archivo. Dependiendo del tamaño, una barra mostrara el progreso de carga (Fig. 3.7). Al finalizar mostrara la lista, y opciones para aceptar o cancelar (Fig. 3.8, 3.9).

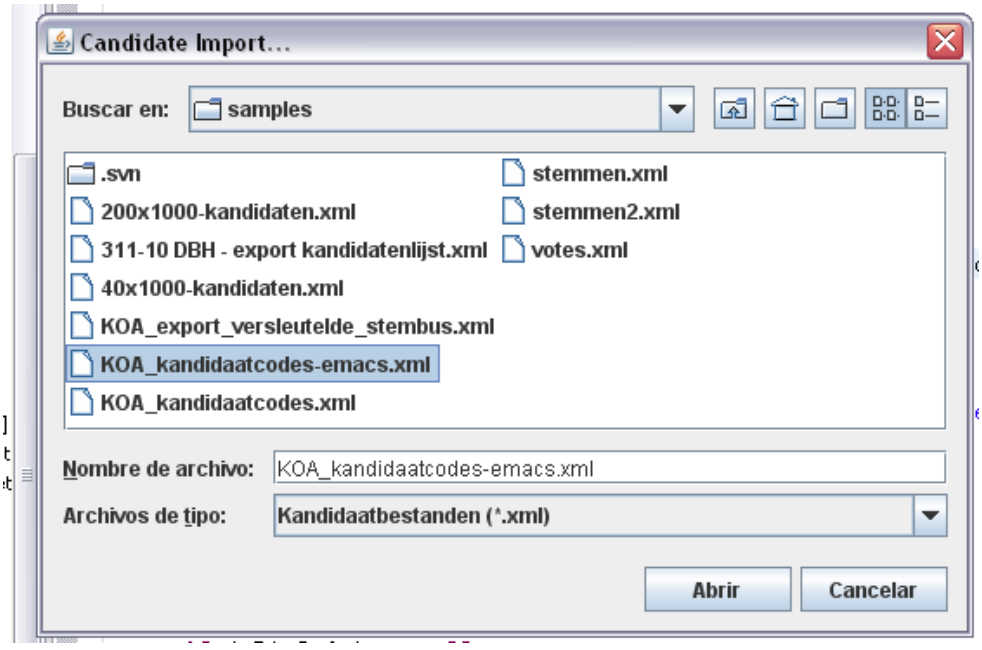


Figura 3.6: Candidate: Selección de archivo

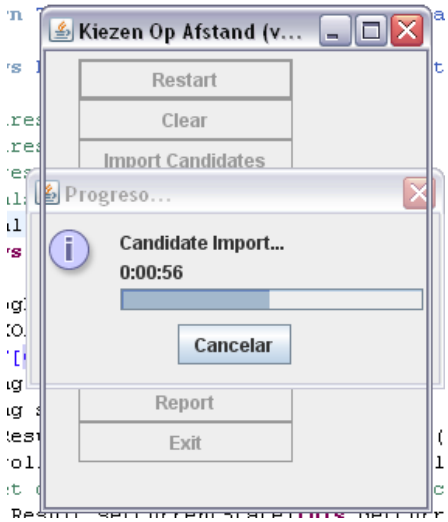


Figura 3.7: Candidate: Progreso de importación

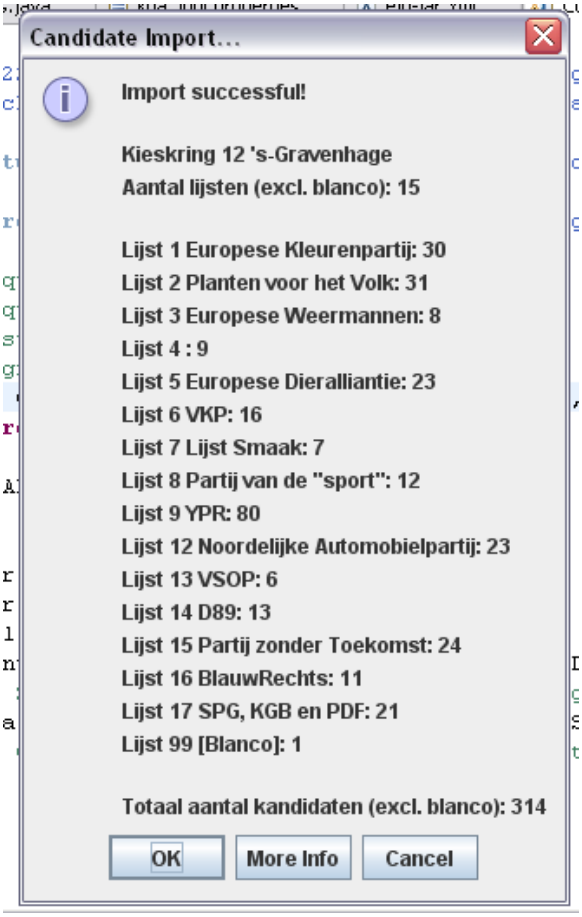


Figura 3.8: Candidate: Confirmación de importación

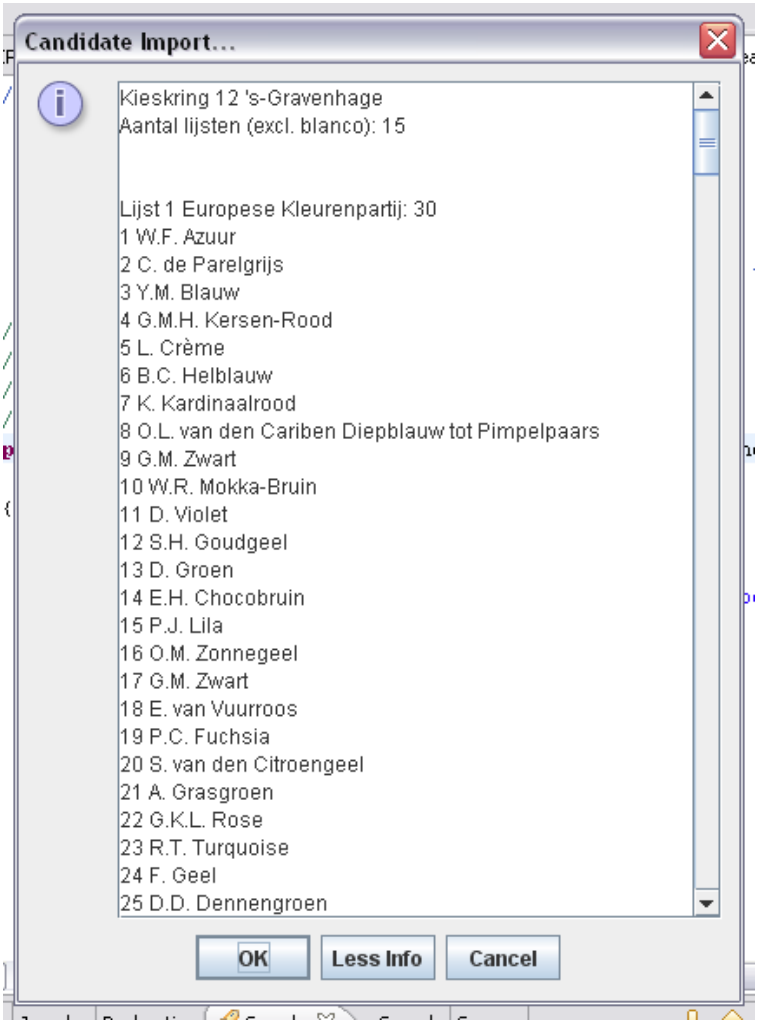


Figura 3.9: Candidate: Información detallada de candidato

Importvotes: Un explorador de archivos es mostrado una vez más (Fig. 3.10). El filtro de archivos sigue siendo XML, si archivo que se escogió tiene un formato incorrecto el sistema lo hará saber (Fig. 3.12), por el contrario un formato válido muestra el número de votos (Fig. 3.11).

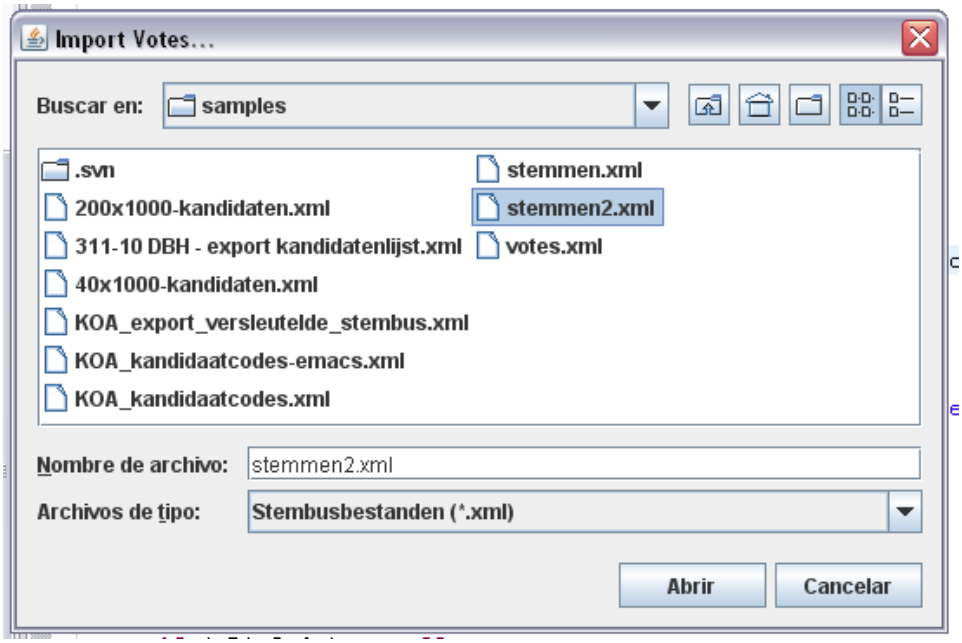


Figura 3.10: Importvotes: Selección de archivo

Figura 3.11: Importvotes: Confirmación de importación

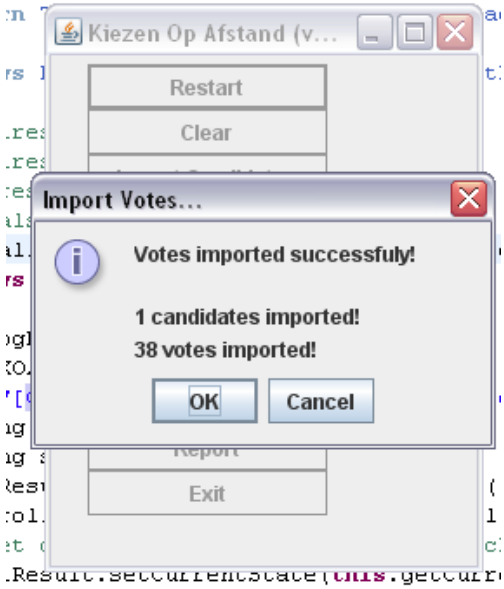


Figura 3.12: Importvotes: Cuadro de Error de Importación

PrivateKeyImport: Se muestra un explorador de archivos que filtra las llaves (Fig. 3.13). Después de seleccionar la llave privada, se le pregunta al usuario la palabra clave (Fig. 3.14). El usuario obtiene un mensaje de si la llave se logro importar con éxito o no (Fig. 3.15).

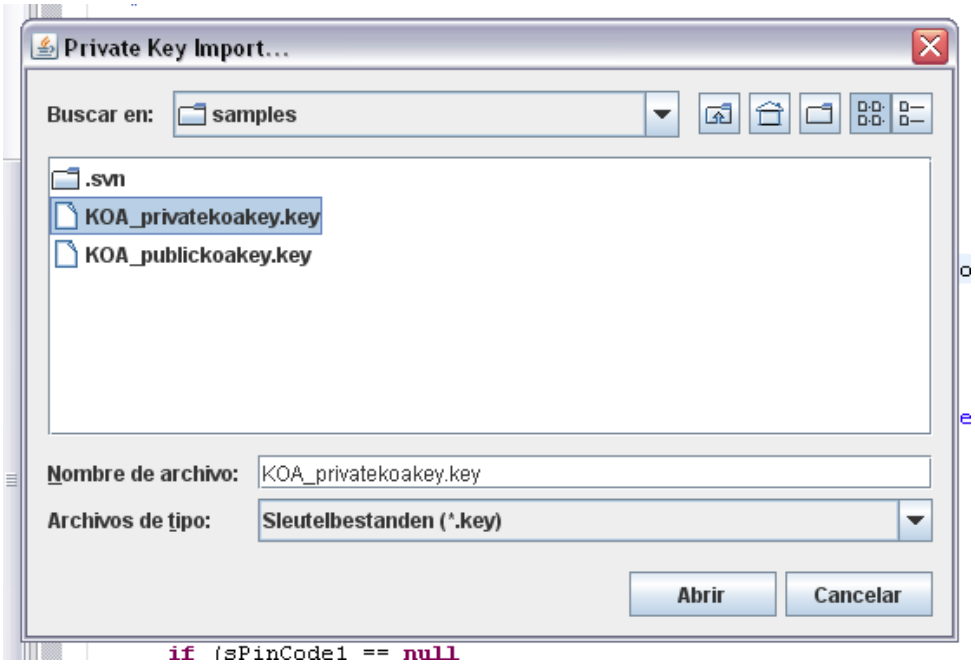


Figura 3.13: PrivateKeyImport: Selección de archivo

Figura 3.14: PrivateKeyImport: Introducción de contraseña

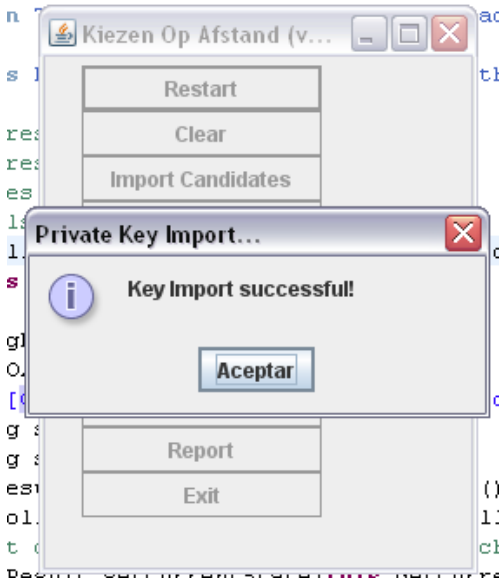


Figura 3.15: PrivateKeyImport: Confirmación de Importación

PublicKeyImport: Se sigue el mismo procedimiento que con la llave privada (Fig. 3.16). Aunque internamente se realiza una tarea extra, se realiza una prueba de que en realidad es una llave pública válida para la llave privada dada (Fig. 3.17).

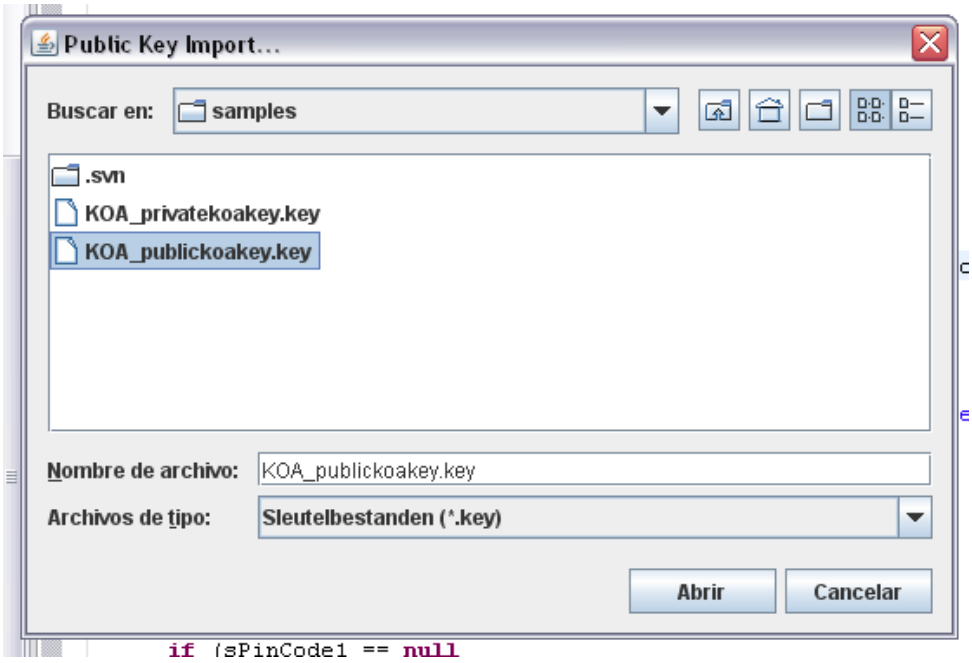


Figura 3.16: PublicKeyImport: Selección de archivo

Figura 3.17: PublicKeyImport: Introducción de contraseña

Decrypt: Esta función utiliza las llaves cargadas en el sistema para leer los votos . En el mensaje de dialogo el número de votos es mostrado (Fig. 3.18). También nos muestra si hubo errores en algunos votos (Fig. 3.19). Esta acción no puede ser cancelada, a lo más puede ser anulada y esto significa que la siguiente vez que ejecutemos descifrar los votos previos serán contados directamente. Un archivo llama decrypted.txt es creado con los votos descifrados.

Figura 3.18: Decrypt: Confirmación de Operación Exitosa

Figura 3.19: Decrypt: Reporte de Errores

Count: Esta función se encarga de contar los votos. En esto intervienen los códigos del candidato. Se checa que el círculo de votantes sea correcto como ya se explico anteriormente. Después de calcular esto el número de votos es mostrado (Fig. 3.20). Una vez que está función ha sido aceptada no puede ser cancelada. Hasta que termine de contar los votos (Fig. 3.21).

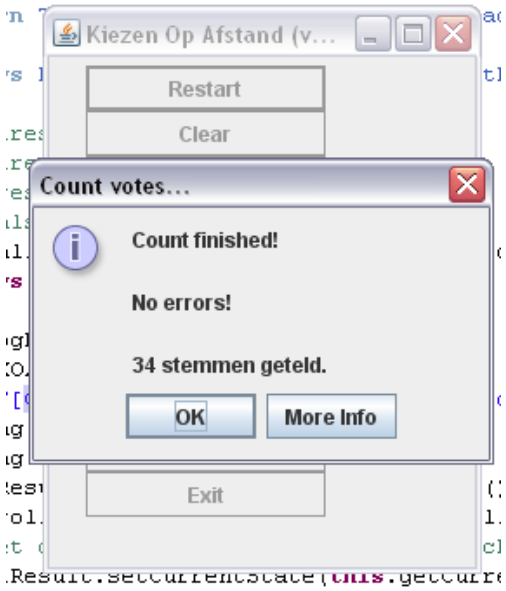


Figura 3.20: Count: Confirmación de Operación Exitosa

Figura 3.21: Count: Reporte

Report: Se encarga de crear reportes en el disco duro (Fig. 3.22, 3.23). Hay dos tipos de reportes: el que se crea mediante el periodo de la elección (por defecto está el periodo de la elección Europea) (Fig. 3.24) y el de fecha actual a ese momento (Fig. 3.25).

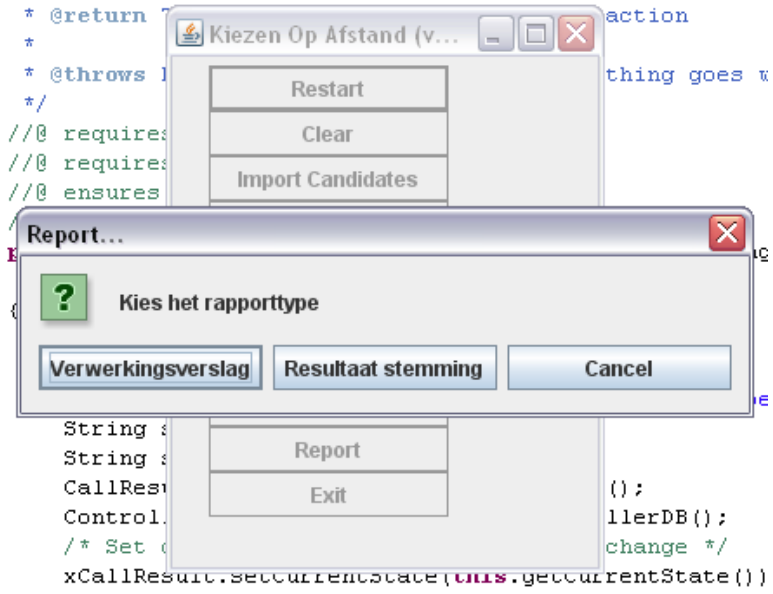


Figura 3.22: Report: Creación de reporte

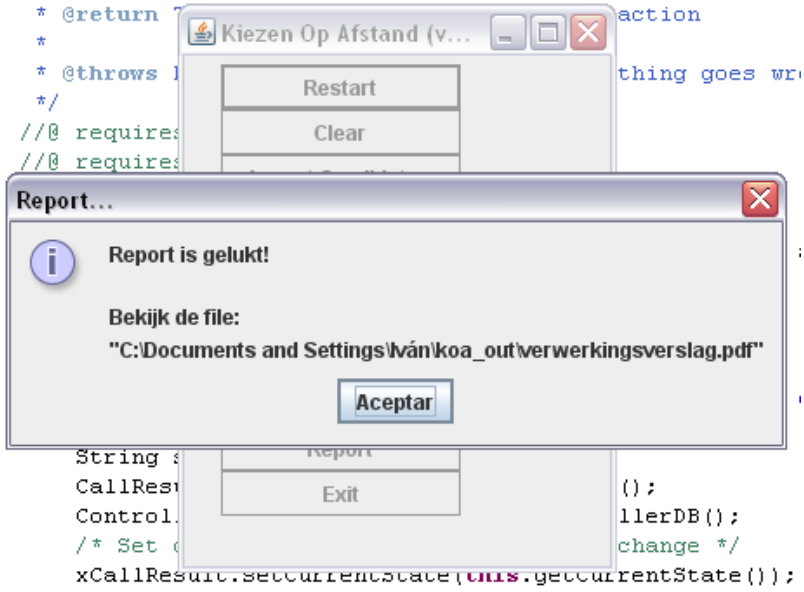


Figura 3.23: Report: Confirmación de Creación de Reporte

Figura 3.24: Report: Introducción de periodo de votación por fechas

Figura 3.25: Report: Introducción de periodo de votación en días

3.1.5. Conclusiones *KOA*

El sistema *KOA* es demasiado grande en su número de clases, lo que hace su entendimiento un tanto difícil [1], además el sistema necesita una configuración elaborada y no se cuenta con todos los recursos como para iniciar una prueba seria. Pongamos un ejemplo, el módulo que genera el par de llaves criptográficas no funciona correctamente. Para poder ejecutarlo tuve que programar algunas cosas y recompilar. Esto lo comente directamente con el líder del proyecto *KOA* que es Joseph Kiniry [62].

Otro ejemplo sería la falta de un script para crear la base de datos *KOA01* (*datasource*) con la cual no se puede realizar casi nada de importancia. Aunque realice ingeniería inversa para generar el script no fue suficiente.

Además tenemos lo siguiente:

- Unicidad se cumple por que las transacciones son atómicas.
- Sólo votantes válidos pueden votar gracias al código de votante generado.

- La integridad de la boleta se mantiene gracias al hash.
- El recuento puede ser hecho por algún programa externo.
- La disponibilidad viene a ser una de las propiedades difíciles de mantener.
- Las complicaciones en el sistema deben de ser solucionadas, en cuanto al tamaño del sistema pienso que es posible reducirlo.

3.2. Sistema de Votaciones Electrónicas del *IMATE*

3.2.1. Introducción

Se van a describir las características particulares del proceso de votación electrónica que se lleva en el *IMATE*, las propiedades de los productos *Plone* que lo conforman y la descripción del protocolo de votaciones. Esta sección está basada en el trabajo de tesis [23], quien desarrollo la primer versión del sistema de votaciones del *IMATE*.

3.2.2. Características del sistema

Los roles de los usuarios que intervienen en el proceso, las fases del proceso, las bitácoras o archivos históricos manejados y los diferentes estados del proceso que pueden visualizar los usuarios del sistema, se describen a continuación.

Enseguida voy a mostrar los casos de uso de cada unos de los actores que intervienen en el sistema de votaciones (Fig. 3.26, 3.27, 3.28, 3.29), para después explicar de forma concisa lo que realiza cada actor [63].

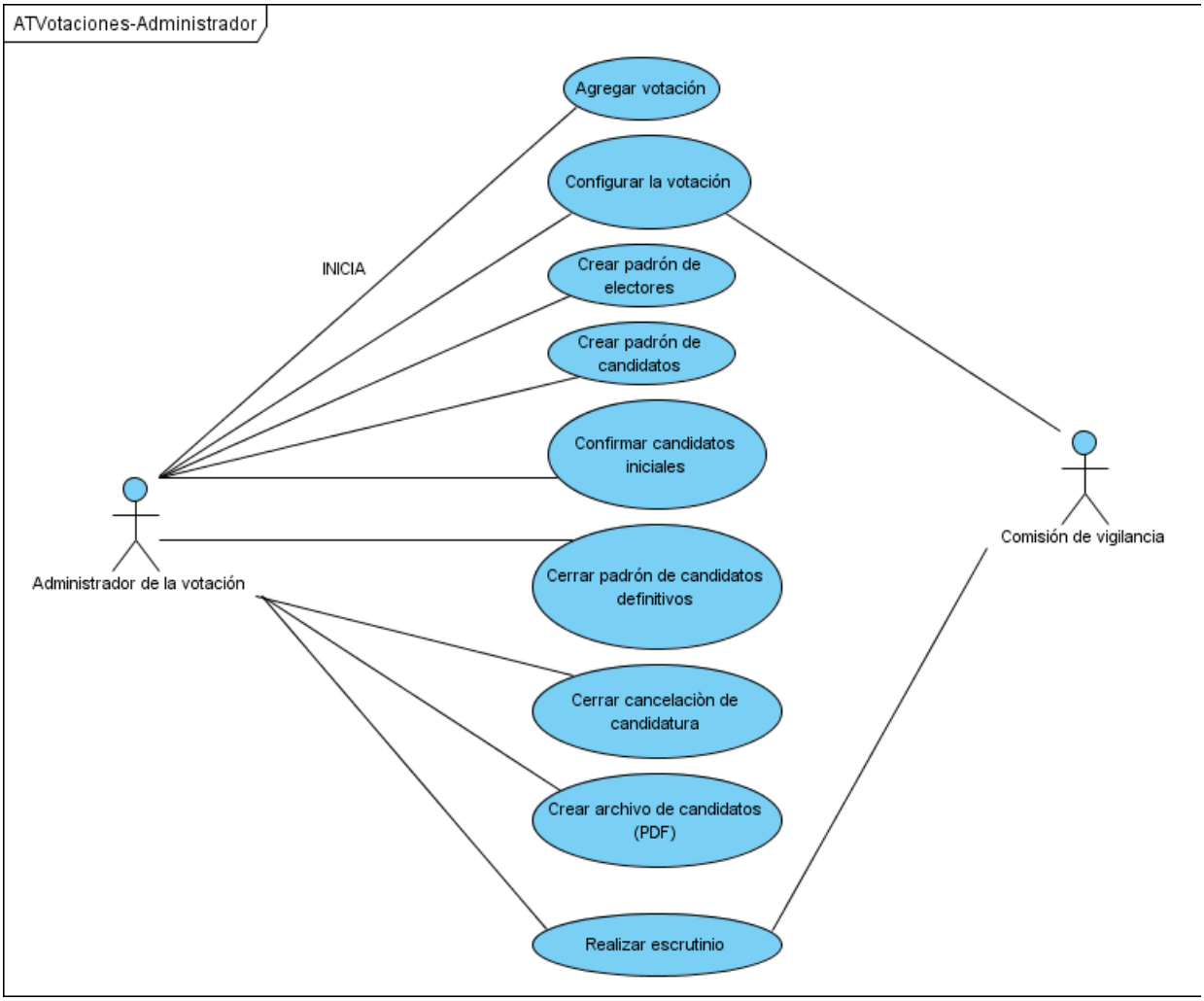


Figura 3.26: Caso de uso - Administrador

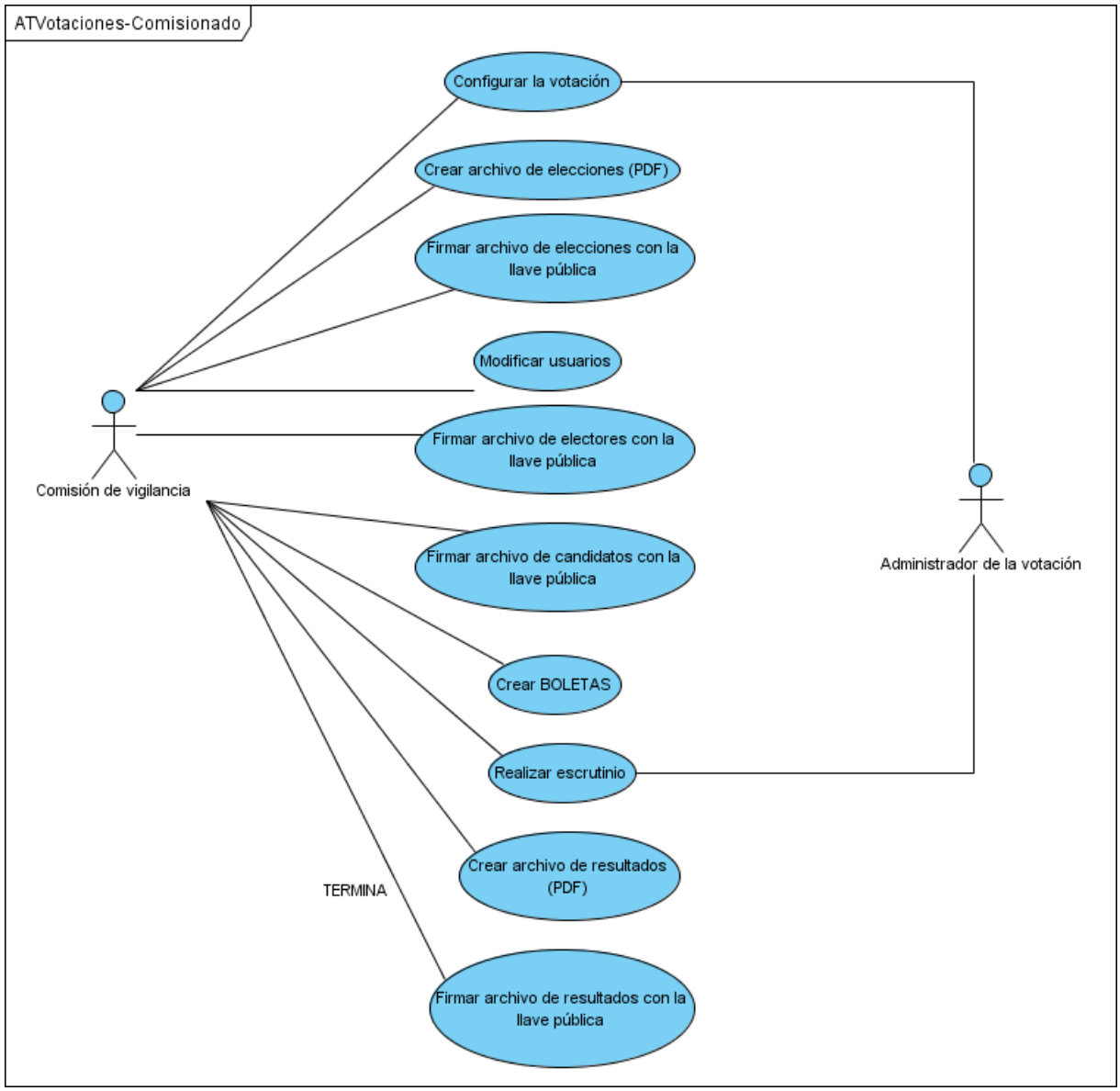


Figura 3.27: Caso de uso - Comisión

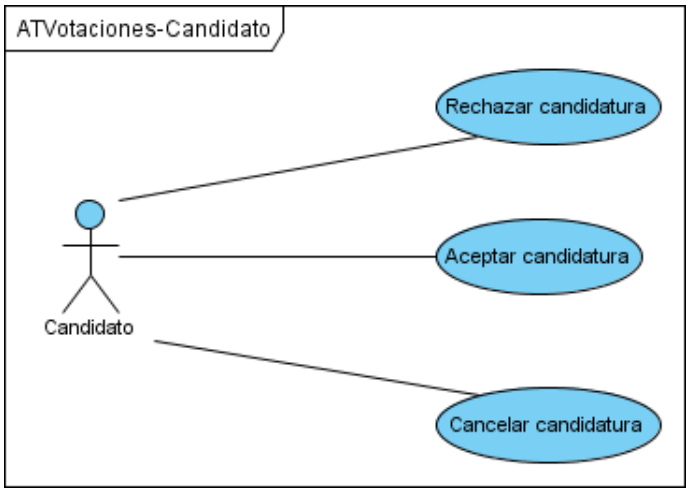


Figura 3.28: Caso de uso - Candidato

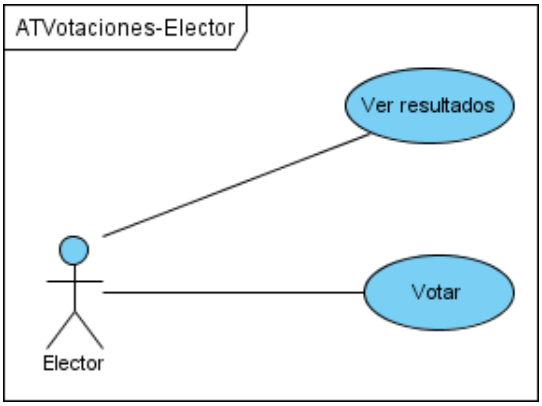


Figura 3.29: Caso de uso - Elector

3.2.2.1. Tipos de usuarios

En el proceso se manejan los siguientes tipos de usuarios:

- Administrador de la votación (Fig. 3.26): corresponde al usuario que agrega el objeto tipo votación, configura todos los parámetros y fechas claves de la misma que sólo pueden ser confirmados por la comisión de vigilancia, registra su clave pública, crea el evento y el foro de preguntas y respuestas de la votación, genera los padrones de electores y candidatos de acuerdo con la convocatoria de la elección pero estos sólo pueden ser confirmados y modificados después de su publicación por la comisión de vigilancia; también puede consultar las bitácoras y apoyar parte del conteo de los votos en lo que respecta a la primera de dos rondas de descifrado de votos de la "urna".

- Comisión de Vigilancia (Fig. 3.27): es un tipo de usuario que le da mayor confiabilidad al proceso, pues es en forma independiente al administrador, asegura que los parámetros de la votación están de acuerdo con la convocatoria de la misma antes de ser confirmados, registra su clave pública, puede consultar las bitácoras, firma todos los archivos claves de la votación que incluyen el archivo de configuración de la misma, y el padrón definitivo de electores y candidatos. Finalmente realiza

el proceso de conteo de votos, descifrando la tabla de códigos aleatorios de candidatos, realizando la segunda ronda de descifrado de votos de la **urna**, y generando y firmando el documento de resultados definitivos.

- Electores / Candidatos (Fig. 3.29, 3.28): corresponde a aquellos que accederán a la votación creada por el administrador ejerciendo el rol de elector o candidato, por lo que podrá verificar si es elegible como candidato o elector y hacer reclamo de caso de alguna inconformidad, rechazar/aceptar/cancelar una candidatura que se le haya propuesto, en el caso de aceptar candidatura registrar su logo y url donde publicará información más detallada, podrá consultar el estado del proceso de votación en cualquier momento, si es elegible como elector podrá ejercer su derecho de voto y verificar los resultados finales de la elección.

3.2.2.2. Fases

En el diagrama de bloques que se presenta a continuación se especifican las fases generales del proceso de votación (Fig. 3.30).

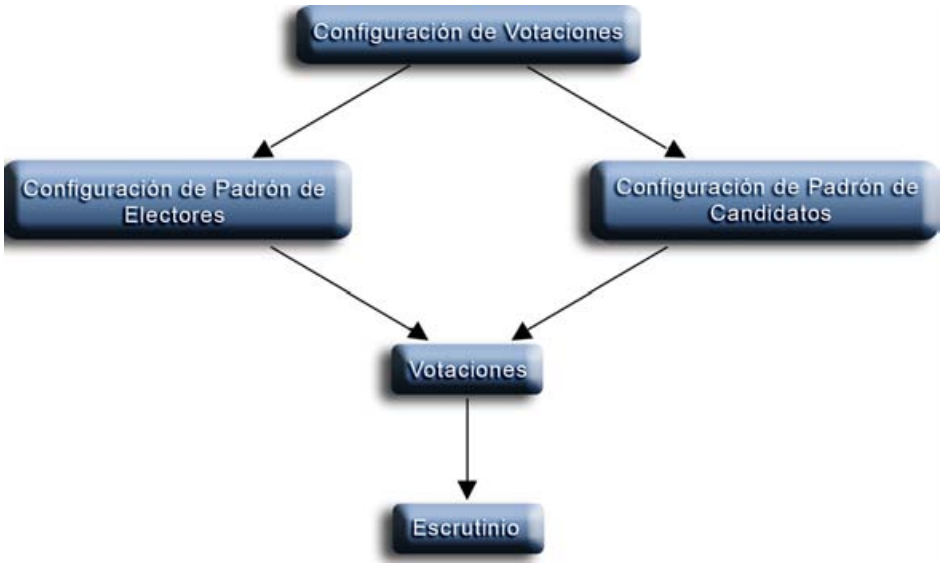


Figura 3.30: Fases

A continuación se presentan los diagramas de cada una las fases principales del proceso, en cada uno se especifica quién participa en las mismas (Administrador, Comisión, Electores y Candidatos) y la plataforma sobre la que están implementadas, así: las actividades realizadas en *Plone* se representan con un rectángulo y las que se soportan externamente en *GnuPG* con una elipse. De otro las actividades realizadas por el administrador o la comisión de vigilancia que usan funciones criptográficas se representan con una elipse, e incluyen un comentario que explica el tipo de clave o función utilizada. Finalmente se colorean con relleno las actividades realizadas internamente por el sistema *Plone* y que usan funciones criptográficas (Fig 3.31, 3.32).

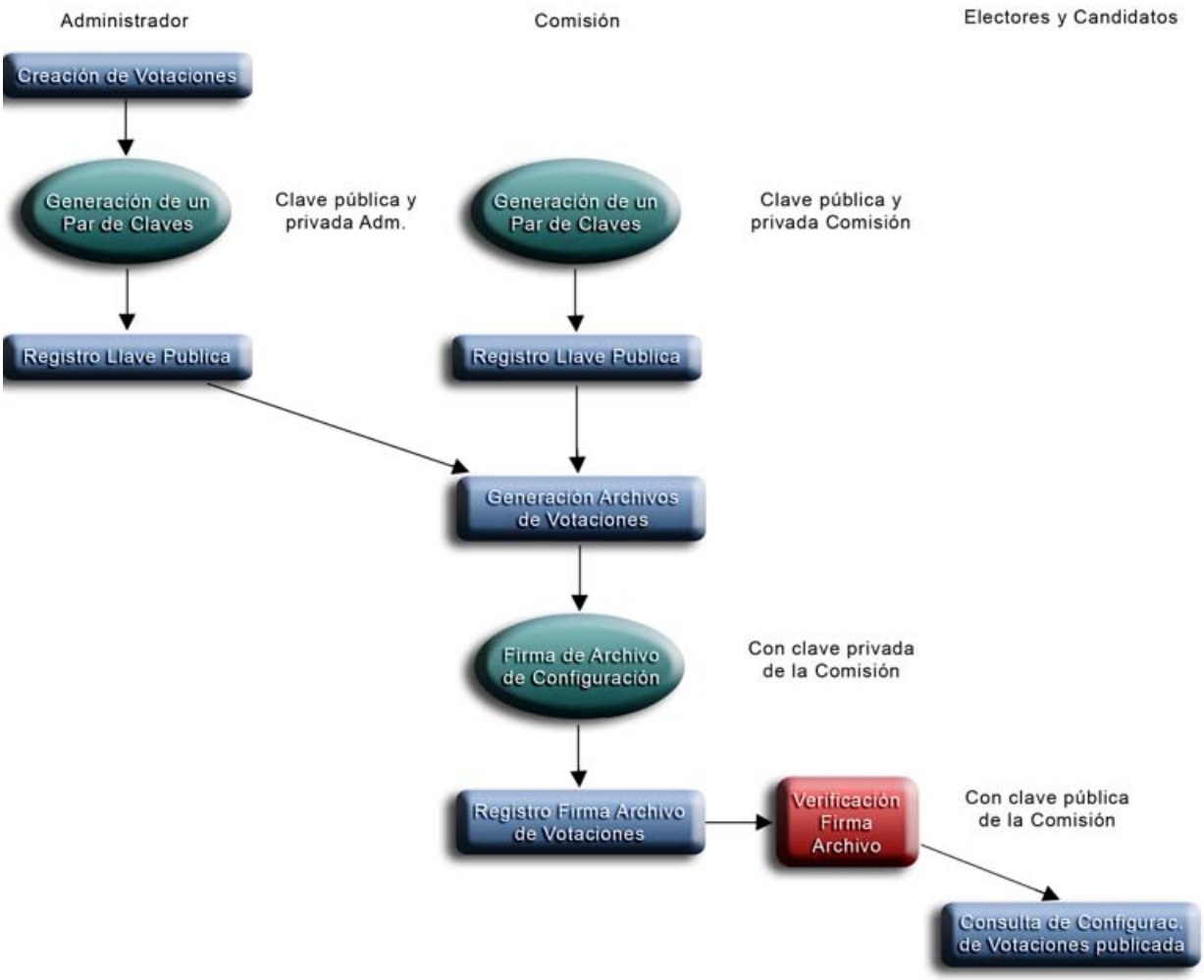


Figura 3.31: Protocolo IMATE

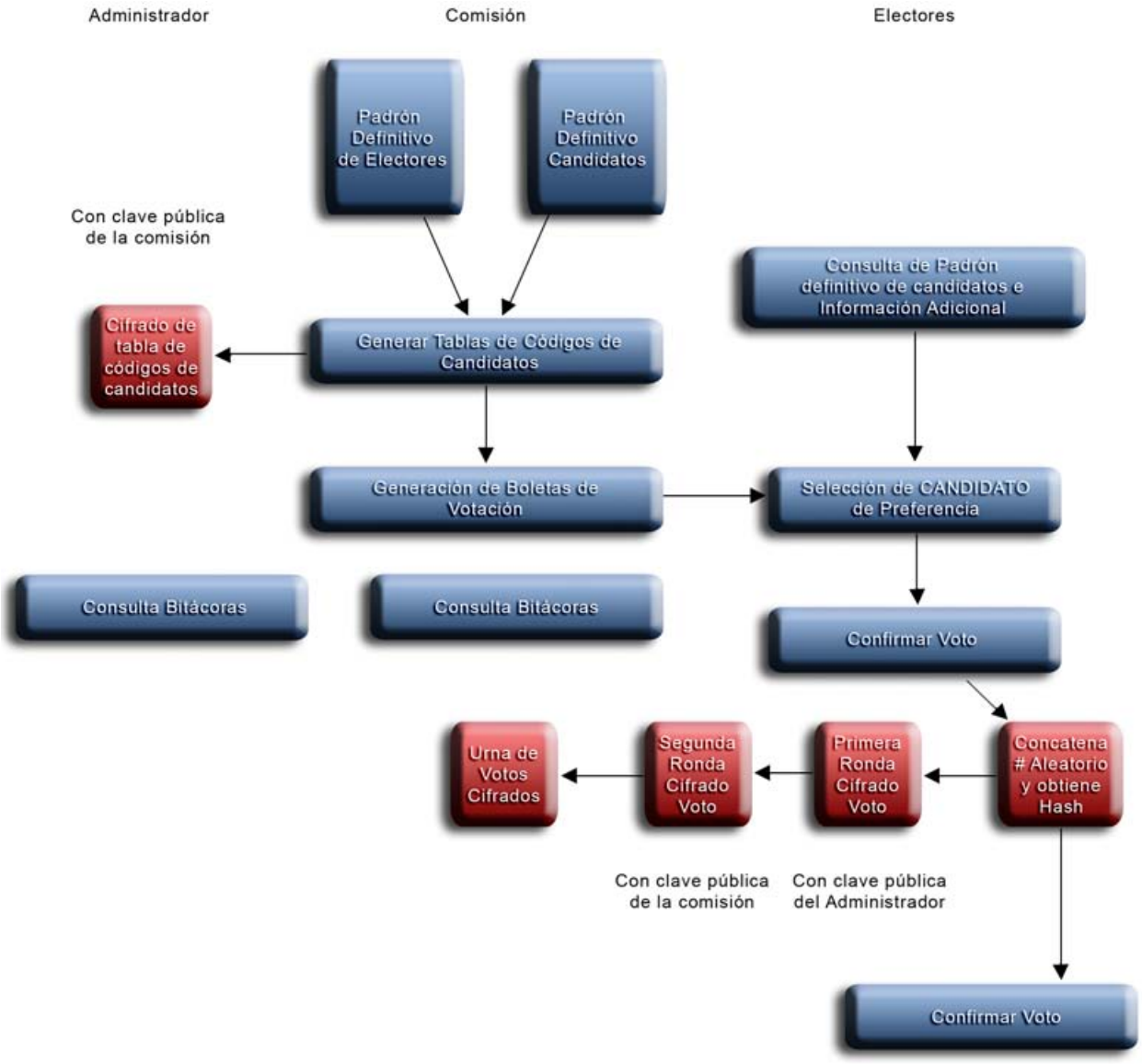


Figura 3.32: Protocolo IMATE

3.2.2.3. Bitácoras

En el proceso de votaciones se manejan tres tipos de bitácoras, las cuales son claves para la confiabilidad del proceso, pues en ellas se registran todas las actividades claves realizadas por el administrador, la comisión de vigilancia y los usuarios del proceso. Estas bitácoras pueden ser accedidas tanto por el administrador, como por la comisión de vigilancia:

- Histórico de Electores: en está bitácora se registran todos los eventos o actividades realizadas para generar el padrón definitivo de electores, que van desde la definición de los tipos de usuarios que podrán votar y las condiciones que deben cumplir, los ajustes realizados por reclamaciones de usuarios, hasta la generación del padrón definitivo y su respectiva firma y publicación.

- Histórico de Candidatos: en está bitácora se registran todos los eventos o actividades realizadas para generar el padrón definitivo de candidatos, que van desde la definición de los tipos de usuarios que podrán ser elegibles como candidatos y las condiciones que deben cumplir, la generación de un padrón preliminar, los ajustes realizados por reclamaciones de usuarios, la aceptación, rechazo o cancelación de candidaturas, hasta la generación del padrón definitivo y su respectiva firma y publicación.

- Histórico de Elecciones: en está bitácora se registran todos los eventos desde la creación del objeto de la votación la configuración y ajuste de cualquier parámetro, la generación del archivo de configuración de votaciones, su correspondiente firma y publicación, la generación de los padrones definitivos de electores y candidatos, la creación del evento de votaciones, la generación de la tabla de códigos de candidatos y de las boletas de votación, el inicio del proceso de votaciones, el registro de los usuarios que votan, la terminación de las votaciones, el reordenamiento aleatorio de la urna con los votos, la realización del proceso de escrutinio incluyendo el descifrado de la tabla de códigos de candidatos y de la urna con los votos, el conteo de los votos y la publicación y firma de los resultados finales.

3.2.2.4. Estados

El proceso de votaciones tiene los siguientes estados, los cuales serán desplegados una vez el usuario ingrese al sistema y dependerán de la fase en que se encuentre el proceso (Fig. 3.33).

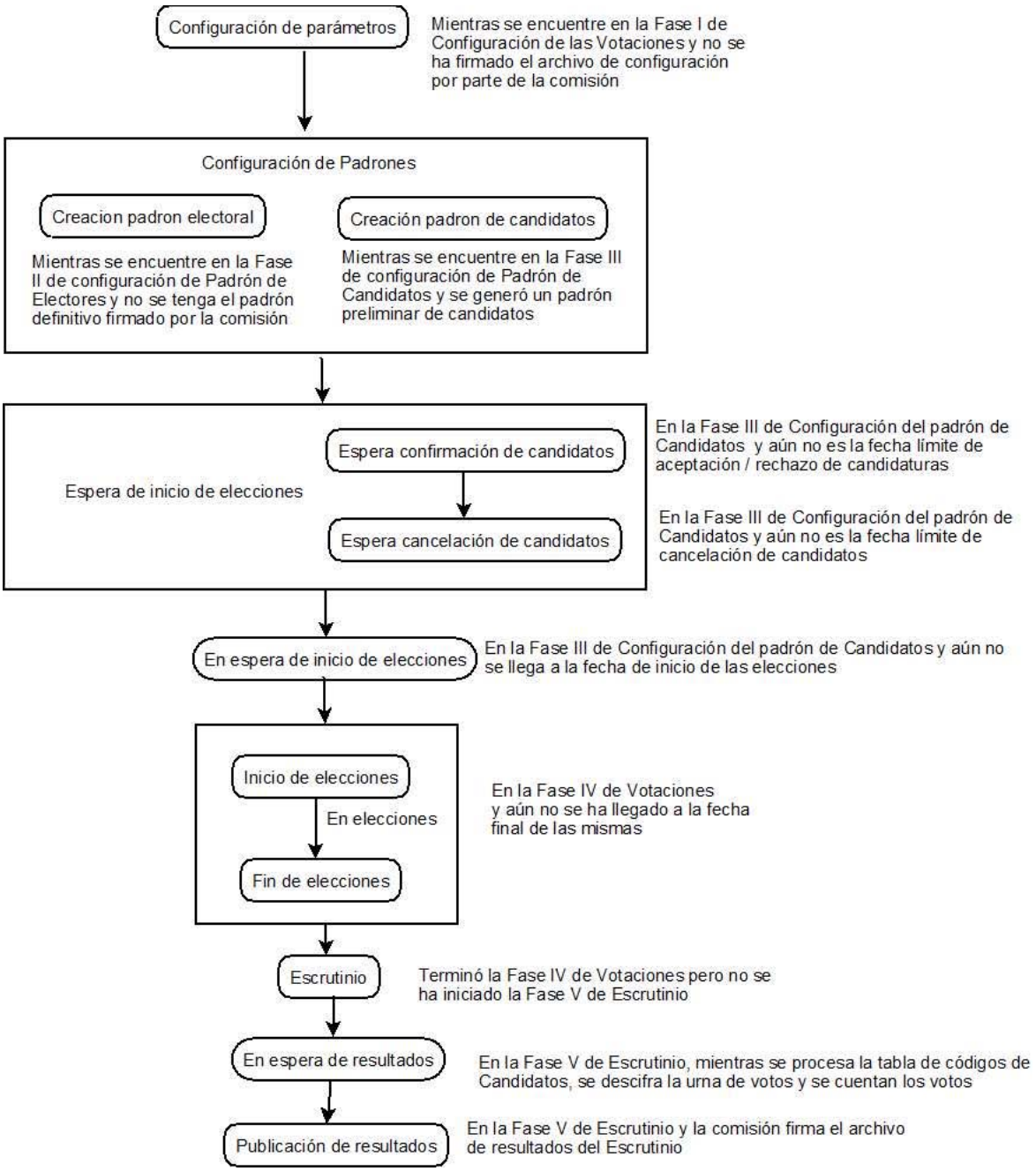


Figura 3.33: Estados de la elección

Se decidió implementar dos productos uno de selección de usuarios y otro de votaciones, con el fin de que el primero pueda ser utilizado en forma independiente del proceso de votaciones, pues se consideró que podría ser útil para procesos donde se requiera seleccionar un conjunto de usuarios de

un sitio *Plone* que cumplan con un grupo de condiciones predeterminadas.

3.2.3. Protocolo de votación

El protocolo implementado para el proceso de votación, considera algunas de las características del esquemas analizado en este capítulo *KOA* y el de Suiza analizado en [23], cumple en gran medida las condiciones requeridas según el análisis comparativo y el proceso de decisión mediante el método Delphi presentado en [23].

El sistema implementado no cumple con las siguientes características del sistema de Votaciones de Suiza y del *KOA*, algunas de las cuales se han establecido como solución en este trabajo:

Sistema de Votaciones de Suiza:

- Separación de la base de datos donde se encuentra la información de los votantes y la de los votos.
- Mecanismos de protección contra ataques de suplantación tipo IP Spoofing.
- Autenticación del servidor de votaciones mediante un certificado digital.
- Mecanismos de protección contra ataques de Denegación de Servicio.
- Uso de dos servidores en paralelo.

KOA (Proceso):

- El votante inicia conexión vía *HTTPs*
- Certificado digital del sitio web de votaciones.

KOA (Características Claves):

- Huella dactilar electrónica de los candidatos y la lista de votantes.

3.2.4. Descripción

El proceso es el siguiente (todos los eventos del mismo quedan registrados en la bitácora de las elecciones):

1. Identificación de los códigos de candidatos que registrarán los electores como votos

   Antes de iniciar el proceso de votación pero después de haber determinado los padrones definitivos de electores y de candidatos se generan un conjunto de n números aleatorios únicos de tamaño configurable entre 0 y 10 para los diferentes candidatos. La idea es que se generen tantos números aleatorios como votantes para cada uno de los candidatos. Por lo que en el caso de que por ejemplo hubiesen X candidatos y Y Electores válidos, por cada uno de los X candidatos se generarán Y números aleatorios, que no pueden repetirse ni en un mismo ni en diferentes candidatos, asegurando que dos votos no sean iguales y se dificulta la técnica de criptoanálisis por texto conocido, además si hay un proceso suplantación de elector al momento de votar se dificultaría escoger un código válido que corresponda a otro candidato pues son números aleatorios, además en el caso que alguien pudiera acceder a la tabla de códigos de candidatos y buscar un valor válido, al asignar exactamente tantos códigos como número de electores ayuda a determinar el caso de una suplantación al tratar de registrar dos votos con el mismo valor.

   Si por ejemplo tuviéramos un proceso de votaciones con 3 candidatos y 5 electores válidos, se podría generar una tabla de códigos asignados a los candidatos así (el tamaño real de los números aleatorios se configura como un parámetro de seguridad de la votación que se recomienda

sea mayor o igual a 40 pues 10^{40} es un poco mayor que 2^{128}):

Elector/Candidato	num1	num2	num3	num4	num5
Ana	27	90	14	25	8
Edith	18	10	45	91	33
Christian	15	2	99	85	72

Cuadro 3.1: Códigos de candidato

Esta tabla sólo puede ser descifrada por la comisión de vigilancia del proceso una vez termina el tiempo de la votación, pues está cifrada con la clave pública de la misma y sólo se habilita la opción de obtener el archivo al inicio del escrutinio.

2. Generación de boletas de votación

   Se debe generar una boleta de votación para cada elector, la cual incluye los códigos asignados para cada candidato que podrá utilizar solamente ese elector al momento de que decida votar, la selección de los códigos para cada candidato es aleatoria y se asegura que ningún código se repita en dos boletas diferentes. Esta boleta de votación es creada como un dato privado que tiene asociado un método que garantiza la privacidad, de tal manera que el objeto sólo podrá ser visualizado por parte del elector al que se le asignó después de haberse autenticado en el sitio *Plone* y de seleccionar la opción de votar, por lo que no podrá ser visualizada por ningún usuario distinto, incluyendo el administrador de las votaciones y la comisión de vigilancia, y en ningún momento distinto al del evento de votar. Este esquema cumple en realidad todas las condiciones de seguridad del estado del arte [23], sólo que no depende de la seguridad y disponibilidad de servicios distintos como el de correo electrónico

   En el ejemplo anterior una posibilidad de las 5 boletas de votación de cada uno de los electores podría ser la siguiente, en la cual se puede notar que la asignación de códigos de candidatos es aleatoria, pues por ejemplo al elector 1 le corresponde el segundo código posible del candidato Luis, el tercer código definido para el candidato María y el quinto código asignado al candidato Carlos; estos códigos de candidatos no podrán ser asignados a ningún otro elector. En el caso del elector 2 le corresponde el primer código del candidato Luis, el quinto del candidato María y el cuarto del candidato Carlos. Como se ve no se sigue ningún patrón reconocible en la asignación de códigos y podemos decir, que este método le da mayor aleatoriedad a la definición de las boletas de votación:
   Elector 1 (E_1) Elector 2 (E_2)

3. Registro de Voto por parte de un elector autenticado

   Una vez que un elector válido se autentique en el sitio *Plone* y decida registrar su voto, se activa un evento que presenta la boleta de votación, que en realidad no muestra los códigos de candidato asignados y sólo presenta sus nombres y fotos (en el caso que hayan sido guardadas).

   En el momento que el elector seleccione el candidato por el cual quiere votar el sistema le pide confirmar su selección, por ejemplo si el Elector 1 (E_1) decide votar por María E_1(M) y confirma su voto, el voto en un inicio corresponde al número 45 (E_1(M)=45) .
   A continuación se genera un número aleatorio de tamaño fijo, con tantos dígitos como el parámetro establecido en la configuración de la votación, y se le concatena al valor inicial del voto,

quedando por ejemplo 4578, donde 78 corresponde al número aleatorio generado.

En este momento se le aplica una función hash MD5 al número conformado por el código del candidato seleccionado y el número aleatorio generado, en el caso del ejemplo sería Hash(4578) y el resultado corresponde al recibo del voto que se explica en el paso siguiente del protocolo.

A continuación se realiza un proceso de doble cifrado al número 4578, primero con la clave pública de la comisión de vigilancia y luego con la clave pública del administrador de las elecciones, éste valor final corresponderá al voto que se guarda en la base de datos (urna de votos doblemente cifrados). El resto de códigos de candidatos no seleccionados en la boleta del usuario se guardan para verificar que no sean utilizados en forma no autorizada como un voto fraudulento, en el caso del ejemplo presentado serían los códigos E_1(L)=90 y E_1(C)=72, los cuales no se repiten para ningún otro elector y no podrían corresponder a un voto válido. Este esquema es tan seguro como el estado del arte pero se requiere que tanto el administrador como la comisión estén presentes para descifrar los votos, por lo que se requieren los n participantes y no k<n como el esquema que utiliza el algoritmo de secreto compartido [44].

4. Recibo del voto para el elector

   Tal como se explicó en el paso anterior del protocolo, antes de cifrar el voto, se le saca el hash, el cual corresponde al recibo del voto, éste número es incluido en un pdf que el elector podrá imprimir o guardar en su carpeta, incluyendo su identificador, la fecha y hora, y el título y descripción de la votación en la que está participando.

   Este esquema garantiza la verificabilidad individual analizada en el estado del arte, pues el elector puede confirmar al final del escrutinio si su voto fue contado apropiadamente, pues para cada candidato se especifica la lista de los recibos de los votos que fueron tenidos en cuenta, dándole mucha confiabilidad en el proceso a los electores. Lo único es que se contradice en parte el anonimato porque el usuario puede demostrar por quien votó, pues como se acaba de mencionar ese número de recibo aparece al final en la lista de votos detallada del candidato que seleccionó y si el elector lo imprimió al momento de votar, aparece su respectiva identificación, esto sucede sólo si es la voluntad del elector hacerlo público, pues él es único que puede acceder al recibo de su voto y cualquier otro participante en el proceso no puede conocer quién efectuó un voto a partir de un número de recibo, pues una función hash es unidireccional.

5. Escrutinio

   Antes de que se inicie el proceso de escrutinio se reordena en forma aleatoria la base de datos de votos (urna de votos doblemente cifrados) para que no exista una forma de asociarlos a los electores de acuerdo con el orden de los eventos de votos registrados en la bitácora de las elecciones.

   Se requiere que tanto el administrador como la comisión de vigilancia utilicen sus claves privadas, en ese mismo orden, para poder leer los votos de la base de datos ("abrir la urna sellada"), después de quitarles la cadena aleatoria de tamaño fijo que se le concatenó.

   Posteriormente, se requiere que la comisión inicie el conteo de votos por cada candidato, pues se toma cada voto descifrado y se le agrega al candidato que tenga dicho número asignado, éste número se marca en el objeto de códigos de candidatos como ya utilizado, en el caso que se repita se anula el voto anterior, el nuevo no es contado y se agrega al objeto de votos anulados con la descripción de **Voto Doble**.

   En el caso que se registre un voto con un código de candidato no asignado o que correspon-

da a una boleta cuyo voto por otro candidato fue ya contado, se desplegarán en el escrutinio como **Voto Inválido**. Es importante aclarar, que la posibilidad de que se ingrese en forma no autorizada un voto doble o un voto inválido es baja debido a que la urna de los votos es un objeto privado que sólo puede ser accedido por la función de votar, la cual sólo se le presenta a los usuarios debidamente autenticados, que no hayan votado y sólo durante el periodo de la votación, además se encuentra doblemente cifrados, lo que dificulta la realización de una modificación íntegra de un voto espurio que se pudiese tomar como válido.

Al finalizar este proceso se tiene el número total de votos por cada candidato con el detalle de los mismos y se compara con el de eventos contados de la bitácora, cualquier diferencia debe quedar registrada para ser analizada por la comisión de vigilancia.
El paso final del escrutinio corresponde a la generación, firma por parte de la comisión de vigilancia y publicación de un archivo pdf con los nombres de los candidatos, el número de votos, incluyendo datos de identificación de la votación, fecha y hora, los votos anulados con su descripción (doble/inválido), y el detalle de los números de hash (recibos de voto) contados para cada uno de los candidato.

Una vez los resultados de la votación han sido publicados, los electores podrán consultar la lista de los candidatos con su número total de votos y el detalle de los números de recibo contados para cada uno, de tal manera que podrá verificar que su voto fue efectivamente considerado para el candidato que seleccionó.

3.2.5. Conclusiones *IMATE*

En ésta sección se han dado las características del sistema de votaciones del IMATE. Éste sistema es el que rediseñe, refactorize y migre.

Hemos visto también los diagramas de casos de uso de cada uno de los roles que intervienen en el sistema. Además vimos las fases y los estados del sistema de votaciones.

Por último hemos visto el protocolo de votación y sus características principales. En el capítulo 5 se verá lo que falta de la arquitectura, pero todo en el contexto del sistema rediseñado.

Capítulo 4

MIGRACIÓN DE UN PRODUCTO *PLONE*

En está parte de la tesis describiré los puntos importantes referentes a una migración de un producto *Plone* cualquiera, en específico de un producto *Plone* 2.5 a *Plone* 3.2. Esta migración básica fue el primer pasó para poder analizar el sistema de votaciones del *IMATE*, comenzar con el rediseño y sus nuevas características. No vamos a dar la migración de un producto específico, si no los lineamientos a seguir para cualquier producto, que quiera ser migrado de forma básica, ya que si no se migra de forma básica el producto en *Plone* 2.5 no va a funcionar en Plone 3.2.

Cuando hablamos de migración de un producto *Plone* nos referimos al proceso de pasar un producto *Plone* de una versión anterior a la nueva versión de *Plone*. La migración fué necesaria, por que la funcionalidad interna de *Plone* cambió, para otorgar nueva funcionalidad o mejoras en la existente. Cuando es el caso de la nueva funcionalidad, el contenido que es almacenado en la instancia de *Plone*, tal vez no se acople a lo que la nueva versión del software espera. Aunque *Plone* tiene una herramienta interna que migra contenido existente a la nueva estructura. Lo anterior no significa que un producto de una versión cualquiera pueda ser migrado con dicha herramienta.

Hay que tomar en cuenta que cuando migramos un producto a una nueva versión de *Plone*, debimos haber actualizado *Plone*, pero como *Plone* no es más que un producto de *Zope*, entonces debimos haber hecho la actualización de *Zope* antes de comenzar todo el proceso de migración.

4.1. Preparación

Antes de comenzar el proceso de migración se debe recabar información como la siguiente:

1. Leer los archivos referentes al release (lanzamiento) de la versión de *Plone* y la sección "What's new in". Esta puede encontrarse en el directorio *CMFPlone* de la nueva distribución de *Plone* a utilizar.

2. Checar las dependencias

   a) De las notas de lanzamiento de la versión de *Plone* a la que se va a migrar debemos obtener:

      1) ¿Que versión de *Python* se requiere?
      2) ¿Que versión de *Zope* se requiere?
      3) ¿Se requieren nuevas bibliotecas para *Python*?

   b) Asegurarnos de que todos los productos add-on que se están utilizando necesitan actualizarce para soportar la nueva versión de *Plone* a la que se va a migrar.

c) Se recomienda comenzar con los productos de terceros que son utilizados en tu sitio. Verificar que ya han sido actualizados o que se ha verificado que trabajan en la nueva versión (test cases), y tenerlos actualizados en la instancia antigua si es posible antes de comenzar la actualización de *Plone*, *Zope* y *Python*.

d) Si *Zope* depende en una nueva versión de *Python*, instalar la nueva versión de *Python* primero.

e) Si la nueva versión de *Plone* depende en una nueva versión de *Zope*, se necesita instalar primero *Zope* antes de proceder con la actualización de *Plone*.

1) Hay que tomar en cuenta que *Zope* tiene sus propios lineamientos para su migración.

f) Leer los siguientes archivos en el directorio *CMFPlone* de la distribución de la nueva versión de *Plone*:

1) README.txt
2) INSTALL.txt
3) UPGRADE.txt (Aunque este usualmente contiene únicamente el procedimiento general descrito arriba)

Es muy importante realizar un respaldo de nuestro sitio *Plone* (incluyendo productos). Así como no es recomendable trabajar directamente en el sitio Productivo hasta que se este seguro que la actualización fue exitosa. Lo que se puede hacer es crear un entorno de pruebas para verificar la actualización. Esta es una buena forma de probar los productos de terceros y dependencias en un entorno igual al Productivo.

4.2. Procedimiento General

En está sección voy a enfocarme en los procedimientos que se necesitan para realizar la mayoría de las migraciones.

Cuando actualizamos a una nueva versión de *Plone*. es importante ejecutar el procedimiento de migración de contenido, debido a que estructuras internas en *Plone* tal vez cambiaron desde la última versión. Veamos el procedimiento general de actualización.

Antes de comenzar la actualización de los productos o del sitio, debemos asegurarnos de tener un respaldo.

El procedimiento manual para lo anterior se describe abajo. Si se está utilizando instalador, podemos saltar la parte referente a reemplazar directorios antiguos con los nuevos (Paso 3 y 4) ya que está tarea es realizada por el instalador.

1. Respaldar el directorio del sitio de *Plone* completo
2. Parar la instancia de *Plone* que se está ejecutando
3. Remover los directorios de productos que deseamos reemplazar
4. Poner los nuevos directorios de productos
5. Iniciar *Plone* (el sitio tal vez sea inaccesible, hasta que realicemos los siguientes pasos)
6. Ir a http://miSitio/manage (aka. el ZMI) y dar click en `portal_migrations`
7. Asegurarte que estas en la pestaña `Upgrade` (en versiones viejas, está pestaña es llamada `Migrate`)
8. Esto significa que se tiene que ejecutar el procedimiento de actualización para que sea 3.1.7
9. Dar click en el botón de `Upgrade`

a) Si se desea ver que pasos debe realizar la actualización sin realizar los cambios actuales, podemos poner la opción `Dry Run`, esto realizara los pasos exactamente igual como una migración normal lo haría, pero sin escribir a la base de datos.

10. El sitio ahora estará actualizado, tomará tiempo dependiendo de las versiones de actualización.

4.3. Actualización de productos *add-on*

Los pasos para migrar un producto de terceros (aka. third party products) son:

1. Parar la instancia de *Plone* que se está ejecutando
2. Navegar al directorio del producto de la instancia de *Plone*
3. Remover los directorios de los productos que queremos o necesitamos reemplazar
4. Copiar los nuevos directorios de los productos, y checar que los permisos en cada directorio de los productos sean correctos
5. Iniciar *Plone* de nuevo (el sitio tal vez sea inaccesible, hasta que realicemos los siguientes pasos)
6. Navegar al quickinstaller en el ZMI, y reinstalar o actualizar los productos si se puede (productos que soportan las dos versiones de *Plone*). Realizar procedimientos de actualización específicos del producto si los hay. Encontraremos esos procedimientos en los documentos de cada producto.

4.4. Problemas

Cuando un problema ocurre durante la migración se recomienda que se tomen los siguientes pasos.

4.4.1. Checar los archivos de bitácora (logs)

Cuando un error en un sitio ocurre, o *Zope* falla al comenzar, hay probablemente una mensaje de error en los archivos de log de *Zope*. Hay que localizar estos archivos de log e inspeccionar `event.log`. Se recomienda ignorar warnings irrelevantes y buscar por palabras como `error`, `exception` y `traceback`.

Cuando *Zope* no inicia y no hay información que nos ayude en los archivos de log, podemos iniciar *Zope* de forma interactiva y verificar los mensajes de salida: `zopectl fg`

Tal vez se pueda encontrar mayor información de mensajes de error en:

- Tips de migración especificas de la versión de nuestra versión de *Plone*
- Referencias de Error
- El `pastebin`, donde mensajes de error y fragmentos de código son compartidos y depurados colectivamente

4.4.2. Pruebas sin personalización

Cuando se tienen templates de páginas o scripts de *Python* personalizados, los cambios que hagas tal vez interfieran con cambios en la nueva versión de *Plone*. Es importante tomar en cuenta está posibilidad, dado que tus personalizaciones son únicas para tu sitio.

Hay que remover temporalmente las personalizaciones, por ejemplo quitar los layers de portal_skins, o quitando archivos de estos layers en el sistema de archivos. Si el problema desaparece, se necesitará checar la personalización. Usualmente es mejor copiar los archivos originales de la nueva versión de *Plone* a tu skin, y volver a personalizar.

4.4.3. Pruebas sin productos

Problemas de compatibilidad o bugs en productos que se hayan instalado pueden causar problemas en *Plone*. Hay que ir a `Site Setup > Add/Remove Products` y remover (desinstalar) todos los productos que no son distribuidos junto con *Plone*. Quitar los productos desinstalados del directorio `Products` de la instancia *Zope* correspondiente.

Si el problema desaparece, se debe volver a checar el producto con problemas:

- ¿Soporta la nueva versión de *Plone*, *Zope* y *Python*? Checar el `README.txt` del producto u otros archivos de información.
- ¿El producto requiere algún procedimiento adicional de migración? Checar el archivo `INSTALL.txt`, `UPGRADE.txt` u otros archivos de información.
- ¿El producto se instalo apropiadamente? Volver a instalar y checar el log de instalación.

4.4.4. Pruebas en una instancia de *Plone* recién instalada

Crear un nuevo sitio de *Plone* con la nueva versión de *Plone* a utilizar. No se necesita una nueva instancia de *Zope*, dado que tu puedes añadir otro sitio de *Plone* en la raíz de *Zope* (Recordemos que *Plone* no es más que un producto sobre *Zope*). Si el problema no ocurre en un sitio recién instalado, la causa del problema es de personalización, y de productos instalados o contenido que no fue migrado apropiadamente.

4.5. Actualización de productos *add-on* para *Plone* 3

Plone 3 viene con una nueva versión de *Zope*, *CMF* y *Archetypes*. Cuando cualquier framework se actualiza, algunas cosas son eliminadas o modificadas. Lo siguiente es una lista de las actualizaciones más comunes que necesitan ser aplicadas a los productos de autor para asegurar que sus productos funcionen en *Plone* 3.

Antes de comenzar con la lista mencionare algunas recomendaciones que nos ayudaran a ahorrar tiempo cuando actualicemos nuestros productos en las siguientes versiones de *Plone* (3.5 y 4.0).

Dependiendo del producto, puede ser difícil incluir compatibilidad tanto para *Plone* 2.5 y *Plone* 3 en el mismo producto. Hay varias razones para esto, pero las principales son:

- La definición del estándar ***workflow*** en ***CMF*** ha cambiado.
- La nueva infraestructura de ***portlet*** (aunque soporta los antiguos portlets, el rendimiento se ve afectado).
- La introducción de ***viewlets*** como el principal camino para renderear fragmentos de contenido en el layout.

La recomendación general es:

- Si tu producto es más complejo que un simple tipo, crea dos versiones, una para *Plone* 2.5 y otra para *Plone* 3.
- Si usas ArchGenXML para crear tu producto, deberías de poder regenerar tu producto desde el modelo UML para obtener una versión compatible con *Plone* 3.

Un tip que tal vez nos salve de modificaciones mayores en *Plone* 3.5 y 4.0, es lo siguiente:

1. Iniciar *Zope* en modo de depuración usando `zopectl fg` y usar el producto normalmente. Verificamos las salidas en la consola para ver si hay algún warning referente a deprecación.

2. Deshabilitar el `plone_deprecated` skin layer y asegurarse de que la aplicación aun se ejecuta (esto deshabilita los métodos deprecados y estilos CSS deprecados).

- Muchos de los nuevos componentes utilizan vistas de *Zope* 3 en lugar de templates. Esto puede ser personalizado a través de la web usando la herramienta `portal_view_customizations`.
- No hay que depender de bibliotecas de Java Script. En su lugar hay que usar abstracciones de KSS, ya que las implementaciones van a cambiar.
- Instalaciones basadas en ***QuickInstaller*** deberían cambiar a perfiles de ***GenericSetup.***
  - Utilizar eventos en lugar de manage_methods (los cuales tal vez desaparezcan en *Plone* 3.5 o 4.0)
- Tecnologías de empaquetado:
  - Utilizar paquetes de *python* en lugar de productos *Zope*
  - Poner los paquetes como ***eggs*** y registrarlos con el ***Python Cheese Shop***.
  - Para crear nuevos paquetes hay que utilizar el comando *`Paste`* de *Python*.

4.5.1. Modificación de CMFCore.permissions

En versiones recientes de CMF, la forma de importar el módulo de permisos ha cambiado. La forma de actualizar el producto para que soporte tanto la forma antigua como la nueva es la siguiente:

```
try: #Nuevo CMF
from Products.CMFCore import permissions as CMFCorePermissions
except ImportError: #Viejo CMF
from Products.CMFCore import CMFCorePermissions
```

Entonces estará este problema arreglado y seras capaz de soportar múltiples versiones en el producto. Hay que notar el bloque try/except el cual es únicamente necesario si queremos soportar *Plone* 2.1, si nuestro *Plone* es 2.5 o superior, con sólo poner lo que hay bajo Nuevo CMF es suficiente.

4.5.2. Módulo *transaction*

En Archetypes 1.3 y 1.4, nosotros importábamos transaction en el modulo principal para trabajar con *Zope* 2.7. Debido a que *Zope* 2.7 ya no es una versión soportada, entonces en Archetypes 1.5 tampoco y como está versión es la que viene con *Plone* 3, tenemos que actualizar nuestro código.

Así que lo que debemos hacer es cambiar las ocurrencias de:

```
from Products.Archetypes import transaction
```

a lo siguiente:

```
import transaction
```

4.5.3. Deprecación de *ContentFactoryMetadata*

Plone 3.0 es la primer versión que forzó este cambio. Veamos como se actualiza nuestro producto para utilizar la nueva sintaxis.

Donde tengamos la siguiente importación:

```
from Products.CMFCoreTypesTool import ContentFactoryMetada
```

hay que pasarlo a:

```
from Products.CMFCore.TypesTool import FactoryTypeInformation
```

4.5.4. Actualizar los workflows para que utilicen perfiles GenericSetup

Para poder instalar workflows en *Plone* 3.0, tenemos que hacer uso de perfiles GenericSetup de CMF. Instalar workflows en alguna otra forma ya no está soportado, desafortunadamente hay cambios en la arquitectura del CMF que no nos permite soportar ambas formas de creación al mismo tiempo.

El error típico que indica que estamos tratando de instalar workflows sin usar GenericSetup es:

```
ImportError: cannot import name addWorkflowFactory
```

Para workflows existentes, el camino más sencillo de hacer que la instalación del producto use GenericSetup es:

- Instalar el producto (y su workflow) usando *Plone* 2.5.
- Usando la herramienta `portal_setup` en el ZMI, exportar un snapshot del perfíl actual del sitio:
  - Click en la pestaña `Export` (Fig. 4.1).

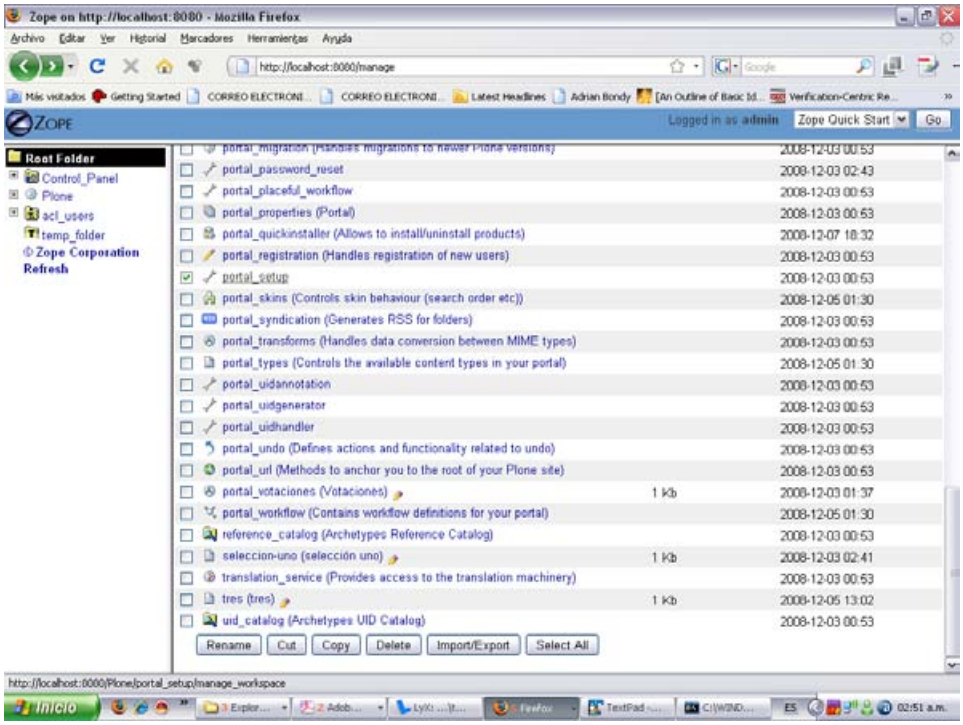


Figura 4.1: portal_setup

  - Seleccionar las partes que se necesitan para exportar la configuración (en este caso, `Workflow Tool`). Fig. 4.2.

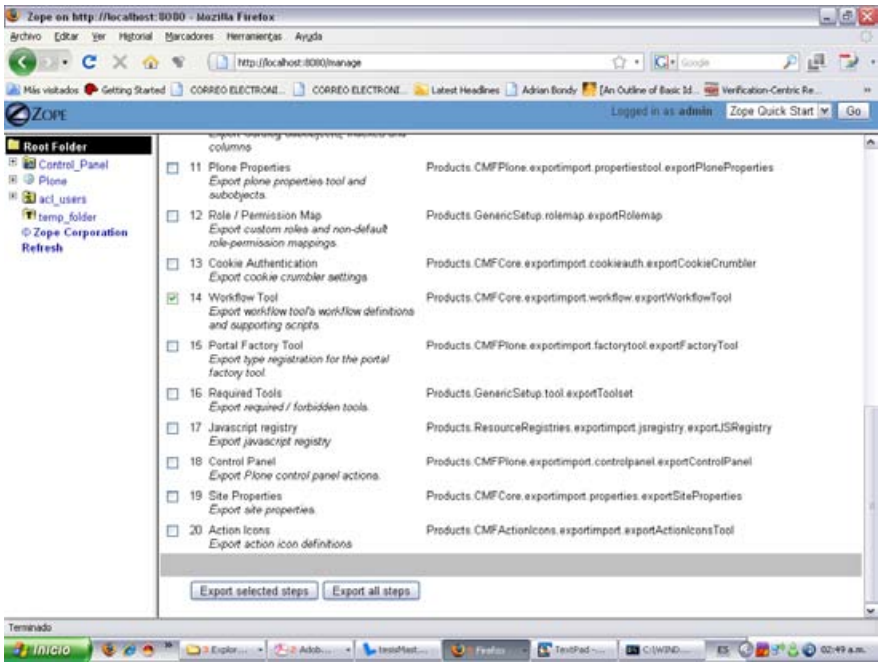


Figura 4.2: Exportar Workflow Tool

- Click en el botón `Export Selected Steps.`
- Lo que nos dará un archivo .tar con un nombre como el mostrado en la siguiente figura 4.3.

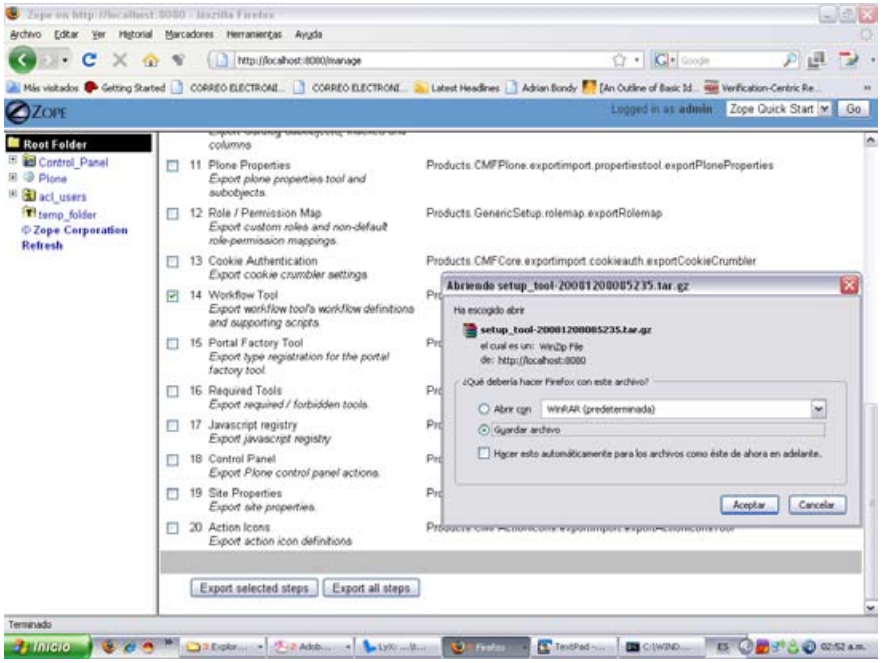


Figura 4.3: Generación del archivo .tar

- Desempaquetar el archivo .tar, y poner los archivos y directorios en el directorio `profiles/default/`

en la raíz del producto.

- Eliminar los directorios en `workflow/` que no son parte del producto, y editar `workflows.xml` tal que únicamente contenga información de tus workflows.
- Eliminar las definiciones viejas en `Extensions`, pero hay que asegurarse de mantener scripts que sean utilizados por el workflow, ya que serán referenciados por el perfíl.
- Añadir un archivo que se llame `configure.zcml` en la raíz de tu producto, debe registrar el perfíl default.
- Remover el código redundante de `Extensions/Install.py` y añadir el machote de código que invoca la configuración de GenericSetup.

4.5.5. Deprecada la búsqueda de usuarios y grupos vía la herramienta de Membership

La búsqueda de usuarios y grupos usando las herramientas portal_membership y portal_groups ha sido deprecada. Hay que utilizar en su lugar funciones de búsqueda de `PAS` directamente o mediante `pas_search` de `PlonePAS`.

En el capítulo 5 se hablara más al respecto.

4.5.6. *Plone* 3 no crea carpetas de miembros

Con la versión 3, los directorios de miembros son opcionales, y no son creados por default. Esto significa que no podemos contar con ellos para almacenar datos o procesar de alguna forma estos directorios. Y dado que es una mala practica no hay que utilizarlas por default.

4.6. Conclusiones

He documentado el proceso de migración de un producto *Plone* de forma muy genérica, no todos los pasos para migrar un producto específico se encuentran en este capítulo, por que cada producto utiliza dependencias diferentes, por ejemplo: el sistema de votaciones del *IMATE* utilizaba el producto *CMFMember*, el cuál ya no tiene una versión para Plone 3 y además a dejado de ser utilizado por la comunidad, aunque apliquemos el proceso de migración básico al sistema de votaciones del *IMATE* de cualquier forma va a seguir inservible por culpa de esta dependencia.

Pero los pasos aquí tratados están bien definidos y sabemos que problemáticas pueden surgir en la migración básica, lo cuál es de gran ayuda para cualquier administrador o desarrollador de *Plone*.

Capítulo 5

REDISEÑO Y NUEVAS CARACTERÍSTICAS DEL SISTEMA DE VOTACIÓN

El proceso de rediseñar el sistema de votaciones incluyó la migración y refactorización, esto trajo consigo desde problemas menores como cambiar el nombre a ciertas bibliotecas, actualizar versiones nuevas de software para tener un mejor rendimiento, hasta problemas mayores como pueden ser que parte de la tecnología utilizada en la versión anterior haya cambiado totalmente, lo cual significa volver a desarrollar el sistema con los lineamientos de la nueva tecnología, etc.

La mayor parte de la migración se explicó en el capítulo anterior. Se refactorizó casi totalmente toda la clase principal del producto de selección de usuarios, así como también algunos métodos de la clase principal del producto de votaciones. Además de esto, se rediseño completamente el producto de selección de usuarios, los permisos del sistema de votaciones y parte de la vista.

5.1. Arquitectura del nuevo sistema

Enseguida se mostraran los diagramas de casos de uso del producto de selección de usuarios (Fig. 5.1, 5.2) diagramas de clase (Fig. 5.3, 5.4), componentes (Fig. 5.6) y despliegue del sistema de votaciones (Fig. 5.7) [63]. Los diagramas de caso de uso del producto de votaciones pueden verse en (Fig. 3.26, 3.27, 3.28, 3.29).

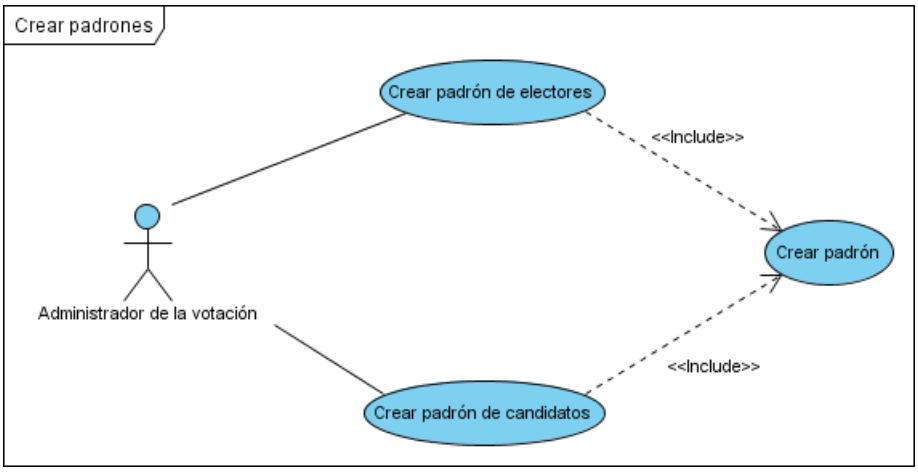


Figura 5.1: Caso de uso - Crear padrones

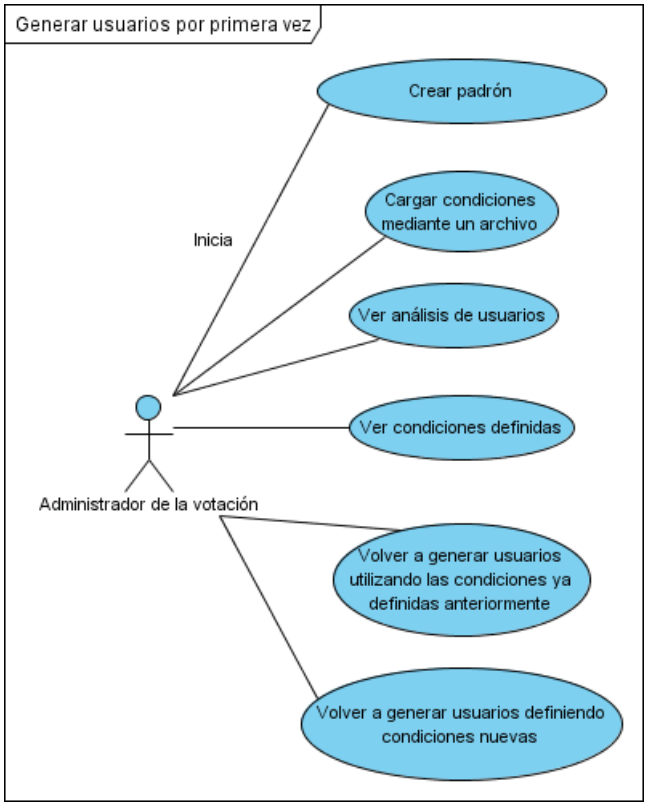


Figura 5.2: Caso de uso - Generar usuarios por primera vez

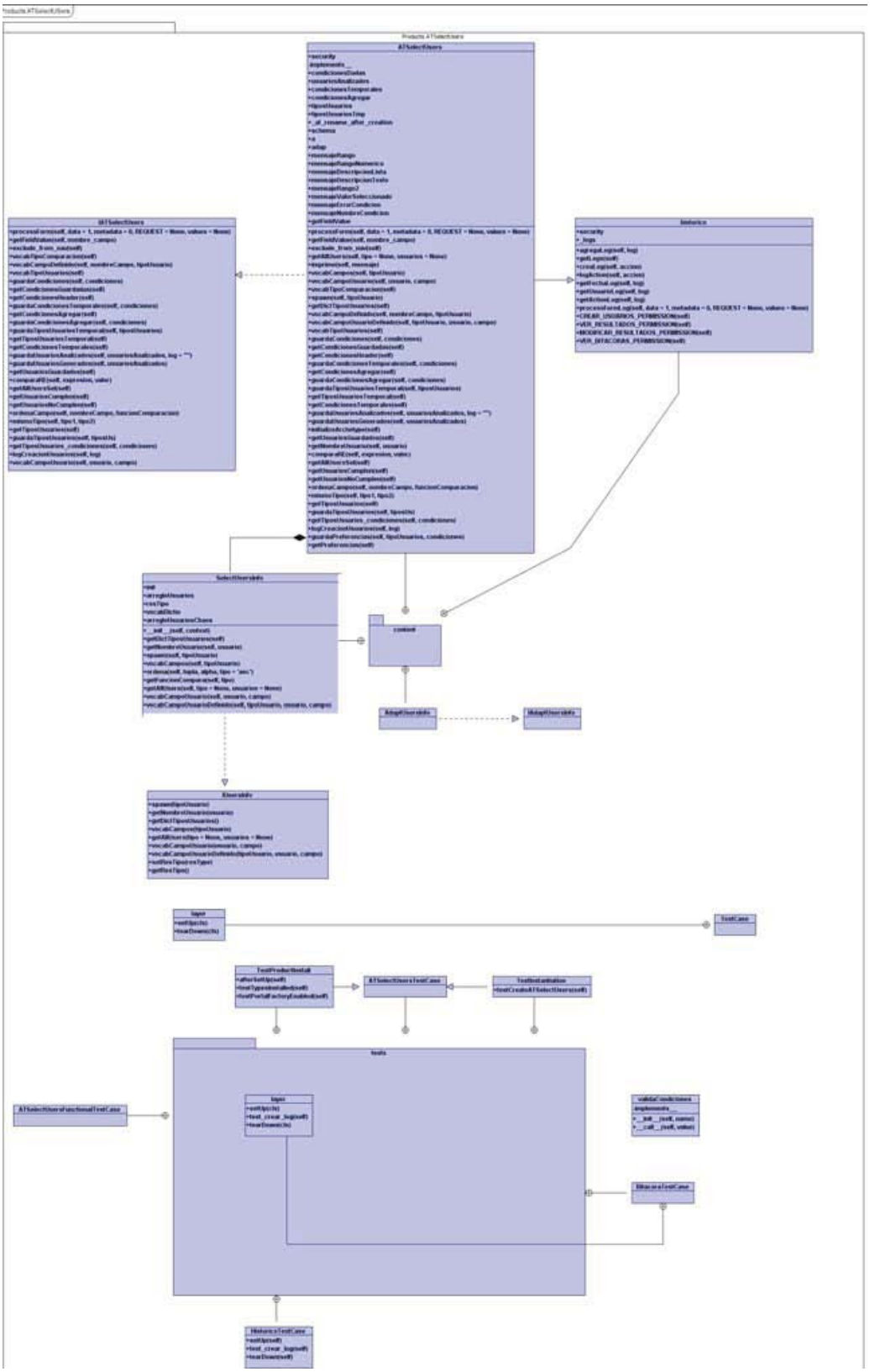


Figura 5.3: Diagrama de clases - ATSelectUsers

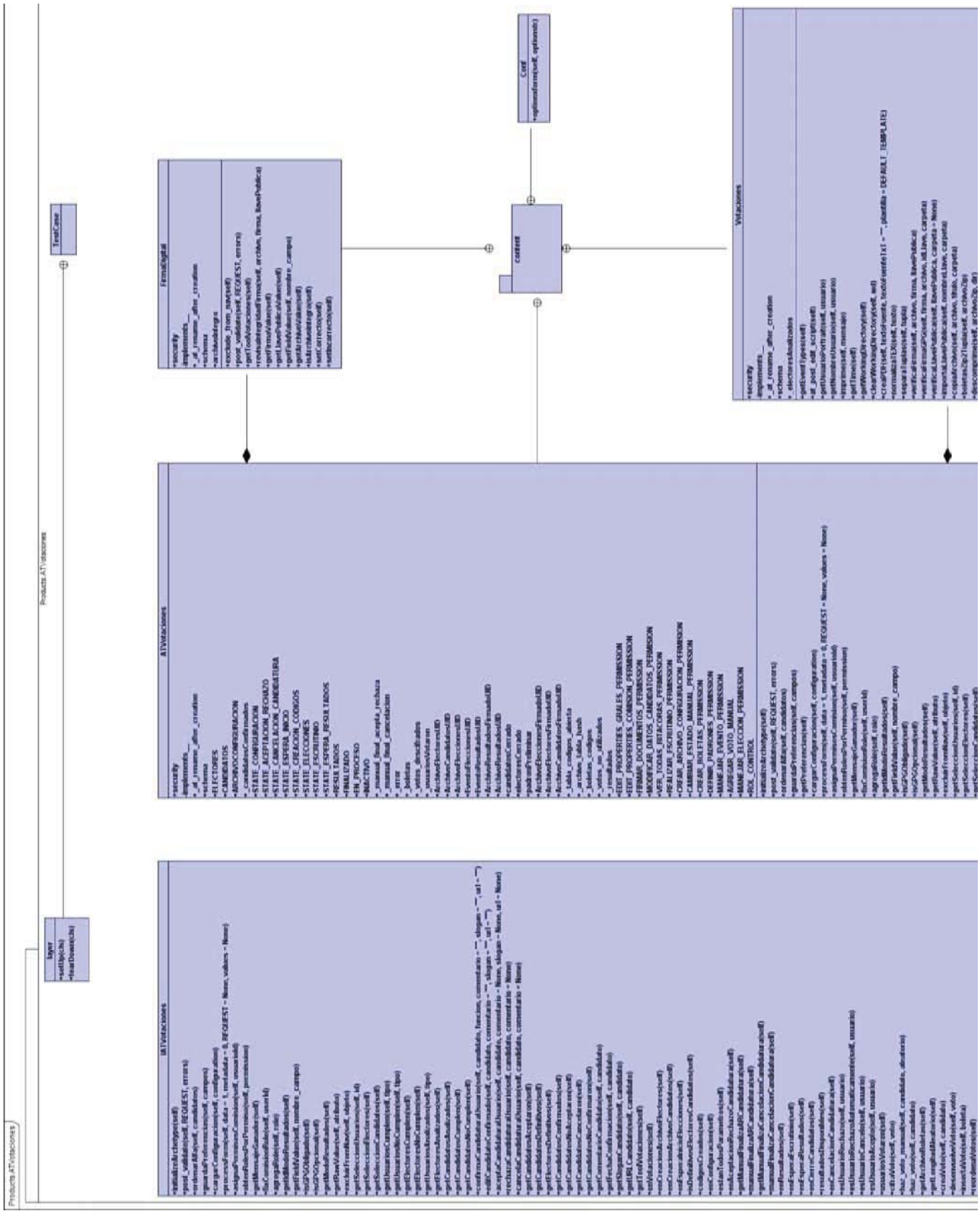


Figura 5.4: Diagrama de clases - ATVotaciones

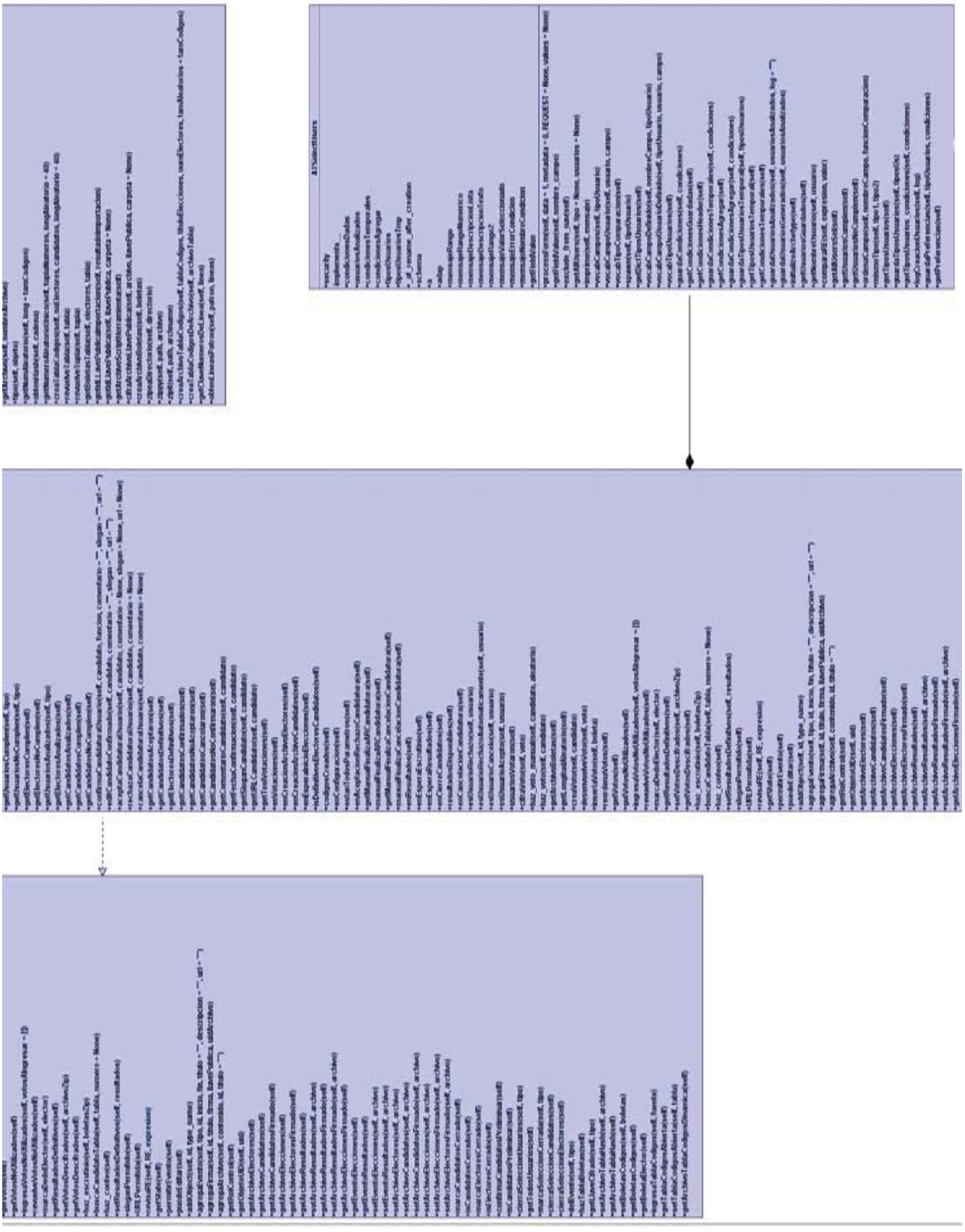


Figura 5.5: Diagrama de clases - ATVotaciones - Parte 2

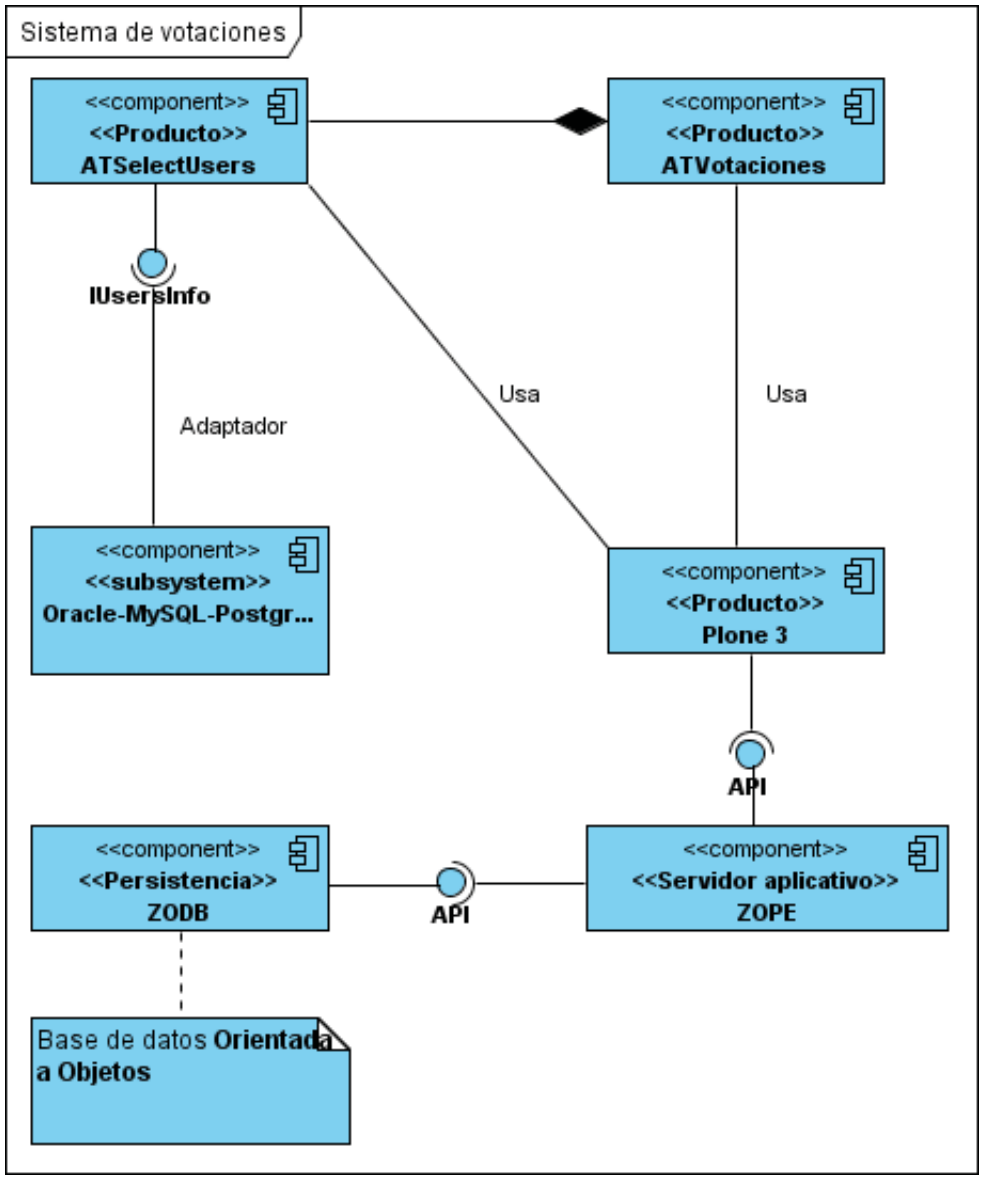


Figura 5.6: Componentes del sistema

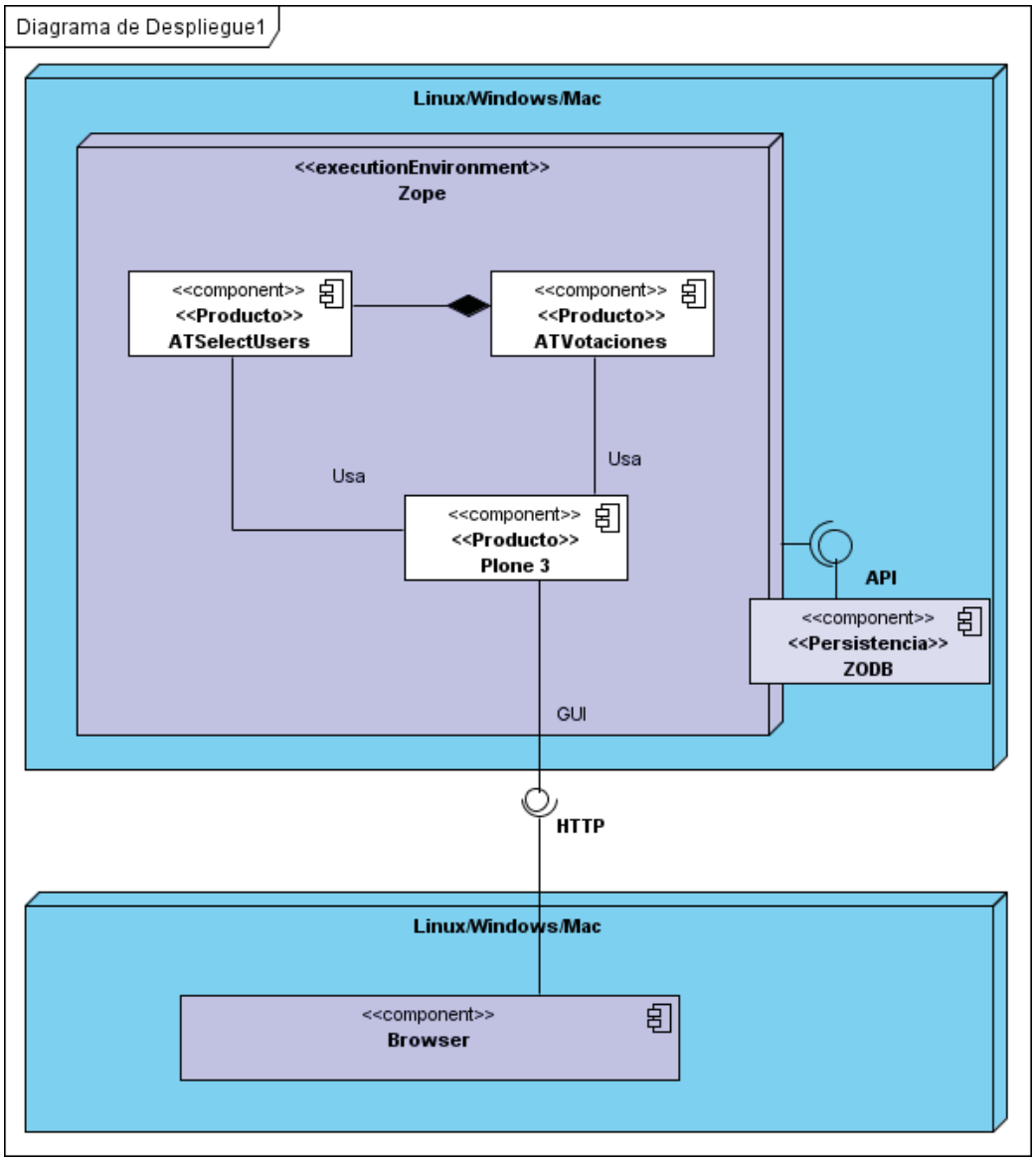


Figura 5.7: Diagrama de Despliegue

5.2. Acerca de las nuevas características, problemas y soluciones

El rediseño del sistema de votaciones del *IMATE* implico tener nuevas características, estas nuevas características generaron problemas al quererlas implementar, se mostrara la forma en que se solucionaron dichos problemas; por otro lado hay características nuevas como las de utilizar patrones de diseño, interfaces y adaptadores que no tienen que ver directamente con el proceso de migración del capítulo anterior, si no con el de rediseñar y refactorizar el sistema.

En seguida se platicara más sobre estas nuevas características. Los productos ATSelectUsers (Fig. 5.3) y ATVotaciones (Fig. 5.4) fueron empaquetados como eggs mediante la creación de una estructura de ZopeSkel `$paster create -t plone Products.ATSelectUsers` y una vez creada está estructura

se pone el directorio del producto que se migro siguiendo los pasos del capítulo anterior y otros que aquí se mostraran.

Los productos también fueron configurados mediante el toolkit basado en *Zope* que administra la configuración del sitio [37]. Esto último basándome en mi migración de la configuración del producto InstantMessage de *Plone* 2 a *Plone* 3 [36]. Para esto hay que hacer cosas como quitar los action dentro de cada clase donde se define un contenido y crear el directorio profiles/default donde estarán los archivos de configuración de GenericSetup.

Una gran parte del producto ATSelectUsers fue rediseñado para utilizar **interfaces**, **adaptadores** y **patrones de diseño**, el fin de esto es crear una modulo mucho más flexible, extensible, y portable entre productos de administración de usuarios. El producto de votaciones también incluyo interfaces.

Veamos una serie de problemas que surgieron como parte de la migración del sistema:

1. Al querer empaquetar los productos ATSelectUsers y ATVotaciones como eggs, ya que el espacio de nombres (namespace) Products causa conflictos.

2. Se tiene en el producto ATSelectUsers una clase la cual lleva el log de las acciones que se han realizado como por ejemplo, que tipos de usuarios se generaron, está clase se muestra como un pestaña en el producto, por lo que se manejaba en la versión de *Plone* 2 como un action dentro de la clase del contenido, pero debido a la nueva forma de configurar los productos (GenericSetup) se tuvo que modificar.

3. Al querer mandar a llamar a portal_votacion con getToolByName, este problema una vez más se debe a que con GenericSetup se tiene que declarar en un archivo de configuración especial para herramientas.

4. El producto ATVotaciones utiliza un tipo de contenido llamado Votaciones el cual como ya hemos mencionado debe declararce también con la configuración de GenericSetup, pero como este tipo de contenido se utilizaba como una herramienta (getToolByName) de las que hablaremos más adelante en este capítulo, entonces se tuvo que declarar en tools.xml.

5. Los archivos del directorio scripts los cuales se registraban por ejemplo con:

   ```
   DirectoryView.registerDirectory('skins/ATSelectUsers/
   widgetsSeleccion',product_globals)
   ```

   Tienen que declararse con GenericSetup en el archivo de skins.xml

6. El producto de votaciones también tuvo que ser modificado con los actions que se declaraban dentro de la clase que definía el tipo de contenido.

7. El tipo de contenido FirmaDigital el cual forma parte del producto ATVotaciones también tuvo que ser puesto con la configuración GenericSetup, pero el problema mayor fue que este producto se declaraba dentro de la clase de votaciones para ser utilizada, lo cual no estaba muy claro como hacerlo con GenericSetup.

5.3. Empaquetar como *eggs*

Los productos ATSelectUsers y ATVotaciones fueron empaquetados como eggs mediante la creación de una estructura de ZopeSkel.

5.3.1. *Python egg*

Es la forma de empaquetar y distribuir paquetes de *Python*. Cada *egg* contiene un archivo setup.py con metadatos, estos metadatos pueden ser: nombre del autor, dirección de correo electrónico e información acerca de la licencia del paquete, así como información acerca de dependencias. *setuptools* es

la biblioteca de *Python* que es la base de las capacidades del mecanismo de los egg, ya que es posible encontrar automáticamente y descargar dependencias para eggs que se desean instalar. Es aun posible para dos eggs diferentes el usar concurrentemente versiones diferentes de la misma dependencia.

5.3.2. Espacio de nombres

Una característica de setuptools es que hace posible distribuir múltiples paquetes separados que comparten un sólo espacio de nombres. Por ejemplo, los paquetes plone.theme y plone.portlets ambos comparten el espacio de nombres "plone", pero son distribuidos como *eggs* separados. Cuando se instalan, cada código fuente dentro de los eggs tiene su propio directorio. El espacio de nombres de los paquetes eliminan la necesidad de distribuir un sólo paquete *plone* enorme.

5.3.3. Espacio de nombres *Products*

Cuando *Zope* encuentra un producto le crea una entrada en Control_Panel/Products en la raíz del ZMI, y ejecuta el método initialize(), que se encuentra declarado dentro del archivo __init__.py en la raíz del producto, esto sucede cada vez que *Zope* inicia. No todos los paquetes usados en un contexto *Plone* necesitan ser un producto, pero se necesita que sea un producto para lo siguiente:

- Perfiles GenericSetup
- Directorios Skin instalados como layers en la herramienta portal_skins.

La forma básica de crear un producto es usar Paster/ZopeSkel para crear un paquete como egg en el espacio de nombres Products. Este espacio de nombres es el utilizado en la versión anterior del sistema de votaciones por lo que se deseaba mantenerlo, y con está característica se pudo lograr. La forma que se menciona en [38] es mediante el template basic_namespace, pero en este trabajo se utilizo el template *plone*, el cual también nos permitió tener el resultado deseado (Fig. 5.3, 5.4).

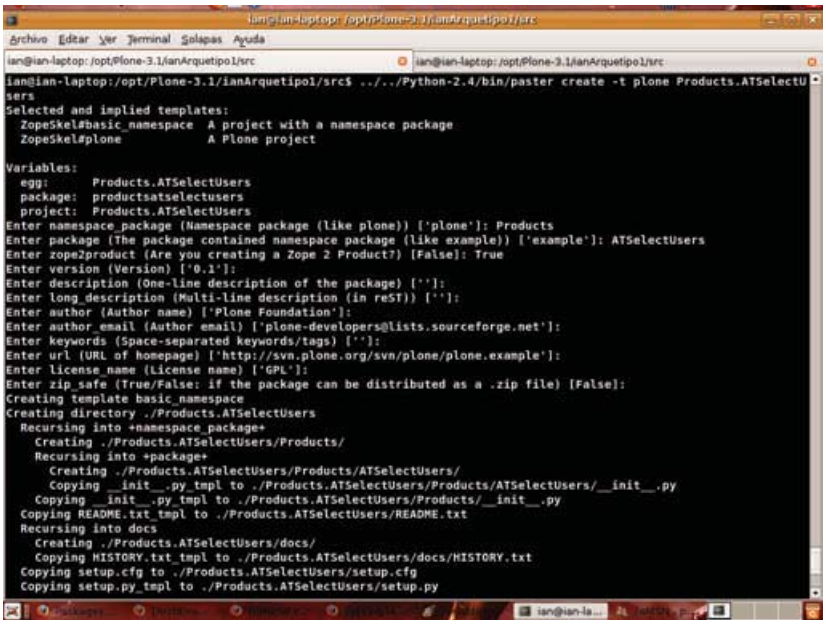


Figura 5.8: Generar template para Products.ATSelectUsers

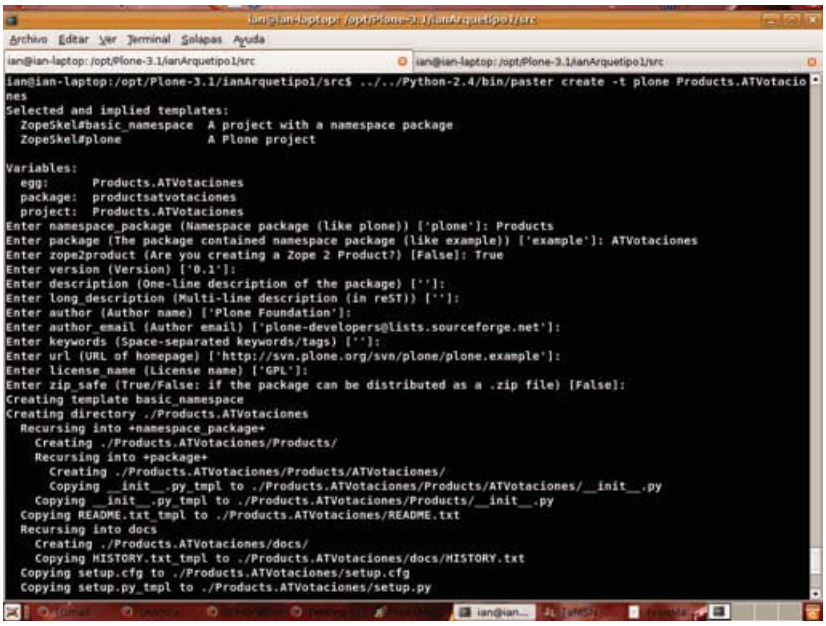


Figura 5.9: Generar template para Products.ATVotaciones

Después de crear los templates para los productos, y dado que también estamos trabajando con buildout (A.2.2) por lo que creamos el template en el directorio src, aparte de esto tenemos que modificar el buildout.cfg.

```
# Add additional eggs here
# elementtree is required by Plone
eggs =
elementtree
FeedParser
Products.ATSelectUsers
Products.ATVotaciones
# Reference any eggs you are developing here, one per line
# e.g.: develop = src/my.package
develop = src/Products.ATSelectUsers
src/Products.ATVotaciones
```

Ya que modificamos el buildout.cfg entonces nos vamos al directorio src/Products.ATSelectUsers/Products/ATSelectUsers y pegamos todo el contenido del producto *Plone* 2 que fue migrado con los pasos del capítulo anterior. Lo mismo para ATVotaciones y por último ejecutamos de nuevo buildout y tenemos la solución al problema 1.

5.4. GenericSetup

Los productos fueron configurados mediante el toolkit basado en *Zope* que administra la configuración del sitio [37]. GenericSetup es un gran avance en la administración de la configuración de un sitio *Plone*, y GenericSetup es una parte primordial de como *Plone* maneja su propio proceso de creación del sitio.

5.4.1. Profiles

En está subsección veremos como GenericSetup realiza su trabajo. GenericSetup introduce la idea de un profile (perfíl) de configuración. Un perfíl es un conjunto de archivos XML que describen la configuración de tu sitio. Hay que notar que la diferencia fundamental entre un perfíl y un método de instalación (como se hacía en la versión anterior del sistema de votaciones); los métodos de instalación definen un conjunto de pasos que deben ejecutarse para obtener un resultado, mientras un perfíl describe el resultado por si mismo. Esto tiene más sentido hablando de semántica. También saca la configuración del sitio de las manos del programador y la pone en un lugar donde los que no son programadores y administradores a la vez puedan mirar la configuración y entender que significa.

Dentro del perfíl tenemos varios archivos de configuración como los que se enlistan enseguida:

- toolset.xml
  - Registra todas las herramientas disponibles en el sitio. Esto debe pasar antes de que las herramientas sean configuradas.
- skins.xml
  - Una lista de todas las rutas en el directorio skins, que deben ser agregadas a la herramienta skins cuando el sitio es configurado.
- skins (directorio)
  - Contiene todos los templates, imágenes, y archivos CSS del producto.
- types.xml
  - Una lista de todos los tipos que deben ser definidos en la herramienta portal_types.
- types (directorio)
  - Un directorio que contiene otros archivos XML, uno por cada tipo listado en types.xml. Cada archivo contiene información de la configuración relacionada al tipo de contenido específico para el cual es nombrado.
- factorytool.xml
  - Se encarga de crear los objetos en listados, los cuales son tipos de contenido.
- metadata.xml
  - Donde se pone la versión.
- rolemap.xml
  - Se declaran los roles a utilizar y el mapeo de permisos generados, podemos generan nuevos permisos.

Estas son algunas de los archivos de GenericSetup hay otros pero no fueron utilizados, por lo que si se desea saber más al respecto puede consultarse [37].

5.5. Guardar y cargar configuraciones

Una de las nuevas características que son de gran ayuda para el usuario final de los productos, tanto de votación como el de selección de usuarios, es poder guardar la configuración de la votación, y poder guardar la configuración de las condiciones para seleccionar a los usuarios.

En el manual de cada producto se muestra como utilizar está característica, lo que nos concierne en la sección actual, es hablar acerca de la forma en que se logro.

5.5.1. *Python*: *cPickle*

Pickling es el proceso mediante el cual un objeto de Python es convertido a un flujo de bytes, y ***unpickling*** es la operación inversa, donde un flujo de bytes se convierte en un objeto. Esto también se conoce como **serialización**, ***marshalling*** o ***flattening*** [45].

El módulo **cPickle** soporta serialización y deserealización de objetos *Python*, otorgándonos una interface y funcionalidad casi idéntica a la del módulo **pickle**. Pero hay varias diferencias, la más importante está en el rendimiento y subclases.

Comencemos diciendo que, cPickle puede ser hasta 1000 veces más rápido que pickle [45], por que está implementado en **C** [48]. Además, en el módulo cPickle, Pickler() y Unpickler() son funciones, no clases. Esto significa que no podemos usarlas para derivar subclases de pickling y unpickling. La mayoría de las aplicaciones no tienen necesidad de derivar estas subclases así que es preferible utilizar cPickle y obtener el alto rendimiento de dicho módulo.

Para utilizar está herramienta lo que hacemos es importar el módulo cPickle, si no se encuentra disponible entonces hay que hacer uso de pickle.

```
security.declareProtected(VER_RESULTADOS_PERMISSION, 'guardaPreferencias')
def guardaPreferencias(self, tipoUsuarios, condiciones):
    try:
        import cPickle as pickle
    except ImportError:
        import pickle
    output = open('data.pkl', 'wb')
    # Pickle the list using the highest protocol available.
    pickle.dump(tipoUsuarios, output, -1)
    print 'Adentro guardaPreferencias tipoUsuarios'
    print tipoUsuarios
    # Pickle the list using the highest protocol available.
    pickle.dump(condiciones, output, -1)
    print 'Adentro guardaPreferencias condiciones'
    print condiciones
    output.close()
```

Como podemos observar utilizamos un archivo para guardar los datos serializados y para cargarlos, las respectivas llamadas son: **dump** y **load**.

5.6. Producto de selección de usuarios

Ya he dado el marco para comenzar con los pasos específicos de cada producto, aunque comparten la mayoría de los pasos los productos de ATSelectUsers y ATVotaciones he preferido separarlos en dos secciones.

Como ya se mencionó en este capítulo, el producto de selección de usuarios se tuvo que empaquetar como un *egg*, pero para que esto pudiera ser, se necesitó un paso anterior, que es haber migrado de

forma básica el producto de *Plone* 2 a *Plone* 3, con esto quiero decir cambiar bibliotecas, dejar de usar productos como ***CMFMember***, y generar un parche para algunas secciones del sistema. Una parte de lo anterior fue cubierto en el capítulo anterior.

Bien, comencemos con el problema llamado ***CMFMember***. Este es un producto de *Plone* 2 el cual administraba los usuarios, este producto ya no es soportado por *Plone* 3 por lo que se tuvo que prescindir de él. Suena un tanto sencillo, pero no lo es.

La mayor parte de las características del producto de selección de usuarios dependían completamente del producto ***CMFMember*** tanto en la clase principal, como en la herramienta principal que se genera de la clase Votaciones del producto de votaciones, así como también de los scripts de *python*, las macros, templates, etc. Como lo dije, casi de todo.

Pero el flujo del proceso de selección de usuarios es bueno, por lo que decidí dejarlo como estaba, por lo tanto no trabaje desde cero si no que rediseñe todo lo mencionado dejando como base el proceso.

Una forma de lidiar con este problema *CMFMember* fue refactorizar la lista de usuarios para que mostrara los grupos de usuarios de *plone*, en vez de llamar a un método de *CMFMember* que regresaba los tipos de usuarios que en realidad son tipos de contenido. Una vez que eliges que grupo o grupos quieres, entonces el producto de selección de usuarios crea todos los usuarios de esos grupos. Esto fue clave en el proceso de rediseño y mejoramiento de los dos productos tanto del de selección de usuarios como el de votaciones ya que el segundo depende del primero.

La forma de lidiar con este dolor de cabeza llamado *CMFMember* es utilizar interfaces y adaptadores, las cuales son características de *Zope* 3, todo esto entra en el rediseño.

5.6.1. Interfaces

Las interfaces son una parte primordial de la mayoría de técnicas de *Zope* 3. Se pueden concebir las interfaces como documentación verificable, descripciones de componentes y su comportamiento el cual puede ser inspeccionado en tiempo de ejecución. La forma más simple de una interface se conoce como una interface de marca (marker interface), está describe el tipo de un componente sin prometer ningún método o atributo.

Debido a que *Python* no tiene interfaces dentro del corazón del lenguaje, las definimos usando clases que se heredan de zope.interface.Interface. Veamos un ejemplo de código:

```
from zope.interface import Interface
class IATSelectUsers(Interface): """ """
```

Por convención, las interfaces se encuentran en un modulo llamado interfaces, y tiene nombres que comienzan con la letra I. El diseño del sistema puede entonces ser modelado usando interfaces, haciendo uso de la especialización (herencia) y la asociación (composición).

Las interfaces pueden también describir métodos y atributos. Hay que tomar en cuenta que los métodos que están declarados en la interfaz no llevan el parámetro self, ni cuerpo de método pero si pueden tener docstrings, que sirven como documentación del método.

Las interfaces son típicamente implementadas por clases. Los objetos de estas clases se dice entonces que proveen dichas interfaces. Esto implica que el objeto posea todos los métodos y atributos que promete la interfaz. Ejemplo:

```
from zope.interface import implements
from Products.ATSelectUsers.interfaces import IATSelectUsers
class ATSelectUsers(BaseContent, historico):
    implements(IATSelectUsers)
```

Estas lineas son parte de la clase ATSelectUsers que es la base del producto, como vemos en el código, la clase implementa la interfaz IATSelectUsers, por lo tanto tiene los métodos de la interfaz, pero falta una cosa y es implementarlos, de lo cual se encarga la clase ATSelectUsers.

5.6.2. Adaptadores

Zope no hace ningún requerimiento especial en componentes de contenido [39]. Ellos no tienen que implementar una interfaz en especial, ni métodos o atributos se requieren para que un componente pueda ser usado en *Zope*. Por un lado, esto nos permite utilizar cualquier componente de terceros en *Zope*. Por otro lado, esto hace difícil para los componentes esperar cierta funcionalidad entre unos y otros. Un ejemplo de adaptador es el siguiente: tenemos por un lado un mouse con un conector USB y tenemos una PC con una entrada PS/2, para poderse comunicar necesitamos un adaptador que conecte estos dos tipos distintos de conectores.

Los adaptadores en *Zope* funcionan como el ejemplo anterior. Aún si un objeto no provee una cierta API directamente, un adaptador puede estar disponible para él. Los adaptadores nos permiten extender componentes existentes sin tener que cambiar el código original.

Veamos el diagrama de clases del patrón de diseño ADAPTER (Fig. 5.10).

5.6.3. Usando adaptadores

Veamos la forma en que se utilizaron los adaptadores en el producto de selección de usuarios. Entonces vamos a adaptar un objeto IAdaptUsersInfo en uno que provea IUsersInfo. Los adaptadores son típicamente clases simples. Un adaptador necesita tomar tantos argumentos como objetos son adaptados. En caso de una adaptación simple, es una convención nombrar el argumento `context` y almacenarlo como el atributo `context` en el objeto adaptador.

Antes de que podamos utilizar el adaptador, se necesita registrarlo usando el API de registro que se encuentra en `zope.component`. Entonces podemos adaptar el objeto AdaptUsersInfo a IUsersInfo. Esto se realiza llamando a la interfaz con el objeto como un argumento. Pensemos en esto como un cast de tipos de un lenguaje de programación.

```
from zope.component import provideAdapter
from Products.ATSelectUsers.content.adaptador import AdaptUsersInfo
from Products.ATSelectUsers.interfaces import IUsersInfo
a = AdaptUsersInfo()
provideAdapter(SelectUsersInfo)
adap = IUsersInfo(a)
```

Ahora tenemos un objeto que provee IUsersInfo, así que podemos llamar los métodos que promete la interfaz. Por como sabemos este es nuestro adaptador así que sabemos que salida esperar. Aquí terminamos con lo referente a está nueva característica del producto de selección de usuarios.

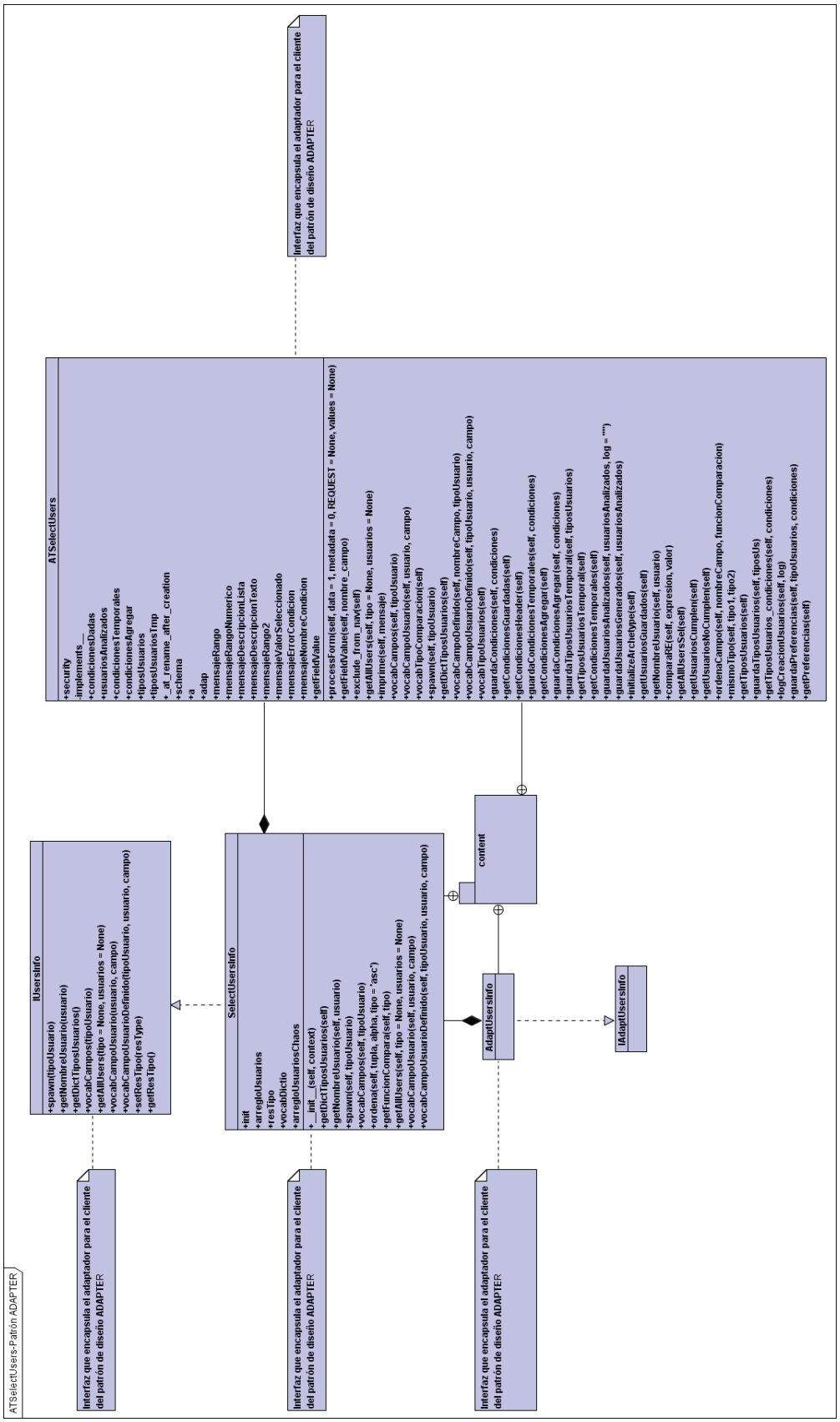


Figura 5.10: Diagrama de clases - Patrón ADAPTER

5.6.4. Usando GenericSetup - ATSelectUsers

Veamos como se configuro el producto de selección de usuarios:

- toolset.xml: En este producto no se declaro ninguna herramienta.
- skins.xml (Fig. 5.11)

```
<?xml version="1.0"?>
<object name="portal_skins" meta_type="CMF Skins Tool">
 <object name="ATSelectUsers" meta_type="Filesystem Directory View"
    directory="Products.ATSelectUsers:skins/ATSelectUsers"/>
 <skin-path name="*">
  <layer name="ATSelectUsers" insert-after="custom"/>
 </skin-path>
</object>
```

Figura 5.11: skins.xml - ATSelectUsers

- skins (directorio)
  - Contiene todos los templates, imágenes, y archivos CSS del producto.
- types.xml (Fig. 5.12)

```
<?xml version="1.0"?>
<object name="portal_types" meta_type="Plone Types Tool">
 <property
    name="title">Controla los tipos de contenido disponibles en el portal</property>
 <object name="ATSelectUsers"
    meta_type="Factory-based Type Information"/>
</object>
```

Figura 5.12: types.xml - ATSelectUsers

- types (directorio)
  - Como se puede ver en la figura 5.13, la clase historico se declara dentro de los action de este archivo de configuración de GenericSetup; historico no se registra sólo se inicializa dentro de su módulo. Los desarrolladores de *Plone* deben de tomar nota especial de esto último.

```
<?xml version="1.0"?>
<object name="ATSelectUsers"
   meta_type="Factory-based Type Information"
   xmlns:i18n="http://xml.zope.org/namespaces/i18n">
 <property name="title">ATSelectUsers</property>
 <property
    name="description"> ATSelectUsers type.</property>
 <property name="content_icon">document_icon.gif</property>
 <property name="content_meta_type">ATSelectUsers</property>
 <property name="product">ATSelectUsers</property>
 <property name="factory">addATSelectUsers</property>
 <property name="immediate_view">vistaBaseSeleccion</property>
 <property name="global_allow">True</property>
 <property name="filter_content_types">True</property>
 <property name="allowed_content_types"/>
 <property name="allow_discussion">False</property>
 <alias from="(Default)" to="vistaBaseSeleccion"/>
 <alias from="edit" to="base_edit"/>
 <alias from="sharing" to="@@sharing"/>
 <alias from="view" to="vistaBaseSeleccion"/>
 <action title="View" action_id="view" category="object" condition_expr=""
    url_expr="string:${object_url}/view" visible="True">
  <permission value="View"/>
 </action>
 <action title="Edit" action_id="edit" category="object" condition_expr=""
    url_expr="string:${object_url}/edit" visible="True">
  <permission value="Modify portal content"/>
 </action>
 <action title="historico" action_id="historico" category="object" condition_expr=""
    url_expr="string:${object_url}/historico" visible="True">
  <permission value="VER_BITACORAS_PERMISSION"/>
 </action>
</object>
```

Figura 5.13: ATSelectUsers.xml

- factorytool.xml (Fig. 5.14)

```
<?xml version="1.0"?>
<object name="portal_factory" meta_type="Plone Factory Tool">
 <factorytypes>
  <type portal_type="ATSelectUsers"/>
 </factorytypes>
</object>
```

Figura 5.14: factorytool.xml - ATSelectUsers

5.7. Producto de votaciones

El producto *ATVotaciones* depende del producto de selección de usuarios, por lo que para poder lograr la primer votación exitosa aunque está fuera solo de usuarios del mismo grupo, se tuvo que dejar el producto ATSelectUsers funcionando. El proceso de migración fue casi idéntico al convertirlo a un empaquetado de tipo *egg*, como ya se menciono con su migración básica adecuada. La configuración mediante GenericSetup también es parecida, pero hay diferencias importantes en este punto ya que se declaran tres tipos de contenido, uno de ellos incluso se declara también como una herramienta.

5.7.1. Usando GenericSetup - ATVotaciones

Es el turno de la configuración del producto de votaciones, hay puntos delicados que los desarrolladores deberían tomar en cuenta para no perder tiempo y esfuerzo.

- toolset.xml
  - Aquí hay algo importante estamos declarando a Votaciones como una herramienta, pero veremos en types que también es un tipo de contenido y que se registra (Fig. 5.15).

```
<?xml version="1.0"?>
<tool-setup>
  <required tool_id="portal_votaciones"
            class="Products.ATVotaciones.content.Votaciones.Votaciones"/>
</tool-setup>
```

Figura 5.15: toolset.xml - ATVotaciones

- skins.xml (Fig. 5.16)

```
<?xml version="1.0"?>
<object name="portal_skins" meta_type="CMF Skins Tool">
 <object name="ATVotaciones" meta_type="Filesystem Directory View"
    directory="Products.ATVotaciones:skins/ATVotaciones"/>
 <skin-path name="*">
  <layer name="ATVotaciones" insert-after="custom"/>
 </skin-path>
</object>
```

Figura 5.16: skins.xml - ATVotaciones

- skins (directorio)
  - Contiene todos los templates, imágenes, y archivos CSS del producto.
- types.xml (Fig. 5.17)

```
<?xml version="1.0"?>
<object name="portal_types" meta_type="Plone Types Tool">
 <property
    name="title">Controla la disponibilidad de tipos de contenido en el
portal</property>
 <object name="ATVotaciones"
    meta_type="Factory-based Type Information"/>
 <object name="FirmaDigital"
    meta_type="Factory-based Type Information"/>
 <object name="Votaciones"
    meta_type="Factory-based Type Information"/>
</object>
```

Figura 5.17: types.xml - ATVotaciones

- types (directorio)
  - Comencemos con el tipo que también es una herramienta.

```xml
<?xml version="1.0"?>
<object name="Votaciones"
   meta_type="Factory-based Type Information"
   xmlns:i18n="http://xml.zope.org/namespaces/i18n">
 <property name="title">Votaciones</property>
 <property
    name="description"> Votaciones type.</property>
 <property name="content_icon">document_icon.gif</property>
 <property name="content_meta_type">Votaciones</property>
 <property name="product">ATVotaciones</property>
 <property name="factory">addVotaciones</property>
 <property name="immediate_view">base_view</property>
 <property name="global_allow">True</property>
 <property name="filter_content_types">False</property>
 <property name="allowed_content_types"/>
 <property name="allow_discussion">False</property>
 <alias from="(Default)" to="base_view"/>
 <alias from="view" to="base_view"/>
 <action title="View" action_id="view" category="object" condition_expr=""
    url_expr="string:${object_url}/view" visible="True">
  <permission value="View"/>
 </action>
</object>
```

Figura 5.18: Votaciones.xml - ATVotaciones

- Veamos los dos restantes pero no menos importantes. Creó que es tiempo de dar el siguiente tip: No confundan la propiedad product con content_meta_type, la primera debe de ser el nombre del producto global y no el nombre del tipo de contenido declarado que en este caso es Votaciones (Fig. 5.18).
- Aquí surgió uno de los problemas que quitaron mucho tiempo y fue que dado que ATVotaciones se declara como un BaseFolder entonces podemos agregar adentro del objeto otros tipos de contenido y es lo que se hacía dentro de la misma clase, pero con GenericSetup no es así, se tiene que poner a False la propiedad filter_content_types para que no filtre los tipos de contenido permitidos, está propiedad está ligada con allowed_content_types (Fig. 5.19, 5.20). La solución a esto no se encontró en documentación en la red así que es un buen tip.

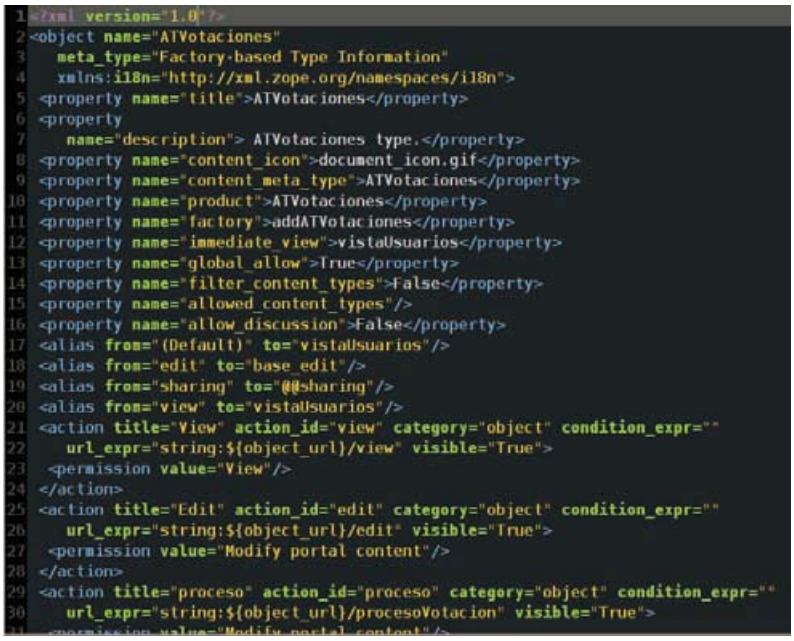


Figura 5.19: ATVotaciones.xml - ATVotaciones

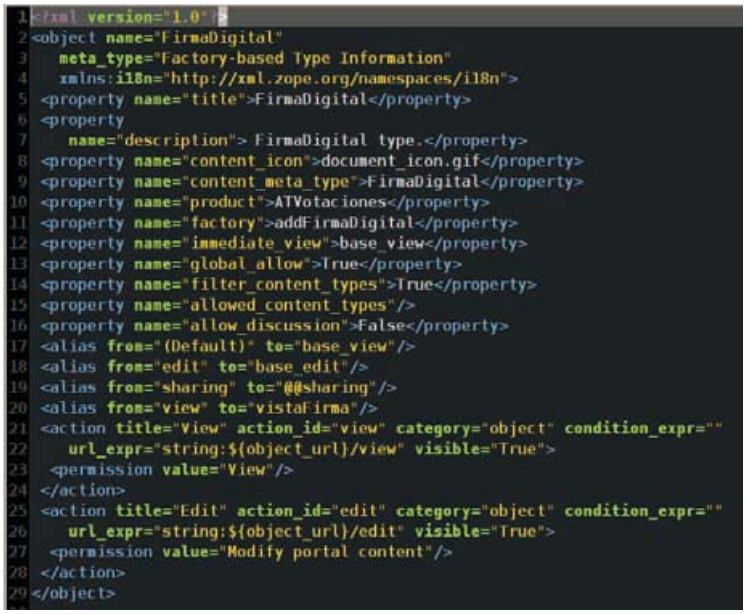


Figura 5.20: FirmaDigital.xml

- factorytool.xml
  - Veremos la diferencia en el número de tipos listados. Esto se debe a que hay más tipos de contenidos declarados en el producto ATVotaciones (Fig. 5.21).

```
<?xml version="1.0"?>
<object name="portal_factory" meta_type="Plone Factory Tool">
 <factorytypes>
  <type portal_type="ATVotaciones"/>
  <type portal_type="FirmaDigital"/>
  <type portal_type="Votaciones"/>
 </factorytypes>
</object>
```

Figura 5.21: factorytool.xml - ATVotaciones

- rolemap.xml
  - PENDIENTE

5.7.2. Votar por N candidatos

En el estado del arte, en específico en el sistema de votacion del *IMATE* en su primer versión, podemos percatarnos de que el elector sólo puede votar por un candidato de los N posibles. En está nueva versión del sistema de votaciones tenemos dos mejoras en el protocolo de votación, podemos, como electores, votar por 1 o más candidatos, y la segunda mejora nos permite asignar a cada elector un número de votos X, los cuales puede repartir entre los N candidatos como desee (Fig. 5.22).

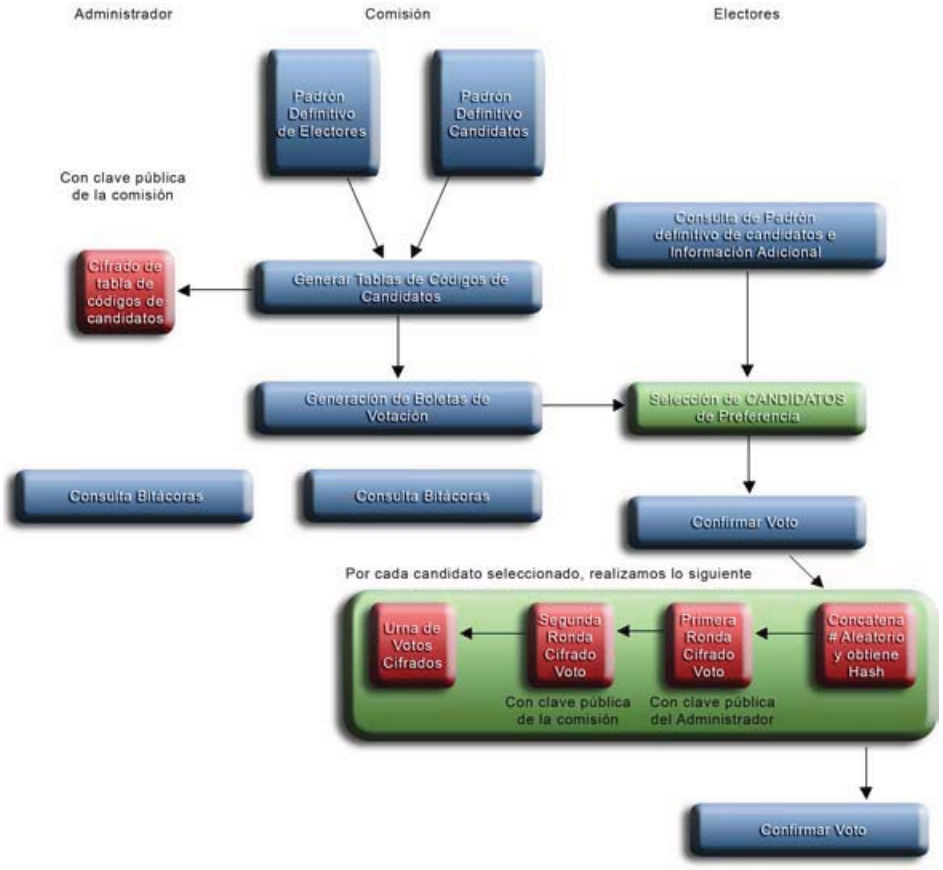


Figura 5.22: Diagrama del nuevo algoritmo

5.7.2.1. Análisis

En el protocolo de votación tenemos 5 partes esenciales [23], que son:

Algorithm 5.1 Protocolo de votación

1. Identificación de los códigos de candidatos que registrarán los electores como votos.
2. Generación de Boletas de Votación.
3. Registro de Voto por parte de un elector autenticado.
4. Recibo del voto para el elector.
5. Escrutinio.

La tabla donde se encuentran los códigos de candidatos (1), no sufre ningún cambio. Lo que tiene la tabla, es un código por candidato para cada elector, por lo tanto tenemos N códigos para un elector, nos podemos dar cuenta que con está tabla es suficiente. Lo que busca la nueva característica, es poder votar por un candidato o más, incluso por todos; y dado que en la tabla ya viene un código por cada candidato, entonces se concluye que no se necesita generar ningún cambio.

Elector/Candidato	num1	num2	num3	num4	num5
Ana	27	90	14	25	8
Edith	18	10	45	91	33
Christian	15	2	99	85	72

Cuadro 5.1: Códigos de candidato

Para generar las boletas (2), se utiliza la tabla de códigos de candidato, a cada elector se le asigna una boleta que contiene un código por cada candidato. Por lo tanto, tampoco hay cambios en la boleta.

BOLETA					
Ana	27	90	14	25	8

Cuadro 5.2: Boleta del usuario **Ana** con códigos de candidato

Cuando el elector se encuentra en la pantalla donde puede ejercer su voto (lo cual es parte del tercer punto principal del protocolo de votación), se le muestra la siguiente información por cada candidato: imagen del candidato (si la tiene), nombre completo y un **radio button**. La interfaz web **no permite seleccionar más de un candidato**, esto es por diseño de la interfaz. Por lo tanto se tuvo que modificar la interfaz para que se pudiese seleccionar más de un candidato, para esto se utilizo checkboxes en lugar de radio buttons, además de modificar el código de javascript que realizaba las validaciones pertinentes, que eran: seleccionar por lo menos un candidato, obtener el código del candidato seleccionado y preguntar al usuario por la confirmación de su voto. El código javascript ha sido modificado para lo siguiente: verificar que por lo menos se seleccione un candidato, obtener el código de todos los candidatos seleccionados y asignarlos a una variable apropiada, y por último preguntar al usuario por confirmación.

Una vez que tenemos los códigos de los candidatos seleccionados, entramos a un método primordial en el protocolo (3): **haz_voto**.

Las principales funciones de este método se enlistan a continuación y en estricto orden de ejecución:

Algorithm 5.2 haz_voto - ATVotaciones

1. haz_voto

   a) Obtener: boleta elector, id del elector y la lista de los electores.

   b) Crear el voto

   c) Obtener el hash del voto

   d) Cifrar el voto

   e) Administrar votos no utilizados

   f) Insertar el voto en la urna

   g) Marcar al elector de que ha emitido su voto

   h) Regresar datos para el recibo

La modificación que se realizó en este método, para que pudiera otorgar la nueva funcionalidad es que, por cada candidato seleccionado por el votante se realizan las tareas: **b**, **c**, **d** y **f**.

Algorithm 5.3 haz_voto modificación 1

a) Obtenemos: la boleta del elector, el id del elector, y la lista de los electores.

Por cada candidato seleccionado por el votante realizamos los siguientes cuatro pasos:
b) Crear el voto
c) Obtener el hash del voto
d) Cifrar el voto
f) Insertar el voto en la urna

Al terminar de insertar todos los votos en la urna realizamos los siguientes tres pasos:
e) Administrar votos no utilizados
g) Marcar al elector de que ha emitido su voto
h) Regresar datos para el recibo (que incluye el hash de cada voto creado)

El único cambio respecto al recibo que se le entrega al elector es el siguiente: antes sólo se regresaba el hash del único voto posible, ahora se regresa una lista con el hash de cada candidato que el elector haya seleccionado, por lo que podrá seguir verificando que su voto haya contado, por lo tanto no afecta las propiedades del protocolo anterior.

El escrutinio no percibe ningún cambio, por que sólo se encarga de descifrar la urna que contiene los votos, contarlos y eliminar los incorrectos.

5.8. Conclusiones

El rediseño del sistema de votaciones trajo consigo mejoras significativas. Las mejoras en el contexto de *Plone* y *Zope* son: el uso de eggs para empaquetar y distribuir cada uno de los productos (selección de usuarios y votaciones), esta forma de distribuir y empaquetar es el estándar en *Plone* 3. Además de utilizar el *framework GenericSetup* para configurar los productos de selección de usuarios y de votaciones.

Para poder cargar y guardar las configuraciones se utilizo en primera instancia cPickle, pero se está analizando dejar cPickle como back-end y poner un widget en el front-end. CPickle es lo que utiliza ZODB que es la base de datos de ZOPE. El producto de selección de usuarios fue mejorado gracias al rediseño y refactorización, ya que mediante la refactorización se mejoraron varios métodos

internamente y mediante el rediseño se logró utilizar patrones de diseño. El producto de votación fue analizado para incorporarle la funcionalidad de poder votar por N candidatos, este fue un análisis de diseño de software no de seguridad.

Capítulo 6

INTEGRACIÓN DEL SISTEMA DE VOTACIONES CON *infoMATEM*

FacultyStaffDirectory es un directorio de personal, un proveedor de espacios de trabajo compartidos para comités, una forma de mantener la pista de las áreas de especialización de las personas. Se integra con la infraestructura de los usuarios y grupos de *Plone*, y soporta un framework extensible para requerimientos personalizados [40].

El diagrama de componentes del sistema de votación incluyendo la integración con *infoMATEM* es el de la Figura 6.1.

6.1. *FacultyStaffDirectory*

FacultyStaffDirectory provee tipos de contenido para la creación y detalles de organización de las personas. Ha sido desarrollado principalmente para directorios de personal en instituciones educativas pero puede ser extendido para utilizar en una variedad de configuraciones.

El tipo de contenido principal es `Person`. Este tipo de contenido incluye una gran variedad de campos (email, número de teléfono, trabajos, etc). Podemos agregar fácilmente las nuestras.

Al instalar el producto *FacultyStaffDirectory* ya vienen tres tipos de clasificaciones (Classifications) que pueden ser asignadas a objetos `Person`: `Faculty, Staff` y `Graduate Student`. Si estas no te funcionan, podemos agregar nuestras propias clasificaciones (e.g. Becarios, Técnicos, o lo que se desee).

FacultyStaffDirectory también provee varios tipos de contenido para agrupar personas: Departments, Specialties y Committes. Si estas etiquetas no tienen sentido en su organización, podemos fácilmente renombrarlas, en cada caso, la asociación entre la persona (instancia de Person) y la agrupación (e.g. la relación Person-Specialty) puede tener una descripción. Así que por ejemplo, si la persona Iván Cervantes está en la especialidad Inteligencia Artificial, podemos darle una descripción a la relación `Iván Cervantes-Inteligencia Artificial` (e.g. "Interesado en la confiabilidad de la prueba de Turing").

FacultyStaffDirectory puede ser configurado tal que las personas agregadas al directorio automáticamente se convierten en miembros (Members) de tu sitio *Plone* y cada persona pude editar su propia página dentro del sitio. También podemos agregar nuevos roles, para facilitar la administración de las personas. Por ejemplo, el rol de `AdministradorDePersonal` puede crear nuevas especialidades (Specialities) y asignarles personas.

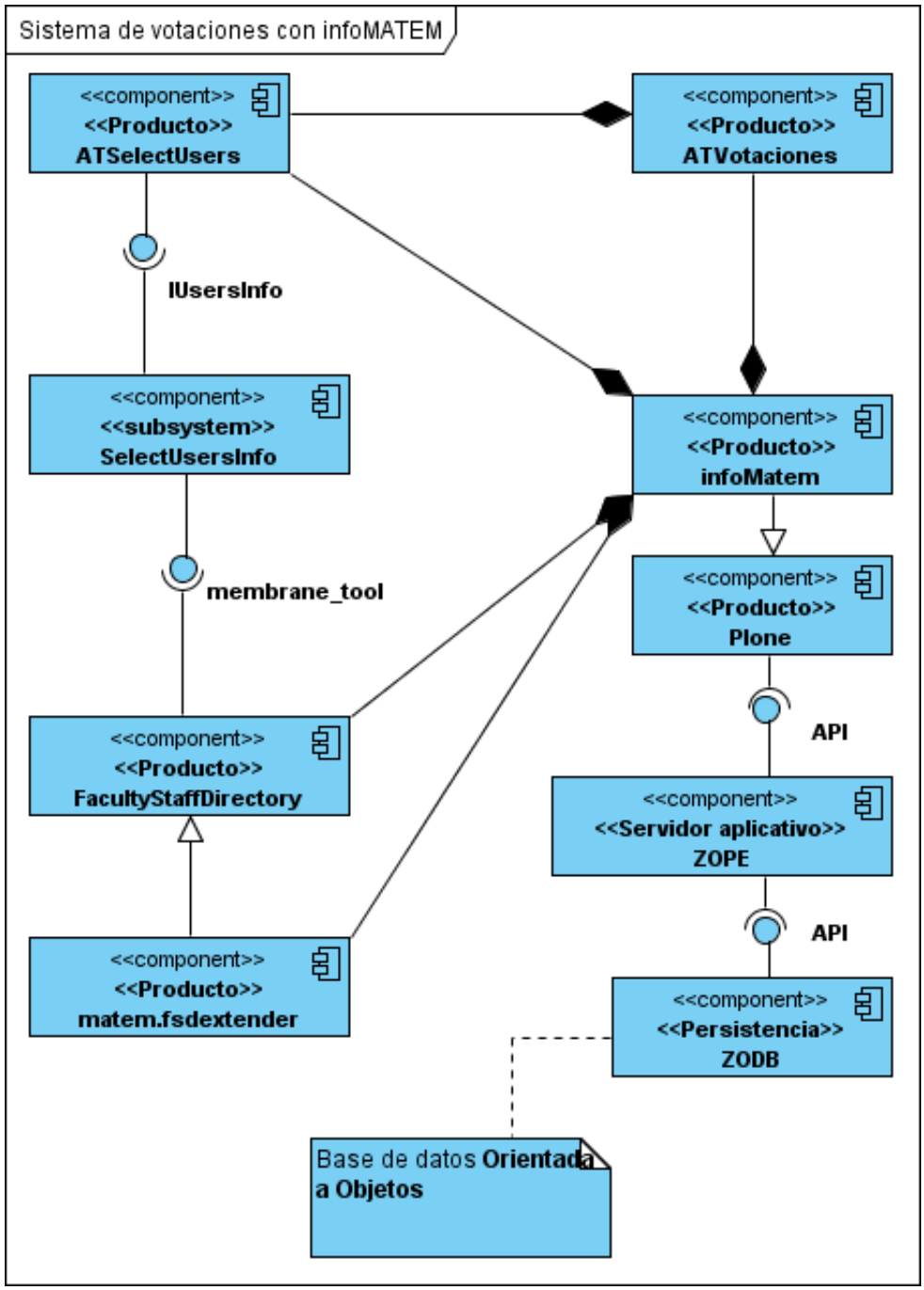


Figura 6.1: Componentes del sistema con infoMATEM

6.1.1. Integración de miembros

FacultyStaffDirectory ofrece la siguiente integración con usuarios y grupos de *Plone*:

- El directorio Faculty/Staff actúa por si mismo como un grupo.
  - Todos los objetos Person creados en el directorio Faculty/Staff son automáticamente con-

siderados miembros de su grupo. Este grupo también provee la opción de asignar un rol global a todas las personas en el directorio. Esta opción debería ser manejada con cuidado. Es generalmente mejor seleccionar el rol `Member`, ya que está es la opción más restrictiva.

- Departamentos, Clasificaciones y Comités actúan como grupos.
  - La asignación de roles globales no está disponible para estos tipos de contenido, pero los grupos que ellos definen tal vez otorguen roles locales a través del sitio *Plone*. Para unidades académicas complejas, esto puede ayudar a ahorrar tiempo, ya que la administración de personal va de la mano con la administración de la seguridad.
- Los objetos que instancian `Person` actúan como usuarios.
  - El configlet del directorio faculty/Staff en la configuración del sitio *Plone* nos permite escoger si los objetos `Person` proveen paswords para la autenticación. Si se está utilizando algún otro plugin PAS, como PloneLDAP, ApachePAS, PubcookiePAS o CAS4PAS, se necesitara deshabilitar el proveer password así que la autenticación sea delegada a estos sistemas mencionados.

Los usuarios definidos por objetos de `Person` tienen automáticamente el rol de `Owner` localmente para el objeto y su contenido. Esto permite a los usuario agregar y editar su biografía, información personal, etc. Ellos también pueden controlar los permisos para compartir sus objetos y pueden entonces permitir asistencias para editar contenido sin necesidad de su password o preferencias de usuario.

La acción `My Folder`, que se encuentra en la barra de herramientas de cada usuario, es alterada por el producto *FacultyStaffDirectory* para tomar usuarios definidos por objetos Person directamente. Los usuarios definidos a través de la UI estándar de *Plone* serán tomados de la ubicación usual (portal/Members). Igualmente, el enlace de preferencias de personal que se encuentra en la barra de herramientas y en el panel de preferencias de miembros de *plone* o el dashboard tomara usuarios `Person` a sus objetos `Person`.

Los propietarios no tienen el permiso para añadir o remover su objeto Person de Departamentos, Comités, Clasificaciones y Especialidades, ya que estas colecciones son utilizadas como grupos de autorización. Este permiso es reservado para administradores de sitio y para el rol nuevo `Personnel Manager`, instalado con el producto *FacultyStaffDirectory*.

6.1.2. Extensibilidad

Dado que cada organización tiene muy pocos requerimientos únicos, *FacultyStaffDirectory* soporta un mecanismo de extensión basado en la librería archetypes.schemaextender. Usándolo, se pueden escribir productos plugin los cuales añaden campos a los tipos de contenidos que nosotros creemos o a los que ya tenemos con la instalación de *FacultyStaffDirectory*.

6.2. Pruebas

Conforme un sistema crece se incrementa el tiempo que toma realizar pruebas manuales. Hay un momento cuando necesitamos o queremos automatizar las pruebas, dejando que la computadora realice dichas pruebas.

Tradicionalmente las pruebas automáticas se definen en tres categorías:

1. Pruebas unitarias (Unit tests): las cuales aislan un componente específico y prueba únicamente la funcionalidad de ese componente. Es el tipo de prueba más común y eficiente, se debe principalmente a que las circunstancias del entorno no afectan la funcionalidad del componente y su prueba.

Confiar en otros componentes para pruebas, significa que cualquier bug potencial que estos componentes contengan, podrían hacer que la prueba falle, aun si el componente que está siendo probado este correcto. Las pruebas unitarias evitan estas dependencias, si un componente tiene que depender o confiar en otro componente debido a su naturaleza (adaptadores), es una practica común escribir un objeto simulado (mock object) que pretenda ser el objeto necesitado pero en realidad sólo implemente cascarones.

2. Pruebas de integración: Estas pruebas aseguran que la interacción entre componentes trabaje como se espera. Mientras las pruebas unitarias cubren la responsabilidad individual, las pruebas de integración cubren un conjunto de componentes integrados en la aplicación. Las pruebas de integración únicamente tienen sentido cuando ya hemos realizado las pruebas unitarias de todos los componentes que interactúan en la prueba. Si no se cuenta con pruebas unitarias y una prueba de integración falla, no podremos decir si falló a causa de uno de los componentes o si falló a causa de que la integración no funciona. Las pruebas de integración en *Zope* son llamadas pruebas funcionales, haciendo difícil diferenciarlos de pruebas funcionales reales.

3. Pruebas funcionales: Tratan una aplicación como una caja negra y no toman encuenta detalles de implementación. Ellas ven lo que los usuarios ven y realizan todo lo que un usuario haría con la aplicación, la prueba podría por ejemplo, operar simplemente como un explorador web operado por el usuario.
   Las pruebas funcionales son independientes de la plataforma de desarrollo y del lenguaje de programación, especialmente en el caso de aplicaciones web. Hay varios kits para pruebas funcionales de aplicaciones web, uno de los más populares es Selenium [49].

Tener pruebas automáticas no es garantía para un producto de software de alta calidad. Los casos de prueba (test cases) pueden no ser efectivos y por lo tanto sin utilidad si ellas no cubren circunstancias realistas. En la comunidad de *Zope*, la siguiente filosofías de prueba han sido probadas con exito:

- Cada vez que haya un cambio en la aplicación, toda la suite de pruebas debe ejecutarse para checar si la aplicación completa sigue funcionando.
- Cuando una nueva característica es añadida, un caso de prueba debe ser escrito para asegurarse que toda la funcionalidad nueva es cubierta por las pruebas. No asegurarse solamente de que funciona ahora si no que también en el futuro.
- Cuando un bug es encontrado debido a las pruebas automáticas y es arreglado, las pruebas cubren esa característica particular debería ser revisada y extendida a ejercicios de acción que lleve a descubrir un bug. Esto asegura que la solución al bug de verdad funciona como es debido.

Esta es una correcta combinación que nos resulta en un buen aseguramiento de la calidad (QA).

6.2.1. Pruebas unitarias

Las pruebas unitarias son el tipo de pruebas más común, sin embargo, no son una característica de *Zope*. *Python* viene con soporte para pruebas unitarias en el módulo unittest, una contribución del proyecto *PyUnit*. Este administra las pruebas agrupándolas en diferentes niveles:

- Una "prueba" es una mínima, prueba atómica para una funcionalidad en particular de un componente.
- Un "caso de prueba" es un grupo de pruebas que prueban la funcionalidad de un componente en particular.
- Una "suite de pruebas" es una colección de casos de prueba, usualmente las del módulo o paquete.

Debido a que los otros tipos de pruebas que veremos confían en la infraestructura provista por el módulo unittest, estas categorías también aplican a ellas.

En la infraestructura de las pruebas basadas en el módulo unittest, los casos de prueba son clases que derivan de unittest.TestCase. Cada método en dicha clase que comience con test es una prueba y deberá ser llamada, cuando el caso de prueba sea ejecutado. Además, TestCase provee un número de métodos útiles para aserciones y otras circunstancias. Consideremos un caso de prueba sencillo, en el que verificamos el nombre del administrador.

```
ian@ian-laptop:/opt/Plone-3.1/juno$ ./bin/zopepy
>>> import unittest
>>> class PruebaTestCase(unittest.TestCase):
... def test_usuario_valido(self):
... self.assertEqual('Ivan'+'Cervantes', 'IvanCervantes')
...
>>> unittest.main()
.
----------------------------------------------------------------------
Ran 1 test in 0.000s
OK
```

6.2.2. Pruebas de integración

Como mencionamos en la introducción de está sección, en el mundo de *Zope* las pruebas funcionales son realmente pruebas de integración. Las pruebas funcionales tratarías a *Zope* como una caja negra. Pero las "pruebas funcionales" en *Zope*, no simulan un programa como *Firefox* y se conectan a través del puerto *HTTP*. Ellas únicamente simulan los objetos necesarios, como son la petición del explorador. Que significa, que son más como pruebas de integración. Las pruebas de funcionalidad de *Zope* son buenas para probar comportamiento que no está cubierto por las pruebas unitarias. La reacción de dar clic en un botón en una forma de HTML no puede ser cubierta por unas pruebas unitarias, pero si por pruebas de integración o funcionales.

6.2.2.1. Capas de prueba

Las pruebas de integración no registran los objetos mock que utiliza. Si está fuera una prueba unitaria, esperaríamos que lo hiciera en los métodos setUp y tearDown. En pruebas de integración, una limpieza de objetos mock como adaptadores es realizada, lo que significa que los registros del adaptador serán eliminados. Todo el alambrado de las pruebas tendrá que ser inicializado y detenido por cada prueba sencilla. Mientras que esto no es un problema en pruebas unitarias, significa rendimiento pobre en pruebas de integración donde el alambrado de la prueba es básicamente la configuración completa de la aplicación.

El que ejecuta las pruebas en *Zope* tiene que lidiar con el problema del rendimiento en pruebas de integración poniendo diferentes tipos de pruebas en diferentes capas. Las pruebas unitarias son ejecutadas sin capa. Como ya he mencionado, ellas se encargan de inicializar y finalizar individualmente. Las pruebas de integración, son ejecutadas en el contexto de una capa que realiza la inicialización y finalización. Las pruebas de una capa particular se ejecutan juntas y generalmente no se espera que realicen la inicialización y limpieza. Para las pruebas de integración que generalmente significa que los objetos mock deberían evitarlas si es posible. Después de todo, una prueba de integración intenta probar la aplicación como un todo. Si los objetos mock son necesarios, ellos deberían ser registrados a nivel de la capa.

La capa en la que las pruebas de integración son ejecutadas, simplemente carga todo el árbol de configuración de *Zope* 3 *ZCML*. Por lo tanto, el camino más sencillo para registrar un objeto mock para pruebas de integración es vía *ZCML*.

6.2.2.2. Doctests funcionales

Regularmente las pruebas de integración son tediosas de escribir por que necesitan simular petición de explorador u otros tipos de interacción del usuario. Hacer esto desde un programa de *Python* no es sólo difícil, también requiere algunos detalles sobre la forma en que *Zope* publica los trabajos.

Para hacer frente a está dificultad, *Zope* nos permite escribir pruebas de integración como doctests (doctests funcionales). Estas son muy explicitas con respecto a la emulación del cliente por que son esencialmente registros literales de sesiones *HTTP* (*request* y *response*).

Claro que no es nada común que uno escriba en un archivo una sesión completa de *HTTP*. En lugar de esto, podemos utilizar una forma mucho más simple de crear doctests funcionales. El programa *Python tcpwatch* puede interactuar como un intermediario entre un explorador web y un servidor web, y guardar la comunicación entre el cliente/servidor de una sesión completa de *HTTP* en archivos de bitácora. El programa dochttp el cual viene con *Zope* puede convertir estas bitácoras en archivos de texto doctests.

Para crear un doctests funcional mediante el registro de un sesión *HTTP*, hacemos lo siguiente:

1. Creamos un directorio temporal donde tcpwatch pueda escribir los archivos de bitácora, por ejemplo /tmp/tcpwatch ó C:\temporal\tcpwatch en Windows. Entonces ejecutamos el programa tcpwatch con los parámetros apropiados:

2. En caso de que las pruebas involucren componentes que son protegidos mediante seguridad, es recomendable crear un administrador principal con el nombre de usuario mgr y password mgrpw. Esta cuenta es configurada por la prueba funcional y únicamente disponible durante las pruebas. Sin embargo, desde que estamos tratando de hacer una prueba desde una sesión *HTTP* guardada, necesitamos simular cosas como la autenticación de usuarios.
   Para configurar el administrador de pruebas principal, añadimos las siguientes lineas a la instancia.

3. Abrimos las páginas que necesitamos probar en un explorador web. Hay que tener cuidado en el puerto, ya que si no ponemos el de tcpwatch no será guardada la sesión en la bitácora. Cuando preguntamos por autenticación, ingresamos con la cuenta mgr. Todo lo que hagamos con el explorador web no será guardado. De cualquier forma deberíamos intentar poner atención de no abrir muchas páginas que no estén relacionadas con la prueba.

4. El directorio temporal que es utilizado para almacenar los registros de tcpwatch debería contener varios archivos, uno por cada request y uno por cada response realizado. The script dochttp.py de *Python* puede ahora ser utilizado para convertir estos archivos en doctest.

5. Como uno de los últimos pasos, editamos el archivo doctest generado. Añadimos los comentarios específicos y eliminamos los pares de request/response que son irrelevantes a la prueba. Finalmente, creamos un modulo de pruebas que inicialice la suite de pruebas para que la prueba sea ejecutable.

6.2.2.3. Test browser

Los doctests funcionales son buenos para simular un cliente *HTTP*. Aunque, si necesitamos describir y probar un comportamiento típico de un explorador web, entonces se vuelven inconvenientes. Por ejemplo, subir formas *HTML* o tratar con cookies involucra datos no legibles de *HTTP*. Una copia tal cual de la sesión *HTTP* no es tampoco la forma ideal para probar comportamiento de la interfaz de usuario, como corregir ligas, botones de las formas, etc.

Para simular de mejor forma un explorador web, *Zope* nos proporciona un test browser. Este componente disponible desde zope.testbrowser nos deja escribir pruebas funcionales que prueba la aplicación desde un punto de vista de explorador. Hay dos sabores disponibles del test browser:

zope.testbrowser.browser.Browser abre conexiones reales de *HTTP* y puede ser usado para pruebas de funcionamiento real en sistemas puestos en operación. Esta variación de test browser nos permite

escribir pruebas como con *Selenium* o frameworks similares, excepto que las pruebas serán ejecutadas en *Python* y no como *JavaScript* en un explorador web. Si tenemos varias pruebas funcionales, esto hará una gran diferencia en cuestiones de velocidad de ejecución. Este sabor de test browser no está totalmente ligado al servidor aplicativo *Zope*. Este puede ser usado para probar cualquier aplicación web, sea basado en *Zope* o no.

zope.testbrowser.testing.Browser hace interfaz con el publicador *Zope* 3 en lugar de abrir conexiones *HTTP*. Aunque este puede ser utilizado para pruebas de integración en *Zope*, como la función http en los doctests funcionales.

El API de las dos variantes es exactamente la misma. Esto lo hace sencillo de intercambiar entre una prueba funcional y una de integración, y viceversa.

Como con los doctests funcionales tradicionales, tenemos un camino conveniente para registrar pruebas de test browser usando el browser test recorder. Esta grabadora de pruebas está escrita en ECMAScript. Debido al modelo de seguridad de ECMAScript, el grabador de pruebas tiene que ser cargado desde el mismo sitio web de donde queremos grabar las pruebas. Cuando instalamos y configuramos como un paquete en nuestra instancia de *Plone*, podemos acceder a las pruebas grabadas vía una URL, usualmente

http://localhost:8080/@@/recorder/index.html

6.3. Internacionalización

La web es un medio multilenguaje y multicultural. Varios sitios web y servicios que se ofrecen en la web necesitan lidiar con usuarios que tienen una diversidad de lenguajes. Es común para los sitios europeos el manejar cuatro lenguaje o más a la vez. Aún en países como los Estados Unidos, el número de sitios web que tienen audiencias alrededor del todo el mundo, proveen el contenido en Español e Inglés. Como es el caso del sistema *infoMatem* y por lo tanto debe serlo del sistema de votaciones.

La internacionalización cubre más que sólo añadir otro lenguaje al sitio web. Hora y fecha tienen un formato diferente en diferentes países, como también los números. Las aplicaciones web necesitan percatarse de las diferencias. Afortunadamente, *Zope* provee un excelente framework para internacionalización (i18n) y localización (l10n).

Cuando internacionalizamos una aplicación, es usual que la aplicación sea la que necesita ser traducida, y no los datos con los que trabaja la aplicación. En términos de *Zope*, significa objetos de contenido que a menudo no son internacionalizados, las vistas por otro lado lo son.

La mayoría del texto que puede ser traducido es parte de la vista, como lo son los Page Templates (PT). Algunas veces, el texto viene de diferentes componentes que no están relacionados con la vista por si mismos. Es el caso de formas autogeneradas, el schema contienes campos que describen sus campos. En todos los casos, es necesario marcar las cadenas que van a ser traducidas tal que la maquinaria de *Zope* pueda reconocerlas.

6.3.1. Mensajes

En el argot de i18n, las cadenas que necesitan ser traducidas las conocemos como mensajes. Ellas son identificadas por un id el cual puede ser la cadena misma, o un valor abstracto único. El id es utilizado para buscar su traducción, que lo conoceremos como la cadena de mensaje (message string). Si la búsqueda falla, la maquinaria de traducción nos regresara el valor por default. Además, las cadenas de mensaje son buscadas mediante cierto contexto, el dominio. Dominios diferentes mantienen mensajes de diferentes grupos de mensajes. En *Zope*, es una convención que cada paquete add-on o aplicación utilice su propio dominio. Por ejemplo *Plone* tiene el dominio *plone* y *Zope* el dominio zope. El producto de votaciones tiene el dominio atvotaciones y el producto de selección de usuarios tiene el dominio atselectusers.

Mientras más largo sea el mensaje que necesita ser traducido, mayor posibilidad de que sea un id único. Sentencias completas son usualmente únicas y pueden ser usadas como su propio id. En

cualquier caso, es preferible incluir el mayor contexto posible en el mensaje.

La mayoría de las traducciones en *Zope* ocurren de una forma automática por que la presentación en el explorador usualmente trabaja a través de *Page Templates*. Una vez que parte de un texto es marcado como un mensaje, los *Page Templates* automáticamente realizan la traducción. Para saber a que lenguaje hacer la traducción, le preguntan al *request* de la sesión *HTTP*. La mayoría de los exploradores web envían un encabezado especial *HTTP*, *Accept-Language* con el cual le dicen al servidor que lenguaje prefieren. Entonces una utilidad compara la lista de lenguajes preferidos con la lista de lenguajes disponibles y escogen la que más parezca.

6.3.2. Dominios

Para dar un vistazo de como los componentes de *Zope* manejan la traducción, es recomendable poner en practica la teoría en un interprete de comandos. Esto no es lo que necesitamos frecuentemente, pero saber como trabaja nos ayudara a comprender mejor los conceptos detrás de todo esto.

Con el fin de poder trabajar con mensajes, necesitamos configurar un poco de maquinaria. En *Zope*, los componentes que son responsables de la traducción son los dominios de traducción (translation domains). Ellos se encargan del mapeo entre los ids y las cadenas de mensaje de un cierto dominio y proveen una API de traducción la cual está documentada en la interfaz zope.i18n.interfaces.ITranslationDomain.

Los dominios de translación don utilidades con nombre [39] registradas para está interfaz y el nombre del dominio que ellos representan. La forma en que los dominios de traducción obtienen las cadenas traducidas no le importa a la maquinaria de traducción. *Zope* 3 tiene soporte para dominios de traducción mediante catálogos de gettext o de datos almacenados en la ZODB.

Para poder hacer que una cadena o un objeto Unicode estén pendientes de la traducción, necesitan ser convertidos en un objeto Message. Este objeto representa el mensaje con *Zope*, este independiente del id del mensaje, también contiene el dominio al que pertenece y un valor por default. Debido a que el dominio de mensajes en un módulo de *Python* es generalmente el mismo y haciendo una instancia de Message por cada cadena en el módulo es desgastante, es una practica común usar una fabrica de mensajes como un atajo. La fabrica crea mensajes del mismo dominio y es simplemente llamada _ (guión bajo) por convención.

6.3.3. Variables de interpolación

Algunas veces, los mensajes no son estáticos e incluyen valores dinámicos. Imaginemos la sentencia "Usuarios del tipo x en el sistema" donde x es el número que va a ser insertado dinámicamente. Esta tarea trivial se convierte en un problema real cuando necesitamos traducir cadenas. Entonces ¿que traduciríamos? sólo "Usuarios del tipo" y "en el sistema" Esto no es deseado ya que el orden de las palabras puede cambiar en otros lenguajes. No hay garantía de que siempre este en el mismo lugar la sentencia.

Afortunadamente, la maquinaria de traducción provee una solución debido a que este es un problema común. Podemos definir variables en nuestras cadenas de mensaje que después sean interpoladas automáticamente. El objeto id del mensaje contiene los valores para ser interpolado. La sintaxis de la variable sigue las de rutas de expresiones embebidas en Page Templates.

```
11 from zope.i18nmessageid import MessageFactory
12 _ = MessageFactory('atvotaciones')
13
14 baseSchema= BaseSchema.copy()
15 baseSchema['id'].widget.visible={'view':'invisible', 'edit':'invisible'}
16 baseSchema['title'].widget.label="Titulo"
17 baseSchema['title'].write_permission=EDIT_PROPERTIES_GRALES_PERMISSION
18
19 descripcion=  TextField('description',
20         required=0,
21         searchable=1,
22         widget=TextAreaWidget(
23             label=_(u"Descripcion"),
24             description=_(u"Escriba una breve descripcion sobre la eleccion."),
25             ),
26         write_permission=EDIT_PROPERTIES_GRALES_PERMISSION,
27         )
28
29 cuerpo=TextField('cuerpo',
30         searchable=1,
31         default_content_type = zconf.ATNewsItem.default_content_type,
32         default_output_type = 'text/x-html-safe',
33         allowable_content_types = zconf.ATNewsItem.allowed_content_types,
34           widget = RichWidget(label = _(u'Cuerpo de las elecciones'),
35                               description=_(u"Escriba una completa descripcion sobre la
   eleccion. Esta es la cara principal que veran los usuarios que entren al modulo de
   elecciones. Puede elegir diferentes formatos.")),
```

Figura 6.2: MessageFactory

6.3.4. Page Templates

Típicamente los Page Templates contienen muchas cadenas que necesitan ser internacionalizadas. Similar a TAL y METAL las cuales son utilizadas para modificar el árbol de salida de XML, un tercer comando es asignado en otro espacio de nombres para realizar la internacionalización.El prefijo del espacio de nombres es i18n por convención. La siguiente lista nos da un panorama de los comandos i18n en los Page Templates:

- translate le dice al interprete TAL que traduzca el contenido del elemento, usando el id del mensaje explicito o el contenido del elemento. Ejemplo:
  <h1 i18n:translate="Usuarios de tipo: "> Usuarios de tipo: </h1>
- domain asigna el dominio de traducción para el elemento actual y todos los elementos hijos del actual. Ejemplo:
  <h1 i18n:domain=”atvotaciones” i18n:translate="Usuarios de tipo: "> Usuarios de tipo: </h1>
- attributes especifica cuales atributos, ya sea estática o dinámicamente insertados, deben ser traducidos. Como con tal:attributes, varias entradas son separadas mediante punto y coma. Especificar un id es opcional. Ejemplo:
  <input class="context" tabindex="" type="submit" name="form.button.guardar" value="Guardar" title="Guardar" i18n:attributes="title; value boton-guardar" tal:attributes="tabindex tabindex/next;">
- name marca un elemento como una variable. Como los mensajes generados en código de *Python*, los mensajes en *Page Templates* también necesitan de estas variables. Ejemplo:
  <h1 i18n:translate="Usuarios de tipo: "> Usuarios de tipo: </h1>

6.3.5. ZCML

La parte menos problemática de todo el proceso de internacionalización es la configuración del ZCML. Debido a la forma en que la maquinaria de configuración funciona, está sabe que valores se espera sean traducidos para que parámetros de directiva. Esto significa que no hay necesidad de especificar lo que debe ser traducido y lo que no, ya que ZCML ya lo sabe. La única cosa que falta es asignar el dominio de traducción. Así que modificamos el archivo de configuración del producto configure.zcml.

```
<configurexmlns="http://namespaces.zope.org/zope"
xmlns:genericsetup="http://namespaces.zope.org/genericsetup"
xmlns:i18n="http://namespaces.zope.org/i18n" i18n_domain="atselectusers">
<i18n:registerTranslations directory="locales" />
```

6.3.6. Catálogos de mensajes

Hemos configurado nuestra aplicación para las traducciones, ahora tenemos que traducir las sentencias. Aunque los desarrolladores no proveen traducciones por ellos mismo si no que se contratan traductores profesionales (en este caso fue mi tutor y yo). De cualquier forma los desarrolladores necesitan decirle al traductor que es lo que va a ser traducido. Entonces ellos también necesitan integrar un conjunto de mensajes traducidos en la aplicación para que el proceso automático de traducción se realice.

El sistema de traducción en *Zope* utiliza un programa bien conocido para obtener las traducciones, el sistema *GNU gettext*. La librería gettext nos da acceso a ids de mensaje y sus cadenas de mensaje almacenados en archivos llamados catálogos de mensaje. Los catálogos de mensaje son archivos de texto pero la biblioteca por si misma sólo puede trabajar con una representación binaria de los archivos.

6.3.6.1. Utilidad de extracción

Como se ha mencionado, *Zope* viene con una utilidad de extracción llamada *i18nextract* la cual puede extraer todos los mensajes de código *Python*, *Page Templates* y *ZCML*. Este escribe el resultado en un machote de catalogo de mensajes (extensión de archivo .pot).

Este machote contiene todos los mensaje que se han definido en el código fuente. La sintaxis del catalogo de mensajes es la de *gettext*. En el archivo machote, la cadena del mensaje simplemente se deja vacía.

```
#. Default: "Campos por tipo de usuario"
#: ./skins/ATSelectUsers/selectCampoUsuarios.cpt:24
msgid "Campos por tipo de usuario"
msgstr ""
```

Ahora podemos darle el machote a un traductor quien traducirá cada mensaje llenando la cadena de mensaje vacía. Esto puede realizarle con un editor de texto o una herramienta especializada como poEdit (la que utilice).

Los catálogos de mensaje en su versión texto tienes la extensión .po. Cuando obtenemos el archivo con las traducciones, tenemos que instalarlo en el directorio locales y compilarlo. El sistema gettext espera un subdirectorio en locales por cada lenguaje disponible en nuestro sistema. Así que si tenemos una traducción al Inglés, debemos crear un directorio llamado **en** dentro de locales. Dentro de ese directorio creamos otro directorio llamado LC_MESSAGES en el cual pondremos nuestro archivo con las traducciones, por ejemplo: atvotaciones.po. Cuando inicializamos *Plone* nos compila este archivo automáticamente.

6.4. Conclusiones

Este capítulo abarca todas las características especificas de la integración de los productos del sistema de votaciones del *IMATE* con *infoMatem*. Como ya se ha mencionado se tuvo que utilizar *FacultyStaff* que es el mejor producto para manejo de usuarios en *Plone*. Las pruebas son una parte importante en la integración ya que el administrador del sistema *infoMatem* puede correr las pruebas de instalación y estar seguro de que la instalación es exitosa. La internacionalización amplia el rango de usuarios y *feedback* que puede recibir el sistema de votaciones. Todo lo anterior permite poner al sistema de votaciones en el repositorio central de *Plone* y ver que en realidad es una aplicación exitosa.

Capítulo 7

CONCLUSIONES

7.1. Conclusiones generales

Todos los objetivos planteados en este trabajo de tesis se cumplieron, en especial la migración, rediseño e integración del sistema. Las siguientes subsecciones describen mas a detalle cada uno de los objetivos principales conseguidos en la tesis. Además concluyo basándome en el estado de arte y el rediseño que, el sistema de votaciones funciona correctamente para escenarios muy específicos, esta diseñado para votaciones pequeñas por lo tanto no tiene el nivel de seguridad que tendría un sistema de votaciones nacionales, y además la tecnología utilizada en este trabajo de tesis desde mi punto de vista, no es apropiada para un sistema como el de votaciones a gran escala pero sí para el tipo de votaciones pequeñas. Por último mencionar que, el diseño de software y en especifico los patrones de diseño y practicas estandarizadas como los contratos de interfaz son de gran ayuda para el arquitecto de software, sea cual fuere la tecnología para implementar, por lo tanto desde la primer versión del sistema se debió tener lo antes mencionado, debido a este legado de un diseño no tan adecuado, sugiero que, el sistema sea desarrollado de nuevo con lo que comento en mente.

7.1.1. Migrar el sistema a *Plone* 3

El requerimiento de la migración básica de los dos productos fue capital. El escenario es el siguiente: Tenemos un sistema de votaciones electrónicas, el cual, funciona con la versión 2 de *Plone*. En la versión actual (*Plone* 3) hay cambios en la API y también en la propia tecnología de *Plone*, este cambio es heredado de la actualización de *Zope*. Todo lo anterior nos deja el sistema incompatible, en otras palabras, inservible en la versión 3 de *Plone* y en las que vengan. Esta sin duda fue la motivación para migrar el sistema y para las mejoras que en este trabajo se muestran. Además *infoMatem* también ha sido refactorizado para *Plone* 3 y futuras versiones.

7.1.2. Integrar el sistema de votacion a *infoMatem* en *Plone* 3

Como ya he mencionado el sistema de votaciones forma parte del *infoMatem*. En la primer versión de *infoMatem* y de votaciones se utilizaba el producto CMFMember para la administración y manejo de usuarios. Este producto CMFMember se ha dejado de utilizar por la comunidad de *Plone*, además de que no existe una versión compatible con *Plone* 3.

infoMatem decidió utilizar *FacultyStaff*, que es un producto de administración y manejo de usuarios, este producto es desarrollado por uno de los grupos más importantes en el mundo *Plone* [31]. Por lo tanto el análisis y diseño del producto de selección de usuarios tuvo que girar entorno a dicho producto.

7.1.3. Internacionalización (i18n)

Los productos de selección de usuarios y de votaciones se pretende tengan los estándares más altos como software y como productos de *Plone*. Uno de estos estándares es la internacionalización del sistema al idioma inglés (en primera instancia). *Plone* cuenta con un fuerte soporte de i18n.

Para resolver el requerimiento se tuvo que modificar ambos productos, ya que en la primer versión del sistema, no se contaba con está característica implementada. Cada uno de los mensajes y letreros del sistema se modificaron de forma tal que, cumplieran con la sintaxis del API de *Plone* que nos permite traducir texto estático y dinámico.

Con esto hemos logrado situar el sistema con el estándar i18n lo que permite su utilización y feedback por parte de una comunidad más amplia.

7.1.4. Nuevos campos y modificación de letreros

La retroalimentación de los usuarios de la primer versión, nos han hecho notar la necesidad de poner más campos de configuración en el producto de votación y de modificar algunos de los letreros que no eran claros.

Se analizo que incluir estos campos nuevos no fuera un riesgo a la seguridad, ya que todo cambio puede traer errores. Al modificar los letreros fueron tocados varios archivos de la vista del sistema.

7.1.5. Eficiencia del producto de selección de usuarios

El producto de selección de usuarios fue el que sufrió una refactorización de su lógica de negocio (back-end) casi total. Debido en parte al cambio de *CMFMember* por *FacultyStaff* y la otra parte fue para mejorar la eficiencia. Un ejemplo de esto último sería: mandar a procesar la información de todos los usuarios de forma repetitiva. Otro punto a mejorar fue dejar de combinar la lógica de negocio con la vista, está parte también pecaba de ineficiente.

Para resolver estos requerimientos se analizo el código y se rediseño el producto implementando patrones de diseño y eficiencia. Este nuevo diseño trajo consigo interfaces y adaptadores como el camino a seguir. El producto de selección de usuarios viene con una implementación predeterminada con *FacultyStaff* se programó de está forma para que estuviera out-of-the-box con *infoMatem*. Pero gracias a los adaptadores y el diseño se puede utilizar diversas fuentes de información de usuarios.

7.1.6. Permitir filtros por expresiones regulares

Esta característica ya se podía realizar en la primer versión pero sólo para campos de texto ahora lo que se implemento es: permitir filtrar cuando el campo sea una lista desplegable. Bien, estos campos a los que hago referencia nos permiten poner valores mediante los cuales vamos a filtrar a los usuarios. Por ejemplo: todos los usuarios que sean mayores de 25 años y menores de 50 años. Esta requerimiento que parece insignificante, no lo es. Esto significaba modificar la estructura de datos que se encontraba cargada en memoria con la información de los usuarios, lo que nos orillaba a refactorizar de nuevo!. Después de analizar de forma más detenida pude encontrar una forma de hacerlo sin tocar nuestra estructura de datos y el flujo del sistema.

7.1.7. Archivos de configuración

Los productos de votaciones y selección de usuarios son sumamente configurables. Por lo tanto hay muchos campos para llenar. La necesidad de agilizar este procedimiento lo que en parte motivo la implementación del requerimiento. En la fase de desarrollo cíclica era sumamente frustrante generar la configuración una y otra vez; para los usuarios del sistema también ha sido frustrante configurar todo, cuando en realidad las votaciones son muy parecidas.

7.1.8. Creación de manuales

Uno de los puntos de mayor importancia en la tesis fue la generación de manuales de el producto de votaciones, selección de usuarios y de la herramienta *GPG*. En la primer versión del sistema ya había manuales, pero se decidió reescribirlos totalmente, para ser más descriptivos e incluir las nuevas características, así como incluir detalles que los usuarios del sistema nos han hecho llegar.

7.1.9. Pruebas

Se ha implementado una suite de pruebas para verificar que los productos de selección de usuarios y de votaciones se instalaron de forma correcta en la instancia de Plone deseada. Además tenemos una prueba para verificar que no hay problemas para crear una instancia tanto del producto de selección de usuarios como el de votaciones en el sitio de Plone donde se instalaron.

7.2. Trabajos a futuro

El sistema de votaciones del *IMATE*, se ha utilizado en votaciones exitosas desde 2007, esto ha permitido tener una retroalimentación por parte de los usuarios muy valiosa. Además de que ya se tiene más claro cual es el rumbo para la próxima versión de *Plone*. Por lo tanto en listo las características que deberían estar contempladas en futuras actualizaciones del sistema:

1. La parte del sistema que concierne a la Vista, debe ser refactorizado para su total compatibilidad con *Plone* 4, debe dejar de utilizarce CMFFormController [53].

2. De alguna forma hacer que el sistema de votaciones sea más flexible; permitir votaciones rápidas, con lo anterior quiero decir que, no se tengan que configurar decenas de parámetros ni tener llaves criptográficas.

3. El sistema de selección de usuarios debería poder crear alianzas entre candidatos, todo esto de forma transparente para el usuario y el sistema de votaciones.

4. Las pruebas de instalación y funcionalidad son capitales, pero una verificación formal del código le daría la sistema de votaciones una carta de presentación más solida.

5. Una refactorización total del sistema que incluya los puntos 1, 2, 3 y 4 sería lo más conveniente.

Apéndice A

Un camino a *Plone* 3

La problemática que se tiene al querer desarrollar productos para *Plone*, es que hay varios caminos para lograrlo por lo que es fácil perderse. Voy a definir un camino que sirve para usuarios básicos e intermedios, es el camino que me funciono y que tuve que descubrir por mi cuenta, así que, es conveniente documentarlo para que los usuarios que comienzan, no pasen por los mismos obstáculos, ahora lo comparto y con ciertos consejos que son de ayuda.

Veamos los vértices de este camino:

- Se tiene que aprender a instalar *Plone*, familiarizarse con los dependencias que hay que instalar como son PIL.
- Una vez realizado lo anterior hay que levantar el servidor (que es *Zope*) el cual contiene nuestro sitio de *Plone* (un producto para *Zope*). NOTA: Hay que recordar que *Plone* no es más que un producto en *Zope*. Con esto podemos ingresar al sitio como admin (usuario definido en buildout.cfg)
- Toca el turno de aprender a manejar el sitio *Plone* como un usuario (añadir, editar, etc)
- Al terminar lo anterior y que ya estamos familiarizados con el entorno debemos comenzar con las tareas administrativas, agregar grupos, usuarios, roles, productos, etc. FAMILIARIZARCE CON LA ESTRUCTURA DE DIRECTORIOS DE NUESTRA INSTALACIÓN Y A GRANDES RASGOS QUE HAY EN CADA UNO.
- Entender que es un producto en *plone* la estructura de directorios de un producto, esto es de gran ayuda ya que si tenemos algo de conocimiento sobre desarrollo web podemos ubicar de forma más clara donde está la vista, el modelo y el controlador de la aplicación (producto)(modelo MVC).

A.1. ¿Que es *Plone*?

Plone es un `sistema de administración de contenido` (*CMS* por sus siglas en ingles) el cual podemos usar para construir un sitio web. Con *Plone*, personas que no tengan conocimiento ni experiencia en sistemas de cómputo pueden contribuir contenido al sitio web sin la necesidad de un experto a su lado. *Plone* se ejecuta sobre la Web, así que no debes instalar ningún software especial en tu computadora, un navegador es suficiente. La palabra contenido tiene un significado genérico, por que podemos publicar muchos tipos de información, incluyendo:

- Texto (páginas web)
- Fotos e imágenes
- Documentos

- Noticias y eventos
- Vídeos
- Archivos de audio

Además podemos crear folders en un sitio web *Plone* para que tenga contenido y para crear una estructura de navegación.

A.2. Instalar *Plone* 3

Veamos dos formas de instalar *Plone* 3, la primera es mediante el instalador (Unified Installer) el cual ya trae las dependencias, este instalador se encuentra para sistemas operativos como lo son Windows, Linux y Mac, debo dejar claro al lector que el instalador es la forma más confiable para Windows. La segunda forma es mediante `buildout`. Si lo que se quiere es tener un entorno de desarrollo de productos para *Plone* recomiendo buildout sobre Linux o Mac.

A.2.1. Unified Installer

El instalador es un kit que contiene *Python*, *Zope*, *Plone* y otras dependencias. Las dos partes más importantes del instalador son:

- Los paquetes fuente para *Python*, *Zope*, *Plone*, un par de librerías de sistema y algunas librerías extra para *Python*.
- Un script de instalación que usa los paquetes para crear un ejecutable, el cual es relativamente autocontenido ya que tanto la instalación de *Python*, *Zope* y *Plone* se encuentran en el mismo directorio, está instalación prediseñada contiene los estándares de mejores practicas de la comunidad de *Plone*.

La nueva instalación que se a conseguido de *Zope* y *Plone* fue realizada con su propia copia de *Python*, y como mencione anteriormente, comparten el mismo directorio, por lo que si tienes otra instalación anterior de *Python* en tu sistema, está no se vera afectada [32].

En *Windows* sólo hay que ejecutar el instalador, decidir en que directorio instalar los componentes y dar un nombre de usuario y contraseña para el administrador.

Figura A.1: Directorio de instalación

Figura A.2: Usuario y contraseña

En Linux hay que tener las siguientes herramientas:

- gcc
- g++
- make
- tar
- gzip

Estas herramientas ya suelen ser parte de cualquier instalación de Linux, pero si no fuera el caso se pueden instalar mediante un administrador de paquetes, por ejemplo en Ubuntu y Debian para instalar `gcc` lo que hacemos es ejecutar la siguiente instrucción:

```
$ apt-get install gcc
```

Una vez que tenemos las herramientas instaladas ejecutamos las siguientes instrucciones en un directorio deseado:

```
$ tar zxf Plone-3.VERSION-UnifiedInstaller.tar.gz
```

Donde VERSION puede variar. Después de esto ingresamos al directorio creado con la instrucción:

```
$cd Plone-3.VERSION-UnifiedInstaller
```

Ahora ejecutamos el siguiente comando como root o mediante sudo para instalar nuestro servidor de *Zope* junto con un sitio *Plone* por default:

```
$ ./install.sh standalone
```

Hay otro argumento para el script el cual es `zeo`, el cual nos crea un servidor con propiedades de cluster.

```
$ ./install.sh zeo
```

Lo que resta es verificar que la instalación fue exitosa, ponemos en un navegador la siguiente url: `http://localhost:8080/Plone`

Aquí es donde pueden surgir los primeros conflictos en la configuración y se debe a que tal vez otro servicio ya este corriendo en el puerto 8080 que es el puerto por default que se asigna. Para resolver lo anterior tenemos que editar el archivo buildout.cfg, poner un nuevo número de puerto y ejecutar el comando `buildout` que se encuentra en el directorio `bin`.

A.2.2. Buildout

Buildout nos ayuda a configurar un entorno de trabajo autocontenido (isolado) para paquetes. Si utilizamos buildout podemos controlar dependencias y las versiones a utilizar. Para realizar lo prometido necesitamos tener el comando buildout y un archivo de configuración.

Pero primero hay que tener lo siguiente:

- Hay que instalar *Python* 2.4, y añadirlo al *PATH*. Lo podemos hacer con el administrador de paquetes o instalarlo manualmente [33]:
  - `$ apt-get install python24`
- Si instalamos *Python* usando un administrador de paquetes, entonces debemos asegurarnos de tener el paquete de desarrollo (ejemplo: python-devel). Este paquete incluye archivos de cabecera de *Python* que se utilizan para compilar *Zope*. Si instalamos *Python* desde el código fuente entonces ya tenemos estos archivos.
- Instalar PIL [34], que es la librería de imágenes de *Python*, podemos hacerlo con un administrador de paquetes como (RPM o apt-get):
  - `$ apt-get install py-pil`

Ahora descargamos ez_setup.py desde `http://peak.telecommunity.com/dist/ez_setup.py` y lo ejecutamos con:

```
$ python ez_setup.py
```

Esto descargara e instalara `setuptools` y el script `easy_install`. Utilizamos easy_install para obtener ZopeSkel, que es una colección de templates para desarrollo de *Zope* y *Plone*.

```
$ easy_install -U ZopeSkel
```

Con lo anterior obtenemos el script Paste y algunas otras dependencias. Finalmente estamos listos para crear un nuevo buildout. Ejecutamos ZopeSkel para crear un directorio que contiene lo básico que se requiere para tener una instancia de *Zope* corriendo un sitio *Plone* 3. En la siguiente instrucción el directorio lo nombro plone3, el lector puede escoger el nombre que guste.

```
$ paster create -t plone3_buildout plone3
```

Nos cambiamos al directorio que acabamos de crear y ejecutamos las instrucciones que se encuentran en el archivo bootstrap.py

```
$cd plone3
$python bootstrap.py
```

El script bootstrap crea un número de directorios, scripts y descarga la última versión del zc.buildout. Por fin ejecutamos

```
$ ./bin/buildout
```

este comando lee el archivo `buildout.cfg` y ejecuta cada una de sus partes, configura *Zope*, crea una instancia de *Zope*, descarga e instala *Plone*. Cada que realicemos un cambio en nuestro `buildout.cfg` hay que volver a ejecutar `buildout`.

Verificamos que la instalación fue exitosa, primero ejecutamos el servidor de *Zope*

```
$ ./bin/instance fg
```

y ponemos en un navegador la siguiente url: `http://localhost:8080/Plone`

A.3. *Plone* 3: Soy usuario

En *Plone* tenemos actividades que los usuarios normales (sin privilegios de administrador) realizan. Cuando terminamos de instalar nuestro sitio *Plone* lo siguiente es aprender a navegar a través del sitio, familiarizarce con las actividades básicas para después dar paso a la administración.

Algunas de las actividades básicas son iniciar sesión, agregar contenido y manejar contenido.

A.3.1. Inicio de sesión

Cuando visitamos un sitio web *Plone*, veremos un botón de inicio de sesión como se muestra en la figura:

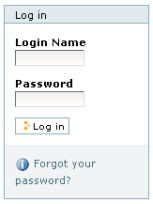


Figura A.3: login

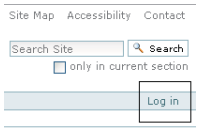


Figura A.4: login 2

Después de haber iniciado sesión en el sitio *Plone* veremos nuestro nombre, en una pequeña barra del encabezado. Podemos dar click en nuestro nombre de usuario para manejar nuestras preferencias dentro del sitio. Cuando instalamos *Plone* ya tenemos un usuario definido en el archivo buildout.cfg el cual tiene el formato `nombre_usuario:contraseña`. Con este usuario podemos ingresar al sitio.

Ya que estamos adentro de nuestro sitio podemos cambiar el idioma del sitio, por ejemplo si está en ingles podemos cambiarlo a español.

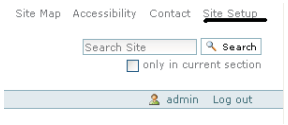


Figura A.5: Cambiar idioma

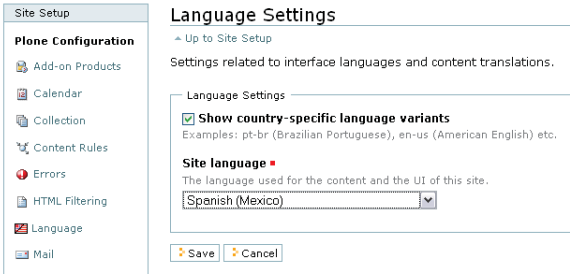


Figura A.6: Español

A.3.2. Agregar contenido

Es muy probable que el lector halla creado folders (carpetas) en el disco duro de su computadora. Las computadoras personales utilizan una jerarquía de folders para estructurar y organizar los archivos y programas en el disco duro. En *Plone* los folders son esencialmente usados de la misma forma, excepto por que se crean en el sitio de *Plone*, para poder organizar el contenido en el sistema de almacenamiento de Plone (*ZODB*).

Los folders se añaden dando click a `Agregar elemento` y en el menú que se despliega seleccionamos `Carpeta`.

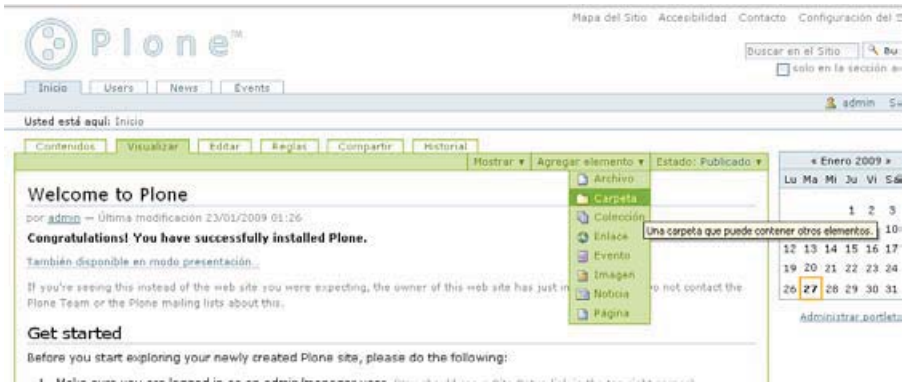


Figura A.7: Agregar una carpeta

Ahora se debería de ver la pantalla de Agregar carpeta:

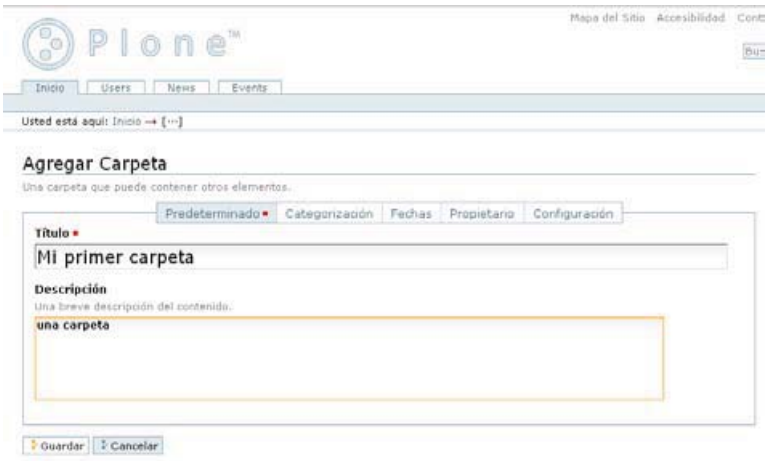


Figura A.8: Agregar carpeta 2

Podemos ponerle un título y una descripción a la carpeta. Además tenemos las siguientes pestañas:

- Predeterminado: para asignar valor a los campos titulo y descripción.
- Categorización: especificar categorías que aplican a la carpeta (se conocen como palabras clave o etiquetas).
- Fechas: para configurar el periodo de tiempo que la carpeta debería estar habilitada para que los demás la vean en el sitio web.
- Propietario: sirve para especificar el creador o contribuyentes al contenido.
- Configuración: para permitir comentarios acerca del contenido agregado y si puede mostrarse en el menú de navegación del sitio web.

Estas pestañas son un estándar, por lo que las verán cuando agreguen otros tipos de contenido. Por último hay que dar click al botón `Guardar`.

A.4. *Plone* 3: Soy administrador

En está sección se mostrara como crear usuarios, grupos, roles, además de la relación entre ellos. Veremos que es muy sencillo de hacer lo anterior desde nuestro sitio *Plone*, como también agregar productos de terceros a nuestro sitio y dar un vistazo a la estructura de directorios de nuestra instalación de *Plone*.

A.4.1. Usuarios y grupos

Los usuarios y grupos se encuentran en una carpeta de usuario, llamada acl_users. Hay una en la raíz de la instancia de *Zope*, típicamente conteniendo al administrador por defecto. La raíz de la instancia de *Zope* se puede acceder mediante la siguiente dirección:

http://localhost:8080/manage

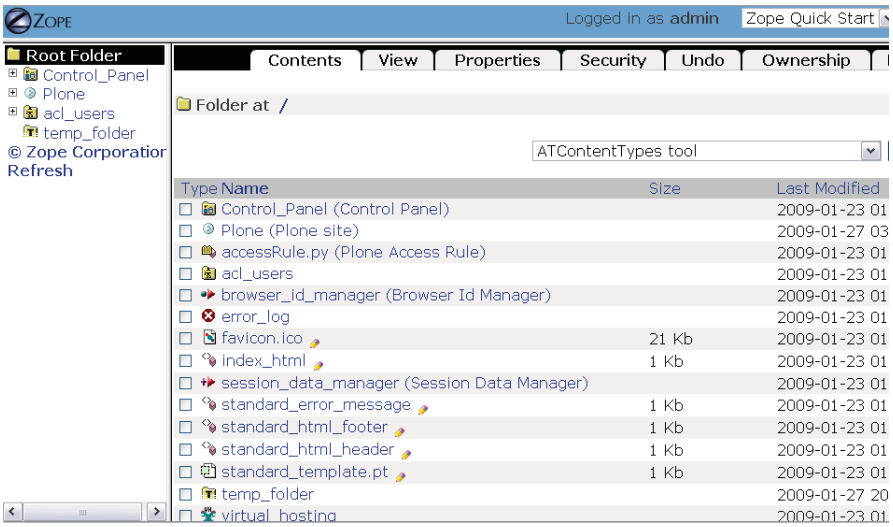


Figura A.9: ZMI

Esta parte de nuestro sitio se llama *Zope* Management Interface (*ZMI*), es la interfaz de *Zope* que nos ayuda a realizar las tareas de administración, como podemos observar en raíz (Root) se encuentra la carpeta acl_user. Pero hay que tener muy claro lo siguiente: el usuario definido en está carpeta es a nivel del servidor de *Zope*, lo que quiere decir que si nosotros ponemos dos o más sitios de *Plone* (instancias) sobre este servidor de *Zope* entonces ese usuario puede administrar todos las instancias. Lo que nos lleva a la pregunta ¿Nuestra instancia de *Plone* tiene su propio acl_users? y la respuesta es sí. Como ya se imaginaran los usuarios definidos en esa carpeta acl_users sólo tendrán efecto en esa instancia. Pero lo anterior para alguien que comienza pude ser confuso, por lo que recomiendo que administren desde `Configuración del Sitio` *Plone* en la parte de `Usuarios y Grupos`.

Figura A.10: Usuarios y Grupos

Sí accedemos a `Usuarios y Grupos` podremos agregar a un usuario

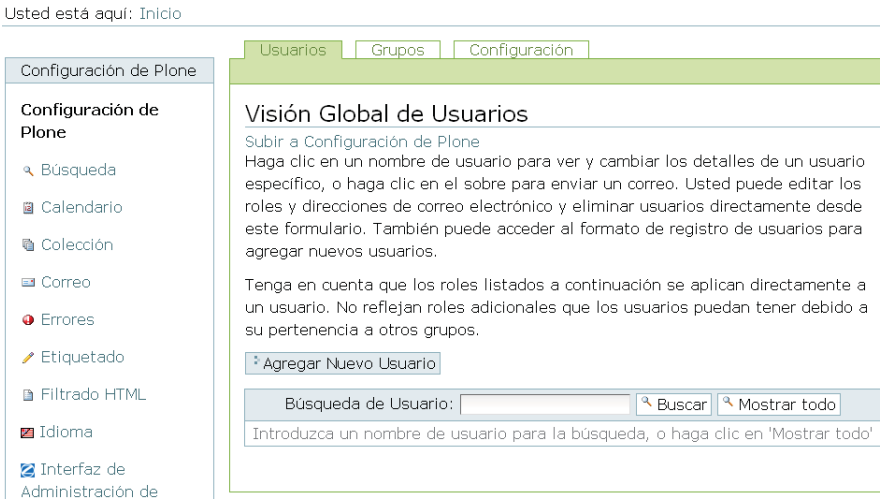


Figura A.11: Agregar usuarios

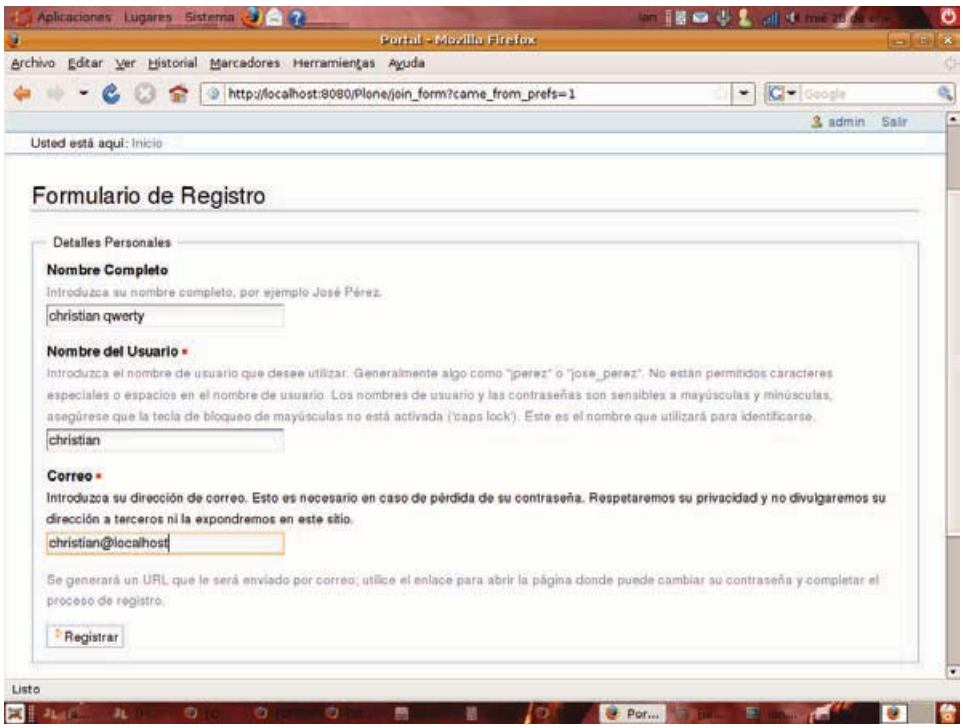


Figura A.12: Agregar usuarios 2

Una vez que asignamos nombre completo, nombre de usuario y correo, entonces creamos el usuario. Ahora podemos buscarlo y asignarle roles.

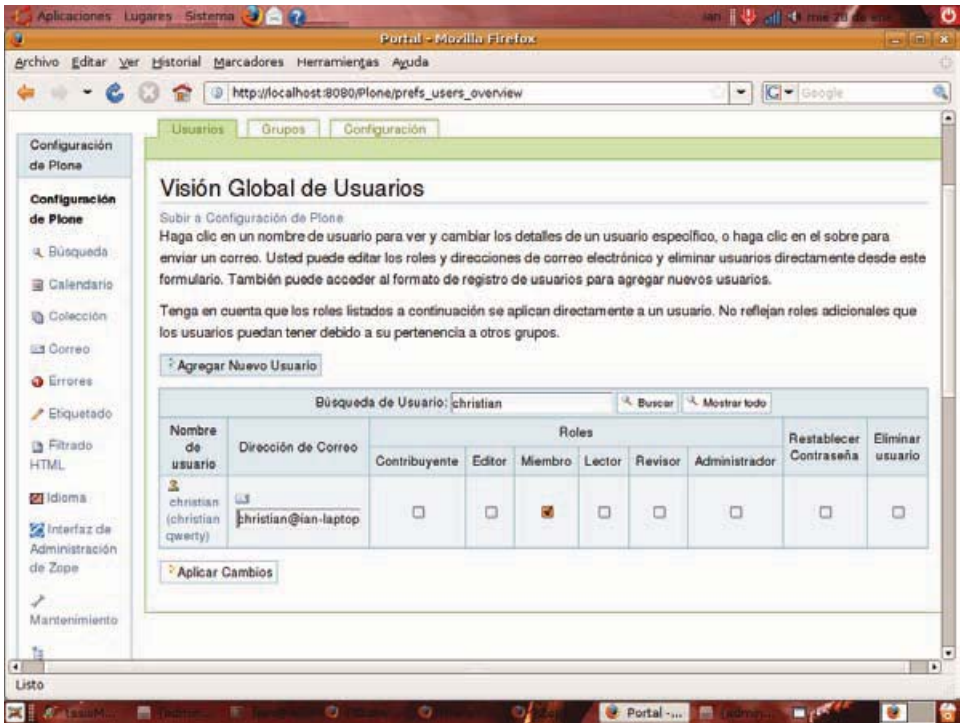


Figura A.13: Roles

Es tiempo de que el lector conozca lo siguiente, los `permisos` no se dan directamente a los usuarios, los permisos se asignan a los `roles`. Los usuarios pueden tener cualquier número de roles, de forma global en el portal, contexto o carpeta en particular. Los roles globales y locales pueden ser asignados a `grupos`, en cuyo caso todos los usuarios en ese grupo tendrán ese rol. Esto ayuda en que la administración de seguridad sea más manejable. Si vamos a la interfaz de administración de *Zope* en la pestaña de Seguridad, podremos ver que permisos tiene cada rol.

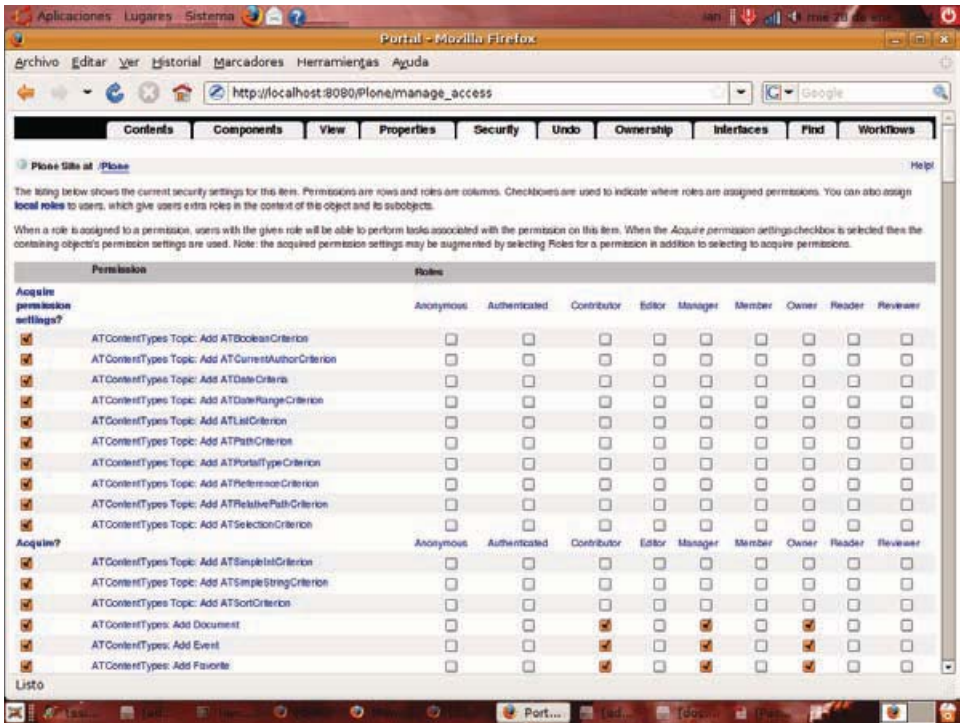


Figura A.14: Permisos * rol

Veamos como agregar un nuevo grupo a nuestro sitio *Plone*; nos vamos a la pestaña `Grupos` y damos click en `Agregar Nuevo Grupo`.

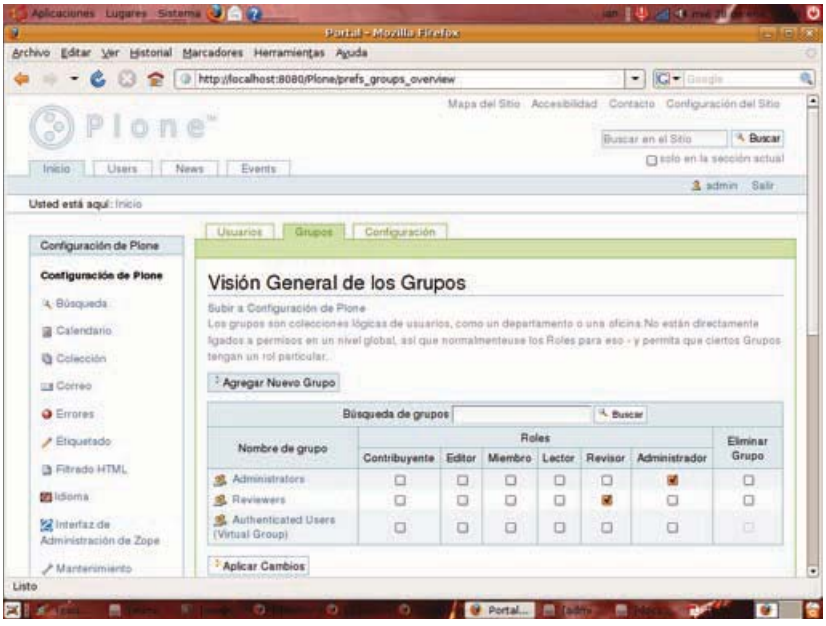


Figura A.15: Grupos

Asignamos los valores deseados a los campos para crear nuestro nuevo grupo.

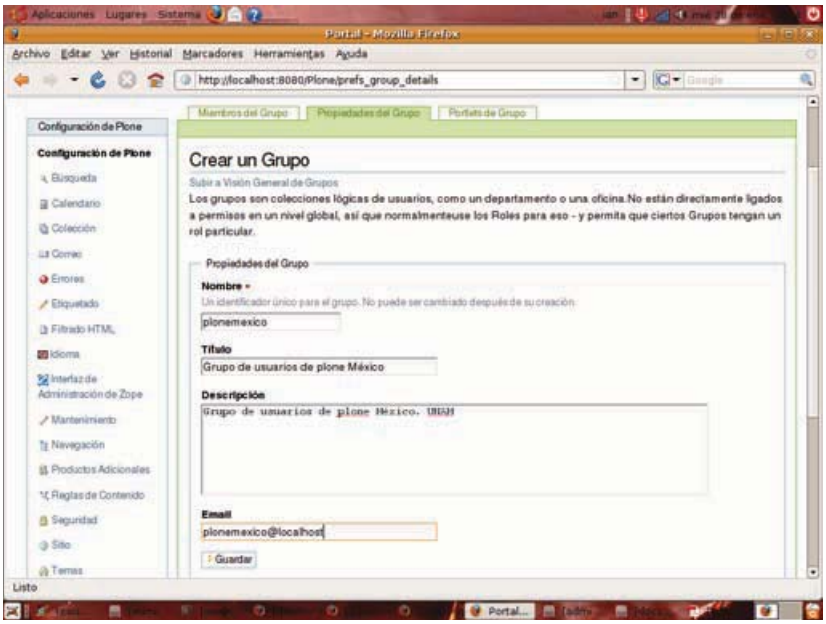


Figura A.16: Grupos 2

Por último podemos asignarle roles a nuestro grupo creado, ya que como vemos en la figura no tiene ningún rol asignado, y damos click en `Aplicar Cambios`.

A.4.2. Instalar un producto de terceros

Como instalar un nuevo producto de terceros depende de si está empaquetado como un `egg`, o como un producto *Zope* 2 tradicional.

Un producto *Zope* 2, es una estructura de directorios, la cual se copia dentro del directorio Products que se encuentra en nuestra instalación de *Plone*. Una vez copiado iniciamos el servidor, nos vamos a nuestro sitio de *Plone* y en la parte de configuración está la opción de agregar productos, ahí magicamente estará enlistado el producto que se puso en Products. Si esto no fuera así quiere decir que el producto instalado tiene algún error.

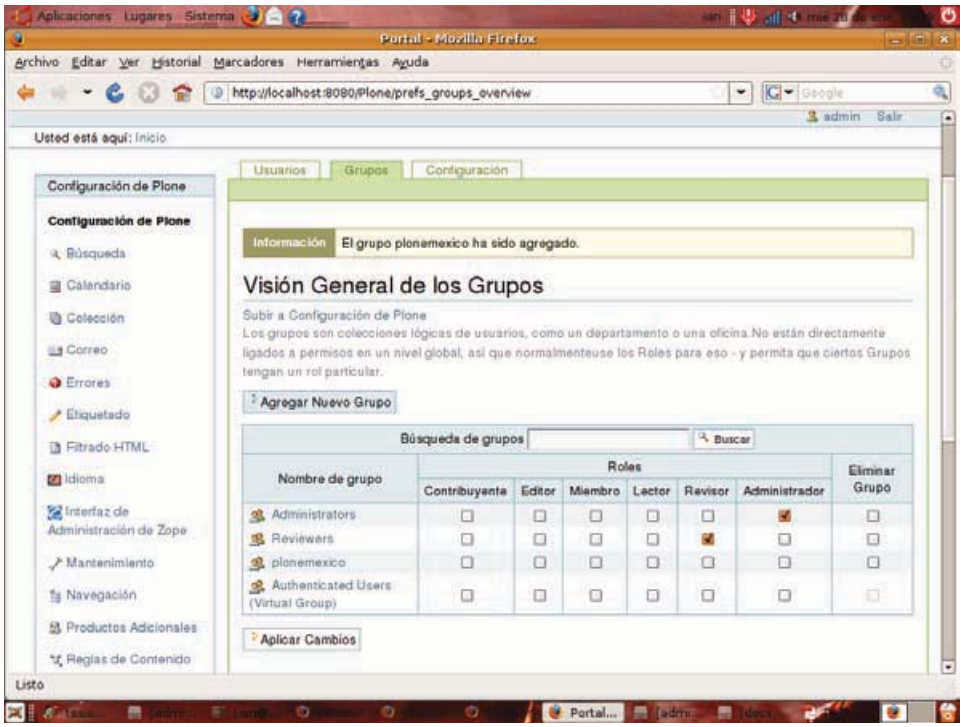


Figura A.17: Roles a un grupo

Los productos pueden encontrarse principalmente en el repositorio de *Plone*: `http://plone.org/products`. Por ejemplo si nosotros descargamos un archivo como:

`algunProducto-0.1.5.tar.gz`.

Lo que hacemos es extraer el contenido del archivo. En sistemas Linux, lo hacemos con:

```
$ tar -zxvf algunProducto-0.1.5.tar.gz
```

En windows podemos usar 7zip o WinZip. Ahora que ya tenemos nuestro directorio descomprimido lo ponemos en el directorio `Products` de nuestra instalación de *Plone*. Hay que asegurarse de que el directorio que acabamos de poner tenga los mismos permisos que los demás directorios de los otros productos.

También podemos instalar un producto *Zope* 2 desde el `buildout.cfg`, sólo tenemos que poner el nombre del archivo en la sección `[productdistros]` en el archivo `buildout.cfg`. Por ejemplo si lo que queremos es instalar un producto llamado ProductoEjemplo:

```
[productdistros]
   recipe = plone.recipe.distros
   urls = http://ejemplo.com/dist/ProductoEjemplo-1.0.tgz
   nested-packages =
   version-suffix-packages =
```

Y como hay que recordar, siempre que modifiquemos nuestro archivo `buildout.cfg` hay que volver a ejecutar `buildout`.

Un *Python egg* es la forma de empaquetar información adicional junto con un proyecto *Python*, esto permite que las dependencias del proyecto sean verificadas y satisfechas en tiempo de ejecución, como también permitir a los proyectos proveer plugins para otros proyectos, estos *Python eggs* se crean con setuptools, lo cual veremos a detalle en el siguiente apéndice.

Para instalar un egg mediante el buildout lo que hacemos es enlistar el nombre del huevo en la sección egg del archivo de configuración (buildout.cfg). Esto siempre y cuando el egg este en el Cheese Shop [35], si no es así, entonces hay que agregar la url de la ubicación en la sección de `find-links`.

```
find-links = http://dist.plone.org
   http://download.zope.org/ppix/
   http://download.zope.org/distribution/ http://effbot.org/downloads
eggs = elementtree
   FeedParser
```

Apéndice B

Manual ATSelectUsers

ATSelectUsers es el producto que se encarga de la selección de usuarios para el padrón electoral y el padrón de candidatos.

Para poder usar el producto hay que instalarlo, esto se realiza en la parte de **Configuración del Sitio -> Productos Adicionales** de nuestro sitio *Plone*. Una vez ahí seleccionamos ATSelectUsers 1.0 y damos instalar.

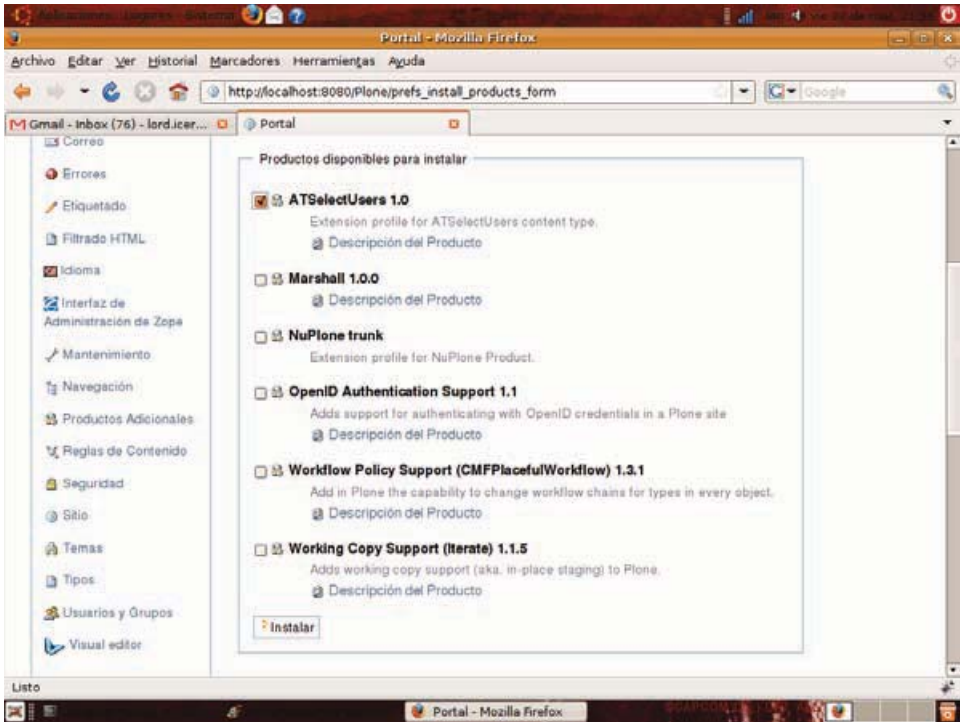


Figura B.1: Instalar ATSelectUsers

Una vez instalado podremos agregar un objeto de este producto.

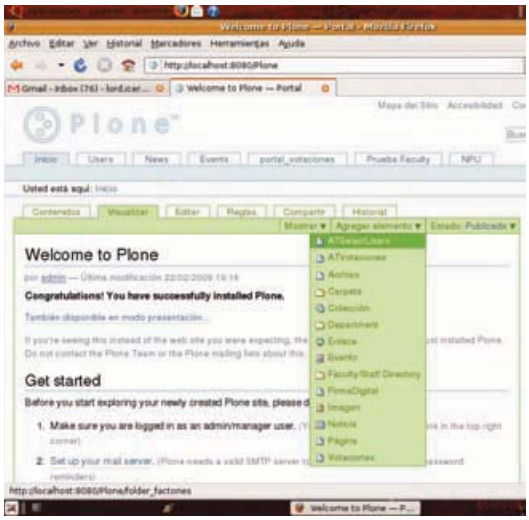


Figura B.2: Agregar objeto ATSelectUsers

Ahora veamos el procedimiento para trabajar con este objeto que hemos agregado.

1. Le asignamos un título y descripción. Le damos clic en guardar.

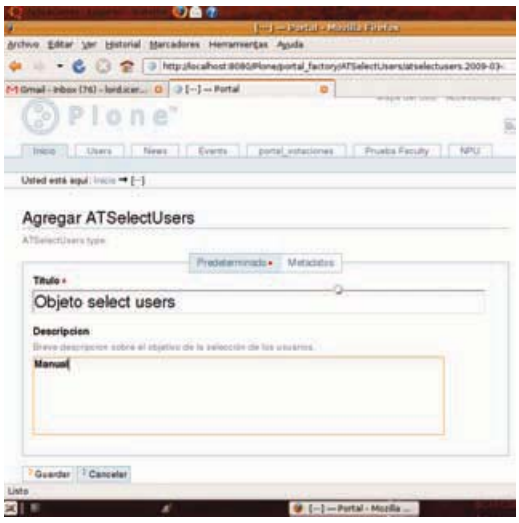


Figura B.3: Título - ATSelectUsers

2. En la página de Proceso de Selección podemos elegir entre:

   a) Ir: Lo cual nos mandara al proceso normal de selección de usuarios

   b) Cargar condiciones: Sirve para cargar condiciones que hayamos generado y guardado con anterioridad, la forma de guardar las condiciones se vera más adelante (12).

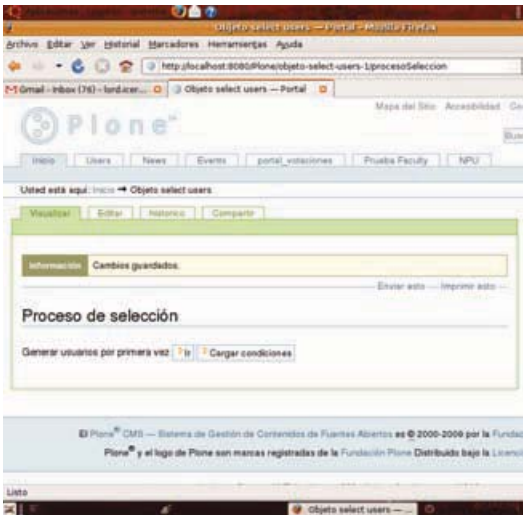


Figura B.4: Proceso Selección - ATSelectUsers

3. Escogemos de la lista el tipo de usuario(s) con los que se quiere trabajar, por ejemplo: becarios. Podemos escoger trabajar con un tipo o varios. Le damos clic en **Siguiente.**

Figura B.5: Tipos de Usuarios - ATSelectUsers

4. Elegimos las propiedades (campos) que tiene el tipo de usuario(s) del punto (3), con las cuales vamos a filtrar a los usuarios. Podemos elegir una o más propiedades.

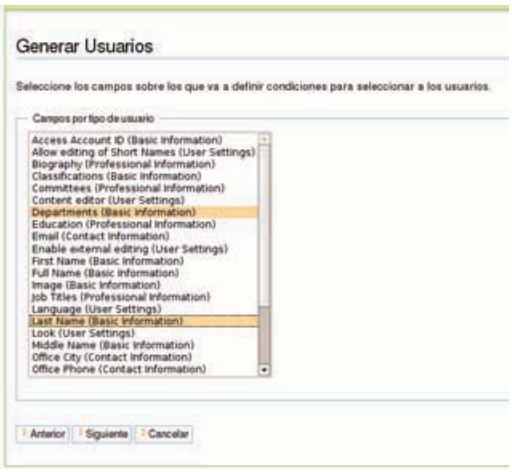


Figura B.6: Propiedades - ATSelectUsers

5. De los campos seleccionados en (4), vamos a darle a cada campo el valor que deseamos cumplan los usuarios, para poder ser parte de la lista de usuarios seleccionados.

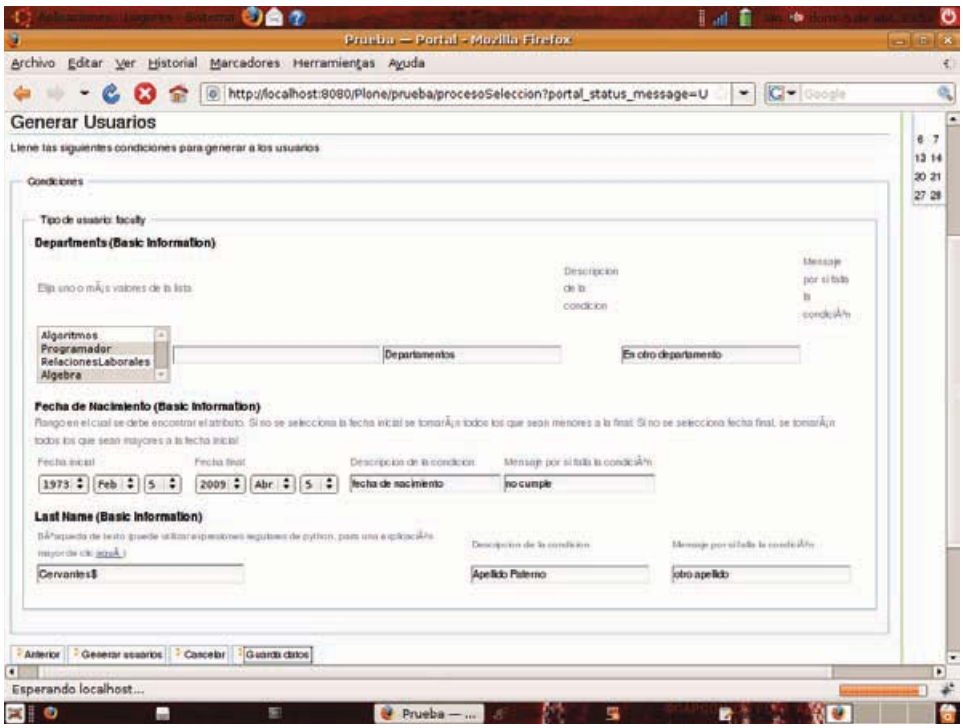


Figura B.7: Valores deseados - ATSelectUsers

En está parte debemos poner el nombre de la condición y el mensaje de error que deseamos aparezca si el usuario no cumple con dicha condición.

Como podemos observar la condición **Last Name** tiene un campo de texto, donde ponemos el

valor que deseamos cumplan los usuarios. Es importante tomar en cuenta que, en estos campos de texto podemos poner expresiones regulares, lo cual nos permite realizar filtros más poderosos. Por ejemplo podemos poner: Cervantes$. Lo cual nos dice que todo usuario que su apellido sea 'Cervantes' cumplirá la condición [46].

El campo **Departments** es una lista desplegable que contiene los únicos valores posibles que puede tener un usuario, tiene la desventaja en la versión anterior de no poder poner una expresión regular, pero dado que estos tipos de campos tienen normalmente un número de valores mucho menor que campos como **Last Name**, entonces por eso se genera una lista desplegable en vez de un campo de texto, tiene la ventaja de que podemos escoger específicamente los tipos de departamentos a los que queremos que pertenezcan los usuarios que vayan a ser parte de la selección. **Nota:** En está versión ya es posible buscar por expresiones regulares o por elementos de la lista.

Podemos tener otro tipo de campos para poder definir nuestras condiciones, por ejemplo: booleano y fechas.
Tenemos dos opciones para seguir con el proceso de la selección que son:

a) Generar usuarios: Con está opción generamos la lista de usuarios que cumplen las condiciones deseadas, pero **no** guardamos las condiciones y sus valores para futuros usos.

b) Guardar datos: La única diferencia con (a) es que está opción guarda las condiciones y sus valores en un archivo, esto se logra mediante la serialización en *Python* [45], la cual es un estándar en otros lenguajes de programación [47]. El archivo donde se guardan las condiciones y sus valores se llama "data.pkl", y se genera en el directorio raíz de nuestra instancia de *Plone*.

6. Cualquiera que haya sido la opción elegida (Generar usuarios o Guardar datos) tenemos la siguiente pantalla.

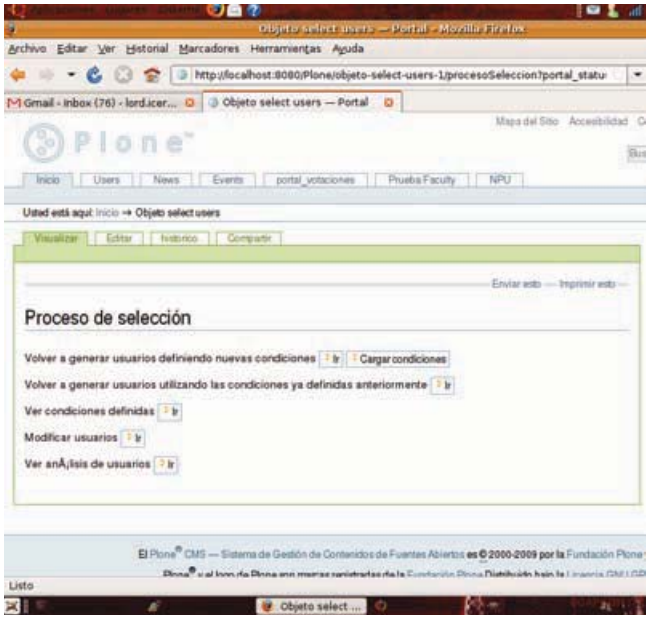


Figura B.8: Opciones - ATSelectUsers

Aquí tenemos varias opciones; para este punto **ya tenemos generado nuestro padrón de electores**. Entonces las opciones que tenemos van hacia modificar usuarios, ver las condiciones definidas, y volver a generar los usuarios, en está última opción podemos generar los usuarios con todas las condiciones nuevas o tomar como base las condiciones que ya se tienen. Pero antes de pasar a describir y ejemplificar todas estas opciones me gustaría mostrarles una de las partes fundamentales de este producto y del de votaciones, me refiero a la bitácora.

a) En la pestaña historico (bitácora) podemos encontrar un informe preciso de lo que ha ocurrido en el producto de selección de usuarios. Estos eventos son: creación del objeto, el título de la instancia, los tipos de usuarios definidos, las condiciones y los mensajes que le asignamos a cada condición. Si llegamos a modificar a los usuarios, eliminar o cambiar alguna condición, la bitácora sera testigo de ello y nosotros podremos siempre llevar un seguimiento de lo que acontece.

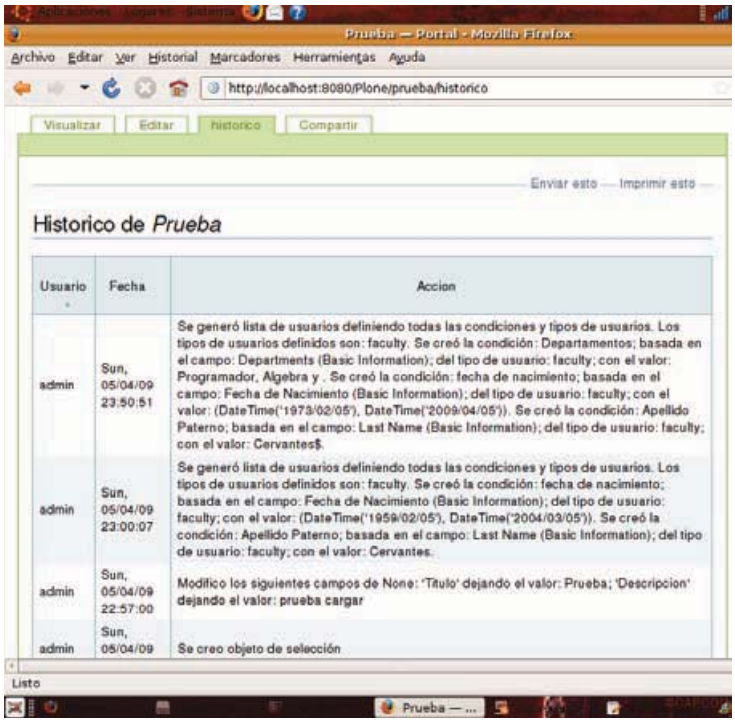


Figura B.9: historico - ATSelectUsers

7. Ahora sí, comencemos con las opciones disponibles una vez que se ha creado la lista de usuarios seleccionados. En la sección de **Usuarios analizados** podemos encontrar la lista de los usuarios que fueron analizados por el sistema, donde la primer columna nos dice si fue seleccionado o no, la segunda columna nos muestra el nombre del usuario y después vienen las condiciones con su explicación de error, está explicación sólo se muestra si no cumplió con dicha condición.

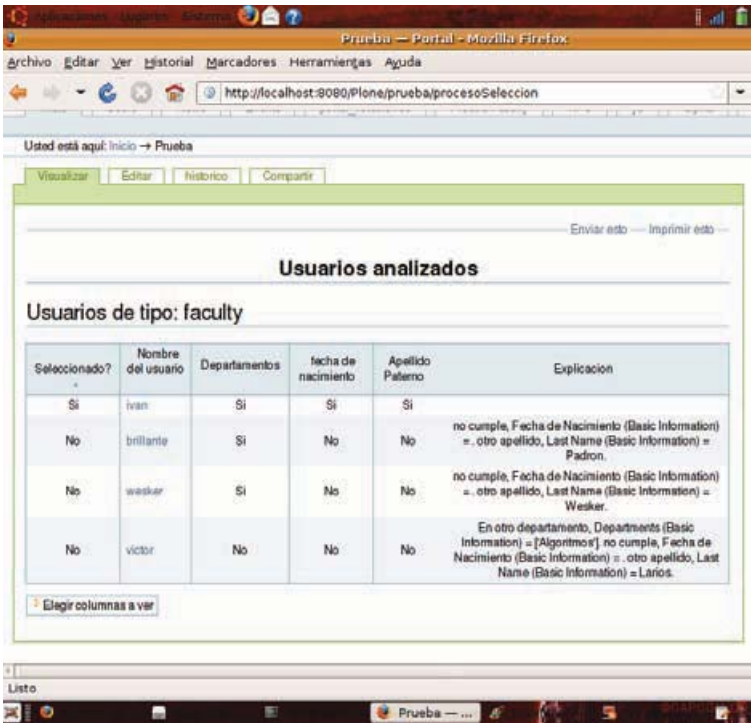


Figura B.10: Usuarios analizados

Otra característica es que podemos ver las condiciones que queramos, esto nos ayuda a enfocarnos en pocas condiciones, por ejemplo: si la selección tuvo demasiadas condiciones, no vendría mal poder visualizar una por una.

8. La sección de **Ver condiciones**, nos muestra los valores que le asignamos a cada una de las condiciones, también vienen los mensajes de error y descripción.

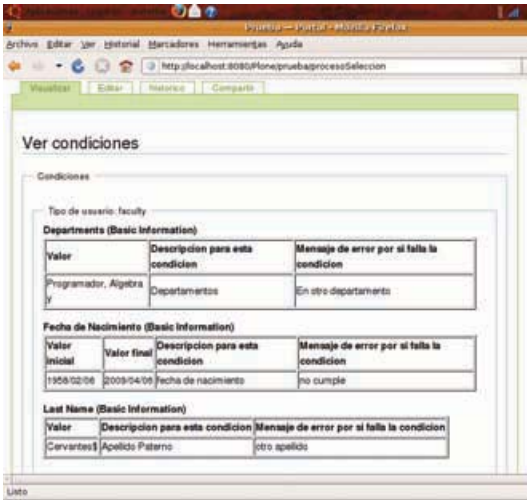


Figura B.11: ver condiciones

9. En **Modificación de usuarios** podemos como se puede adivinar, modificar los usuarios y forzar a que sea seleccionado y que cumpla con las condiciones que en primera instancia no pudo ser. Veamos los siguientes usuarios y su análisis.

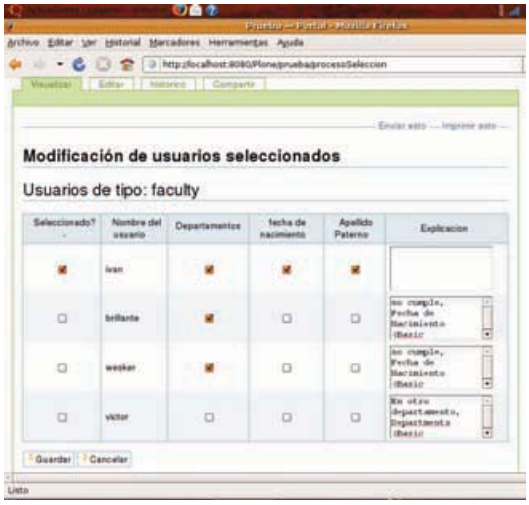


Figura B.12: modificación de usuarios 1

Podemos observar que el único usuario seleccionado es el que cumplió con las tres condiciones: **ivan.** Por otro lado **brillante** y **wesker** cumplieron con la condición llamada **Departamentos**, y **victor** no cumplió con ninguna. Vamos a forzar a que el usuario **victor** sea seleccionado y es sumamente importante que pongamos en **explicación** el por que de este cambio, ya que será guardado en el **historico**.

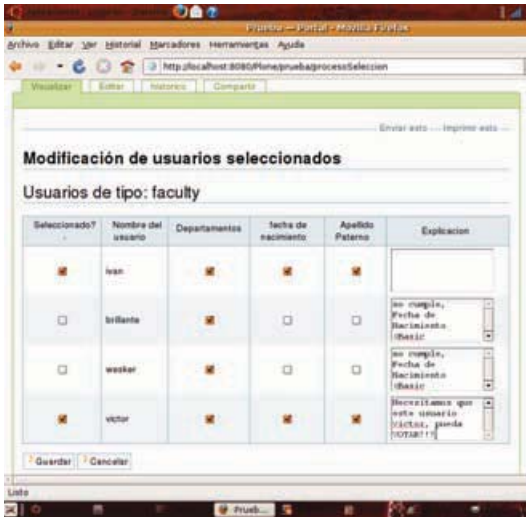


Figura B.13: modificación de usuarios 2

Podemos comprobar que nuestra modificación es exitosa, ingresando a la sección de **Usuarios analizados.**

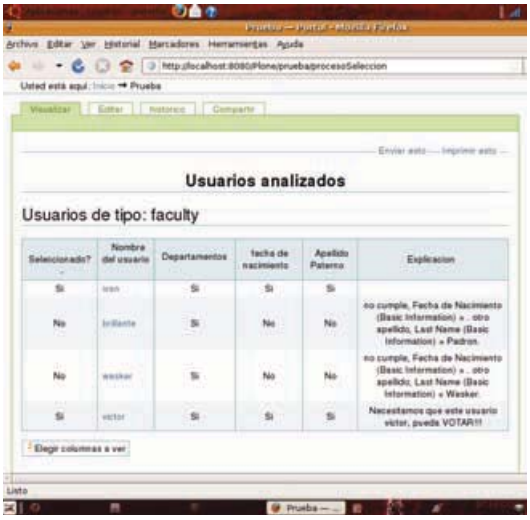


Figura B.14: modificar usuarios 3

10. **Generar usuarios utilizando condiciones existentes**, nos permite volver a generar los usuarios pero... la gran ventaja es que podemos mantener las condiciones existentes o sólo las deseadas.

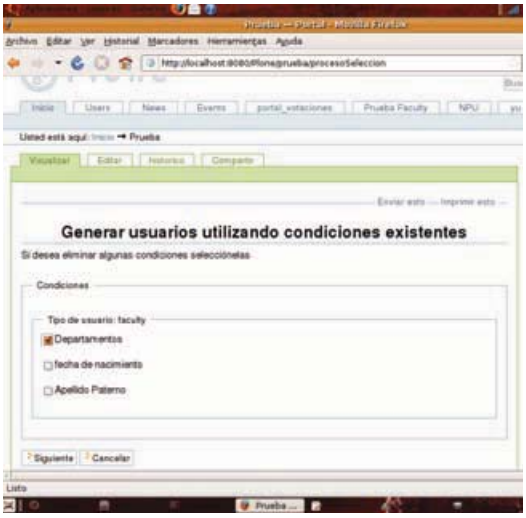


Figura B.15: generar usuarios con condiciones existentes 1

Como el letrero muestra, podemos eliminar las condiciones que queremos, ya no sean parte del filtro de selección, lo hacemos seleccionando la casilla correspondiente. En este ejemplo eliminamos la condición **Departamentos**.

Figura B.16: generar usuarios con condiciones existentes 2

Aquí podemos o no cambiar los tipos de usuarios sobre los que vamos a trabajar. Pero si damos **Omitir este paso** entonces trabajaremos con los mismos tipos de usuarios que teníamos. Es un atajo en realidad ya que podríamos darle otra vez faculty y trabajar con los anteriores, pero internamente ahorra mucho trabajo de procesamiento. Además de que ya no entraríamos a la pantalla de selección de condiciones. Vamos a seleccionar becarios y faculty para agregar emoción, y damos clic en **Siguiente.**

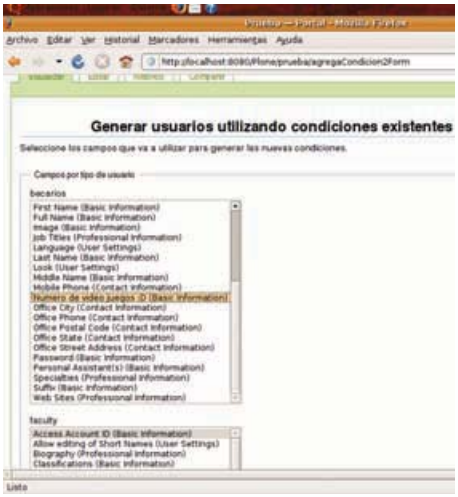


Figura B.17: generar usuarios con condiciones existentes 3

Como nos muestra la imagen seleccionamos una propiedad del tipo de usuario **becario** y una propiedad del tipo de usuario **faculty.** Por lo tanto **faculty** tendrá las propiedades anteriores las cuales elegimos no borrar y la nueva propiedad, y **becario** tendrá sólo la que acabamos de escoger. Llenamos los campos de las nuevas condiciones y damos **Generar usuarios** o **Guardar datos.**

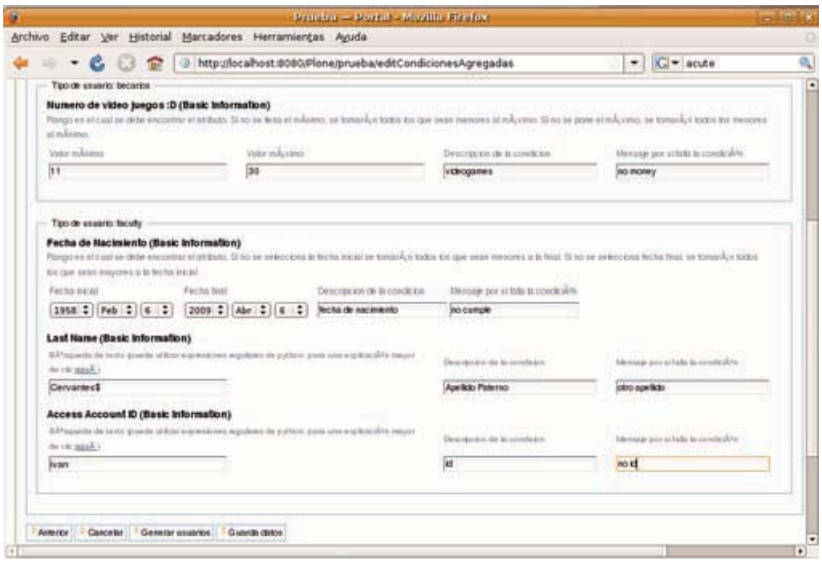


Figura B.18: generar usuarios con condiciones existentes 4

Para ver la lista de usuarios seleccionados como ya sabemos lo consultamos en **Usuarios analizados**.

11. Volver a generar usuarios definiendo nuevas condiciones, es lo mismo que comenzar desde la selección de tipos de usuarios (3).

12. Como mencionamos en el paso (2), podemos cargar datos que hayamos guardado. Damos clic en **Cargar condiciones** y el sistema se encargara de dejar todo como estaba cuando guardamos las condiciones y sus valores (5). Esta funcionalidad es muy útil en términos prácticos.

Figura B.19: cargar condiciones 1

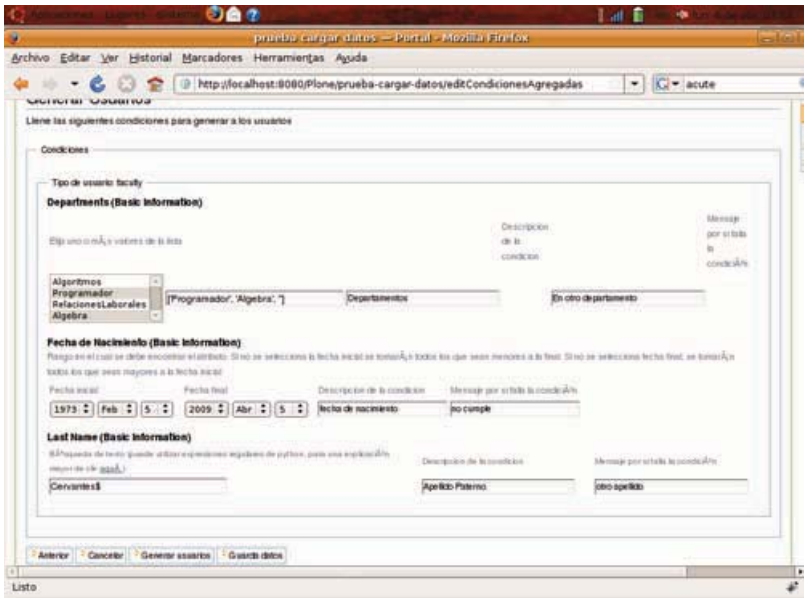


Figura B.20: cargar condiciones 2

Como se muestra en la figura podemos editar las condiciones y volverlas a guardar, lo cual sobreescribirá el archivo donde se guardan las condiciones.

Apéndice C

Manual ATVotaciones

ATVotaciones es el producto que se encarga de llevar acabo el proceso de votación, desde la configuración hasta la muestra de los resultados. Los actores en este sistema son los siguientes: **Administrador**, **Comisión**, **Electores** y **Candidatos**; estos actores sólo son usuarios de nuestro sitio *Plone*, lo que realmente los diferencia entre ellos, es su rol dentro del sistema de votación, y los permisos que deben de tener dentro del sitio *Plone* para poder cumplir su rol.

Para poder usar el producto hay que instalarlo, esto se realiza en la parte de **Configuración del Sitio -> Productos Adicionales** de nuestro sitio *Plone*. Una vez ahí seleccionamos ATVotaciones 1.0 y damos instalar. Las dependencias que deben estar instaladas para que el producto de votaciones funcione correctamente son: ATSelectUsers y MasterSelectWidget. Los productos instalados para está prueba se muestran en la figura G.1.

Figura C.1: productos - ATVotaciones

[Administrador] Una vez instalado podremos agregar un objeto de este producto.

Figura C.2: Agregar - ATVotaciones

Ahora veamos el procedimiento para trabajar con este objeto que hemos agregado.

1. **[Administrador]** Una vez agregado el objeto ATVotaciones se nos muestra la siguiente pantalla:

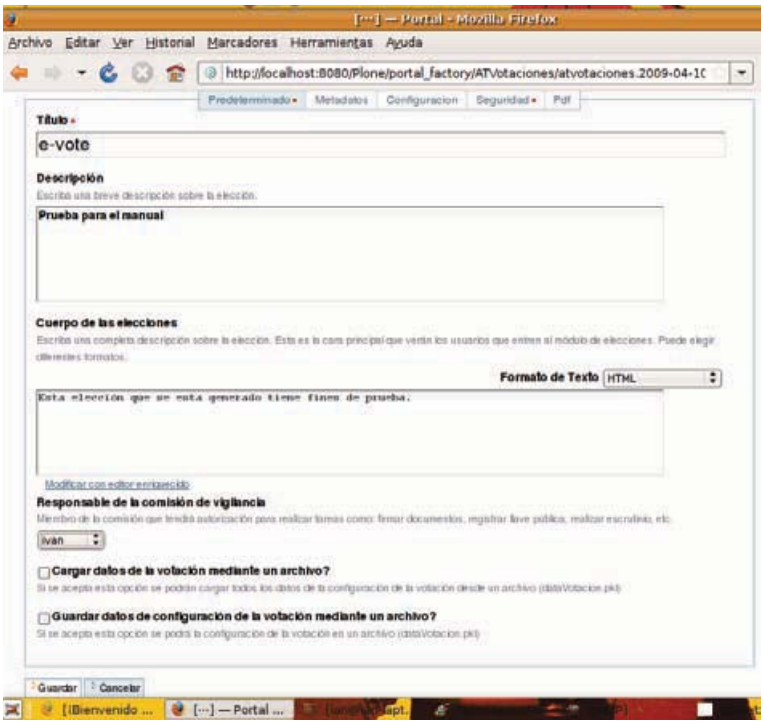


Figura C.3: Predeterminado - ATVotaciones

Esta pantalla forma parte de un grupo de cuatro (Predeterminado, Configuración, Seguridad y Pdf) donde en cada una, debemos poner información sobre la configuración de la elección. Podemos **Guardar** en cualquiera de las cuatro pantallas pero recomiendo que se llenen todas las pantallas, ya que si falta algún dato obligatorio, no se podrá seguir con los demás pasos de la elección.

a) Predeterminado: Esta es la primer pantalla en el camino para configurar la elección; debemos llenar los campos **Título**, **Descripción**, **Cuerpo de las elecciones** y **Responsable de la comisión de vigilancia** (quien será el actor **Comisión**).

Hay dos campos más que sirven para guardar o cargar una configuración.

1) **Guardar datos de configuración de la votación mediante un archivo?** al seleccionar está casilla le estamos ordenando al sistema que guarde en un archivo llamado dataVotacion.pkl la configuración actual, esto es para poder utilizarla en otras elecciones.
2) **Cargar datos de la votación mediante un archivo?** si seleccionamos está casilla entonces, no necesitamos llenar los datos de configuración de la elección, por que el sistema se encargara de cargarlas con la información que contenga el archivo dataVotacion.pkl que se encuentra en el directorio raíz de nuestra instancia de *Plone*.

b) Configuración: Una vez que llenamos los datos en Predeterminado, lo que sigue es dar clic en la pestaña **Configuración**.

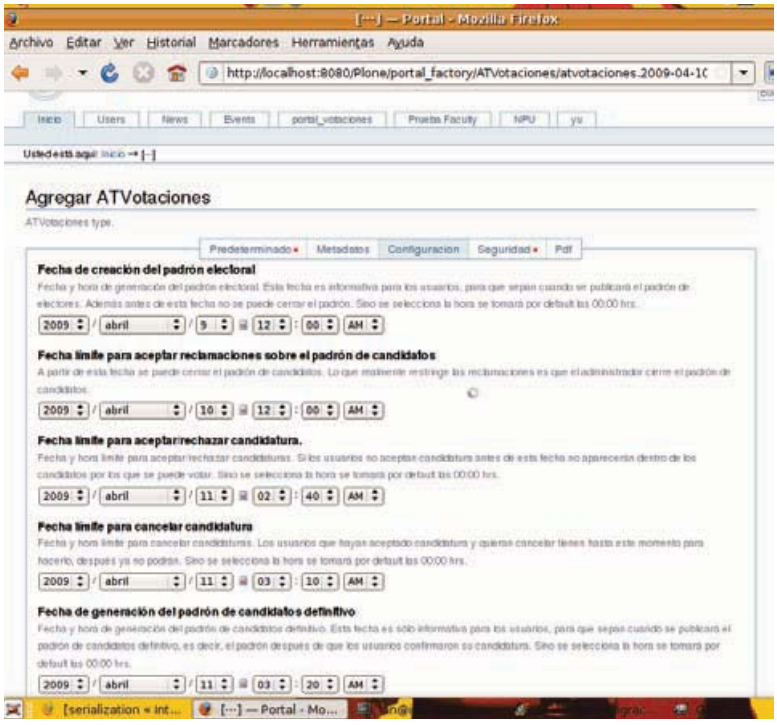


Figura C.4: Configuración 1 - ATVotaciones

En está primer parte de la pantalla configuración tenemos los campos:

1) Fecha de creación del padrón electoral: Esta fecha es informativa para los usuarios, para que sepan cuando se publicará el padrón de electores. Además antes de está fecha no se puede cerrar el padrón de electores.

2) Fecha límite para aceptar reclamaciones sobre el padrón de candidatos: A partir de está fecha se puede cerrar el padrón de candidatos. Lo que realmente restringe las reclamaciones es que el administrador cierre el padrón de candidatos.
3) Fecha límite para aceptar/rechazar candidatura: Si los usuarios no aceptan candidatura antes de está fecha entonces, **no** aparecerán dentro de los candidatos por los que se puede votar.
4) Fecha límite para cancelar candidatura: Los usuarios que hayan aceptado candidatura y quieran cancelar, tienen hasta este momento para hacerlo, después ya no podrán.
5) Fecha de generación del padrón de candidatos definitivo: Esta fecha es sólo informativa para los usuarios, para que sepan cuando se publicará el padrón de candidatos definitivo, este padrón definitivo está formado por los usuarios que **confirmaron** su candidatura.

En la segunda parte de la pantalla de Configuración tenemos lo siguiente:

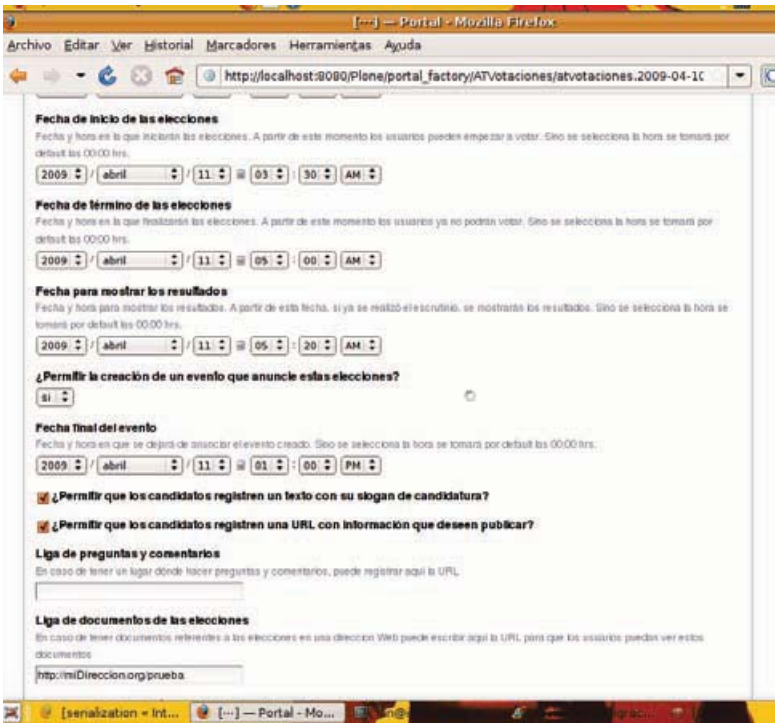


Figura C.5: Configuración 2 - ATVotaciones

1) Fecha de inicio de las elecciones: A partir de este momento los usuarios pueden empezar a votar.
2) Fecha de término de las elecciones: A partir de este momento los usuarios ya no podrán votar.
3) Fecha para mostrar los resultados: A partir de está fecha, si ya se realizó el **escrutinio**, se mostrarán los resultados.
4) ¿Permitir la creación de un evento que anuncie estas elecciones?. Si se selecciona crea un evento dentro del sitio *Plone*.
5) Fecha final del evento: Fecha y hora en que se dejará de anunciar el evento creado.
6) ¿Permitir que los candidatos registren un texto con su slogan de candidatura?. El slogan del candidato aparecerá en la pantalla donde el elector realiza su voto.

7) ¿Permitir que los candidatos registren una URL con información que deseen publicar?
8) Liga de preguntas y comentarios: Si se tiene un lugar dónde hacer preguntas y comentarios (un foro, blogs, etc), puede registrar aquí la URL.
9) Liga de documentos de las elecciones: En caso de tener documentos referentes a las elecciones en una dirección Web puede escribir aquí la URL para que los usuarios puedan ver estos documentos.

En la última parte de la pantalla de Configuración tenemos lo siguiente:

Figura C.6: Configuración 3 - ATVotaciones

1) Forma de ver los resultados: En está lista podemos seleccionar las opciones deseadas incluso las tres, están le dirán al sistema como presentar el resultado de las votaciones, por porcentaje, por número de votos. Al menos se mostrará al ganador de las elecciones.
2) ¿Permitir crear votos de prueba?: Si se acepta está opción se podrán ingresar votos de prueba en la urna generados manualmente.

c) Seguridad: En está parte de la configuración nos enfocamos en información sobre *GPG* y la llave pública del **Administrador**.
Lo primero que tiene que hacer el Administrador es generar un par de llaves criptográficas. Ver Manual *GPG* .
Una vez que el Administrador tiene su par de llaves, debe obtener la llave pública, la cual puede obtener con el siguiente comando: **gpg –export -a usuario**. Debe copiar la salida de ese comando y pegarlo en el campo **Llave pública GPG del administrador de las elecciones**.

Enseguida viene la **Longitud de los números aleatorios**, donde debemos poner la longitud que deseamos tenga la cadena aleatoria, esto ayuda a que sea difícil que el protocolo de votación sufra un ataque por texto conocido. Por lo tanto, mientras más grande... mejor.

En **Forma de utilizar *GPG*** recomiendo que se ponga **Obligatorio (siempre).** Esto lo digo por que todas las pruebas del sistema han sido realizadas con *GPG*.

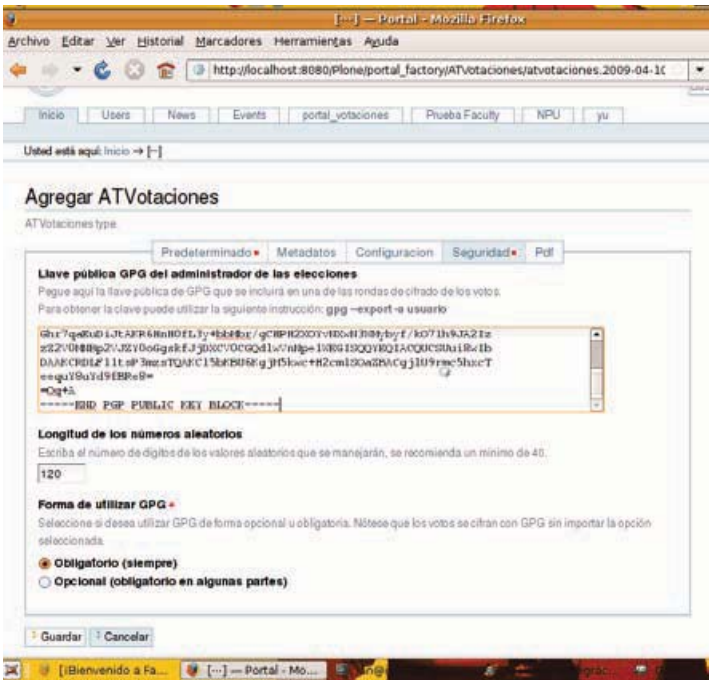


Figura C.7: Seguridad - ATVotaciones

d) Por último tenemos **Pdf**, que contiene un único campo **Título de los PDF's**, donde introducimos el título que se va a poner en cada archivo PDF creado en el proceso de la elección; además podemos incluir varias líneas, incluso código Latex.

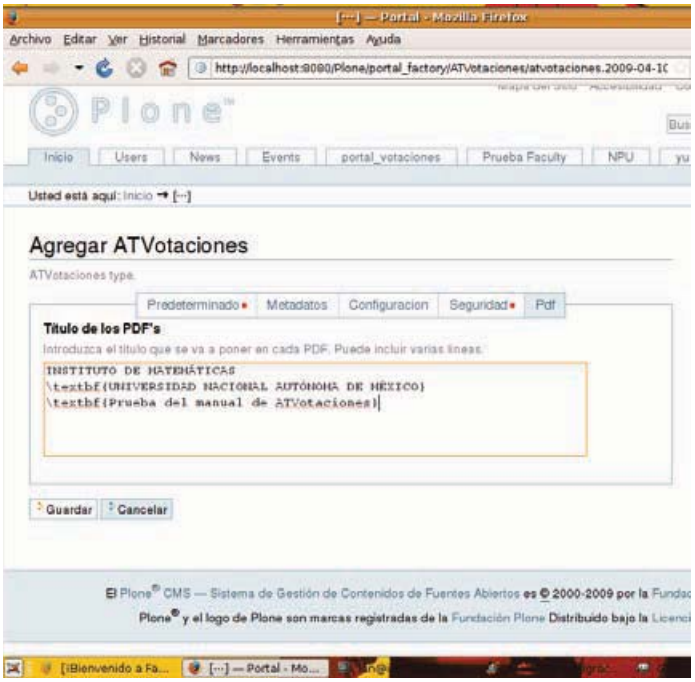


Figura C.8: Pdf - ATVotaciones

Cuando terminamos de configurar y damos clic en **Guardar**, aparece la siguiente pantalla, que corresponde a la pestaña **Visualizar.**

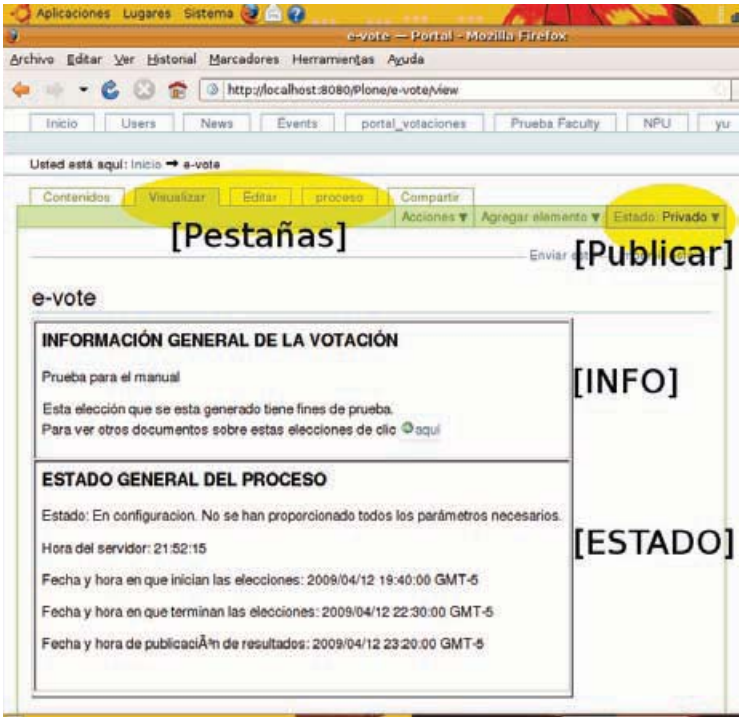


Figura C.9: Información y Estado de la votación

- **[INFO]**: Esta sección nos muestra información sobre la elección, está información es tomada de la que ingreso el administrador en la configuración.
- **[PUBLICAR]**: El administrador debe publicar la elección, si es que no se encuentra en estado: Publicado.
- **[ESTADO]**: Aquí podemos verificar las fechas en las que se realiza cada una de las etapas del proceso de elección, estas fechas son tomadas de la configuración de la votación. Algo importante es que también nos muestra el **estado** de la elección.
- **[PESTAÑAS]**: Como podemos observar en la imagen, tenemos marcadas tres pestañas que son: Visualizar, Editar y proceso.
  - **Visualizar**: Es la página que contiene información sobre la elección, donde podemos ver el estado de la elección, además donde al elector se el mostrara la liga para votar y al candidato se le mostrara la liga para que confirme o rechace su candidatura.
  - **Editar**: Esta pestaña nos envía a configurar la elección [1], esto es para quienes no llenaron todos los campos o para quienes quieren editar la información de la configuración, esto último no se puede realizar cuando uno quiera, si se acepta la configuración inicial (veremos más adelante como hacerlo y en que momento) entonces ya no se podrá editar la configuración de la elección.
  - **proceso**: Esta pestaña nos muestra varias secciones muy importantes para el **administrador** y el **comisionado**, ya que sólo ellos pueden acceder a está sección y además a cada uno se le van mostrando diferentes tareas a realizar, todo esto depende de si eres el administrador o el comisionado. En está parte de la pantalla de **proceso** podemos ver dos secciones:

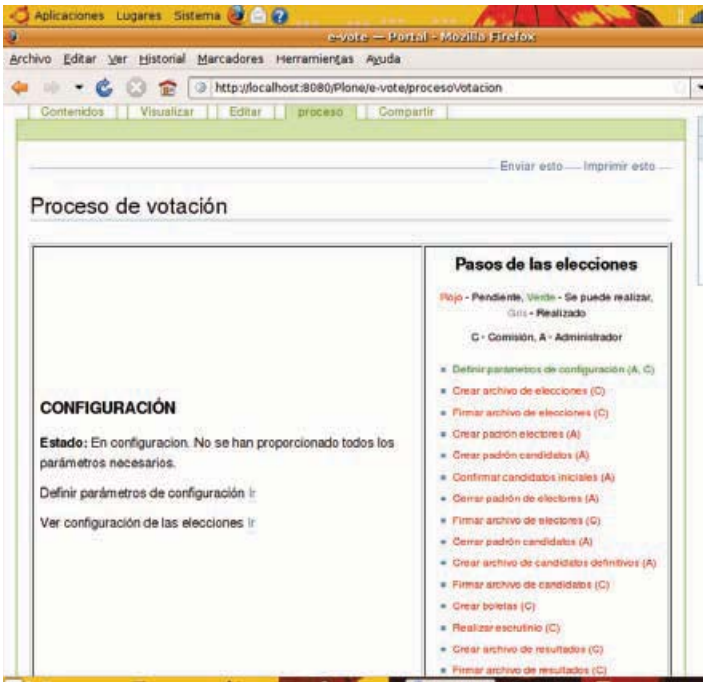


Figura C.10: Pantalla 1: Proceso de Votación

- CONFIGURACIÓN: Hasta este momento en está sección sólo aparece información sobre el estado de la elección, nos da una liga para editar la configuración (1). Más adelante aparecerán más opciones.

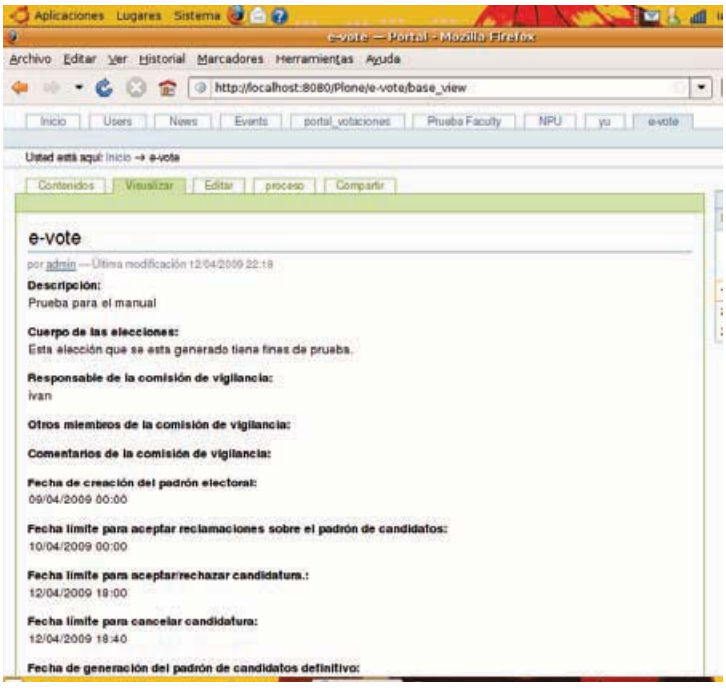


Figura C.11: Ver configuración de las elecciones

- Pasos de las elecciones: Esta sección está diseñada para guiar al administrador y comisionado en el proceso de elección, se enlistan las tareas capitales, así como un identificador que muestra a quien le toca dicha tarea, **A** para el administrador y **C** para el de la comisión. También podemos verificar si ya se puede realizar la tarea (**Verde**), si ya se realizó (**Gris**) o que está pendiente (**Rojo**).
  Como podemos observar en el listado de tareas, la primer tarea **Definir parámetros de configuración** es la única que se realiza por las dos partes (administrador y comisionado). La parte del administrador ya la realizamos al configurar todos los campos requeridos de la elección. La parte del comisionado se explicara en el siguiente paso (2).

Veamos la segunda parte de la pantalla **proceso** que se muestra en la siguiente figura.

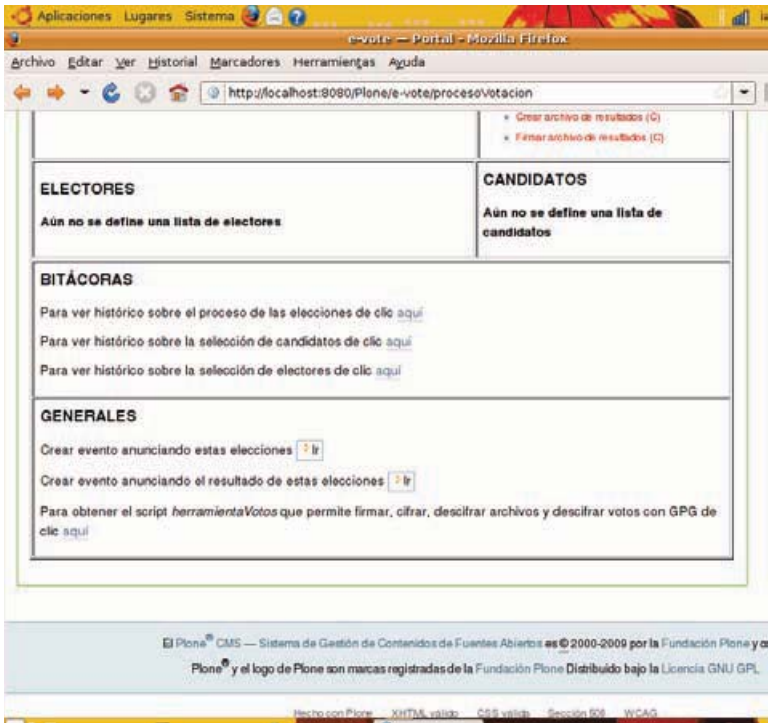


Figura C.12: Pantalla 2: Proceso de Votación

- ELECTORES: En está sección vamos a tener ligas para crear el padrón de electores, confirma el padrón de electores, firmar el archivo con la lista de electores, etc. Cada una de estas ligas irá apareciendo dependiendo si eres el administrador o el comisionado. Por el momento sólo hay un mensaje informativo.
- CANDIDATOS: En está sección vamos a tener ligas para crear el padrón de candidatos, confirma el padrón inicial de candidatos, firmar el archivo con la lista de candidatos definitivos, etc. Cada una de estas ligas irá apareciendo dependiendo si eres el administrador o el comisionado. Por el momento sólo hay un mensaje informativo.
- BITÁCORAS: Las bitácoras son un mecanismo que le otorga mucha confiabilidad al proceso de elección, podemos verificar en cualquier momento cambios al padrón de electores, al padrón de candidatos y al proceso de elección mismo.
- GENERALES: En está sección hay una liga para crear un evento anunciando las elecciones y otra para anunciar los resultados. Además de una liga para obtener un

script de *Python* con el cual descifrar la urna de votos, pero eso será llegando al final del proceso de elección.

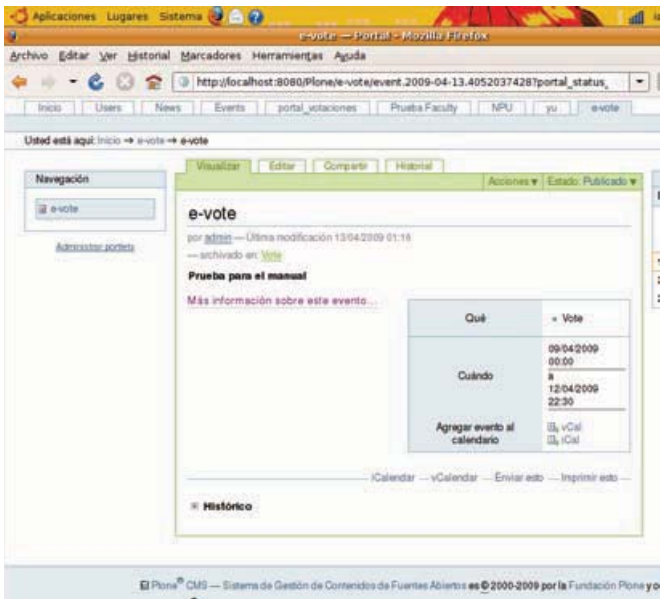


Figura C.13: Crear evento anunciando estas elecciones

2. **[COMISIÓN]** La primer tarea del comisionado es terminar de configurar la elección. El usuario que haya sido asignado como el miembro de la comisión en el paso anterior (1), debe entrar con su cuenta al sitio *Plone*, y dar clic a la pestaña de la votación correspondiente o al evento que se creó.

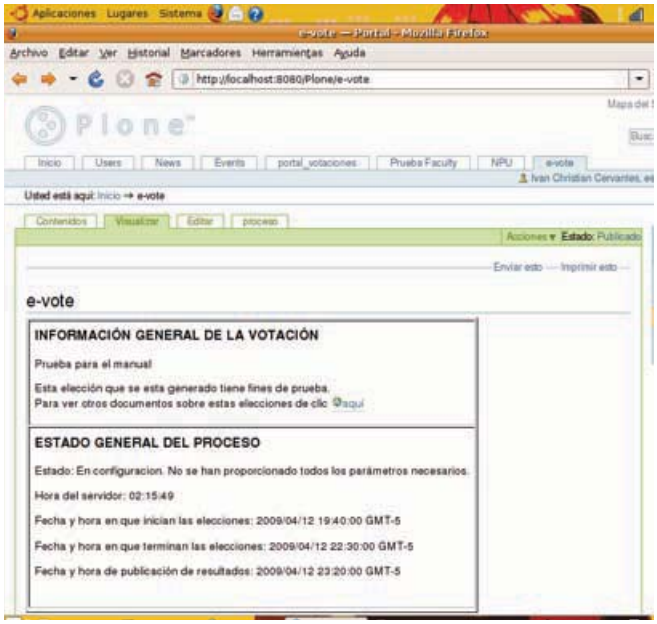


Figura C.14: Bienvenida - Comisionado

Como muestra la figura, el miembro de la comisión también tiene las pestañas: Visualizar, Editar y proceso.

a) Visualizar: Ver la explicación en el paso anterior.

b) proceso: Ver la explicación en el paso anterior.

c) Editar: El miembro de la comisión tiene que ingresar la siguiente información:

1) Otros miembros de la comisión de vigilancia: Hay que escribir el nombre de los miembros de la comisión de vigilancia, no tienen por que ser usuarios del sitio *Plone*. Sólo es un campo informativo.
2) Comentarios de la comisión de vigilancia: Podemos agregar algún comentario que de preferencia este relacionado con la comisión de vigilancia.
3) Llave pública *GPG* de la comisión: El miembro de la comisión tiene que crear su par de llaves criptográficas con *GPG*, el comando para ello es: gpg –gen-key. Después debe obtener su llave pública con el comando: gpg –export -a usuario. Copia la salida del último comando y lo pega en este campo. Para una explicación más detallada lea el manual *GPG* de este trabajo de tesis.

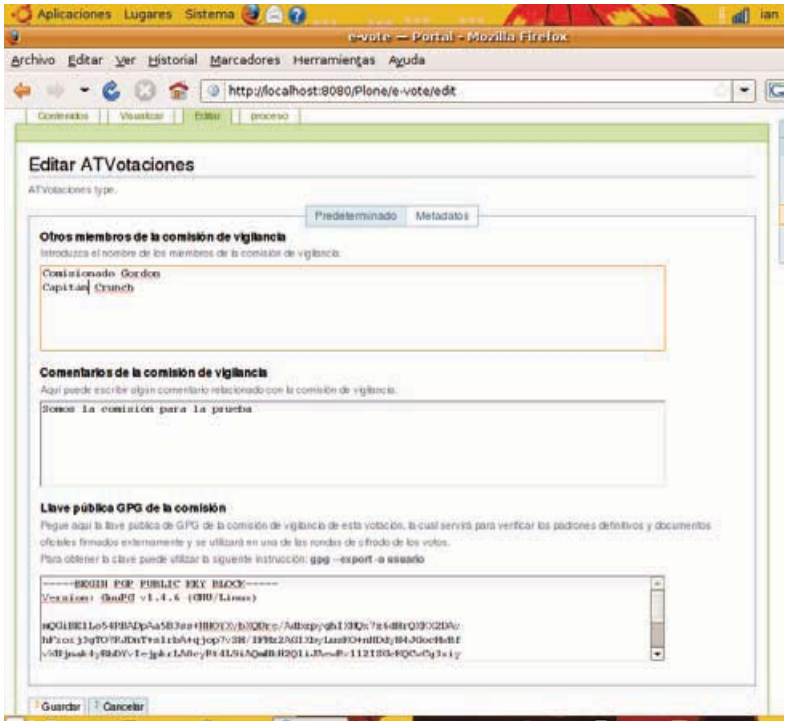


Figura C.15: Editar elección - Comisión

Damos clic en **Guardar** y nos dirijamos hacia proceso para ver el paso 3 del proceso de elección.

3. **[COMISIÓN]** Antes de cada paso a realizar tanto por el miembro de la comisión o por el administrador, es buena práctica checar la pestaña de **proceso**, para verificar que tarea es la que sigue, en que estado se encuentra la tarea, y el estado de la elección.

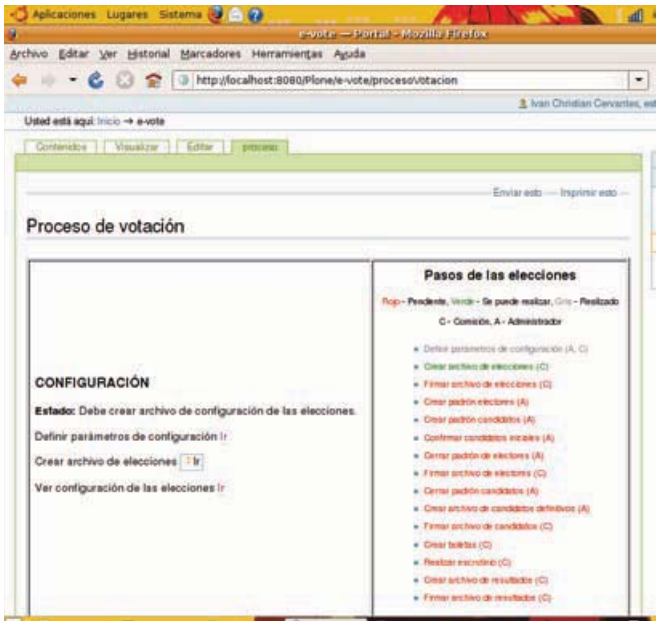


Figura C.16: Crear archivo elecciones - Comisión

Como nos muestra la imagen, la tarea **Definir parámetros de configuración (A, C)** ha sido realizada, por lo tanto está en color gris, ahora tenemos en verde la tarea **Crear archivo de elecciones (C)**, que le corresponde al miembro de la comisión.
En la sección de CONFIGURACIÓN vemos que hay un botón al lado de *Crear archivo de elecciones*. Al darle clic nos preguntara si realmente queremos crear el archivo. Damos clic en aceptar y enseguida se actualiza la página mostrándonos lo siguiente (paso 4).

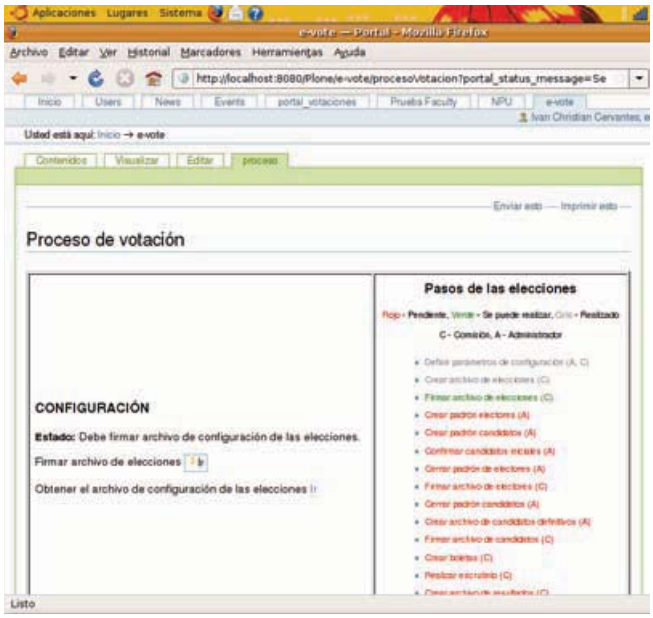


Figura C.17: Firma archivo elecciones

4. **[COMISIÓN]** Como podemos observar en la última imagen del paso anterior (3), tenemos que la tarea **Crear archivo de elecciones (C)** ya está en color **Gris**, y que en la sección CONFIGURACIÓN ya no se encuentra el botón referente a *Crear archivo de elecciones* en su lugar tenemos un nuevo botón referente a *Firma archivo de elecciones* que es la tarea de este paso 4. Damos clic en Ir. Lo cual nos manda a la siguiente página.

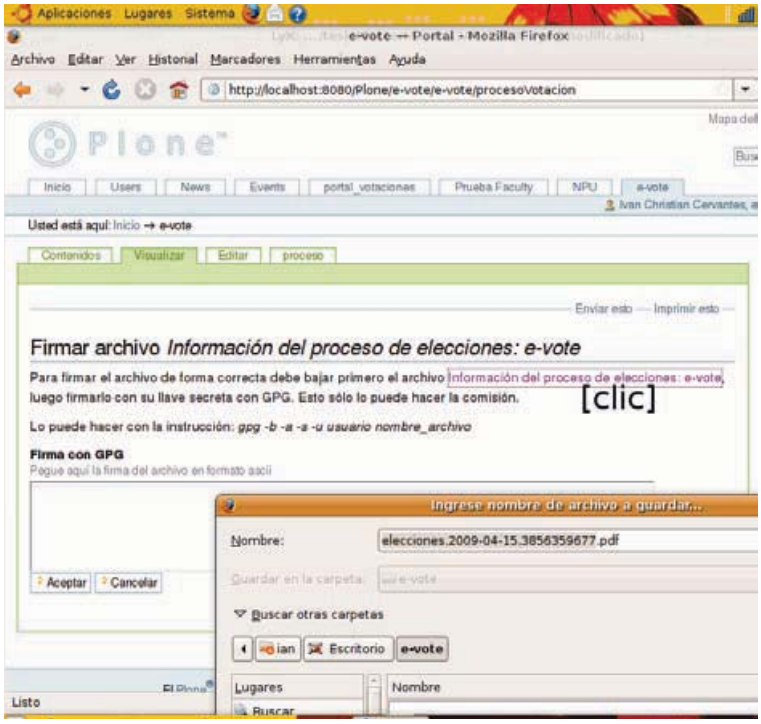


Figura C.18: Firma archivo de elecciones

En está parte del proceso el miembro de la comisión tiene primero que obtener el archivo **Información del proceso de elecciones: e-vote** esto lo logra dando clic en la liga. Lo siguiente es firmar el archivo (Manual *GPG*). Una vez que se tiene el contenido del archivo **.asc** lo que hacemos es pegarlo en el campo **Firma con *GPG***.

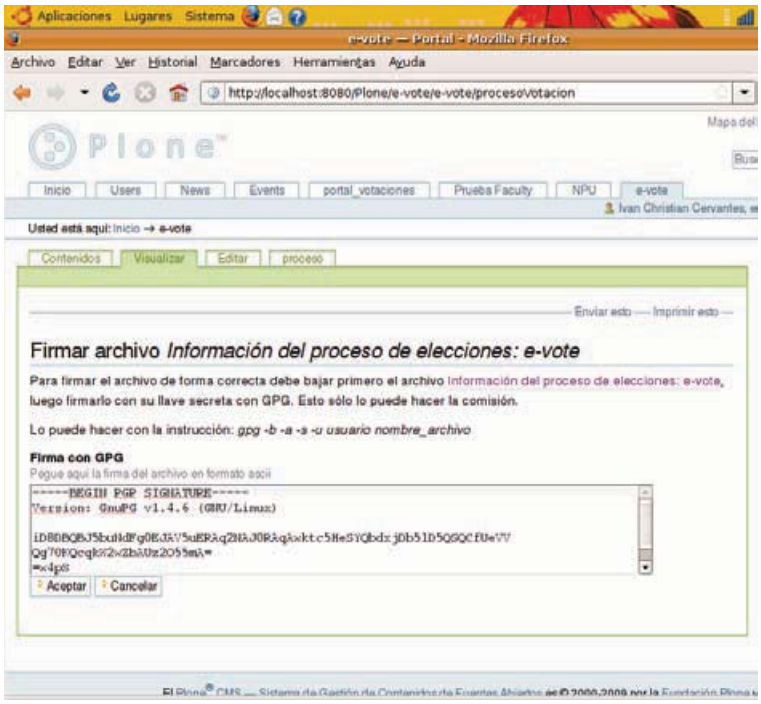


Figura C.19: Firma archivo de elecciones - 2

5. **[ADMINISTRADOR]** Ahora tenemos que crear el padrón de electores y el padrón de candidatos.

Figura C.20: Crear padrones - ATVotaciones

Damos clic en **Ir** de la sección de electores y el sistema nos mandará a nuestro ya bien conocido producto de selección de usuarios. Una vez creado nuestro padrón de candidatos podemos regresar a la pestaña proceso para generar el padrón de candidatos.
Antes de crear el padrón de candidatos podemos como lo muestra la imagen cerrar el padrón de electores, para que ya no pueda ser modificado el padrón electoral. NOTA: Podemos cerrar el padrón más adelante si se desea.

Figura C.21: Cerrar padrón electoral - ATVotaciones

6. **[ADMINISTRADOR]** Vamos a crear el padrón de candidatos. De la misma forma que el paso anterior. No olvidemos publicar el archivo de candidatos generado en este paso.

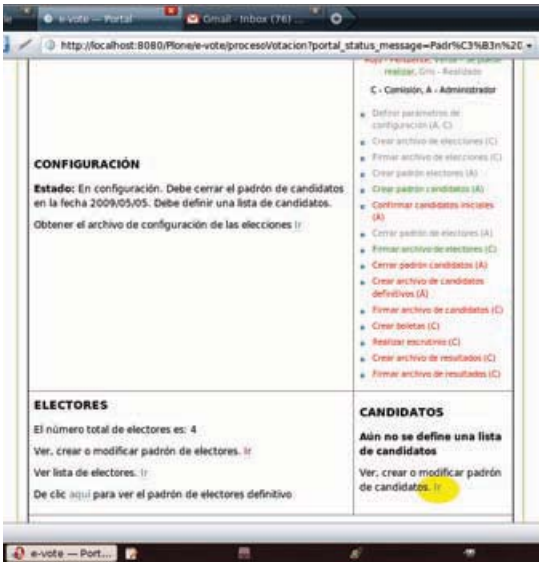


Figura C.22: Crear padrón de candidatos - ATVotaciones

7. **[ADMINISTRADOR]** Podemos ahora confirmar el padrón de candidatos inicial.

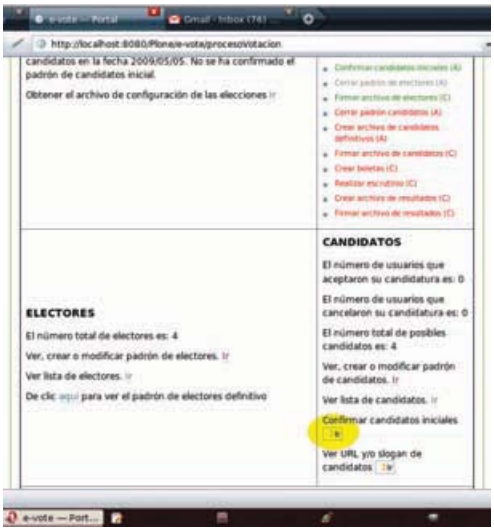


Figura C.23: Confirmar padrón candidatos inicial - ATVotaciones

Una vez que hemos confirmado el padrón inicial de candidatos, los usuarios que hayan sido seleccionados en el padrón de candidatos, podrán Aceptar o Rechazar su candidatura, aparte podemos realizar otras tareas como se muestra en la imagen.

8. **[ADMINISTRADOR]** Ahora vamos a cerrar el padrón de candidatos definitivos, lo que significa que ya no se podrá cambiar la lista de candidatos iniciales.

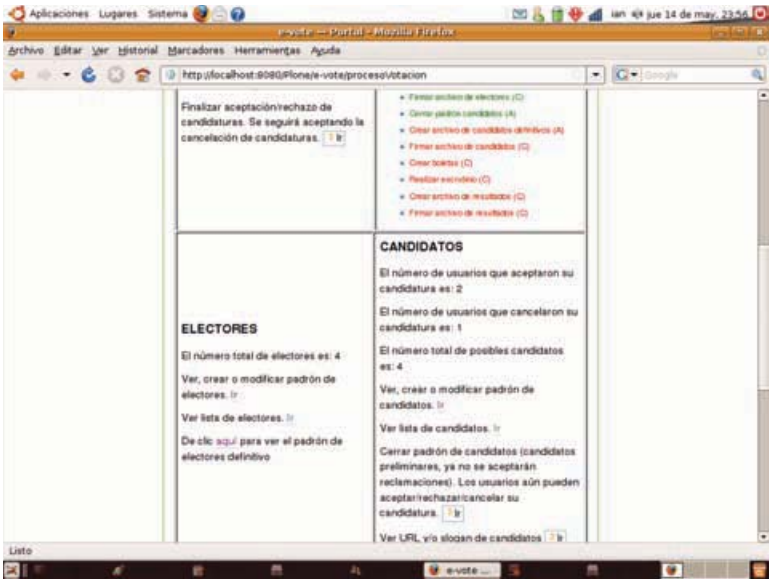


Figura C.24: Cerrar padrón candidatos - ATVotaciones

9. **[CANDIDATOS]** En está parte del proceso me ocupare en describir todo lo que puede realizar

un candidato: aceptar candidatura, rechazar candidatura y cancelar candidatura.
Aceptar candidatura: Una vez que se ha cerrado el padrón de candidatos iniciales, cada uno de los usuarios que pertenezcan al padrón de candidatos iniciales, podrá ingresar al sitio *Plone*, ingresar a la parte de la votación en donde se encontrara con que hay una sección (Estado como candidato), en la que vemos habilitado el botón que nos conduce a la pantalla de aceptación o rechazo de la candidatura.

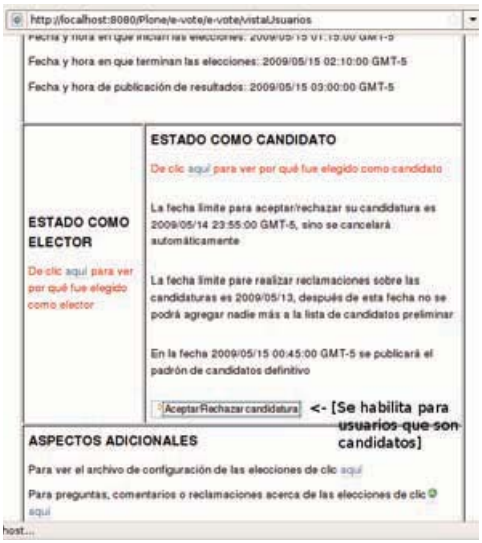


Figura C.25: Estado como candidato - ATVotaciones

Damos clic y nos encontramos con la pantalla para aceptar o rechazar la candidatura, damos clic en el radio button con la leyenda "Aceptar candidatura", además podemos poner un comentario, una URL y un slogan.

Figura C.26: Aceptar candidatura - ATVotaciones

Damos clic en **Aceptar** y eso es todo para aceptar una candidatura. Después de que damos clic en Aceptar nos regresa a la pantalla donde tenemos la sección "Estado como candidato", podemos cerciorarnos de que el botón con la leyenda "Aceptar/Rechazar candidatura" ha sido reemplazado por el de "Cancelar candidatura".

Figura C.27: Cancelar candidatura habilitado - ATVotaciones

Rechazar candidatura: Al igual que **Aceptar candidatura**, damos clic en el botón con la leyenda "Aceptar/Rechazar candidatura", ahora en la siguiente pantalla damos clic en **Rechazar candidatura**, además podemos poner comentarios.

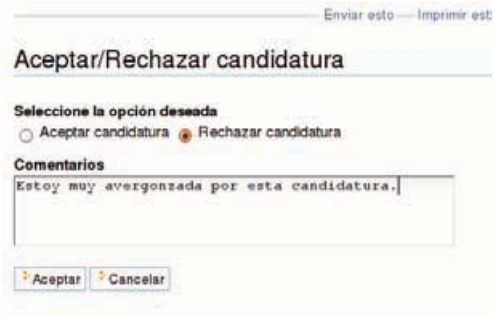


Figura C.28: Rechazar candidatura - ATVotaciones

Una vez que rechazamos la candidatura podemos darnos cuenta que a diferencia de aceptar candidatura en la que nos habilito el botón para poder cancelar la candidatura, en este caso ya no tenemos habilitada ninguna opción.

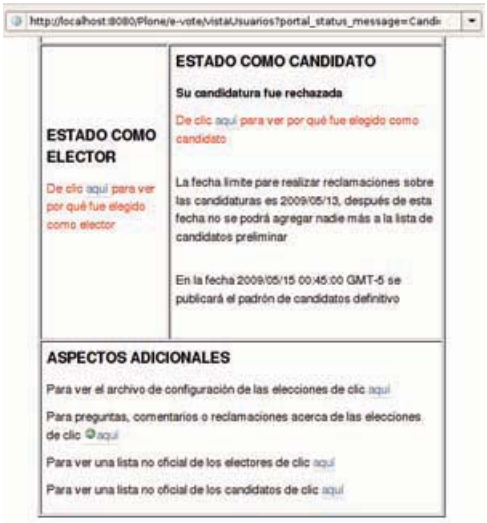


Figura C.29: Rechazar candidatura - ATVotaciones

Aceptar y después cancelar la candidatura: Una vez que aceptamos la candidatura se no habilita la opción de Cancelar la candidatura. Damos clic en dicho botón lo que nos conduce a la siguiente pantalla.

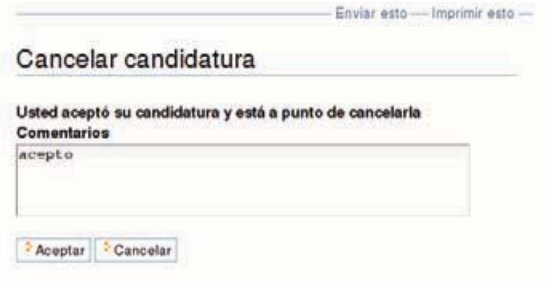


Figura C.30: Aceptar entonces cancelar - ATVotaciones

En la parte de comentarios, tendremos los comentarios que agregamos cuando aceptamos la candidatura.

10. **[ADMINISTRADOR]** Podemos finalizar la cancelación de candidaturas, si lo hacemos entonces los candidatos que no hayan cancelado su candidatura ya no podrán hacerlo.

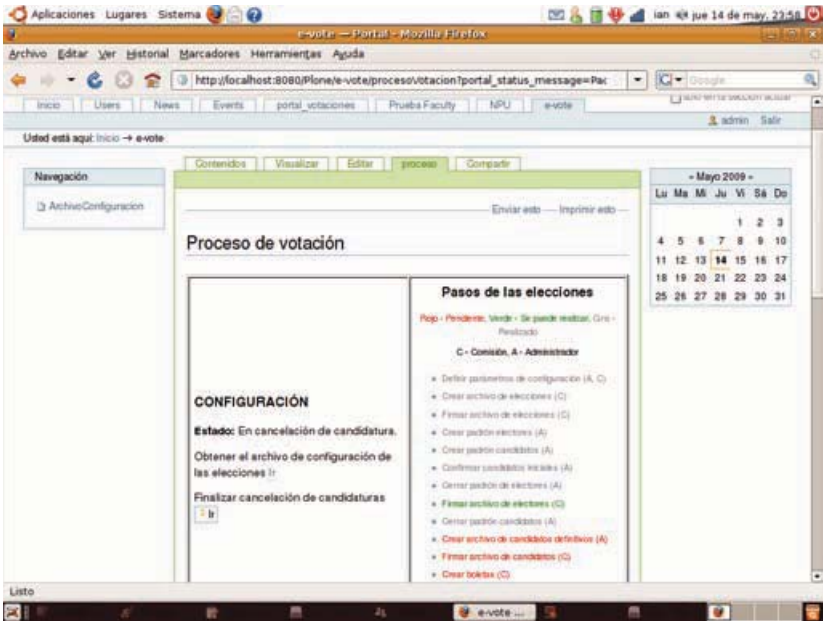


Figura C.31: Finalizar cancelación candidatura - ATVotaciones

11. **[ADMINISTRADOR]** Ahora lo que tiene que hacer el administrador es crear el archivo candidatos, este archivo contiene la lista de los candidatos DEFINITIVOS.

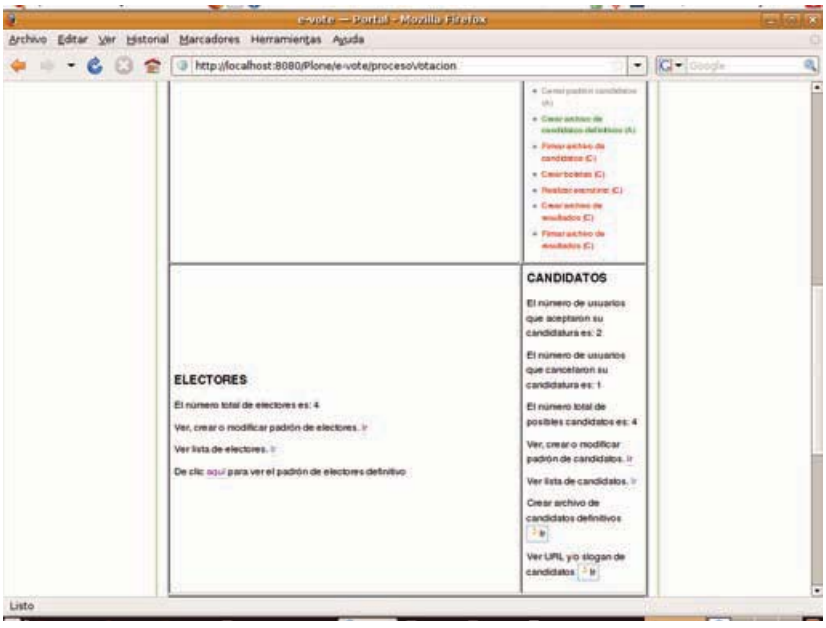


Figura C.32: Crear archivo de candidatos - ATVotaciones

12. **[COMISIÓN]** El miembro de la comisión puede firmar el archivo de electores (aunque pude hacerlo desde que se creó el archivo de electores). Bueno, damos clic en Ir.

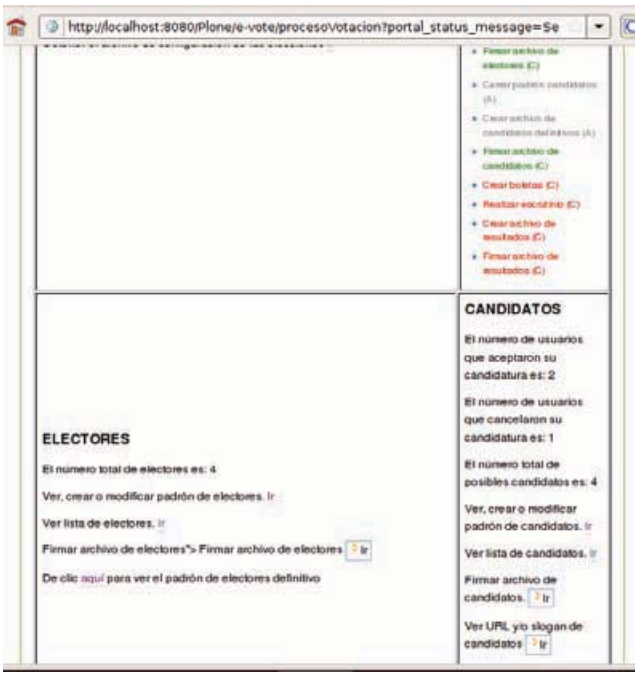


Figura C.33: Firmar archivo electores 1 - ATVotaciones

Después en la siguiente pantalla, damos click en la liga de descarga para poder bajar el archivo

Figura C.34: Descargar archivo electores - ATVotaciones

y lo firmamos como dice el manual *gpg*. Una vez firmado pegamos la firma y damos en aceptar.

Figura C.35: Agregar firma del archivo de electores - ATVotaciones

13. **[COMISIÓN]**Seguimos con la firma del archivo de candidatos. Es el proceso anterior. Damos clic en Ir.

Figura C.36: Firmar archivo candidatos - ATVotaciones

Después en la siguiente pantalla, damos click en la liga de descarga para poder bajar el archivo

Figura C.37: Descargar archivo electores - ATVotaciones

y lo firmamos como dice el manual *gpg*. Una vez firmado pegamos la firma y damos en aceptar.

Figura C.38: Agregar firma del archivo de electores - ATVotaciones

14. **[COMISIÓN]**Uno de los pasos de mayor importancia del protocolo de seguridad es crear boletas, y lo logramos con tan sólo dar clic en Ir. Y esperamos a que inicien las elecciones.

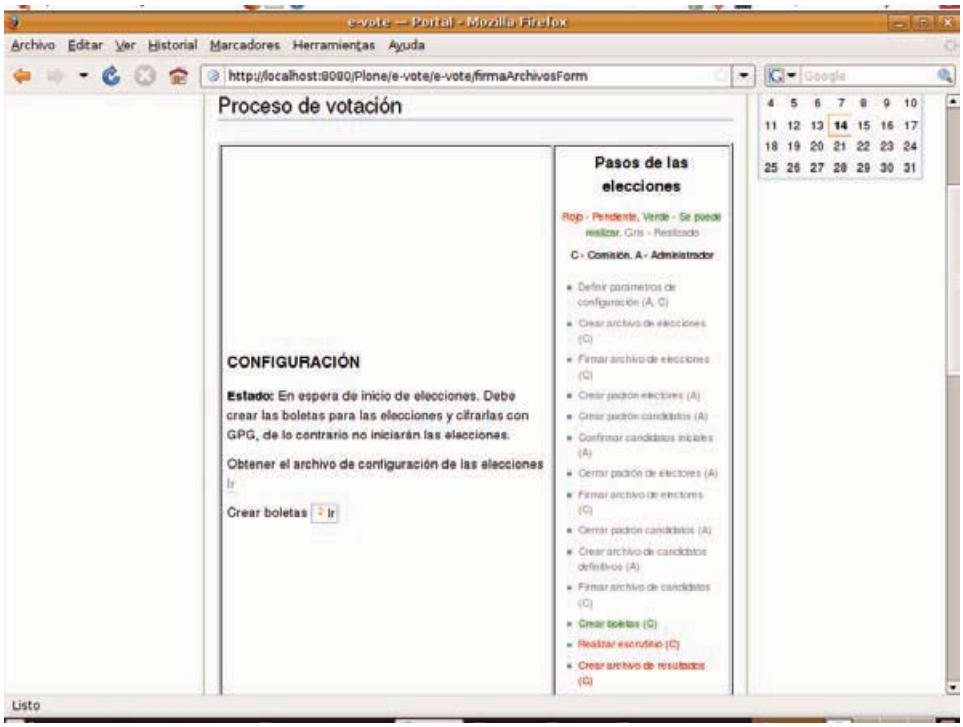


Figura C.39: Crear boletas - ATVotaciones

15. **[ELECTORES]** Es tiempo de que los electores tomen el poder de la votación, cada elector puede ingresar al sitio *Plone*, y dirigirse a la votación mediante el evento creado o en la pestaña correspondiente. Ejercer nuestro voto en el sistema sólo se puede hacer cuando la fecha de inicio de las votaciones llegué, está fecha la definimos al configurar el proceso electoral. Entonces, una vez que ingresamos a la pantalla de la votación, esto lo logramos dando clic en el botón color verde que tiene la leyenda "Votar",

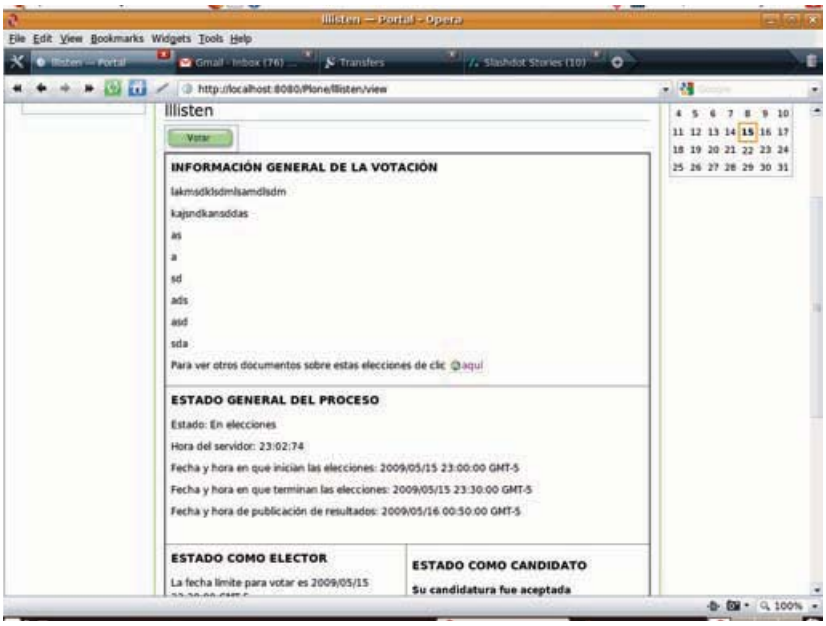


Figura C.40: Ingresar a la pantalla de votación - ATVotaciones

bien, entonces veamos la pantalla donde ejercemos nuestro voto.

Figura C.41: Pantalla boleta - ATVotaciones

En la pantalla se nos muestra cada uno de los candidatos finales, podemos ver su foto, su nombre, su slogan, y tenemos un radio button en cada candidato, es excluyente así que sólo podemos escoger UN candidato. Damos clic en "Votar por candidato elegido".

16. **[COMISIÓN]** Ahora vamos a comenzar la etapa de escrutinio, está comienza cuando termina el tiempo para votar, el cual definimos al configurar el proceso electoral. Damos clic en Ir.

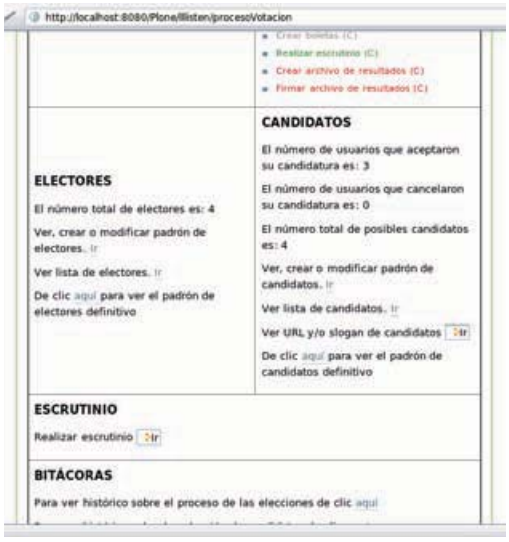


Figura C.42: Comenzar escrutinio - ATVotaciones

Bajamos el archivo de votos y el de herramientas,

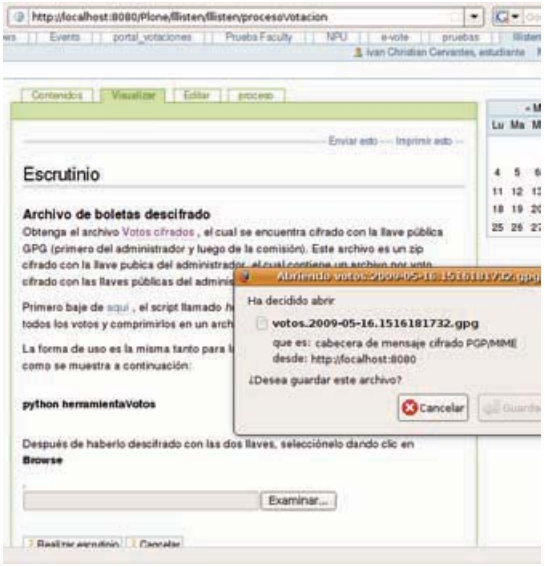


Figura C.43: Descargar archivo de votos - ATVotaciones

ESTE PUNTO ES MUY IMPORTANTE: Hay dos rondas de descifrado una por parte del administrador y otra por parte del comisionado. El miembro de la comisión debe darle al **administrador** el archivo de votos y el de herramientas que descargo.

a) **[ADMINISTRADOR]** Una vez que el miembro de la comisión le dio el archivo de votos y la herramienta (la herramienta puede ser descargada por su parte), entonces comenzamos a descifrar el archivo de votos (vease D.5 en la página 155). Una vez que descifro debe mandarle el archivo de salida al comisionado.

b) **[COMISIONADO]** El comisionado una vez que recibio el archivo debe llamar otra vez a la herramienta con la opción 4 para la segunda ronda de descifrado (vease D.5 en la página 155), y el archivo que resulta es el que se sube al sistema.

17. **[COMISIÓN]** Crear archivo de resultados y obtener el archivo de la tabla de códigos de los candidatos.

18. **[COMISIÓN]** Firmar archivo de resultados como lo muestra el manual de *GPG* de está tesis.

19. **[TODOS] Ver los resultados cuando llegue la fecha de publicación.**

Figura C.44: Resultados - ATVotaciones

Apéndice D

Manual *GPG*

Esta sección está escrita pensando en las personas que llevaran los roles de administrador y comisión del sistema de votaciones electrónicas, pero puede servir para cualquier usuario que quiera firmar archivos, firmar email, etc.

GPG provee funcionalidad para cifrar y descifrar datos, y para crear y verificar firmas, esto lo puede realizar gracias a la criptografía de llave pública.

D.1. Instalación

GPG está disponible para Windows, Linux, otros sabores de Unix y Mac OSX. Pueden descargarlo de la siguiente página: http://www.gnupg.org/download/

D.1.1. Windows

Hay una versión compilada para Windows (Instalador) por lo que no hay mayor problema. Hay que notar que está es una versión de línea de comando y viene con una herramienta gráfica de instalación (todo lo que se necesita).

D.1.2. Linux

La mayoría de las versiones de Linux ya vienen con la instalación de *GPG*, si no fuera el caso, entonces podemos instalarlo de forma automática mediante comandos como **apt** o **yum**.

También podemos descargar el código fuente y compilarlo:
#./configure
#make
#sudo make install

D.1.3. Mac

En Mac se recomienda bajar el código fuente y compilarlo:
#./configure
#make
#sudo make install
Para mayor información http://macgpg.sourceforge.net/

D.2. Crear nuestro par de llaves

El administrador y el miembro de la comisión deben realizar este proceso.

GPG utiliza criptografía de llave pública para cifrar y firmar mensajes (archivos). La criptografía de llave pública engloba tu llave pública la cuál puede ser distribuida al público y es usada normalmente por otras personas para cifrar mensajes que son enviados a ti y para descifrar firmas que tú hayas creado, y la llave privada la cual complementa tu llave pública te permite descifrar mensajes que recibes y para cifrar firmas. Juntas, se conocen como tu par de llaves.

D.2.1. Crear Llaves mediante –gen-key

Una vez instalado *GPG* en nuestro equipo, lo que debemos hacer para crear el par de llaves es seguir cada uno de los pasos aquí mostrados:

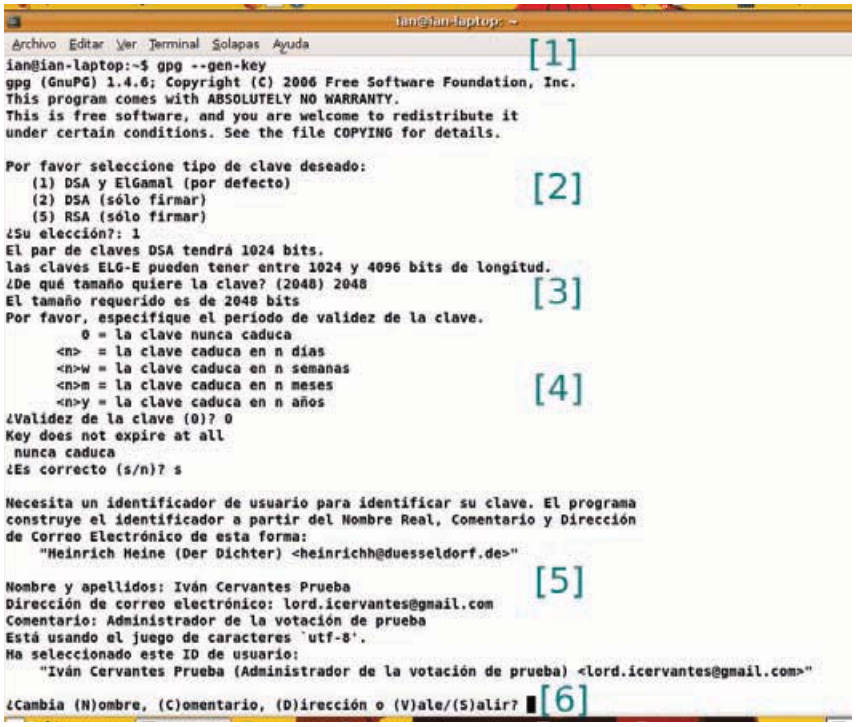


Figura D.1: Generar Llaves - GPG

1. En la línea de comandos tecleamos **gpg –gen-key** lo cual le dice a *gpg* que queremos generar un par de llaves criptográficas.

2. El tipo de llave deseado será **DSA y ElGamal** ya que las otras opciones sólo nos permiten firmar, y además ElGamal nos permite tener más de 1024 bits.

3. El tamaño de la llave se recomienda que sea de **2048** bits o más, aunque hay que decir que mientras más grande, cada operación que la utilice será más lenta.

4. El periodo de validez por lo menos debe de ser igual al tiempo que dura todo el proceso de la elección, pero también podemos decirle que siempre sea válida, esto lo hacemos dándole el valor **0**. Enseguida nos preguntara si es correcto.

5. Ahora hay que ingresar nuestro nombre real, un comentario y nuestra dirección de email, la concatenación de estos tres campos formara nuestro **id**, el cual será utilizado para poder identificar nuestra llave dentro de la colección de llaves.

   Si no nos sentimos cómodos poniendo nuestro nombre real a la llave, entonces debemos poner algo que pueda ser único, para que nuestra llave no se confunda con la de nadie más. Daré un ejemplo de problemas con el id de la llave más adelante.

6. Nos da la opción de cambiar el nombre, comentario, dirección o si queremos salir o continuar con el proceso de la generación de las llaves. Le damos **V** para continuar.

7. Por último hay que ingresar la contraseña y confirmarla, como ya se sabe mientras más compleja sea la contraseña mas difícil de romper; *GPG* no nos dará ninguna información de la contraseña mientras la estamos tecleando, quiero decir que, no imprime asteriscos o espacios. Esto es por medida de seguridad.

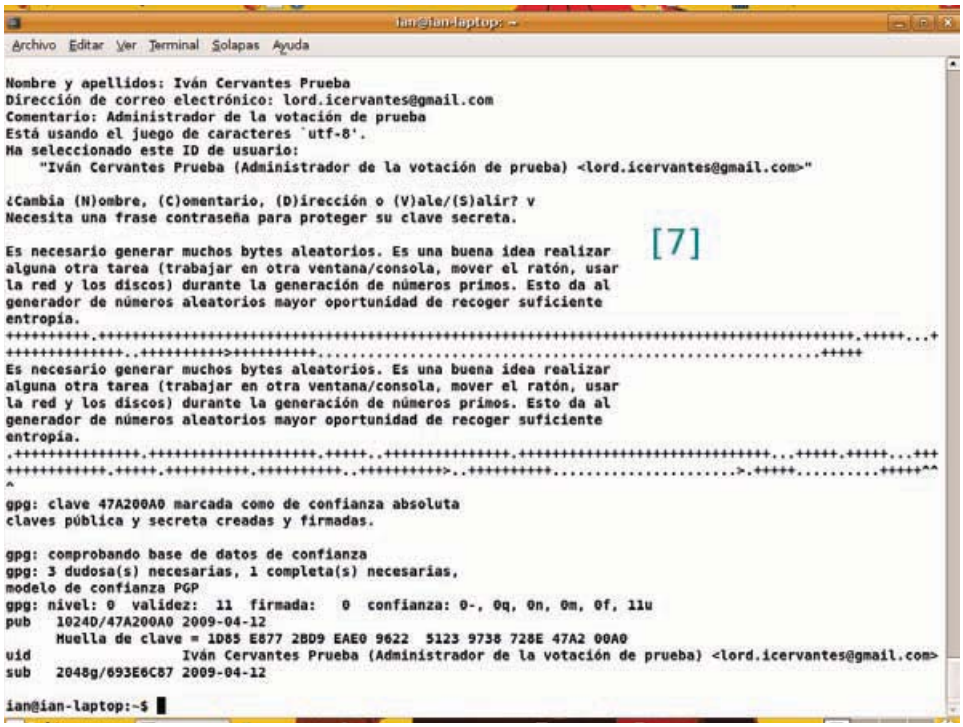


Figura D.2: Contraseña y entropía - GPG

D.3. Obtener llave pública

Una vez creado nuestro par de llaves, lo que sigue es poder obtener nuestra llave pública para poderla registrar en la **elección**. Esto lo debe hacer el administrador y el miembro de la comisión.

D.3.1. Listar llaves

Antes de obtener la llave pública sería conveniente poder en listar todas las llaves que se hayan creado para poder verificar la información perteneciente a la nuestra, en especial nuestro id y el quinto

campo separado por comas que corresponde a la huella de la llave. Para listar todas las llaves ejecutamos el comando: **gpg –list-keys –with-colons**

Lo cual generara un listado con información sobre todas las llaves.

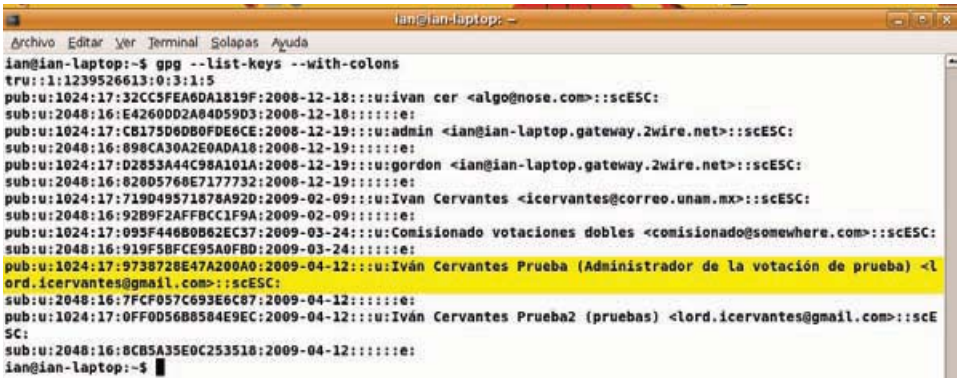


Figura D.3: Listado de llaves - GPG

La sección marcada es la que corresponde a nuestra llave pública creada anteriormente.

D.3.2. Exportar llave pública

Para poder obtener nuestra llave pública debemos ejecutar el siguiente comando: **gpg –export -a *cadena***

Donde cadena puede ser el nombre, comentario, email o huella de la llave. En la siguiente sección explicare los problemas que pueden surgir al tener, por ejemplo, compartir el mismo comentario con otra llave. Por lo tanto el mejor camino para especificar ***cadena*** es mediante la huella de la llave.

Figura D.4: Exportar - GPG

Lo que el administrador y el comisionado deben copiar y pegar en el sistema de votaciones es lo marcado en la imagen.

D.4. Firmar archivos

En el proceso de elección se pide que los archivos que se van generando sean firmados por el miembro de la comisión.

Una vez que obtuvo el archivo, debe ejecutar el siguiente comando: gpg -b -a -s -u cadena archivo. Como está parte es exclusiva del miembro de la comisión debo recordarle generar su par de llaves antes de querer firmar. Ver 11.2.

Por ejemplo si tenemos el archivo de configuración de las elecciones **elecciones.2009-04-15.3856359677.pdf** el cual lo guardamos en /home/ian/Escritorio/e-vote/ entonces lo que tenemos que hacer es abrir una consola y posicionarnos en ese directorio. Lo siguiente es ejecutar el comando gpg **-b -a -s -u** cadena archivo. En este ejemplo **cadena** será **Noam** y **archivo** será **elecciones.2009-04-15.3856359677.pdf**

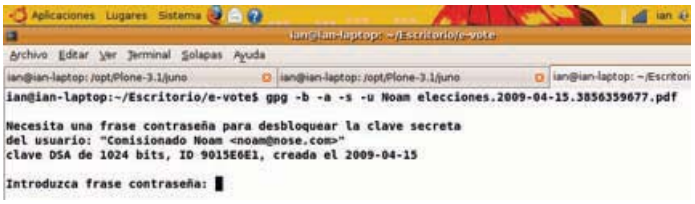


Figura D.5: Firma archivo con GPG

El comando nos va a solicitar la contraseña que hayamos puesto cuando creamos el par de llaves de Noam. Si la contraseña es correcta, nos crea un archivo en el mismo directorio y con el mismo nombre que el archivo que firmamos, pero le concatena la terminación **.asc** que significa que la firma está en formato ASCII.

Lo que nos queda es copiar el contenido del archivo con terminación **.asc**

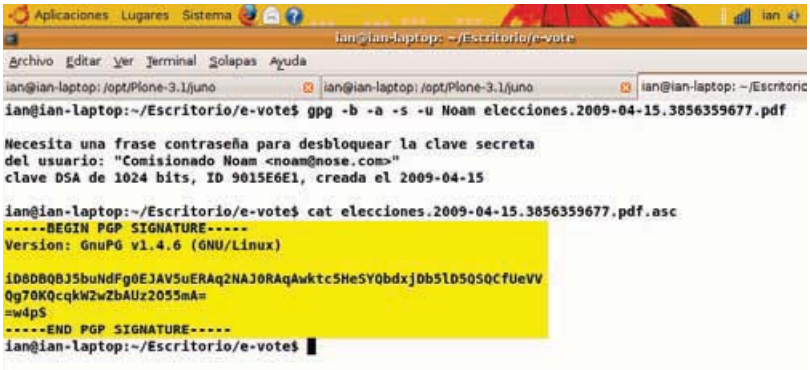


Figura D.6: Firma archivo con GPG - 2

D.5. Descifrar votos

Esta sección es muy importante en el buen desarrollo del proceso de votaciones. Una vez que se comienza el escrutinio y el miembro de la comisión descarga el archivo de votos, este comisionado debe mandarle el archivo de votos al administrador de las votaciones, quien debe descifrar los votos en su

primer ronda. Esto lo puede hacer con ayuda del archivo herramientaVotos que puede descargar de la sección **Generales** de la pestaña **proceso** de la votación.

Figura D.7: Obtener la herramienta de votos - ATVotaciones

Una vez que tenemos los dos archivos (votos y herramienta), vamos a la linea de comandos y nos posicionamos en el directorio donde se encuentre el archivo de votos. Ejecutamos el script herramientaVotos para esto necesitamos tener instalado *Python*.

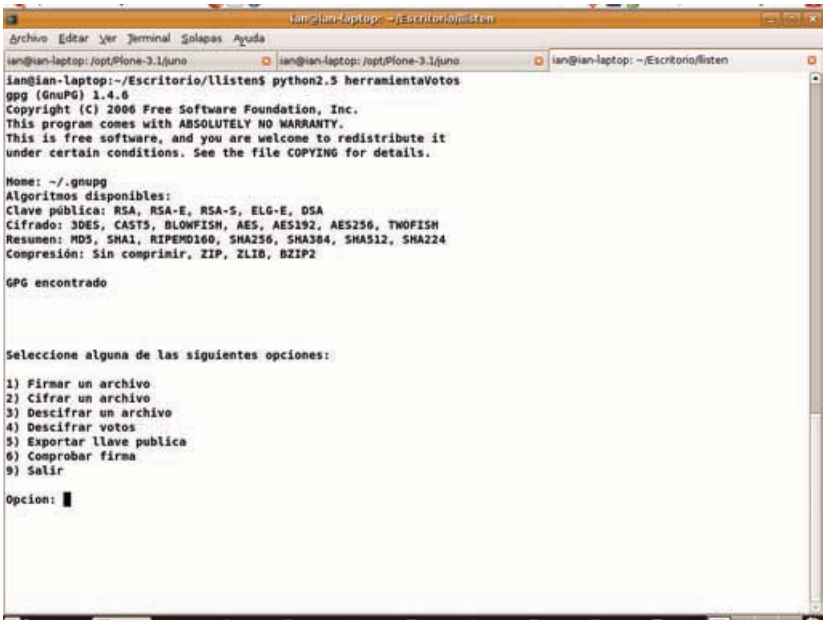


Figura D.8: Proceso descifrar votos 1 - ATVotaciones

Escogemos la opción 4, ponemos el nombre del archivo que contiene los votos y el nombre del archivo de salida. Este archivo de salida generado por el administrador debe ser entregado al miembro de la comisión para realizar la segunda ronda de descifrado. NO olvidemos ingresar la contraseña de la llave privada del Administrador.

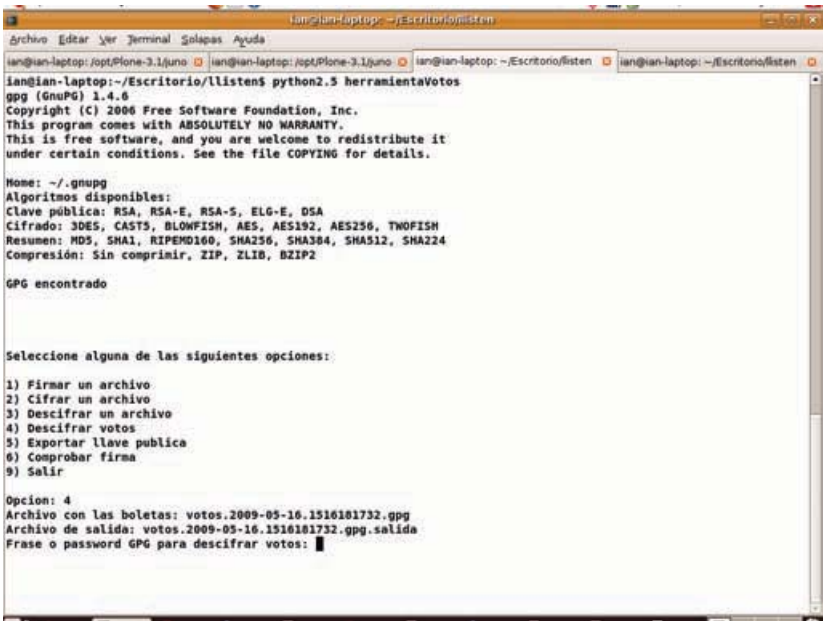


Figura D.9: Proceso descifrar votos 2 - ATVotaciones

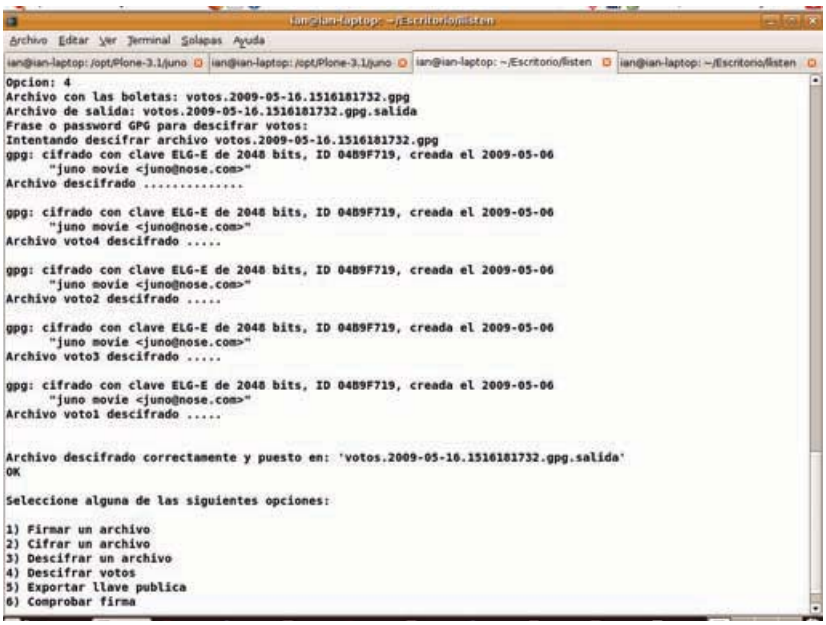


Figura D.10: Proceso descifrar votos 3 - ATVotaciones

El archivo **votos.2009-05-16.1516181732.gpg.salida** debe ser entregado al miembro de la comisión. Una vez que le miembro de la comisión tiene el archivo de votos descifrado por el administrador, es tiempo de que el comisionado descifre por segunda vez los votos.

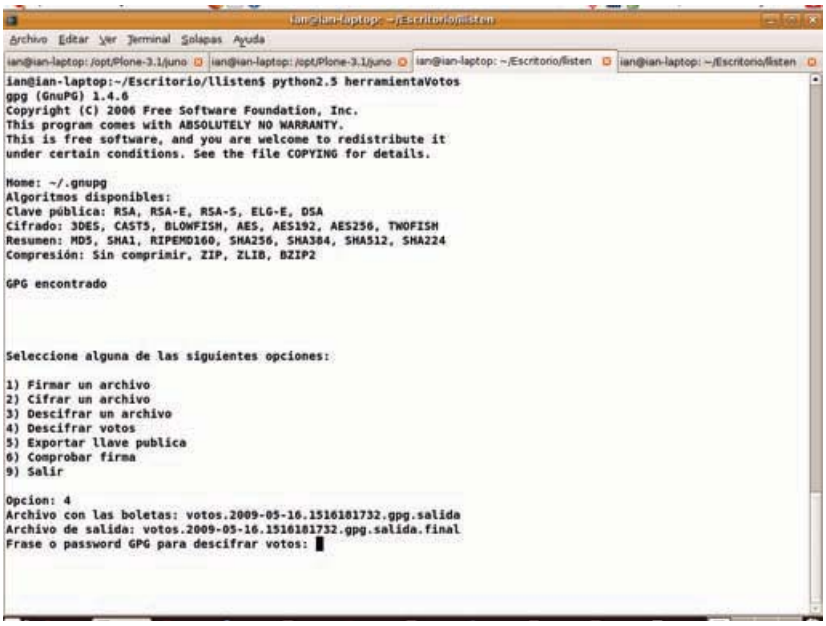


Figura D.11: Proceso descifrar votos 4 - ATVotaciones

Igual que con el Administrador debemos poner el nombre del archivo de votos que está vez es **votos.2009-05-16.1516181732.gpg.salida** y el nombre del archivo de salida que está vez será **votos.2009-05-16.1516181732.gpg.salida.final** este archivo es el que debe ser cargado por el comisionado.

D.6. Posibles detalles en la creación de las llaves

Bueno... hay detalles que pueden surgir a la hora de crear nuestro par de llaves, uno es la generación de entropía para que *GPG* pueda obtener suficientes bits aleatorios y la otra es elegir un buen id.

- Problema con la entropía: Cuando ingresamos nuestra contraseña y la verificamos (paso 7) *GPG* comenzará a generar todos los bits aleatorios que necesita para crear tu par de llaves. Para esto utiliza todo tipo de fuentes dentro de la computadora para simular aleatoriedad [entropía], incluyendo entrada en la consola, usar la red, el ratón, etc. Por lo tanto si no estamos generando suficiente aleatoriedad, *GPG* nos pedirá que la generemos realizando mas tareas. Así que si nos topamos con un mensaje como el siguiente: **Not enough random bytes available. Please do some other work to give the OS a chance to collect more entropy! (Need 300 more bytes)**. No hay que exaltarnos por que ya sabemos que lo que hay que hacer es: abrir mas consolas, ejecutar comandos, abrir pestañas en el explorador, abrir archivos de música, vídeo, etc.

- Problema con el id: Vamos a suponer que ya generamos un par de llaves con los datos del ejemplo de está sección, entonces el id es: Iván Cervantes Prueba (Administrador de la votación de prueba) <lord.icervantes@gmail.com>
  Como mencione anteriormente el id es la concatenación del nombre completo, comentario y correo. Ahora vamos a crear otro par de llaves con el id: Iván Cervantes Prueba2 (pruebas) <lord.icervantes@gmail.com>. Entonces a la hora de obtener la llave mediante la palabra *cervantes* vamos a tener problemas. Por lo tanto es muy importante tener un identificador único.

Bibliografía

[1] Misc KindSoftware: Software Engineering with Applied Formal Methods, K. S. E. http://kind.ucd.ie/products/opensource/archives/koa2.0.0.zip 2008

[2] Article Kiniry, J. R.; Morkan, A. E.; Cochran, D.; Fairmichael, F.; Chalin, P.; Oostdijk, M. & Ubres, E. KOA Remote Voting System: A Summary of Work To Date. Trusted Global Computing, 2006

[3] Thomas Lotze & Jan Ulrich Hasecke. A User's Guide to Plone. gocept gmbh & co. kg. 3rd Edition, February 2009

[4] Sitio de Plone. http://www.plone.org

[5] Grupo de usuarios de Plone en México. http://www.facebook.com/group.php?gid=27280942099

[6] Martin Aspeli. Professional Plone Development. Packt Publishing Ltd. 2007

[7] Relicenciar Plone. http://plone.org/foundation/newsitems/plone-foundation-membership-vote-framework-components-relicensing-policy. Mayo 2009

[8] Joomla CMS. http://www.joomla.org/

[9] aboutJoomla. http://www.joomla.org/about-joomla.html

[10] Lenguaje de programación PHP. http://www.php.net/

[11] Base de datos relacional MySQL. http://www.mysql.com/

[12] Sistema de Votaciones Electronicas de la UNAM. https://www.jornadaelectoral.unam.mx/sve.html

[13] Carta de auditoria informática sve 3.5. http://www.procesoselectorales.unam.mx/Documentos_Instructivos/Seguridad_SEV.pdf UNAM 2009.

[14] Misc None of the Above, NOTA Voters for None of the Above tm, http://www.nota.org/ 2006.

[15] Charles P. Pfleeger and Shari Lawrence Pfleeger, Security in Computing 4th Edition, Prentice Hall, 2006.

[16] Bruce Schneier, What's Wrong With Electronic Voting Machines?. http://www.schneier.com/essay-068.html, 2004.

[17] Bruce Schneier, Applied Cryptography: Protocols, Algorithms, and Source Code in C, John Wiley & Sons, Inc., 1996.

[18] Ronald R. Rivest & Warren D. Smith, Three Voting Protocols: ThreeBallot, VAV and Twin. Electronic Voting Technology Workshop, Boston, MA, August 6, 2007

[19] Misc Wikipedia Protest vote - Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/Protest_vote 2008.

[20] Alexandre Alapetite, Content accessibility of Web documents: Overview of concepts and needed standards, Risø-R-1576(EN), 2006.

[21] Welke, S., et al. "A Taxonomy of Integrity Models, Implementations, and Mechanisms." Proc National Computer Security Conf, 1990, p541551.

[22] Mayfield, T., et al. "Integrity in Automated Information Systems." C Technical Report, 7991, Sep 1991.

[23] Alexader Zapata Leis, Implementación de un sistema de votación electrónica como un producto sobre la plataforma Plone, IIMAS-UNAM, 2008.

[24] Michael Ian Shamos, Ph.D., J.D, Electronic Voting - Evaluating the Threat CFP'93, 1993.

[25] Dominique Cansell, J. Paul Gibson & Dominique Méry, Formal verification of tamper-evident storage for e-voting, Software Engineering and Formal Methods, 2007. SEFM 2007. Fifth IEEE International Conference on Volume , Issue , 10-14 Sept. 2007 Page(s):329 - 338.

[26] David Frith, E-voting security: hope or hype?, Network Security Volume 2007, Issue 11, November 2007.

[27] Gary T. Leavens & Yoonsik Cheon, Design by Contract with JML, 2006.

[28] Lilian Burdy, Yoonsik Cheon, David R. Cok, Michael D. Ernst, Joseph R. Kiniry, Gary T. Leavens, K. Rustan M. Leino, Erik Poll, An overview of JML tools and applications, Software Tools for Technology Transfer, 2003.

[29] Bracho Carpizo Javier. Convocatoria ordinaria para elegir al representante del personal académico del instituto de matemáticas ante el consejo técnico de la investigación científica para el periodo 2006-2009. IMATE, 2006.

[30] Sistema de Información del Instituto de Matemáticas. https://info.matem.unam.mx

[31] WebLion. http://weblion.psu.edu/

[32] Installing Plone 3. http://plone.org/documentation/tutorial/installing-plone-3-with-the-unified-installer

[33] Python 2.4. http://www.python.org/download/releases/2.4.6/

[34] PIL. http://www.pythonware.com/products/pil/

[35] pypi. http://pypi.python.org/pypi

[36] Archetypes. http://plone.org/documentation/manual/archetypes-developer-manual

[37] GenericSetup. http://plone.org/documentation/tutorial/genericsetup/

[38] http://plone.org/documentation/tutorial/buildout/packages-products-and-eggs

[39] Philipp von Weitershausen. Web Component Development with Zope 3. Springer, 2007.

[40] Penn State product FacultyStaffDirectory. http://plone.org/products/faculty-staff-directory/

[41] Voto de protesta. http://en.wikipedia.org/wiki/Protest_vote

[42] None of the above. http://www.nota.org/

[43] Lambrinoudakis C, Tsoumas V, Karyda M, Ikonomopoulos S. Secure E-Voting The Current Landscape. Dept. of Information and Communication Systems. University of the Aegean Karlovassi, Dept. of Informatics, Athens University of Economics and Business, p 1-19.

[44] Shamir A, "How to share a secret", Communications of the ACM, 22(1), p 612– 613, 1979.

[45] cpickle. http://docs.python.org/library/pickle.html

[46] Expresiones regulares - Python. http://docs.python.org/library/re.html

[47] Kathy Sierra & Bert BatesHead, First Java. O'Reilly Media Inc, 2005.

[48] The C Programming Language. http://plan9.bell-labs.com/cm/cs/cbook/

[49] Selenium software. http://seleniumhq.org/

[50] Elisabeth Freeman, Eric Freeman, Bert Bates & Kathy Sierra. Head First Design Patterns. O'Reilly, 2004.

[51] Servidor aplicativo Zope. http://www.zope.org/

[52] Implementación de OpenPGP. http://www.gnupg.org/

[53] CMF Form Controller. http://plone.org/products/cmfformcontroller

[54] Content Ratings. http://plone.org/products/contentratings

[55] Lista de productos para rating. http://www.contentmanagementsoftware.info/plone/rating-and-evaluation

[56] Proyecto ACE. http://aceproject.org/

[57] Repositorio notablesoftware. http://www.notablesoftware.com/evote.html

[58] Sitio oficial del IFE. http://www.ife.org.mx/

[59] Computational Social Choice. http://staff.science.uva.nl/~ulle/COMSOC/

[60] Sitio Oficial de Drupal. http://drupal.org/

[61] Drupal Voting API. http://drupal.org/project/votingapi

[62] Sitio Web de Joseph Kiniry. http://secure.ucd.ie/~kiniry/

[63] Ivar Jacobson, Grady Booch, James Rumbaugh. The Unified Software Development Process. Addison Wesley, 1999.

Índice de figuras

Índice de algoritmos