UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Posgrado en Ciencia e Ingeniería de la Computación

Elecciones en Plone 4: collective.elections

T E S I S

QUE PARA OBTENER EL GRADO DE:
MAESTRO EN CIENCIAS (COMPUTACIÓN)
PRESENTA:
HUGO ALBERTO RODRÍGUEZ PEÑA

DIRECTOR DE TESIS:
DR. SERGIO RAJSBAUM GORODEZKY
INSTITUTO DE MATEMATICAS

MÉXICO, D.F. AGOSTO 2014

Índice general

Resumen

En este trabajo se representa una reimplementación del protocolo de elecciones diseñado por Cervantes (2009)[1], se desarrolló el producto collective.elections para el CMS Plone en su versión 4. A diferencia de la implementación anterior este producto es independiente a las tecnologías utilizadas dentro del Infomatem[2], pero con la opción de adaptarse a estas así como a otras tecnologías similares. La nueva implementación se hizo teniendo en mente que el producto sea usado por la comunidad internacional de Plone con un esquema más general para las elecciones y posteriormente sea mantenido por la misma, de esta forma el IMATE tendrá un producto de votaciones sin necesidad de seguir su desarrollo para cada nueva versión de Plone.

Capítulo 1

Introducción

1.1. Objetivos de este trabajo

En este trabajo presenta la implementación de un producto desarrollado para Plone en su versión 4[3]. Versiones anteriores de este producto han sido implementadas para Plone 2 y 3, éstas fueron presentadas en los trabajos "Implementación de un sistema de votación electrónica como un producto sobre la Plataforma Plone" por Zapata (2008)[4] y "Migración y nuevas características del sistema de votación electrónica del instituto de matemática en la UNAM" por Cervantes (2009)[1]

En su versión mas reciente, los productos de elecciones se han presentando en congresos de Plone y atrajeron interés por parte de la comunidad pero dichos productos eran demasiado ad hoc a las necesidades del Instituto de Matemáticas (IMATE)[2] por lo que no eran fácilmente integrables dentro de otras instancias de Plone con diferentes configuraciones por ejemplo, si algún miembro de la comunidad de usuarios de Plone quiere hacer uso de estos productos de elecciones es necesario que adecué su instancia de Plone a las dependencias requeridas para el manejo de usuarios o bien realizar cambios al código fuente de los productos para ajustarlos a sus necesidades.

El objetivo de este trabajo es desarrollar un producto para realizar elecciones en línea, que cubran las necesidades del IMATE y ser lo suficientemente flexible para que pueda ser adoptado por otros miembros de la comunidad e integrado en sus instancias de Plone de una manera sencilla. La relevancia de que el producto sea adoptado por mas miembros de la comunidad radica en que se colocará en un repositorio público donde el mantenimiento se realizara por miembros de la comunidad, de esta forma el IMATE tendrá un producto de elecciones sin necesidad de encargarse de las nuevas versiones.

1.2. Elecciones

Las elecciones es un método tradicional para llegar a un consenso al elegir entre diferentes opciones, es usado en muchos niveles de complejidad desde elegir cosas sencillas entre amigos como un restaurante para ir comer, hasta elecciones a nivel nacional para elegir un presidente.

Para los diferentes niveles de complejidad de las elecciones se implementan diversos sistemas dependiendo de las características que pueda requerir (costos, privacidad, rapidez, etc), existen sistemas tan sencillos como levantar las manos y realizar un conteo de las manos levantadas por opción, esta es una forma sencilla de realizar una elección de forma rápida que se usa de forma cotidiana cuando sea desea una elección con costo prácticamente nulo y donde la privacidad de los electores no importa, por ejemplo cuando deseamos ponernos de acuerdo entre amigos para elegir los ingredientes de una pizza.

Otro sistema que se usa ampliamente es el de escribir la opción elegida en un papel y depositarlo en una urna donde posteriormente serán contados los votos que tuvo cada opción, este esquema generalmente lo usamos cuando queremos que la elección de cada elector sea secreta, por sencilla que parece una votación de este tipo dependiendo de la importancia de la privacidad, la cantidad de participantes, etc, puede requerir la implementa de un sistema sencillo o uno muy complicado.

Un ejemplo de sistema sencillo es el implementado dentro de un salón de clases usando un bote para recolectar los votos, repartir pedazos de papel y marcadores a los electores para pedir que anoten la opción deseada y la introduzcan dentro del bote. Sin embargo cuando queremos hacer una elección de algún representante a nivel estatal, este sistema es despreciable debido a las características de seguridad mínimas inherentes a éste, con un sistema como este sería muy fácil corromper los resultados de la votación, por ejemplo se podrían crear votos falsos debido a la falta de un registro de votantes.

Para una elección estatal nos encontramos con que el proceso de votación es el mismo pero por la cantidad de personas que necesitan acudir a votar y la importancia de de la privacidad de la elección, la implementación anterior no es suficiente, es necesario una mejor organización, como la creación de boletas especiales, identificaciones para que cada persona pueda votar una sola vez, comités de vigilancia, etc. Lo cual hace que para elecciones similares en las cuales utilizamos el mismo esquema de votación los sistemas implementados para cada una de ellas sean muy diferentes.

Debido a la necesidad de diseñar sistemas para elecciones cada vez mas confiables y seguros, los sistemas de elecciones han ido incorporando a su funcionamiento el uso de nuevas tecnologías que les permiten ser mejores en alguno de estos aspectos. Por ejemplo

el"Electric Voting-Recorder" diseñado por Thomas A. Edison en 1869[5], para su uso en el congreso, el cual nunca fue usado debido a que era demasiado rápido.

El sistema de elecciones de nuestro país es un buen ejemplo de la incorporación de diferentes tecnologías para su mejor funcionamiento, puesto que en diversas partes del proceso se han incorporado diferentes tecnologías, que permiten tener elecciones mas confiables. Para que un elector pueda emitir su voto es necesario que se identifique con una credencial especial diseñada con medidas de seguridad para que sea infalsificable, emite sus votos en hojas especiales que evitan le emisión de múltiples votos, las urnas donde se deposita el voto de igual forma están diseñadas con sellos para que no puedan ser abiertas y los votos no sean modificados, entre otras, Instituto Nacional Electoral[6].

En diferentes partes del mundo se han realizado estudios para encontrar la mejor manera de aprovechar las nuevas tecnologías. El California Institute of Technology y Massachusetts Institute of Technology tiene un proyecto llamado "Voting Technology Project" el cual fue iniciado en el año 2000 se encargan de estudiar todos los aspectos de un proceso de elección y como este puede ser mejorado haciendo el uso de tecnologías emergentes[7].

1.2.1. Elecciones electrónicas

Las elecciones tradicionales realizadas con el esquema de voto en boletas de papel, es confiable siempre que cada una de sus etapas sea llevada a cabo correctamente y de manera honesta, algunas desventajas de este sistema son la necesidad de presentarte en un lugar específico pare que los electores emitan su voto, la lentitud al realizar el conteo de los votos, la necesidad de lugares físicos designados para realizar las elecciones, personas que se encarguen de revisar presencialmente el proceso, etc.

El desarrollo de las computadoras ha permitido que se utilicen para diseñar e implementar sistemas que se encarguen de llevar a cabo diferentes procesos de una elección. Existen sistemas que se utilizan para emitir los votos, para el conteo de votos, encargados de emisión y conteo de votos, hasta los que se encargan de todo el proceso de elecciones. Sistemas para escanear y contar marcas en papel fueron utilizados por primera vez para revisar exámenes como los que se hacen para entrar a las escuelas en los años 50, el primer sistema de este tipo que se utilizo para votaciones fue el denominado Votronic en 1965[8].

Actualmente existen sistemas denominados maquina de votación DRE (direct-recording electronic voting machine)[9], que como su nombre lo indica, son usadas para emitir votos, dependiendo del sistema éstos pueden ser usados solamente para emitir votos, o bien para emitirlos y contabilizarlos. Unas de las ventajas de las maquinas DRE son su

fácil adaptación para que personas con discapacidades pueden emitir sus votos. Estas máquinas han tenido gran aceptación, en particular en Brasil donde la totalidad de las elecciones se realizan haciendo uso de uno de estos sistemas, su uso fue progresivamente aumentando desde 1996 cuando se usaron por primera vez[10].

Un característica que comparten todos los esquemas de elecciones antes mencionados es la necesidad presencial del elector para emitir su voto, este requerimientos restringe la posibilidad de que una persona emita su voto, por que no siempre tiene la posibilidad de asistir a la emisión de su voto por diversos motivos tales como trabajo, enfermedad, inclemencias del clima, etc.

Para evitar la no emisión del voto por motivos de movilidad se han implementado diferentes soluciones: la posibilidad de emitir un voto vía correo postal, emitir voto en un lugar ajeno al que te es asignado, voto vía correo electrónico, entre otros. Las dos primeras opciones son posibilidades con las que se cuenta en el sistema electoral mexicano, la última es una opción que un elector puede elegir en Alaska [11].

Estonia en el 2005 fue primer país que permitió el voto en línea como una opción a nivel nacional para elecciones locales, en ese año 9,317 personas votaron en línea lo que representa el 1.9 % de los votos, en el año 2011 para las elecciones del parlamento, 140,846 personas votaron en línea el 24.3 % de los votos totales. Esta sistema de elecciones permite el voto en línea como una opción, dicho voto puede ser cambiado en cualquier momento siempre y cuando este cambio sea hecho en las fechas asignadas o bien puede ser cancelado si el elector emite un voto presencial, esto se hace con la finalidad de evitar la coerción de votos. También se necesita una tarjeta especial así como un dispositivo para la lectura de la misma[12].

Se han diseño e implementado sistemas electorales en línea que se encargan completamente de las diferentes etapas de la elección, desde la selección de electores y nominados hasta el conteo de los votos y entrega de resultados. Un ejemplo de estos sistemas es Helios Voting[13], aunque los creadores de dicho sistema no recomiendan que éste sea usado para elecciones estatales, debido a que no están diseñados para lidiar con un gran intento de defraudar o coaccionar a los electores.

Las elecciones electrónicas todavía son un gran tema de debate por la desconfianza que se tiene a los sistemas electrónicos, debido a que requieren de un conocimiento técnico para entender su funcionamiento, no pueden ser auditados por cualquier persona, el temor a que sean modificados, etc. Ciertamente estas dudas no son injustificadas, estudios hechos sobre sistemas para escanear boletas y maquinas DRE, han probado que dichos sistemas son susceptibles a ataques que pueden modificar los resultados[14][15], de igual forma al sistema para emitir votos en línea de Estonia se le han encontrado vulnerabilidades[16].

A pesar de lo anterior, no a todos los sistemas de elecciones electrónicas se le han encontrado vulnerabilidades, sin embargo es un hecho que no todo sistema pueda llevar a cabo cualquier tipo de elección cada sistema está diseñado para ser usado para cierto tipo de elecciones dependiendo de las garantías de seguridad que las mismas requieran.

Los sistemas de elecciones en línea son una buena alternativa al esquema tradicional de votar en boletas de papel, mucha logística se puede delegar al sistema que tiene implementando el sistema de elección, los votantes no deben presentarse en un lugar en específico para emitir su voto, etc. Sin embargo se debe tomar especial cuidado en la seguridad del sistema ya que no queremos que una elección se pueda ver comprometida, es deseable que el sistema de elecciones usado sea lo mas seguro posible y cumpla con las garantías de seguridad requeridas.

1.2.2. Propiedades de seguridad en elecciones

Un sistema de elecciones en línea debe de cumplir varias propiedades de seguridad para garantizar la integridad de las mismas, este punto ya se ha estudiado ampliamente en los trabajos de Zapata(2008)[4] y Cervantes(2009)[1] realizados para crear productos para las elecciones dentro del IMATE. Las propiedades de seguridad que son deseables en un sistema de elecciones son:

1. Correctitud: el resultado de la elección debe ser congruente con los votos emitidos por los electores.
2. Privacidad: ningún voto debe de ser asociado a un elector.
3. Verificabilidad: se debe asegurar que los votos fueron contados correctamente y sean congruentes con los votos emitidos.
4. Justicia: no debe ser posible obtener resultados parciales de una elección para ningún usuario antes de emitir su vote.
5. Robustez: el sistema debe producir un resultado y mantener su funcionamiento bajo cualquier circunstancia.
6. Democracia: siempre que alguien tenga derecho a votar debe ser capaz de emitir su voto y que esta sea contado.
7. Incoercibilidad: nadie de ver capaz de obligar a un elector a votar de cierta forma.

Es importante notar que estas propiedades se pueden satisfacer en diferentes grados, acorde a las necesidades que tenga el sistema implementado.

1.3. Trabajo Previo

El desarrollo de productos en Plone para realizar las elecciones dentro del IMATE se debe al Infomatem que está desarrollado en esta Plataforma en la cual ya se tiene la información del personal del IMATE y de aquí es fácil tomar la información necesaria para organizar una elección sin realizar un mayor esfuerzo, las ventajas de Plone sobre otros manejadores de contenido ya fueron expuestas en Zapata(2008)[4] Cervantes(2009)[1].

1.3.1. Implementación de un sistema de votación electrónica como un producto sobre la plataforma Plone

Esta es la tesis de maestría realizada por Alexander Zapata Lenis, en donde se revisaron distintos sistemas para realizar elecciones poniendo especial atención en los sistemas de elecciones en línea y el grado en que estos satisfacía las garantías de seguridad que debe de tener una elección en línea.

Se desarrolló un protocolo para elecciones en línea tomando como referencias principales el sistema de elecciones del gobierno de Suiza y el sistema Kiezen op Afstand (KOA)[17] que su traducción literal seria Votación a distancia, KOA es un sistema para elecciones desarrollado por el gobierno de Holanda, se eligieron estos dos sistemas debido a la detallada información con la que se contaba acerca de ellos así como las garantías de seguridad que dichos sistemas cumplían.

Dicho protocolo fue puesto a revisión por expertos para que éstos dieran su aprobación, asegurando de esta forma que se tiene un protocolo que cumple los requerimientos de seguridad óptimos pare realizar una elección en línea.

En el trabajo desarrollado por Zapata(2008)[4] se hace una revisión del CMS Plone[3] poniendo especial énfasis en la seguridad y como dicho CMS nos permite tener e implementar las diferentes requerimientos de seguridad necesarios para garantizar una correcta implementación del protocolo de elección que se desarrolló.

1.3.2. Migración y nuevas características del sistema de elección electrónica del Instituto de Matemáticas de la UNAM

Esta es la tesis de maestría realizada por Iván Christian Cervantes Coronado en el 2009, el trabajo desarrollado por Cervantes(2009)[1] es una migración del producto de elecciones en línea realizado por Alexander Zapata para el sistema Plone en su versión 2 a el sistema Plone su versión 3, así como una revisión del protocolo de elecciones reafirmando las garantías de seguridad. Por otro lado, también se realizo una comparación entre

el producto de elecciones en línea desarrollado para el Plone y con productos parecidos para otros CMS, al igual que la descripción los cambios que se tuvieron que realizar para la migración debido a que varios paquetes de los que dependía dejaron de ser soportadas en la nueva versión de Plone.

1.3.3. ATVotaciones

ATVotaciones es el producto de elecciones desarrollado en Zapata(2008)[4] Cervantes(2009)[1], debe su nombre a ArcheTypes[18] la herramienta que se usaba principalmente para el desarrollo de productos para Plone 2 y 3, sin embargo este producto depende completamente de Faculty/Staff Directory (FSD)[19] para su funcionamiento.

FSD es un producto de Plone para el manejo de usuarios, dicho producto es mantenido por al comunidad que hace uso de Plone, para cada versión nueva de Plone la comunidad realiza la migración de el producto para continuar su uso.

ATVotaciones depende fuertemente de una extensión de FSD desarrollada por el IMATE para ajustarlo mejor a a sus necesidades. Estos productos se encargan de el gestionamiento de los usuarios del Sitio Plone, siendo éstos los que se toman en cuenta como votantes, lo cual es ideal para el uso interno del IMATE ya que es exactamente lo que se desea, pero a su vez interpone una limitación muy grande para otros personas que deseen hacer uso de este producto para realizar elecciones en línea.

También tiene otra gran limitación al hacer uso de ATSelectUsers para realizar la lista usuarios del sistema que tendrán derecho a participar en las elecciones, así como para hacer la lista de los candidatos, siendo ésta la única forma de realizar este proceso lo cual de nuevo nos limita a hacer uso de FSD.

ATVotaciones también falla al no hacer uso *Workflows* para llevar a cabo el proceso de elecciones, siendo estos uno de los de componentes principales de Plone.

ATVotaciones cumple los siguientes propiedades de seguridad:

1. Solamente se permite votar a usuarios autorizados
2. Ningún voto se puede contar mas de una vez
3. Los votos son guardados de manera segura
4. Los votos no pueden ser modificados o removidos sin ser detectado
5. Es posible verificar que todos los votos fueron contados
6. Lo votos no se pierden

1.4. Contribuciones de la tesis

Actualmente el CMS Plone cuenta con muchos productos para su uso que son mantenidos por la comunidad debido al interés en común en tener ciertas funcionalidades en sus sitios. Realizar elecciones es una actividad común para casi todas las organizaciones, realizan estos ejercicios al tomar decisiones que afectarán a las personas dentro de la misma, por lo que es una funcionalidad deseable para un CMS, actualmente no existe un producto que haya sido acogido por la comunidad para cubrir esta funcionalidad.

Hacer un producto que cumpla estas expectativas no es tarea fácil, pues los productos anteriores han servido para realizar elecciones dentro del IMATE satisfactoriamente, pero no han sido acogidos por la comunidad, por lo que se necesita desarrollar uno más general, sin perder las funcionalidades que le son de gran utilidad al IMATE.

Se tiene la experiencia de los productos anteriores que no utilizaban satisfactoriamente todas las funcionalidades que nos brinda Plone para realizar un proceso de elecciones, debido a que estos productos también dependen de otros productos que no son de uso estándar dentro de un sitio Plone. Dichas dependencias son indispensables para el funcionamiento del sitio del IMATE por lo que fue necesario desarrollar un producto que no tengas las dependencias, pero puedan agregarse a su funcionamiento de una manera sencilla y aprovechar al máximo las funcionalidades de Plone para implementar el proceso de elecciones.

Por lo tanto, se desarrollo un producto para el CMS Plone, este nos permite llevar a cabo un proceso de elecciones completo basado en el sistema KOA y cuenta con las garantías de seguridad adecuadas para realizar elecciones en línea. El proceso va desde la configuración de la elección hasta la entrega de resultados, siguiendo un protocolo para que cada uno de los estados por los que pasa la elección pueda ser revisado por cualquier usuario y de presentar alguna irregularidad ésta sea corregida.

Se documentará el proceso de creación del producto y la selección de las tecnologías usadas para su desarrollo, de tal forma que la tesis será un documento que pueda ser consultado por personas que quieran desarrollar nuevos productos para el sitio Plone del IMATE, con el fin de que la curva de aprendizaje de nuevos desarrolladores sea menos pronunciada. Plone es bien conocido por ser un CMS para el cual es complicado desarrollar debido a su curva de aprendizaje y por esta razón las implementaciones anteriores del producto de elecciones no aprovechaban todas sus capacidades.

1.5. Apoyo de la comunidad internacional de Plone

Para el desarrollo de este nuevo producto de elecciones se contó con la colaboración de Hector Velarde, es miembro de la Plone Foundation[20] y miembro activo de la comunidad desde el 2005. Hector ha trabajado con anterioridad en productos para el sitio del IMATE y fue uno de los interesados en el productos de votaciones. El fue quien señalo las deficiencias que tenia dicho producto en cuanto su implementación usando Plone.

Hector fue muy importante para le desarrollo de este trabajo, pues gracias a su experiencia y colaboración, se evitaron prácticas incorrectas al desarrollar productos para Plone. Además me guió en el desarrollo y aconsejó sobre productos que se podían utilizar sin cometer errores de las implementaciones anteriores, manteniendo un producto de uso general.

1.6. Estructura de este trabajo

En el primer capitulo se presentó un panorama muy general de las elecciones electrónicas, así como de los trabajos previamente desarrollados para dar una solución a este problema dentro del IMATE.

El segundo capítulo introduce las tecnologías que se consideran mas importantes en el desarrollo de collective.elections, veremos qué funciones cumplen, la importancia de cada una de ellas y una breve explicación de su uso.

El tercer capitulo se dedicará a examinar la implementación de ATVotaciones, explicaremos brevemente los errores de diseño cometidos en su implementación y la solución que daremos a estos errores .

En el cuarto capitulo se detallará el proceso llevado a cabo para la implementación del nuevo producto.

El quinto y último capítulo servirá para conclusiones.

Capítulo 2

Tecnologías

En este capitulo revisaremos las tecnologías que se consideran mas importantes para el desarrollo de un producto en Plone, el uso que tiene en el proceso de desarrollo y sencillos ejemplos de su utilización. Muchas de estas tecnologías son modernas y no existían cuando se desarrollaron los productos anteriores de elecciones.

2.1. ZopeSkel

ZopeSkel[21] es una herramienta para el desarrollo de proyectos para Plone y Zope[22], sirve para la creación de plantillas a partir de las cuales es posible desarrollar desde productos con Dexterity[23] o ArcheTypes, hasta instancias de Plone. Para desarrollar productos con Zopeskel y Dexterity se debe incluir el paquete zopeskel.dexterity en la instalación de Plone.

Parar crear una plantilla para desarrollar un producto de Plone se debe ejecutar el comando: `../bin/zopeskel dexterity`, esto ejecuta un script que solicita Nombre, descripción y otro detalles sobre el proyecto que se va a crear como se muestra en 2.1 y crea un paquete con el nombre del proyecto con el código necesario.

Para agregar un *Content Type* se debe incluir el producto creado a la instalación de Plone y desde el directorio del proyecto se debe ejecutar el comando: `../../bin/paster addcontent dexterity\_content`, este nuevamente solicita la información acerca del *Content Type* como en 2.2 y crea los archivos necesarios.

Después de estos 2 pasos se compila el sitio Plone y es posible agregar el *Content Type* creado al sitio, este es un sencillo que solo cuenta con título y descripción.

```
qb@qbs:~/Downloads/PlonePrueba/zinstance$ bin/zopeskel dexterity

dexterity: A Dexterity-based product



This template expects a project name with 1 dot in it (a 'basic
namespace', like 'foo.bar').

Enter project name: example.project

If at any point, you need additional help for a question, you can enter
'?' and press RETURN.

Expert Mode? (What question mode would you like? (easy/expert/all)?) ['easy']:

Version (Version number for project) ['1.0']:
Description (One-line description of the project) ['Example Dexterity Product'
]:
Grok-Based? (True/False: Use grok conventions to simplify coding?) [True]:
Use relations? (True/False: include support for relations?) [False]:
Creating directory ./example.project
Replace 0 bytes with 148 bytes (0/0 lines changed; 4 lines added)
Replace 912 bytes with 1375 bytes (1/32 lines changed; 12 lines added)
```

FIGURA 2.1: Creación de un producto usando ZopeSkel

```
qb@qbs:~/Downloads/PlonePrueba/zinstance/src/example.project$ ../../bin/paster addcontent dexterity_content
Enter contenttype_name (Content type name ) ['Example Type']: Example Type
Enter contenttype_description (Content type description ) ['Description of the Example Type']: Example
Enter folderish (True/False: Content type should act as a container ) [False]:
Enter global_allow (True/False: Globally addable ) [True]:
Enter allow_discussion (True/False: Allow discussion ) [False]:
  Copying +content_class_filename+.py_tmpl to /home/qb/Downloads/PlonePrueba/zinstance/src/example.project/ex
ample/project/example_type.py
  Recursing into +content_class_filename+_templates
File 'sampleview.pt' already exists: skipped
  Copying +contenttype_classname+.txt_tmpl to /home/qb/Downloads/PlonePrueba/zinstance/src/example.project/ex
ample/project/ExampleType.txt
  Inserting from configure.zcml_insert into /home/qb/Downloads/PlonePrueba/zinstance/src/example.project/exam
ple/project/configure.zcml
```

FIGURA 2.2: Creación de un Content Type usando ZopeSkel

2.2. Dexterity

Dexterity es un *framework* de Plone para el desarrollo de *Content Types* creado como alternativa a ArcheTypes, el cual, era el *framework* preferido en versiones anteriores de Plone y ha sido sustituido paulatinamente por Dexterity.

Las razones por las cuales Dexterity ha sido la opción de preferencia de muchos desarrolladores son:

1. Permite crear *Content Types* sin tener ningún conocimiento de programación en Python[24] usando herramientas disponibles en un sitio Plone.
2. La creación de *Content Types* por medio de código Python es mas rápida y fácil, sin perder la capacidad de personalizar cualquier aspecto de estos.
3. Soporta el uso *Behaviors* que pueden ser activados o desactivados a conveniencia. Un *Behavior* puede ser un conjunto de *Fields* como, título, identificador, meta datos, entre otros.
4. Es más consistente con versiones más modernas de Zope y Plone.

5. Tiene mejor documentación que Archetypes.

6. Es compatible con tecnologías modernas que permiten desarrollar productos más rápido.

En general un producto desarrollado con Dexterity es mas rápido y eficiente que el mismo producto desarrollado con Archetypes. La popularidad de Dexterity ha sido tal que muchos productos desarrollados con ArcheTypes han sido reescritos usando Dexterity para aprovechar las ventajas que este ofrece. Se prevé que a largo plazo Dexterity sustituya completamente a Archetypes.

2.3. plone.supermodel

El producto plone.supermodel permite especificar el *Schema* de un *Content Type* haciendo uso de archivos XML[25], este método nos otorga la misma funcionalidad que si se definiera el *Schema* escribiendo código en Python. La ventaja de definirlos de esta manera es que hace más sencillo el mantenimiento y más legible la revisión de código.

En las siguientes figuras se muestra un ejemplo elemental de un *Schema* definido mediante código Python en 2.3 y el mismo usando plone.supermodel en 2.4.

```
title = schema.TextLine(
        title=_(u"Program name"),
    )
description = schema.Text(
        title=_(u"Program summary"),
    )
start = schema.Datetime(
        title=_(u"Start date"),
        required=False,
    )
end = schema.Datetime(
        title=_(u"End date"),
        required=False,
    )
details = RichText(
        title=_(u"Details"),
        description=_(u"Details about the program"),
        required=False,
    )
```

FIGURA 2.3: Ejemplo de un Schema definido con código Python

2.4. Collective

Collective es un repositorio comunitario para productos y software relacionado con Plone manejado por Plone.org y y dónde otros usuarios pueden compartir sus proyectos, para que otros usuarios puedan encontrarlos fácilmente. En él se encuentran muchos productos, en ocasiones, debido a su gran uso pasan a ser parte del Core de Plone. En vista de lo anterior y de que uno de los objetivos es que el producto de elecciones sea

```
<schema>
    <field type="zope.schema.TextLine" name="title">
        <title>Title</title>
        <description>Session title</description>
    </field>
    <field type="zope.schema.Text" name="description">
        <title>Program summary</title>
    </field>
    <field type="zope.schema.Datetime" name="start">
        <title>Start date</title>
        <required>False</required>
    </field>
    <field type="zope.schema.Datetime" name="end">
        <title>End date</title>
        <required>False</required>
    </field>
   <field type="plone.app.textfield.RichText" name="details'
        <title>Details</title>
        <description>Details about the program</description>
        <required>False</required>
    </field>
</schema>
```

FIGURA 2.4: Ejemplo de un Schema definido con plone.supermodel

utilizado por el mayo número posible de usuarios, se trata de la comunidad indicada para acercarse a compartir el producto de elecciones ya que uno de los objetivos

Los repositorios de Collective hacen uso de GitHub[17], es un servicio de almacenamiento de proyectos de desarrollo de software que hace uso del control de versiones Git[26].

Mediante el uso de GitHub, un proyecto en Collective puede ser fácilmente descargado por otros usuarios y agregado a sus instancias de Plone, también cuenta con herramientas que facilitan la colaboración, en diferentes esquemas, cuando se cuenta con el permiso del dueño del proyecto para hacer cambios directos y en los que no se cuenta con éste, en el segundo caso es posible clonar el proyecto para realizar cambios y solicitar la revisión del bueno para su consideración y posiblemente la fusión de la propuesta al proyecto original.

El colaborador Hector Velarde recomendó hacer uso de este repositorio para el proyecto basándose en las razones expuestas y es la razón del nombre del producto: collective.elections.

2.5. Grok

Cuando se escribe un componente en Python es necesario registrarlo en un archivo de configuración ZMCL. Grok[27] es una herramienta de Zope diseñada para facilitar el desarrollo que realiza el registro usando código Python.

El registro es necesario para poder realizar la conexión entre los componentes de un producto, Grok permite crear las conexiones de cada uno de los componentes en el mismo archivo donde se define, de esta forma es mas fácil de entender las conexiones de cada componente y reduce la necesidad de tener registros en diferentes archivos.

Las figura 2.5 presentan el código de un adaptador y su registro dentro de un archivo ZCML, en la figura 2.6 tenemos el mismo ejemplo usando Grok para registrarlo extendiendo a la clase `grok.Adapter`.

```python
from zope.interface import implements
from zope.component import adapts

from my.package.interfaces import IMyType
from zope.size.interfaces import ISized

class MyTypeSized(object):
    implements(ISized)
    adapts(IMyType)

    def __init__(self, context):
        self.context = context

    def sizeForSorting(self):
        return 'bytes', 0

    def sizeForDisplay(self):
        return u'nada'
```

```
<configure xmlns="http://namespaces.zope.org/zope
 i18n_domain="my.package">

    <adapter
        for=".interfaces.IMyType"
        provides="zope.size.interfaces.ISized"
        factory=".size.MytypeSized"
        />

</configure>
```

FIGURA 2.5: Ejemplo de un adaptador y su registro usando ZCML

```python
from five import grok

from my.package.interfaces import IMyType
from zope.size.interfaces import ISized

class MyTypeSized(grok.Adapter):
    grok.provides(ISized)
    grok.context(IMyType)

    def sizeForSorting(self):
        return 'bytes', 0

    def sizeForDisplay(self):
        return u'nada'
```

FIGURA 2.6: Ejemplo de un adaptador y su registro usando Grok

five.grok es la implementación de Grok que se usa para Plone. Nos permite hacer el registro de BrowserViews, Adapters, Viewlets, Event Suscribres, con lo que se obtiene un código mas fácil de mantener.

2.6. Membrane

Por defecto, los usuarios en Plone sólo tienen *Fields* para guardar la siguiente información: nombre, correo, pagina personal, biografía y foto. En muchas ocasiones es necesario ser capaz de almacenar más información acerca del usuario como género y ocupación. El problema es que los usuarios en Plone no están definidos como *Content Type*, son un objeto dentro de Zope, esto impide que se pueda hacer simplemente una extensión al usuario para agregar los nuevos *Fields*.

Membrane es una herramienta que permite el manejo de los usuarios como *Content Type* es un producto sencillo de usar y su implementación requiere poco código. Por sí solo no representa una implementación de usuario como *Content Type*, pero permite hacer

la conexión entre un Cotent Type para el manejo de usuarios y los usuarios por defecto de Plone.

Faculty Staff Directory hace uso de Membrane para extender a los usuarios con la información necesaria para su manejo dentro de una institución académica.

En la figura 2.7 se muestra un ejemplo de un *Content Type* llamado Employee que se registra con membrane y marca con las interfaces:

- IPropertiesProvider: Permite la conexión entre el *Content Type* y las propiedades de los usuarios de Plone.
- IUserAuthProvider: Para poder realizar iniciar sesión en el sitio.
- IGroupsProvider: Puede ser agregado a grupos de usuarios.
- IRolesProvider: Se le puede asignar Roles de usuarios.

```
<?xml version="1.0"?>
<object name="membrane_tool" meta_type="MembraneTool">
    <membrane-type name="Employee">
        <active-workflow-state name="active" />
    </membrane-type>
</object>
```

```
from Products.membrane.interfaces import IUserAuthProviderfrom
from Products.membrane.interfaces import IPropertiesProviderfrom
from Products.membrane.interfaces import IGroupsProviderfrom
from Products.membrane.interfaces import IGroupAwareRolesProvider
...
class Employee(ExtensibleSchemaSupport, BaseContent):
"""A borg employee.    Employees are also users.    """
    implements(IEmployeeContent,
               IUserAuthProvider,
               IPropertiesProvider,
               IGroupsProvider,
               IGroupAwareRolesProvider,
               IAttributeAnnotatable)
    ...
    ...
```

FIGURA 2.7: Ejemplo Content Type que hace uso de Membrane

2.7. Collections

Las *Collections* de Plone permiten hacer búsquedas en todo el contenido del sitio, muestran los resultados dinámicamente y ordenados, funciona como una consulta en una base de datos.

Las búsquedas se pueden hacer usando criterios como: *Content Type*, Fechas, Autor, Estado, Título, Palabras clave, Contenido de *Fields*, etc.

De esta forma se pude buscar dentro del sitio todo los contenidos que fueron publicados antes de alguna fecha específica, que tenga algún autor específico, alguna palabra dentro de cierto *Field* y contenido específico. Es posible realizar una búsqueda por tantos criterios como se desee y aparecerán todos los contenidos que cumplan con éstos en la *Collection*.

La capacidad de realizar búsquedas utilizando criterios por defecto o alternativo es especialmente útil cuando se crean nuevos *Content Types* y los deseamos listar con respecto al contenido especifico de algún *Widget* o *Field*.

En la figura 2.8 se muestra un ejemplo donde se realiza una búsqueda de todos los *Content Types* que sean páginas, por tratarse de un sitio Plone recién instalado solo muestra la página principal en los resultados ya que es la única existente.

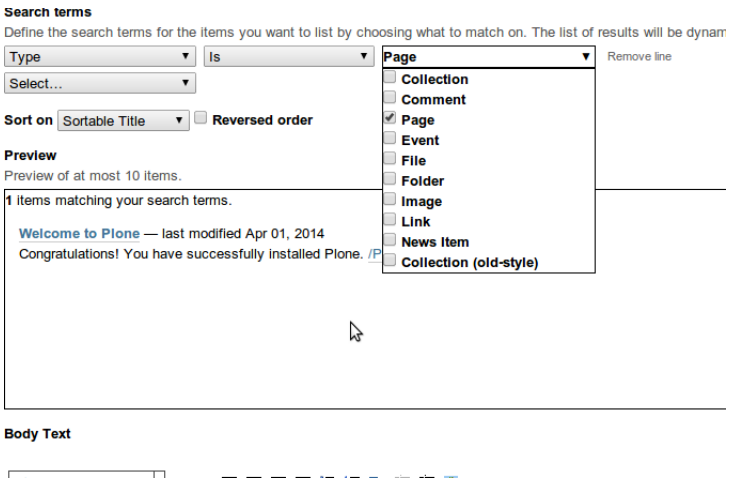

FIGURA 2.8: Ejemplo de búsqueda usando Collection

2.8. Roles y Permisos

Los Roles en Plone son conjuntos de permisos que tienen los usuarios dentro del sitio, los Roles junto con los grupos y *Workflow* forman parte del robusto modelo de seguridad de Plone.

Por defecto Plone contiene una serie de Roles con permisos específicos:

- Member: la mayoría de usuarios del sitio van a pertenecer a este rol y únicamente pueden ver los contenidos que son públicos.
- Reader: este tipo de usuarios pueden ver el contenido que esta en estado privado pero no pueden realizar ningún cambio.
- Contributor: puede hacer lo mismo que los usuarios con el rol de Member, pero puede agregar contenidos propios y modificarlos, aunque no es capaz de modificar contenidos creados por otros usuarios.
- Owner: este rol es automáticamente asignado cuando un usuario agrega un contenido al sitio, le es asignado este rol únicamente para ese contenido, lo que le permite hacer cualquier cosa con él sin importar el estado en el que se encuentre.

- Editor: los usuarios con este rol pueden cambiar las propiedades de los contenidos y mandar el contenido para su publicación.
- Reviewer: este rol permite revisar contenido que fue asignado para su publicación y cambiar su estado a publico o retractarlo.
- Manager: los usuarios con este rol pueden realizar cualquier acción. Son los administradores del sitio.

La forma más sencilla de administrar permisos es crear grupos de usuarios y asignar los roles a los grupos, todos los usuarios que pertenecen al grupo tendrán los roles asignados a el, en la figura 2.9 e muestra un ejemplos de asignación de roles por grupos.

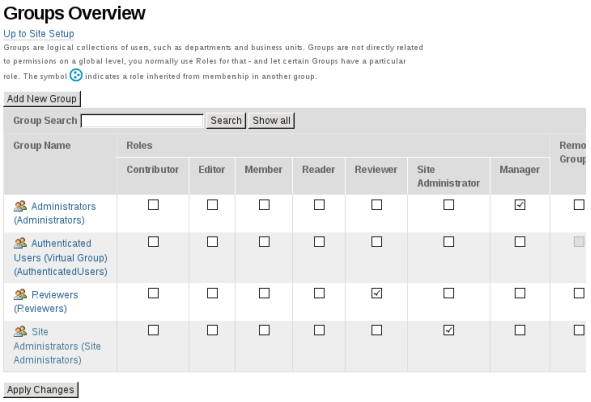

FIGURA 2.9: Ejemplo de asignación de roles por grupos

Los permisos puede ser asignados localmente para cada contenido, esto resulta de utilidad cuando se desea que un usuario pueda modificar un contenido especifico, en ese caso, es posible asignarle un permiso de edición únicamente para el ese contenido como se muestra en la figura 2.10.

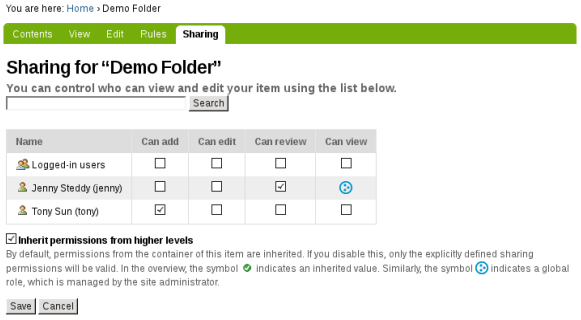

FIGURA 2.10: Ejemplo de asignación de permisos localmente

También es posible crear permisos personalizados para restringir de forma mas especifica lo que pueden ver y hacer los usuarios.

Cuando ninguno de los roles anteriores cubra las necesidades de permisos para los usuarios existe la posibilidad de crear nuevos roles y designarle el subconjunto de permisos deseados.

2.9. Workflow

Un *Workflow* es una serie de acciones o eventos que se llevan acabo para completar una tarea. Se puede ver cómo una gráfica bien definida que describe como un contenido pasa por diferentes estados a través del proceso de su publicación.

Esta herramienta es ideal para realizar todo tipo de productos debido a que un *Workflow* facilita la separación de los diferentes aspectos de un producto como son la lógica, contenido y presentación, es una de las herramientas más poderosas y complejas con las que cuenta Plone.

Con un *Workflow* de Plone se puede controlar cuáles usuarios pueden ver el contenido, modificarlo, crear, etc. Permite definir procesos de trabajo sobre un contenido, dando diferentes niveles de acceso a los usuarios en cada etapa del proceso, el nivel de acceso depende de las acciones que puede efectuar el usuario en la etapa que se encuentra el contenido.

El *Workflow* tiene 2 componente principales:

- Estados: para cada estado del contenido se pueden definir diferentes permisos para los grupos de usuarios que se relacionarán con el. En todo momento el contenido debe estar en alguno de estos estados.
- Transiciones: por medio de éstas el *Workflow* cambia de estado. Se le puede designar nombre a cada transición, condiciones a cumplir antes de poder ser activada y acciones a realizar antes o después de que se completa la transición.

El *Workflow* por defecto de Plone tienes 3 estados:

- Private: después de que el contenido es creado queda en este estado y tiene 2 transiciones:
  - Submit for publication: manda al contenido a ser revisado para su publicación.
  - Publish: si el creador tiene permisos de publicación puede hacerlo público directamente.

- Pending review: en este estado el contenido está en espera de ser revisado y publicado. Cuenta con 3 transiciones:
  - Send back: un usuario con rol de Review regresa al estado privado en caso de que algo este mal.
  - Retract: el usuario que creó el contenido regresa al estado privado para modificar el contenido.
  - Publish: Si no existe ningún problema, un usuario con rol de Review pude publicarlo.
- Published: En este estado el contenido es público. Cuenta con 2 transiciones:
  - Send back: Un usuario con rol de Review regresa al estado privado en caso de que exista alguna anomalía o error.
  - Retract: El usuario que creó el contenido regresa al estado privado para modificar el contenido.

En la figura 2.11 se muestra la gráfica definida por este *Workflow*.

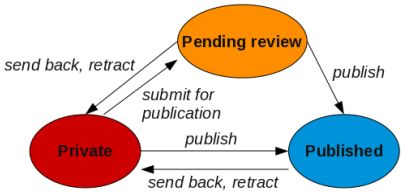

FIGURA 2.11: *Workflow* por defecto de Plone

2.10. GTK+

GTK+[28] es una herramienta para crear interfaces gráficas con C/C++, con más de una década de desarrollo, es multi-plataforma, compatible con Linux, MacOS X y Windows, entre otras características.

En particular se utiliza PyGTK[29] que emplea GTK+ y brinda la misma flexibilidad multi-plataforma pero haciendo uso de Python.

Capítulo 3

Producto ATVotaciones

ATVotaciones está desarrollado para Plone 3. Este es el segundo esfuerzo para realizar un producto para el manejador de contenidos Plone para realizar elecciones internas dentro del IMATE, se derivó de un primer producto de elecciones realizado para Plone 2 al que se le hizo una migración de versión así como un rediseño para su mejor funcionamiento.

Este producto tiene 3 componente principales: ATSelectUsers el cual es un producto que permite crear una lista de usuarios para ser usados como electores, ATVotaciones que es el producto que se encarga de la mayor parte del proceso de elección y el último componente es la herramienta de ayuda para realizar las funciones criptográficas. A continuación se analizara mas a fondo cada uno de los componentes.

3.1. ATVotaciones

En ATVotaciones se configura y controla las diferentes etapas de la elección, desde la configuración de los parámetros de la elección (fechas, tipo de seguridad, etc), hasta la entrega de resultados.

Una de las herramientas mas básicas al desarrollar un producto para Plone es el uso de *templates* para personalizar la vista que se desea mostrar al usuario, contenidos, etc. También se hace uso de pestañas para organizar el contenido, evitando sobre saturar las vistas mostrando todo el contenido. Por defecto siempre se muestra la pestaña de visualizar.

El producto hace uso de lo mencionado anteriormente, se muestra la mayor parte del contenido usando 2 pestañas:

1. Visualizar: muestra el estado actual de la elección, también se pueden llevar acabo acciones como emitir un voto cuando la elección este en la etapa de votación 3.1.

2. Proceso: esta pestaña muestra la etapa actual de la elección y las acciones que se deben realizar para continuar con el proceso de elecciones 3.2.

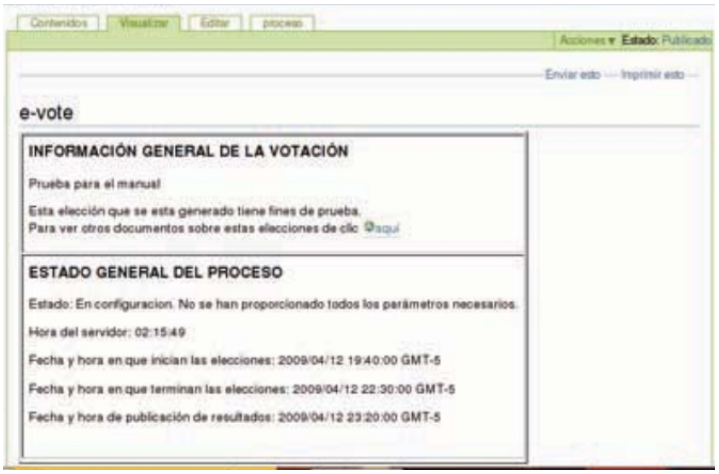

FIGURA 3.1: Pestaña de Visualizar

FIGURA 3.2: Pestaña de Proceso

Lo que un usuario puede ver en cada una de las pestañas depende del rol que tenga en las elecciones, los posibles roles son: Administrador, Comisión de vigilancia, Electores y Candidatos.

Dependiendo de la etapa en la que se encuentre, se usan diferentes *templates*[30] para desplegar la información acerca de la elección, cuando se realizan ciertas acciones también cambia el *template* usado para visualizar la elección. Éstos muestran únicamente la información requerida en cada una de las etapas y habilitan acciones necesarias para seguir avanzando en el proceso de la elección.

Un componente muy importante del producto es una máquina de estados interna que es usada para llevar el control de las diferentes etapas por las que pasa la elección, la información que se mostrará en las pestañas, los permisos que tiene cada usuario dependiendo de su rol, las acciones que pueden realizar los usuarios y los permisos con los que cuentan.

Al llevar a cabo algunas acciones en pestañas separadas, se dificulta la visualización, lo que hace poco amigable el uso del producto. La falla principal es no utilizar tecnologías disponibles en Plone y lo poco intuitivo que resulta para el usuario.

Si bien el producto resultante cumple de forma correcta con su cometido, falla al no utilizar tecnologías disponibles en Plone y su uso es poco intuitivo.

El *Workflow* usado para ATVotaciones es el que tiene por defecto Plone para todos los contenidos. Esto es de llamar la atención, como vimos en el capitulo pasado un *Workflow* es la herramienta ideal para modelar un proceso de elecciones, nos permite manejar vistas, permisos, etc. Al hacer uso de un *Workflow* eliminaríamos la necesidad de la máquina de estados creada para este producto.

Una de las principales razones por las que se decidió rehacer completamente el producto es que al sustituir la máquina de estados por un *Workflow* todo el código escrito dependiente de la máquina deja de ser útil.

Para collective.eletions se diseñará un WorkFlow personalizado, cada una de las etapas de la elección tendrá su propio estado, eliminaremos la pestaña extra para el seguimiento del proceso, en cada uno de los estados del *Workflow* se indicará la acción que se debe realizar y todos los cambios a la elección se llevarán acabo en la pestaña de edición, la pestaña de visualización únicamente se usará para desplegar información acerca de la elección y emitir votos.

3.2. ATSelectUsers

ATSelectUsers depende del producto Faculty Staff Directory (FSD) para realizar una selección de usuarios por medio de filtros, primero deja elegir qué tipo de usuarios queremos filtrar 3.3. Dependiendo del tipo que hayas seleccionado nos muestra una lista de todos los *Fields* que tienen éstos y posteriormente seleccionamos los *Fields* que deseamos usar como filtro 3.4.

Después nos deja definir filtros para los valores que deben tener los usuarios que deseamos seleccionar 3.5 y generamos la lista de usuarios.

FIGURA 3.3: Filtrado por tipo de usuario

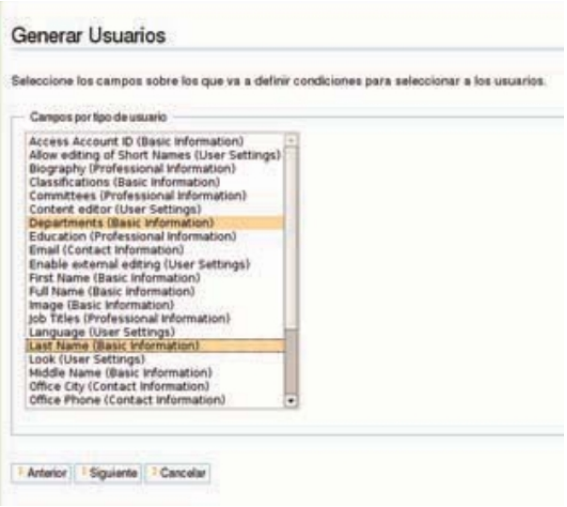

FIGURA 3.4: Filtrado por Fields de los usuarios

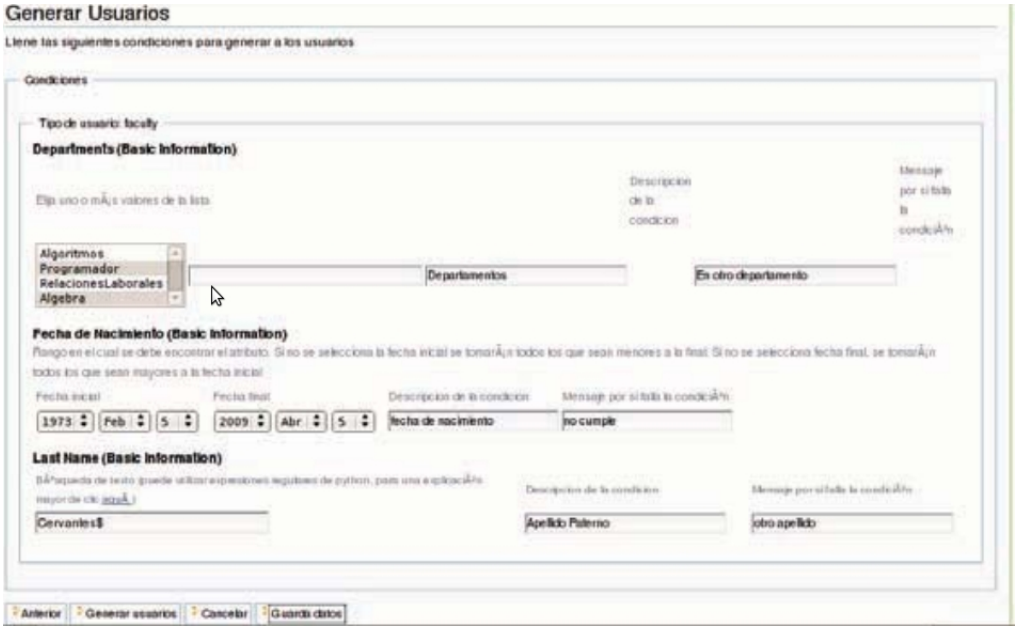

FIGURA 3.5: Filtrado por valores

El producto también tiene la opción de guardar la configuración de los filtros, cambiarlos, etc.

ATSelectUsers se diseñó pensando en las necesidades específicas del IMATE, el cual se hizo estrictamente dependiente de Faculty Staff Directory, creando una limitación para el uso del producto. Si otras personas que estén interesadas en hacer uso de ATVotaciones y no usan Faculty Staff Directory para la gestión de usuarios dentro de su sitio, no pueden usar ATSelectUsers y por lo tanto tampoco ATVotaciones.

Aunque Faculty Staff Directory es un producto muy popular dentro de la comunidad hay sitios que hacen su propia implementación de usuarios o simplemente se quedan con los usuarios que por defecto de Plone. Cabe mencionar que si el producto Faculty Staff Directory sufriera un cambio drástico en su implementación el producto ATSelectUsers tendría que ser rediseñado, dejando ATVotaciones sin poder utilizarse.

Con Plone 3 se agregaron las *Collections*, que permiten hacer una lista de contenidos de un sitio Plone que cumplan ciertas características, lo que las hace una buena opción para realizar el mismo proceso de filtrado para el cual ATSelectUsers está diseñado pero siendo *Collections* una herramienta mas poderosa.

Cuando se diseñó el producto de selección de usuarios para Plone 2 no se contaba con esta herramienta, por lo que se requería implementar un proceso de filtrado específico. Ahora que se tiene *Collections* no es necesario hacer uso de este producto e implica una dependencia innecesaria.

El inconveniente que surge al hacer uso de *Collections* para el filtrado, es que solo permite filtrar *Content Types*, usuarios dentro de un sitio Plone que por defecto no son *Content Types*, en el caso del IMATE que usa Faculty Staff Directory para los usuarios no hay problema, pues este producto convierte a los usuarios de Plone en un *Content Type* usando Membrane. Los sitios que hacen su propia implementación de usuarios usan Membrane o herramientas parecidas que convierte a los usuarios en un *Content Type*.

Para la implementación de collective.elections haremos uso de *Collections* para el filtrado de usuarios, con esto perdemos la dependencia que tenia ATSelectUsers de Faculty Staff Directory, sin perder funcionalidad, teniendo un producto de uso mas general.

3.3. Elementos Criptográficos

En ATVotaciones para llevar acabo las tareas criptográficas por parte de la comisión de vigilancia y administradores se describe en los manuales los comandos a ejecutar en una

terminal que cuente con GnuPG y Python para crear las llaves GPG, firmar archivos, cifrar archivos, etc.

Este proceso requiere que el usuario que vaya a realizar estas funciones tenga un conocimiento técnico del uso de una terminal y ejecución de programas en Python.

Como se requieren conocimientos más allá del uso de un navegador o interfaz gráfica, se quiere evitar la comisión de vigilancia recurra a personas externas, pues las integridad de las elecciones se podría ver comprometida.

Para collective.elections se desarrollará una herramienta gráfica que facilite las herramientas necesarias para llevar acabo estas tareas, esta debe ser de fácil uso, con esto trataremos de evitar que nuestro producto sea depreciado por sus elementos criptográficos.

A continuación se revisarán características que serán modificadas en el producto de elecciones.

3.4. Selección de Electores y Candidatos

Como mencionamos anteriormente ATVotaciones depende de ATSelectUsers para el proceso de de la selección de usuarios y candidatos, siendo ésta una característica que se va eliminar haciendo uso de *Collections* para este proceso, sin embargo cuando un sitio haga uso de los usuarios por defecto de Plone queremos que el producto pueda seguir siendo usado.

Por lo que tendremos que considerar las 2 opciones, cuando se usen lo usuarios por defecto y cuando los usuarios hayan sido convertidos a un *Content Type*.

3.4.1. Usuarios como Content Type

Si se tiene un producto que haga uso de Membrane o una herramienta parecida para convertir a los usuarios a un *Content Type*, haremos uso de *Collections* para realizar el filtrado de usuarios.

Las *Collection* de usuarios podrán ser usadas como electores o candidatos, éstas se podrán crear antes de iniciar el proceso de elecciones o durante el mismo, pues las *Collections* podrán ser reutilizadas en distintas elecciones.

Para poder filtrar un *Content Type* haciendo uso de *Collections* se deben indexar los *Fields* que se van a usar como criterio de búsqueda y registrar el índice en el catálogo de

Plone. Esto no presenta mayor problema, ya que el crear y registrar estos índices es una tarea sencilla, en el producto se incluirá un manual de como hacerlo, asi como ejemplos que hagan el registro e indexado.

Crear una extensión al producto de usuarios para crear los índices o agregarlos en los mismos representa un trabajo mucho menor que desarollar un producto de elecciones. De no querer hacerlo podrán hacer uso de la herramienta para Usuarios por defecto perdiendo todas las ventajas que representa el poder hacer filtrado con *Collections*.

En el caso específico de las elecciones para el IMATE se desarrollará una extensión a FSD para crear y registrar los índices de los *Fields* sobre los que se desea hacer filtrado de usuarios para las elecciones, teniendo de esta forma el mismo funcionamiento que tenia ATSelectUsers.

3.4.2. Usuarios por defecto de Plone

Si un sitio no hace uso de ningún producto para convertir a los usuarios en *Content Types* queremos que de alguna forma se pueda seguir usando el producto de elecciones.

En estos casos usaremos los usuarios por defecto de Plone, se hará un listado de todos lo usuarios registrados en el sitio, permitiendo seleccionar manualmente a los usuarios, usaremos la lista resultante de esta selección.

Esta es una solución demasiado sencilla en comparación al uso de *Collections* para filtrar usuarios, pero es mejor ofrecer una opción alterna que permita que el producto sea usado en mas sitios. Es una buena opción práctica para aquellos sitios que no tienen una implementación robusta de usuarios como Faculty Staff Directory, es seguro que tampoco necesiten un aplicación robusta para el filtrado de sus usuarios.

3.4.3. Candidatos

ATVotaciones solo permite hacer elecciones con candidatos que sean usuarios del sitio, pues esta pensado como un producto para elecciones intrainstitucionales, no se pueden hacer elecciones con ningún otro tipo de candidatos. Lo anterior restringe a los posibles usuarios para el producto, ya que no siempre que se realizan elecciones con el fin de elegir a una persona para ocupar un cargo, como elecciones para elegir servicios, comidas, presupuesto, mascotas, etc.

El nuevo producto de elecciones permitirá hacer la selección de candidatos de la misma forma en la que se realiza el proceso de electores o se podrá introducir a los candidatos

como cadenas de texto, permitiendo hacer elecciones teniendo como candidatos cualquier cosa.

Para las nuevas funcionalidades que tendrá collective.elections para la selección de electores y candidatos, tendremos que crear un *Widget* personalizado, pues los *Widgets* que vienen por defecto con Plone no permiten la funcionalidad deseada.

3.5. Archetypes

Al ser ATVotaciones un producto para Plone 3 que fue desarrollado haciendo uso del *framework* Archetypes, cuyo nombre lleva al inicio las letras AT, por convención de los productos que hacen referencia al uso de dicho *framework*

Archetypes sigue siendo usado para el desarrollo de productos en Plone, pero a partir de Plone 4 ya se ha hecho la migración de muchos productos a Dexterity. La tendencia en la comunidad de Plone es desarrollar los nuevos productos con Dexterity, por lo que desarrollaremos collective.elections con Dexterity.

Capítulo 4

Implementación

En este capitulo se describirá como se implementaron los diferentes componentes el producto collective.elections. Para el producto fue creado un *Content Type* llamado Election, este permite realizar el proceso de elección, también se creo un *Widget* personalizado.

4.1. Creación del Producto

Pare crear un producto en Plone usando Dexterity se usara la herramienta ZopeSkel.

Primero usando ZopeSkel se crea la plantilla para un producto basado en Dexterity, mediante el `comando: zopeskel dexterity` tal como se muestra en la figura 4.1. El comando muestra un menú, se ingresan los datos acerca del proyecto, el nombre (collective.elections), la versión, una descripción del proyecto y, dependiendo del caso, si se desea utilizar Grok. Una vez ingresados los datos, automáticamente se crea una carpeta con el producto y se genera un mensaje que notifica sobre los comandos disponibles con Paster para agregar contenido al producto.

La ejecución del comando da como resultado un producto que solo permite el registro y la verificación de la instalación.

4.2. Content Type

Los Conten Types permiten agregar nuevos contenidos al sitio, por defecto tiene varios *Content Type* definidos: Event, File, Image, News Item, Page etc, cada uno de estos tiene sus propias características y funcionalidades. Cuando se agrega un Event este nos pide información acerca del evento como título, descripción, locación, fecha de inicio,

```
qb@QBs:~/Plone/zinstance/src$ ../bin/zopeskel dexterity

dexterity: A Dexterity-based product



This template expects a project name with 1 dot in it (a 'basic
namespace', like 'foo.bar').

Enter project name: collective.elections

If at any point, you need additional help for a question, you can enter
'?' and press RETURN.

Expert Mode? (What question mode would you like? (easy/expert/all)?) ['easy']:
Version (Version number for project) ['1.0']:
Description (One-line description of the project) ['Example Dexterity Product']: Elections Product
Grok-Based? (True/False: Use grok conventions to simplify coding?) [True]:
Creating directory ./collective.elections
Replace 0 bytes with 153 bytes (0/0 lines changed; 4 lines added)
Replace 912 bytes with 1375 bytes (1/32 lines changed; 12 lines added)
-------------------------------------------------------------------------------
The project you just created has local commands. These can be used from within
the product.

usage: paster COMMAND

Commands:
  addcontent  Adds plone content types to your project

For more information: paster help COMMAND
```

FIGURA 4.1: Creación de un prodcuto

fecha de termino, entre otras. Cuando se agrega una pagina pide título, resumen y el contenido de la pagina.

Cada *Content Type* pide los datos necesarios para describir al contenido que ésta creando, los datos son guardados en *Fields* para posteriormente mostrarlos o hacer las operaciones requeridas con ellos, para cada dato se crea un *Field*, los *Fields* de un *Content Type* se definen dentro de un *Schema*. Para el producto collective.elections se crea un *Content Type* Election, con los *Fields* necesarios para guardar toda la información necesaria para implementar el proceso de elecciones.

Para crear el *Content Type* se emplea uno de los comandos de Paster que queda disponibles después de crear el paquete. Se utiliza el comando:
`paster addcontent dexterity\_content` como se muestra en 4.2. Igual que el primer comando de ZopeSkel este presenta un menú interactivo donde pide el nombre del *Content Type* que en este caso sera Election, una descripción y otras opciones de los *Content Types* como ser agregado globalmente, si permiten discusiones y si el *Content Type* podrá ser usado como contenedor. Después de proporcionar los datos requeridos crea los archivos necesarios para que cuando se instale el producto collective.elections se instale el *Content Type* Election en el Sitio.

Hay 3 archivos que son creados que definen al *Content Type*, type.xml dentro del directorio profiles/default que sirve para registrar los nuevos tipos dentro de portal_types, collective.election.xml dentro del directorio profiles/default/types que define las características del *Content Type* y el archivo election.py en la raiz del producto que es donde se escribe el código en Python para darle vistas, métodos, *Schema*, etc.

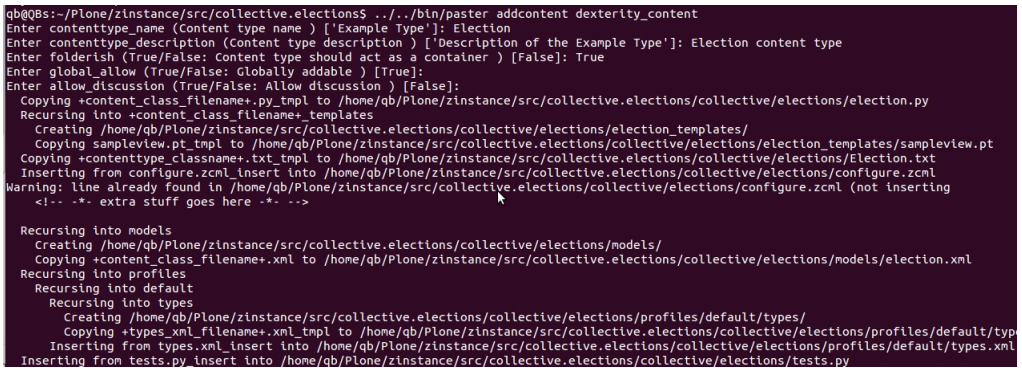

FIGURA 4.2: Creación de un Content Type

Los archivos xml son leídos al instalar el producto y de estar forma agrega el *Content Type* al sitio.

4.3. Schema

El *Schema* es la parte central de los *Content Types* en Plone, aquí se definen los *Fields* que tendrá el *Content Type* y si utiliza un *Widget* diferente al predeterminado se especifica que tipo de *Widget* va a utilizar.

4.3.1. Field

Un *Field* en Plone es definido dentro del *Schema*, es un de tipo dato que se desea almacenar, aparte del tipo, los *Fields* tienen otras propiedades como título, descripción, si el *Field* es requerido, permisos de lectura, vocabulario (si es necesario), *Widget*.

4.3.2. Widget

Un *Widget* es un pequeño programa que especifica como se visualizará un *Field*. Para un mismo tipo de *Field* se puede tener diferentes modos de visualización dependiendo del *Widget* que tenga definido, también definen en qué forma se visualizará el *Field* si estará en modo sólo lectura o si se le puede hacer modificaciones, el *Widget* es el que nos permite hacer cambios a los valores que se tienen guardados dentro del *Field*.

Un *Schema* se define en la figura 4.3. Se toma como base ATContentTypeSchema y agregándole con `atapi.Schema` *Fields* extras que se deseen definir para este *Content Type*, en este ejemplo se agrega un StringField que tiene como nombre miembroComision, lleva definido un vocabulario, permisos de escritura, un valor por defecto y un *Widget* tipo SelectionWidget. El *Widget* se debe de especificar por que un StringField

por defecto usa un StringWidget que cuando el *Field* esta en modo editable permite ingresar cualquier cadena, al cambiar a un SelectionWidget sólo se podrá seleccionar del vocabulario definido para este *Field* el valor que se desea ingresar.

```
.
.
.
Schema = schemata.ATContentTypeSchema.copy() + atapi.Schema((

miembroComision=StringField('miembroComision',
        searchable=0,
        widget=SelectionWidget(label=_(u"Responsable de la comision de vigilancia"),
           description=_(u"Miembro de la comision que tendra autorizacion para realizar
                          tareas como: firmar documentos, registrar llave publica,
                          realizar escrutinio, etc."),
           ),
        vocabulary='getTodosUsuarios',
        default="Ninguno",
        write_permission=EDIT_PROPERTIES_GRALES_PERMISSION,
       )
.
.
.
```

FIGURA 4.3: Ejemplo de definición de un Schema

Ésta es la forma estándar para definir un *Schema* en Plone, así es como están definidos en los productos de elecciones desarrollados previamente en el IMATE, actualmente existen mejores métodos para definir los *Schemas*, uno de ellos es utilizando super.model que es el método mostrado en la figura 4.4. super.model permite definir los *Schemas* haciendo uso de XML en el ejemplo se esta definiendo un *Field* de tipo Choice (ChoiceField) de nombre chief_electoral_officer así como otros parámetros como permisos de escritura, omitirlo de ciertos modos de vista, titulo, vocabulario, una descripción vacía y lo define como un *Field* requerido.

```
<model xmlns="http://namespaces.plone.org/supermodel/schema"
       xmlns:form="http://namespaces.plone.org/supermodel/form"
       xmlns:security="http://namespaces.plone.org/supermodel/security">
 <schema>
.
.
.
  <field name="chief_electoral_officer" type="zope.schema.Choice"
     security:write-permission="collective.elections.chief_electoral_officerEditable"
     form:omitted="z3c.form.interfaces.IForm:false z3c.form.interfaces.IEditForm:false">
   <title>Chief electoral officer</title>
   <description></description>
   <required>True</required>
   <vocabulary>plone.principalsource.Users</vocabulary>
  </field>
.
.
.
 </schema>
</model>
```

FIGURA 4.4: Ejemplo de definición de un Schema con supermodel

4.4. Workflow, Roles y Permisos

El *Workflow* de 3 estados que usan *Content Types* por defecto no es suficiente para cubrir las necesidades de el *Content Type* de Elecciones.

Se creo un *Workflow*, permisos y roles adecuados a las necesidades del proceso de elecciones. Estos 3 componente permiten cumplir con la seguridad requerida para la elección, teniendo un buen diseño de estos se puede asegurar que únicamente los usuarios con roles adecuados tenga acceso a las partes sensibles del procesos de elecciones.

4.4.1. Creación de Permisos

Para crear un permiso se debe registrar en el sitio, se hace usando los archivos ZCML, se crea un archivo en la raíz del producto llamado permissions.zcml y se incluye en el archivo configure.zcml 4.5 que es el archivo que es leído al instalar el productor para cargar los registros hechos con ZCML.

```
<configure
    xmlns="http://namespaces.zope.org/zope"
    xmlns:five="http://namespaces.zope.org/five"
    xmlns:genericsetup="http://namespaces.zope.org/genericsetup"
    xmlns:grok="http://namespaces.zope.org/grok"
    xmlns:i18n="http://namespaces.zope.org/i18n"
    i18n_domain="collective.elections">
 .
 .
 .
  <include file="permissions.zcml" />
 .
 .
 .

</configure>
```

FIGURA 4.5: Agregamos permissions.zcml

Dentro del archivo permissions.zcml se definen los permisos como se muestra en 4.6, se definieron dos permisos con sus respectivos id y título, uno de los permisos es el que debe de tener un usuario para poder emitir un voto, el otro es el permiso que debe de tener un usuario para poder modificar el *Field* chief_electoral_officer.

```
<configure xmlns="http://namespaces.zope.org/zope"
    i18n_domain="collective.elections">
    .
    .
    .
    <permission
        id="collective.elections.canCastVote"
        title="collective.elections: Can cast a vote"
        />

    <permission
        id="collective.elections.chief_electoral_officerEditable"
        title="collective.elections: Election chief_electoral_officer field editable"
        />
    .
    .
    .
</configure>
```

FIGURA 4.6: Definicion de permisos

Se crearon permisos de acceso para cada uno de los *Fields* para proteger el acceso a ellos y garantizas que solo los usuarios con el debido permiso podrán acceder a ellos. Cuando se definen los *Fields* en el *Schema* de el *Content Type* se puede definir permisos específicos, para poder modificar el *Field* chief_electoral_officer se debe contar con el permiso correspondiente 4.7.

```
<field name="chief_electoral_officer" type="zope.schema.Choice"
   security:write-permission="collective.elections.chief_electoral_officerEditable"
   form:omitted="z3c.form.interfaces.IForm:false z3c.form.interfaces.IEditForm:false">
 <title>Chief electoral officer</title>
 <description></description>
 <required>True</required>
 <vocabulary>plone.principalsource.Users</vocabulary>
</field>
```

FIGURA 4.7: Permiso para poder modificar un Field

4.4.2. Creación de Roles

Para el proceso de elecciones se tendrán 3 diferentes roles:

- Administrador de la elección.
- Oficial de la elección.
- Usuario o elector.

Para agregar estos roles al sitio de Plone, se definen en el producto collective.elections dentro de la carpeta profiles/default se encuentra un archivo que se llama rolemap.xml a este archivo se le deben agregar las líneas como se muestra en 4.8, cuando se instala el producto en el sitio se registran los roles y permisos que estén definidos en el archivo.

```
<rolemap>
.
.
.
 <roles>
  <role name="Election Administrator"/>
  <role name="Election Officials"/>
  <role name="General Users"/>
 </roles>
.
.
.
</rolemap>
```

FIGURA 4.8: Agregamos nuevos Roles al Sitio

4.4.3. Creación de Workflow

Para agregar el *Workflow* que implementará el protocolo de elecciones, se crea el *Workflow* y posteriormente se le asigna dicho *Workflow* al *Content Type*.

De acuerdo al protocolo definido para las elecciones se debe de tener los siguientes estados en el *Workflow*:

- Privado: el estado al que pase inmediatamente después de haber sido creada una elección, tendrá oportunidad de revisar que todo este en orden antes de pasar a revisión interna.
- Revisión Interna: el oficial de elección que la configuración sea la correcta de no ser así puede regresar la elección de nuevo al estado privado, si esta todo correcto puedo pasar la elección e Revisión Publica.
- Revisión Publica: en este estado todo los usuarios pueden revisar la elección y en caso de haber algo erróneo notificaran al administrador u oficial para que regresen al estado Privado y reconfigurar la elección. Si todo esta correcto la elección pude pasar a Selección de electores y nominados.
- Selección de Electores y Nominados: en esta estado el administrador hace la selección de electores y candidatos la de acuerdo a los lineamientos que se designaron en la configuración inicial. Cuando termina este proceso puede pasar a Revisión de Electores y Nominados.
- Revisión de Electores y Nominados: en este estado el oficial debe revisar la lista de electores y candidato, si existe alguna anomalía puede regresar de nuevo al estado de Selección, si esta esta todo correcto puede pasar la elección al estado de Publico.
- Publico: en este estado todo los usuarios pueden revisar quienes a los electores y candidatos de la elección en caso de haber algún error deben notificar al administrador para que se regrese al estado de Selección y arreglar el error, si todo esta bien las votaciones pueden empezar.
- Votaciones: en este estado todos los usuarios que son electores deben de ser capaces de emitir su voto. Al terminar las votaciones debe pasar al estado de conteo.
- Conteo: en este estado se debe llevar a cabo el conteo de todos los votos emitidos. Cuanto el proceso finalice pasará el estado de Resultados.
- Resultados: se publican los resultados de la elección.
- Cerrada: se dan por terminadas las elecciones.

De acuerdo al funcionamiento de los estados que mencionamos se necesitan definir las siguiente transiciones:

- Del estado Privado puede pasar Revisión Interna
- De Revisión interna puede avanzar a Revisión Publica o ser Regresado Privada
- De Revisión pública debe poder ser regresada a Privada
- De Revisión pública pueda avanzar a Selección de Electores
- De Selección de Electores puede pasar a Revisión de electores.
- De Revisión de electores puede ser regresada a la Selección de Electores
- De Revisión de electores puede continuar a estado Publico
- De Publico se puede regresar a Selección de Electores
- De Publico puede pasar a Votaciones.
- De Votaciones solo puede pasar a Conteo.
- De Conteo pasa a Resultados.
- De Resultados pasa Cerrada.

La figura 4.9 muestra un diagrama del *Workflow* así como de las transiciones que se definieron.

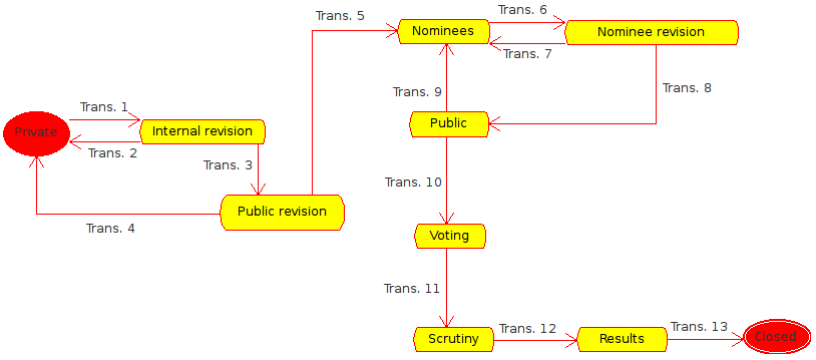

FIGURA 4.9: Diagrama del Workflow

A continuación revisaremos el proceso para crear un *Workflow* con la definición de estados y transiciones vistos anteriormente.

Se debe crear dentro del directorio profiles/default/workflow una carpeta nombrada como el identificador que tendrá el *Workflow* y dentro de esta carpeta crear un archivo definition.xml, este archivo contendrá la definición del *Workflow*.

El primer tag de este XML (`<dc-workflow>`) como se muestra en 4.10 se definen como atributos del tag, las propiedades principales del *Workflow*:

- workflow_id: identificador del *Workflow*.
- title: el nombre de tu el *Workflow*.
- description: una breve descripción.
- state_variable: la variable definida aquí guardare el estado de la elección.
- initial_state: estado inicial del *Workflow*.

```
<?xml version="1.0"?>
<dc-workflow workflow_id="election_workflow"
             title="Election workflow"
             description="Defines the workflow of an election."
             state_variable="review_state"
             initial_state="private"
             manager_bypass="False">
.
.
.
</dc-workflow>
```

FIGURA 4.10: Definición del Workflow

Dentro de `<dc-workflow>` se tendran definidos estados, transiciones y variables del *Workflow*.

La definición de un estado se muestra en la figura 4.11, se definen como atributos del tag `<state>` el identificador y el titulo del estado, dentro del tag se tendrán las transiciones que se pueden cambiar de estado usando el tag `<exit-transition>`, como atributo de éste se define el id de la transición, por último se tiene una serie de permisos que se asignara localmente dentro del tag `<permission-map>`.

```
.
.
.
<state state_id="internal_revision" title="Internal Revision">
 <exit-transition transition_id="retract_to_private"/>
 <exit-transition transition_id="submit_to_public"/>
 <permission-map>
  .
  .
  .
 </permission-map>
</state>
 .
 .
 .
```

FIGURA 4.11: Definición de un estado del *Workflow*

Para asignar permisos localmente se debe listar los permisos que van a ser manejados en el *Workflow* como se muestra en 4.12, antes de empezar a definir los estados y transiciones.

```
<?xml version="1.0"?>
<dc-workflow workflow_id="election_workflow"
             title="Election workflow"
             description="Defines the workflow of an election."
             state_variable="review_state"
             initial_state="private"
             manager_bypass="False">
 <permission>Access contents information</permission>
 <permission>Modify portal content</permission>
 <permission>Review portal content</permission>
 <permission>View</permission>
 <permission>collective.elections: Can cast a vote</permission>
 <permission>collective.elections: Election chief_electoral_officer field editable</permission>
 <permission>collective.elections: Election configuration_pdf field editable</permission>
 <permission>collective.elections: Election configuration_pdf_signature field editable</permission>
 .
 .
 .
</dc-workflow>
```

FIGURA 4.12: Permisos que manejare el Workflow

Dentro de cada estado se tiene que definir para cada permiso los roles que recibirán dicho permiso localmente. En la figura 4.13 se puede ver que para el estado internal_revision se le esta asignado los siguientes permisos:

- Acces content information: asignado a los usuarios con roles
  - Election Administrator
  - Election Officials
  - Owner
  - Site Administrator
- Modify Portal content: asignado a los usuarios con roles
  - Election Officials
- Review portal content: asignado a los usuarios con roles
  - Election Administrator
  - Election Officials
- View: asignado a los usuarios con roles
  - Election Administrator
  - Election Officials
  - Owner
  - Site Administrator

```
<state state_id="internal_revision" title="Internal Revision">
 <exit-transition transition_id="retract_to_private"/>
 <exit-transition transition_id="submit_to_public"/>
 <permission-map name="Access contents information" acquired="False">
  <permission-role>Election Administrator</permission-role>
  <permission-role>Election Officials</permission-role>
  <permission-role>Owner</permission-role>
  <permission-role>Site Administrator</permission-role>
 </permission-map>
 <permission-map name="Modify portal content" acquired="False">
  <permission-role>Election Officials</permission-role>
 </permission-map>
 <permission-map name="Review portal content" acquired="False">
  <permission-role>Election Administrator</permission-role>
  <permission-role>Election Officials</permission-role>
 </permission-map>
 <permission-map name="View" acquired="False">
  <permission-role>Election Administrator</permission-role>
  <permission-role>Election Officials</permission-role>
  <permission-role>Owner</permission-role>
  <permission-role>Site Administrator</permission-role>
 </permission-map>
  .
  .
  .
</state>
```

FIGURA 4.13: Asignación de permisos en un estado

Los permisos que pudiera tener un usuario globalmente son sobreescritos localmente para *Content Type* dejando únicamente a los roles que se definieron con los permisos asignados.

El proceso de transición se muestra en la Figura 4.14 4.14, como atributos del tag de `<transiton>` se define:

- transiton_id: define el id de la transición.
- title: titulo de transición.
- new_state: estado al que lleva la transición.
- trigger: la forma en que se activara la transición.
- before_script: acciones a ejecutar antes de realizar la transición.
- after_script: acciones a ejecutar después de realizar la transición.

Dentro del tag que define la transición se pueden usar otros 3 tipos de tags: description, action y guard.

Se usa el tag `<action>` para definir una acción que se realizara al ejecutar la transición. En 4.15 se muestra un método que se marca usando Grok para ser ejecuta cuando se active una acción, acción del *Workflow* sea `'start'`, de ser así ejecuta el código definido posteriormente de otra forma regresare sin hacer nada, la acción que logra ejecutar este

```
.
.
.
<transition transition_id="submit_to_public"
            title="Submit to public"
            new_state="public_revision"
            trigger="USER"
            before_script=""
            after_script="">
 <description>Submits the election for public revision</description>
 <action url="%(content_url)s/content_status_modify?workflow_action=submit_to_public"
         category="workflow" icon="">Submit to public</action>
 <guard>
  <guard-permission>Review portal content</guard-permission>
  <guard-expression>here/@@can-submit-to-public</guard-expression>
 </guard>
</transition>

<transition transition_id="start"
            title="Start voting"
            new_state="voting"
            trigger="USER" before_script=""
            after_script="">
 <description>Moves the election to the voting state.
It shouldn't be able to execute if the given date was not reached.</description>
 <action url="%(content_url)s/content_status_modify?workflow_action=start"
         category="workflow" icon="">Start voting</action>
 <guard>
  <guard-permission>Review portal content</guard-permission>
  <guard-expression>here/@@can-be-started</guard-expression>
 </guard>
</transition>
.
.
.
```

FIGURA 4.14: Definición de un transición del Workflow

código esta definida en la transición con id `'start'`, se puede modificar la acción del *Workflow* en los atributos de la etiqueta.

```
@grok.subscribe(IElection, IActionSucceededEvent)
def generate_random_numbers_for_candidates(obj, event):
    """
    Here we will generate the random numbers for each candidate and each
    voter
    """

    .
    .
    .

    if event.action != 'start':
        # If this is not the transition where the voting starts, then just
        # return
        return

    .
    .
    .
```

FIGURA 4.15: Definicion de una accion de IElection

Dentro del tag `<guard>` se definen condiciones bajo las cuales la transición puede ser ejecutada usando el tag `<guard-permission>`, para la transición `"submit\_to\_public"`

se debe cumplir que el usuario a ejecutar la transición tenga el permiso Review Portal Content y que el método registrado como `can-submit-to-public` sea verdadero, la definición de este tipo de método se hace dentro de un archivo llamado transition_guards.py, en la figura 4.16 podemos ver el código del método del ejemplo, es registrado usando Grok, este método sólo regresa verdadero cuando el usuario este registrado como CEO y se hayan cargado el archivo de configuración y la firma de este archivo.

```
class CanSubmitToPublic(grok.View):
    grok.context(IElection)
    grok.name("can-submit-to-public")
    grok.require("cmf.ReviewPortalContent")

    def __call__(self):
        pm = getToolByName(self.context, 'portal_membership')

        auth_member = pm.getAuthenticatedMember()
        ceo = self.context.chief_electoral_officercu

        proper_user = auth_member.getMemberId() == ceo

        if not proper_user:
            return False

        if not (getattr(self.context, 'configuration_pdf') and
                getattr(self.context, 'configuration_pdf_signature')):
            return False

        return True
```

FIGURA 4.16: Definicion de un guard

El tag `<description>` permite definir una descripción para la transición.

Ya que se tiene todo definido, se registra que se utilizará el *Workflow* en el *Content Type* Election , se hace dentro de la carpeta profiles/default en un archivo llamado workflows.xml 4.17.

```
<?xml version="1.0"?>
<object name="portal_workflow">
 <object name="election_workflow" meta_type="Workflow"/>
 <bindings>
  <type type_id="collective.elections.election">
   <bound-workflow workflow_id="election_workflow"/>
  </type>
 </bindings>
</object>
```

FIGURA 4.17: Registro del Workflow para el Content Type

Por último se revisara un ejemplo completo de uno de los estados definidos para el *Workflow* (Ver figura 4.18), en el estado Revisión de Nominados se necesita que únicamente el Oficial de la elección verifique que las listas de usuarios nominados y electores sean correctas, si así debe de firmar el archivo y subirlo al sitio junto con el archivo contiene las listas.

Se dará el permiso para modificar el contenido a usuarios con el rol de Oficial de Elección y únicamente se le da permiso para modificar los *Fields* que permite subir los archivos requeridos. Todos los otros *Fields* les serán inaccesibles ya que no cuenta con los permisos

de accesos, en este estado estos permisos no se lo otorgan a nadie con lo que asegura que no podrán ser modificados.

En todo momento cualquier usuario puede ver estado actual en el que se encuentra la elección por lo que se la los permisos de Acces Contents Information y View a todos lo usuarios del sitio.

Por último se le da los permisos de Review Portal Content a los usuarios con roles de Administrador de la elección y Oficial de la elección para que estos puedan realizar una transición para cambiar de estado a la elección ya sea pasarla al estado de revisión publica o retractarla a la Selección de nominados.

```
<state state_id="nominee_revision" title="Nominee Revision">
 <exit-transition transition_id="select_nominees"/>
 <exit-transition transition_id="send_to_public"/>
 <permission-map name="Access contents information" acquired="False">
  <permission-role>Anonymous</permission-role>
 </permission-map>
 <permission-map name="Modify portal content" acquired="False">
  <permission-role>Election Officials</permission-role>
 </permission-map>
 <permission-map name="Review portal content" acquired="False">
  <permission-role>Election Administrator</permission-role>
  <permission-role>Election Officials</permission-role>
 </permission-map>
 <permission-map name="View" acquired="False">
  <permission-role>Anonymous</permission-role>
 </permission-map>
 <permission-map name="collective.elections: Can cast a vote" acquired="False">
 </permission-map>
 <permission-map name="collective.elections: Election chief_electoral_officer field editable" acquired="False">
 </permission-map>
 <permission-map name="collective.elections: Election configuration_pdf field editable" acquired="False">
 </permission-map>
 <permission-map name="collective.elections: Election configuration_pdf_signature field editable" acquired="False">
 </permission-map>
 <permission-map name="collective.elections: Election electoral_roll field editable" acquired="False">
 </permission-map>
 <permission-map name="collective.elections: Election gpg_key_admin field editable" acquired="False">
 </permission-map>
 <permission-map name="collective.elections: Election gpg_key_ceoEditable field editable" acquired="False">
 </permission-map>
 <permission-map name="collective.elections: Election gpg_mode field editable" acquired="False">
 </permission-map>
 <permission-map name="collective.elections: Election nominations_roll field editable" acquired="False">
 </permission-map>
 <permission-map name="collective.elections: Election nominee_selection_date field editable" acquired="False">
 </permission-map>
 <permission-map name="collective.elections: Election publication_date field editable" acquired="False">
 </permission-map>
 <permission-map name="collective.elections: Election results_mode field editable" acquired="False">
 </permission-map>
 <permission-map name="collective.elections: Election rolls_pdf field editable" acquired="False">
  <permission-role>Election Officials</permission-role>
 </permission-map>
 <permission-map name="collective.elections: Election rolls_pdf_signature field editable" acquired="False">
  <permission-role>Election Officials</permission-role>
 </permission-map>
 <permission-map name="collective.elections: Election text field editable" acquired="False">
 </permission-map>
 <permission-map name="collective.elections: Election votes_count_zip field editable" acquired="False">
 </permission-map>
 <permission-map name="collective.elections: Election votes_count_zip_signature field editable" acquired="False">
 </permission-map>
 <permission-map name="collective.elections: Election voting_end_date field editable" acquired="False">
 </permission-map>
 <permission-map name="collective.elections: Election voting_start_date field editable" acquired="False">
 </permission-map>
</state>
```

FIGURA 4.18: Definición completa de un estado

Haciendo uso de un *Workflow*, roles y permisos se definió la seguridad al acceso a cada uno de los *Fields* del *Content Type* de forma que únicamente los usuarios con los roles adecuados puedan accesarlos durante el estado de la elección que así lo requiera y en ningún otro momento, manteniendo la integridad de la elección dentro del *Content Type*.

4.5. Views

Hasta ahora se tiene lo necesario pare implementar el proceso de elecciones de una manera segura, pero al visualizar el *Content Type*, se muestra la visualización que por defecto esta definida para cada uno de los *Widgets* asignados a los *Field* definidos en el *Schema*.

La visualización por defecto no es la deseada, simplemente muestra los *Fields* y su contenido en cada estado del Worflow, para modificar esto se hara uso *Views* personalizadas para el *Content Type*.

Para crear un vista personalizada se escribe una clase en Python que herede de la clase plone.directives.dexterity.Displayform, se registra haciendo uso de Grok para que sea una vista de IElection y se define que para acceder a ella se tenga el permiso View, dentro de la clase se pueden definir métodos para que la vista pueda interactuar con el *Content Type*, en la figura la figura 4.19 se tiene la definición de la clase antes mencionada con un método que devuelve el valor del estado en el que se encuentra el *Workflow*. Al nombrar la clase con el nombre View sobre escribe la vista por defecto que se usa para mostrar el contenido de el *Content Type*.

```
class View(dexterity.DisplayForm):
    grok.context(IElection)
    grok.require('zope2.View')

    .
    .
    .

    def get_election_state(self):
        wf_tool = getToolByName(self.context, 'portal_workflow')
        chain = wf_tool.getChainForPortalType(self.context.portal_type)
        status = wf_tool.getStatusOf(chain[0], self.context)

        state = status['review_state']

        return state
```

FIGURA 4.19: Definición de una clase de vista

Después de crear la clase para la vista se debe que crear en el directorio raíz de el Conten Type una carpeta nombrada ¡nombre del Content Type¿_templates, para el *Content Type* se debe llamar election_templates. Dentro de esta carpeta se crea un *template* con el nombre de clase de la vista que visualizara, para el ejemplo se nombrara al *template* view.pt. En la figura 4.20 se muestra un ejemplo de un *template* que lo único que visualiza es el título de la elección.

Dentro del sitio de Plone el *Content Type* se vería como en la figura 4.21 si usa el *template* del ejemplo anterior.

Para el *Content Type* se definen varias *Views* que permitirán mostrar el debido contenido de la elección en los diferentes estados dela misma, dependiendo del estado en que se encuentre la elección desplegara la vista correcta.

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
      xmlns:tal="http://xml.zope.org/namespaces/tal"
      xmlns:metal="http://xml.zope.org/namespaces/metal"
      xmlns:i18n="http://xml.zope.org/namespaces/i18n"
      lang="en"
      metal:use-macro="context/main_template/macros/master"
      i18n:domain="collective.elections">

<body>

<metal:main fill-slot="main">
    <tal:main-macro metal:define-macro="main">

        <h1 class="documentFirstHeading" tal:content="context/title" />

    </tal:main-macro>
</metal:main>

</body>
</html>
```

FIGURA 4.20: Definición de un template

FIGURA 4.21: Vista del Content Type

Las *Views* también permiten cierta interacción entre los usuarios y el *Content Type*, haciendo uso de estas se emiten los votos, el uso de *Views* no compromete la seguridad de la elección, las *Views* respetan en todo momento los permisos designados por el *Workflow*.

4.6. Creación de Widget personalizado

Para las diferentes opciones que se permiten al seleccionar a los usuarios que serán electores y candidatos de la elección se necesito que crear un *Widget* personalizado.

Los desarrolladores más experimentados recomienda que para crear un *Widget* personalizado se tome como base un *Widget* de Plone que tenga un comportamiento similar al que se requiere, el *Widget* requiere que se pueda hacer 2 tipos de seleccione: seleccionar un *Collection* o seleccionar a mano los usuarios, por lo anterior usamos como base a SelectionWidget.

Se crea en el directorio raíz del producto un archivo llamado electionswidget.py donde se define al *Widget*. Primero se crea la interfaz del *Widget* 4.22, extiende a la interfaz

ISequenceWidget como lo hace la interfaz de SelectWidget, ISequenceWidget es la definición mas general de *Widget* que hacen multiseleccion por esto SelectionWidget y el *Widget* que se creara la extienden.

```
class IElectionsWidget(interfaces.ISequenceWidget):
    """Text lines widget."""
```

FIGURA 4.22: Interfaz del Widget

Después se crea la clase 4.23, extiende a las clases `widget.HTMLSelectWidget` y `SequenceWidget`, en la primera se defines los atributos para una selección múltiple *Widget* e implementados la interfaz antes definida, en la clase se definen los siguientes atributos:

- klass: define el nombre.
- size: la cantidad mínima de valores a mostrar para la selección.
- multiple: define que la selección siempre sera múltiple.
- items: una lista de todos los elementos a listar.
- selectedItems: todo los elementos seleccionados de la lista.
- notselectedItems: los elementos no seleccionados.

```
class ElectionsWidget(widget.HTMLSelectWidget, SequenceWidget):
    """ Widget for elections
    """
    zope.interface.implementsOnly(IElectionsWidget)
    klass = u'election-widget'
    size = 5
    multiple = u'multiple'
    items = ()
    selectedItems = ()
```

FIGURA 4.23: Clase del Widget

Dentro de la clase también se definirán todos lo métodos necesarios para la gestión la de los elementos seleccionados, un método importante a definir es update este se ejecutara cuando se guardan cambios del *Widget*.

En el archivo se define la factory del *Widget* 4.24, las factorys son lo que permiten crear *Widget*s de este tipo por lo que es muy importante.

```
@zope.interface.implementer(interfaces.IFieldWidget)
def ElectionsFieldWidget(field, request):
    """ IFieldWidget factory for ElectionsWidget.
    """
    return FieldWidget(field, ElectionsWidget(request))
```

FIGURA 4.24: Factory del Widget

Se crean 2 *templates* electionswidget_display.pt y electionswidget_input.p, en estos *templates* se define como se ve el *Widget* en modo view y edit respectivamente, en 4.25 se

muestra la definición del modo view que simplemente lista los usuarios seleccionados. Se realiza el registro de los *templates* haciendo uso de ZCML en el archivo configure.zcml como en la figura 4.26.

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:tal="http://xml.zope.org/namespaces/tal"
      tal:omit-tag="">
<span id="" class=""
      tal:attributes="id view/id;
                      class view/klass;
                      style view/style;
                      title view/title;
                      lang view/lang;
                      onclick view/onclick;
                      ondblclick view/ondblclick;
                      onmousedown view/onmousedown;
                      onmouseup view/onmouseup;
                      onmouseover view/onmouseover;
                      onmousemove view/onmousemove;
                      onmouseout view/onmouseout;
                      onkeypress view/onkeypress;
                      onkeydown view/onkeydown;
                      onkeyup view/onkeyup"><tal:block
    repeat="value view/listUsersSelected"
    ><span class="selected-option"
           tal:content="value"
    /><tal:block condition="not:repeat/value/end">, </tal:block
  ></tal:block
></span>
</html>
```

FIGURA 4.25: Contenido de electionswidget_display.pt

```
<z3c:widgetTemplate
    mode="input"
    widget="collective.elections.electionswidget.IElectionsWidget"
    layer="z3c.form.interfaces.IFormLayer"
    template="electionswidget_input.pt"
    />

<z3c:widgetTemplate
    mode="display"
    widget="collective.elections.electionswidget.IElectionsWidget"
    layer="z3c.form.interfaces.IFormLayer"
    template="electionswidget_display.pt"
    />
```

FIGURA 4.26: Registro de las vistas del Widget

Con los pasos anteriores se crea el *Widget* y puede ser usado en la definición de esquemas.

4.7. Integración con Infomatem y otros sitios

Con todo lo anterior se tiene definido el *Content Type*, solo falta por revisar como se puede agregar la funcionalidad de filtrado usando *Collections*.

Para hacer el uso de *Collections* para filtrar un *Content Type* específico y alguno de sus *Fields* se tiene que hacer un par de cosas. Para ejemplificar este proceso se creó un producto con un *Content Type* sencillo para el manejo de usuarios llamado ElectionPerson, los indexers son creados en otro producto.

Primero Indexar el *Field* sobre el que se desea definir un criterio. En el ejemplo de la figura 4.27 se está creando un método para el *Content Type* ElectionPerson dentro de un archivo llamado indexers.py haciendo uso del marcador `@indexer(IElectionPerson)`, el método `electionperson\_firstname` definido dentro del archivo, se usara para indexar el *Field* `firstName` del *Content Type*.

```python
from example.electionperson.interfaces import IElectionPerson
from Products.CMFCore.utils import getToolByName
from plone.indexer import indexer


@indexer(IElectionPerson)
def electionperson_firstName(object, **kw):
    catalog = getToolByName(object, 'portal_catalog')
    return object.firstName
```

FIGURA 4.27: Ejemplo de creación un método de indexado

En la figura 4.28 se registran adaptadores haciendo uso de ZCML en el archivo configure.zcml, indicamos que la factory de dichos adaptadores serán los métodos creados en el archivo indexers.py.

```
<adapter name="electionperson_firstName" factory=".indexers.electionperson_firstName" />
<adapter name="electionperson_lastName" factory=".indexers.electionperson_lastName" />
<adapter name="electionperson_gender" factory=".indexers.electionperson_gender" />
```

FIGURA 4.28: Ejemplo de Registro un método de indexado

Se hace el registro del índice que se ha creado en el catálogo Plone como se muestra en 4.29, dentro del archivo catalog.xml, que debemos crear dentro del directorio profiles/-default.

```
<index name="electionperson_firstName" meta_type="FieldIndex">
 <indexed_attr value="electionperson_firstName"/>
</index>

<index name="electionperson_lastName" meta_type="FieldIndex">
 <indexed_attr value="electionperson_lastName"/>
</index>
<index name="electionperson_gender" meta_type="FieldIndex">
 <indexed_attr value="electionperson_gender"/>
</index>
```

FIGURA 4.29: Ejemplo de Registro del índice en el catalogo de plone

Por último se crea un criterio de búsqueda específico para el índice que se definió con el nombre de `ElectionPerson FirstName`, de esta manera se tendrá el criterio dentro del menú de *Collections*. El registro del criterio se pone dentro del archivo registry.xml dentro de la carpeta profiles/default 4.30.

De esta forma se pueden crear todos los criterios de búsqueda de que se necesiten usar para filtrar el *Content Type*.

La integración con Infomatem se realizó de la misma manera, se creó un producto que extiende al Faculty Staf Directoy usado por el Infomatem y se crearon los índices para

```
<records interface="plone.app.querystring.interfaces.IQueryField"
         prefix="plone.app.querystring.field.electionperson_firstName">
  <value key="title">ElectionPerson FirstName</value>
  <value key="description">First Name</value>
  <value key="enabled">True</value>
  <value key="sortable">False</value>
  <value key="operations">
      <element>plone.app.querystring.operation.string.is</element>
  </value>
  <value key="group">Election Filters</value>
</records>
```

FIGURA 4.30: Ejemplo de Registro del criterio de búsqueda para la Collection

todos los *Fields* que se desean utilizar, se necesita instalar el producto creado para que se registren los índices y poder ser utilizados.

4.8. Herramienta de GPG para elemento criptográficos

Se creo un Script de Python el cual para poder ser ejecuta correctamente depende de 3 módulos:

1. gtk: Tiene las funciones para poder crear una interfaz gráfica.
2. gnupg: Nos da las funciones criptográficas necesarias.
3. zipfile: Contiene las funciones para usar archivos Zip.

Los módulos son fáciles de instalar en todas las plataformas gtk es parte pyGTK, gnupg es un proyecto de software libre disponible multiplataforma y zipfile que viene incluido con las distribuciones de Python.

Se creó una clase CriptoTool que extiende a `Gtk.Window`, donde se definió la interfaz gráfica, la interfaz tendrá barra menús de donde podremos elegir la acción a realizar 4.31.

Si se elige alguna de las acciones por ejemplo Delete Key, se muestra la ventana donde se crean los *Fields* necesarios con la información qué se necesita para ejecutar el comando de la herramienta GPG, se crea un botón para que una vez que se tenga llena la información ejecutar la acción 4.32.

Cuando el botón es apretado se ejecuta la función de la herramienta GPG, cuando se finaliza muestra una ventana que confirma la acción 4.33.

De forma análoga fueron creadas ventanas para todas las funciones criptográficas necesarias.

```
<ui>
  <menubar name='MenuBar'>
    <menu action='KeyMenu'>
      <menuitem action='CreateKey' />
      <menuitem action='DeleteKey' />
      <menuitem action="ListKeys" />
      <menuitem action='ExporPK' />
      <separator />
      <menuitem action='FileQuit' />
    </menu>
    <menu action='OptionMenu'>
      <menuitem action='SignFile' />
      <menuitem action='EncryptFile' />
      <menuitem action='DecryptFile' />
      <menuitem action='DecryptVote' />
      <menuitem action='VerifySign' />
    </menu>
  </menubar>
</ui>
"""

class CriptoTool(Gtk.Window):

    def __init__(self):
        Gtk.Window.__init__(self, title="Menu Example")

        action_group = Gtk.ActionGroup("my_actions")


        self.add_key_menu_actions(action_group)
        self.add_option_menu_actions(action_group)

        uimanager = self.create_ui_manager()
        uimanager.insert_action_group(action_group)

        menubar = uimanager.get_widget("/MenuBar")

        self.box = Gtk.Box(orientation=Gtk.Orientation.VERTICAL)
        self.box.pack_start(menubar, False, False, 0)


        self.add(self.box)

    ...
    ...
```

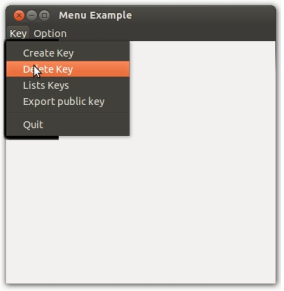

FIGURA 4.31: Código y Ventana principal

```
def on_deletekey(self, widget):

    self.remove_all()
    grid = Gtk.Grid()

    keys_combo = Gtk.ComboBoxText()
    keys_combo.set_entry_text_column(0)
    keys = gpg.list_keys()
    dic_keys = []
    for key in keys:
        keys_combo.append_text(key['uids'][0])
    grid.attach(keys_combo, 0,0,8,1)

    button = Gtk.Button("Delete")
    button.connect("clicked", self.on_delete_clicked, keys_combo)

    grid.attach(button, 0,1,3,1)

    self.box.add(grid)
    window.show_all()
```

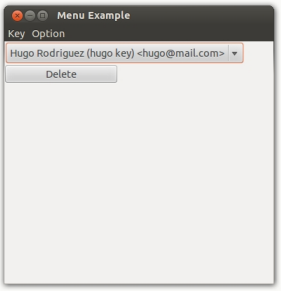

FIGURA 4.32: Código y ventana de elimina llave

```
def on_delete_clicked(self, button, combo):
    if combo.get_active_text() is None:
        self.error_dialog("Input Error ", "Selection cant be empty")
        return
    keys = gpg.list_keys()
    keytodel = None
    for key in keys:
        if key['uids'][0].rfind(combo.get_active_text()) >= 0:
            keytodel = key

    gpg.delete_keys(keytodel['fingerprint'], True)
    gpg.delete_keys(keytodel['fingerprint'])

    self.succes_dialog("Delete key for "+combo.get_active_text())

    self.on_deletekey(button)
```

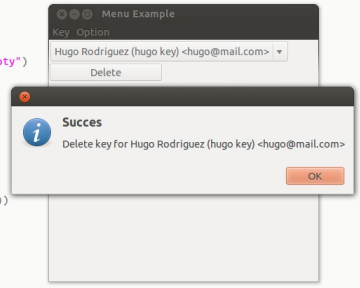

FIGURA 4.33: Código y mensaje cuando se elimina una llave

Capítulo 5

Conclusiones

Por ultimo hablaremos brevemente las conclusiones y resultados obtenido en este trabajo, así como trabajo a futuro para tener un producto de elecciones en linea mas robusto.

5.1. Conclusiones y Resultados

El resultado de esta tesis cumple los mismos requerimientos de seguridad que las implementaciones anteriores, para tener un producto más amigable, el cual tiene la posibilidad de ser adoptado por una mayor cantidad de usuarios, pues el proceso de elecciones se simplificó sin perder ninguna garantía de seguridad.

Por otro lado, el funcionamiento de collective.elections no depende de ningún otro producto que no sea instalado junto con Plone, por lo que puede ser usado en cualquier sitio Plone. Además permite el uso de *Collections* que es la mejor herramienta que se puede usar para el filtrado de contenido.

En cuanto al flujo de trabajo y esquema criptográfico, el *Workflow* que fue diseñado específicamente para las elecciones nos permite garantizar la seguridad de las elección dentro del sitio. Mientras que el esquema criptográfico se mantuvo sencillo para que la complejidad del mismo no presentara un obstáculo para su difusión.

El producto de elecciones collective.election desarrollado en este trabajo está en el repositorio:

https://github.com/collective/collective.elections

En dicho repositorio se encuentra también un tutorial del uso del producto, este tutorial está escrito en inglés para que se accesible a una mayor cantidad de usuarios de la comunidad de Plone.

En el repositorio:

https://github.com/HRodriguez/example.electionperson

Se encuentra un producto que para el manejo de usuarios con solo un par de campos extras y en:

https://github.com/HRodriguez/example.electionpersonindex

Esta la extensión del mismo producto que permite el uso de lo usuarios definidos por el producto en collective.electios. Estos 2 productos fueron creados para probar y ejemplificar el uso de usuarios definidos como *Content Types*.

Con los 3 productos anteriores se pueden realizar pruebas del producto de elecciones con las diferentes opciones de selección de electores.

Con este trabajo tenemos un producto de elecciones para Plone que es de utilidad para el IMATE y de un uso más general para que puede ser usado por otro usuarios de Plone. Así como un pequeño manual de desarrollo de un producto para Plone usando Dexterity y otras tecnologías antes mencionadas.

5.2. Trabajo a futuro

Para tener un producto con mejores garantías de seguridad, necesitamos cambiar el actual esquema criptográfico y sustituirlo por el presentado en Clapp(2011)[31], el cual está diseñado para llevar acabo elecciones en línea. Es necesario modificar collective.elections a través de la adición de estados al *Workflow*, además de sustituir el código en donde se usa GPG para encriptar por las funciones de la biblioteca que fueron desarrolladas en Clapp(2011)[31] y por último crear otro herramienta para las funciones criptográficas.

Con la experiencia obtenida al desarrollar esta tesis, la complejidad para realizar el proceso anterior sera mucho menor.

Bibliografía

[1] Iván Christian Cervantes Corona. Migracrón y nuevas caracteŕisticas del sistema de votación electrónica del instituto de matematica en la unam, tesis de maestría, universidad nacional autónoma de méxico. 2009.

[2] Sitio del instituto de matematicas de la unam. URL http://www.matem.unam.mx/.

[3] Sitio de plone. URL http://www.plone.org.

[4] Alexander Zapata Lenis. Implemetancion de un sistema de votación electrónica como un producto sobre la plataforma plone, tesis de maestría, universidad nacional autónoma de méxico. 2008.

[5] Ronald L. Rivest. Electronic voting technology workshop, boston, ma. 2007.

[6] Instituto nacional electoral. URL http://www.ine.mx/.

[7] California institute of technology and massachusetts institute of technology. voting techlogy project. 2005. URL http://www.vote.caltech.edu/.

[8] Douglas W. Jones. On optical mark-sense scanning. 6000:175–190, 2010. doi: 10.1007/978-3-642-12980-3_10. URL http://dx.doi.org/10.1007/978-3-642-12980-3_10.

[9] Dre voting machine. URL http://en.wikipedia.org/wiki/DRE_voting_machine.

[10] Tribunal superior eleitoral. URL http://www.tse.jus.br/.

[11] Voting by-mail and electronic transmission, state of alaska division of elections. URL http://www.elections.alaska.gov/vi_bymail_byfax.php.

[12] Vabariigi valimiskomisjon (comisión electoral de estonia). URL http://www.vvk.ee/.

[13] Helios voting). URL https://vote.heliosvoting.org/.

[14] Willem-Jan Hengeveld Rop Gonggrijp. Nedap/groenendaal es3b voting computer a security analysis. 2006.

[15] Matt Bishop David Wagner, David Jefferson. Security analysis of the diebold accubasic interpreter. 2006.

[16] Independent report on e-voting in estonia. URL https://estoniaevoting.org/.

[17] JosephR. Kiniry, AlanE. Morkan, Dermot Cochran, Fintan Fairmichael, Patrice Chalin, Martijn Oostdijk, and Engelbert Hubbers. The KOA Remote Voting System: A Summary of Work to Date. 4661:244–262, 2007. doi: 10.1007/978-3-540-75336-0_16. URL http://dx.doi.org/10.1007/978-3-540-75336-0_16.

[18] Plone archetypes. URL http://plone.org/products/archetypes.

[19] Faculty/staff directory. URL http://plone.org/products/faculty-staff-directory.

[20] Plone foundation. URL https://http://plone.org/foundation/.

[21] Zopeskel. URL https://github.com/collective/zopeskel.

[22] Zope. URL http://www.zope.org/.

[23] Dexterity (plone.app.dexterity). Dexterity (plone.app.dexterity). URL http://plone.org/products/dexterity.

[24] Python. URL https://www.python.org/.

[25] Xml. URL http://www.w3.org/TR/REC-xml/.

[26] Git. URL http://git-scm.com/.

[27] Grok. URL http://grok.zope.org/.

[28] Gtk+. URL http://www.gtk.org/.

[29] Pygtk. URL http://www.pygtk.org/.

[30] Tal page templates. URL http://docs.plone.org/adapt-and-extend/theming/templates_css/template_basics.html.

[31] Lázaro Clapp Jimenéz. Plonevotecryptolib: Una biblioteca critográfica para la implementación de elecciones en línea secretas y verificables por electores, tesis de licenciatura, universidad nacional autónoma de méxico. 2011.