Universidad de las Ciencias Informáticas

Facultad 3

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Título: Sistema de Votación Electrónica para la Universidad de las Ciencias Informáticas

Autores: Yoandra Reyes Agüero

Lino A. Martell Guevara

Tutores: Ing. Yelena Hernández Estrada

Ing. José Miguel Cazorla Santana

Ciudad de La Habana, 2012. "Año 54 de la Revolución"

Declaración de autoría

Declaramos ser autores del presente trabajo y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales del mismo, con carácter exclusivo.

Para que así conste firmamos el presente a los ____ días del mes de _____ del año 2012.

_____ _____

Yoandra Reyes Agüero.

(Autor)

Lino Alejandro Martell Guevara.

(Autor)

_____ _____

Ing. Yelena Hernández Estrada.

(Tutor)

Ing. José Miguel Cazorla Santana.

(Tutor)

Dedicatoria y agradecimientos

De Yoandra:

Ante todo le agradezco a mis padres por haberme dado la vida, en especial a mi mamá por ser madre, padre y amiga, por haberme encaminado en la vida y hacer de mí una mujer de principios, ya que por ella es que estoy aquí, por seguir sus consejos, porque todo lo que hago es porque se sienta orgullosa de mí, y quien más para darle esa alegría si soy su única hija, a ti, te dedico este logro porque no sería nadie si no contara con tu apoyo. A mis abuelos por su confianza y por creer que para mí nada es imposible, por su cariño y dedicación. A mis familiares, tíos, tías, primos, primas por darme el ánimo que siempre he necesitado para seguir adelante, por apoyarme y confiar en mí, por mimarme y considerarme el orgullo de la familia. A mi novio, por ayudarme en todo, por preocuparse por mí, por ser compañero y amigo y por brindarme su amor. También les dedico este trabajo a todos mis compañeros por hacer más fácil la estancia en esta institución, porque recuerdo que antes de tenerlos como amigos me pase 1 mes llorando y con deseos de regresar a casa. A Julito, por las tantas veces que necesite de tu ayuda y me la ofreció. Al loco, por acompañarme a estudiar hasta las 4 de la mañana cuando en primer año no sabíamos ni donde estábamos parados, porque a pesar de ser la persona más pesimista del mundo me daba ánimos para estudiar. A Maide, Ana, Ariadna, Lisandra y Rances, Butin, por ser de los pocos que quedamos de ese 3108 y por mantenernos unidos, por su cariño y amistad. Esto que considero un triunfo también va dedicado a las nuevas amistades por compartir buenos y malos momentos y por su ayuda incondicional. A mis tutores por preocuparse para que este trabajo se hiciera lo mejor posible y hasta por enojarse cuando no lo teníamos en cuenta para tomar cualquier decisión. A los integrantes del tribunal porque con sus críticas y sugerencias contribuyeron a que este trabajo saliera adelante. A todos ustedes, muchas gracias.

De Lino:

A mi madre y mi abuela por ayudarme siempre y estar atentas en todo momento de lo que me pueda faltar.

A mi padre, a Mayte, a mi hermana melisa, a mi tia Midiala, por saber entenderme o aceptarme.

A los profes que una vez fueron Yasmany, Madelis, Kiosmy.

A Mónica y Alayna por ser buenas compañeras y demostrarme su amistad y afecto incondicional en todo momento.
A mis compañeros de la escuela, Isabel, Dario, Frank, Jose, Yosmany, Manuel, Raúl, Carlos, Amilcar, Ricardo, Ariel, entre otros tantos los que serian interminable si me pusiera a nombrarlos.
A Titi y Duvergel, ellos saben por qué.
A los diferentes profesores y tutores que me han guiado a través de todos estos años en la universidad. Por mencionar algunos Lourdes, Yausmary, Rodolfo, Susana.
A los tutores de la tesis José Miguel y Yelena, por brindar un soporte y dar su opinión en cada aspecto de este trabajo de diploma.
A la revolución por permitir que esta escuela exista y que yo pueda asistir a ella.

"Tecnología y política son los dos grandes modos de obrar con los que el hombre se desenvuelve a la búsqueda de modificaciones en su entorno..."

José Luis González Quiroz

Resumen

El presente trabajo de diploma se desarrolla debido a la necesidad de la existencia de un software donde se informaticen los procesos eleccionarios en la Universidad de las Ciencias Informáticas (UCI), de modo que permita optimizar recursos materiales y humanos. Por tal motivo, se determinó implementar un Sistema de Votación Electrónica que contribuya a mejorar las actividades relacionadas con las elecciones de la FEU, ya que estas se realizan actualmente de forma manual.

Se realiza un profundo estudio de los aspectos y requisitos legales que se deben tener presentes a la hora de realizar el voto electrónico, para asegurar el buen funcionamiento del sistema desde el punto de vista jurídico y se abordan diferentes sistemas de voto electrónico.

La idea que impulsó llevar a cabo esta tarea, se debe a que en el curso 2010-2011 se desarrolló un trabajo de diploma que llevó por título Sistema de Registro Electrónico Directo para las votaciones en la UCI, donde se propuso la implementación de un sistema Web que tratara este tema, pero no se materializó, no se tuvieron en cuenta los principales aspectos legales que rigen el voto electrónico, ni las condiciones que debe presentar un sistema en cuestiones de seguridad para poder desarrollar el voto electrónico.

Para la realización de este trabajo, se estudiaron las técnicas, herramientas y metodologías necesarias, con el objetivo de seleccionar las adecuadas para llevar a cabo la implementación del sistema, el cual consiste en una aplicación de escritorio.

Palabras claves

Sistema de Votación Electrónica, Procesos Eleccionarios, Voto Electrónico.

Contenido

Capítulo 1: Fundamentación Teórica 13

1.1 Introducción del capítulo. 13

1.2 Antecedentes tecnológicos. 13

1.3 Requisitos legales del sufragio. 17

1.4 Criptografía en el voto electrónico 19

1.5 Sistemas de votación electrónica. 20

1.6 Implicaciones de Carácter Informático 27

1.7 Fortalezas del voto electrónico. 27

1.8 Desventajas del voto electrónico 28

1.9 Sistemas informáticos. 28

1.10 Tecnologías, Metodologías y herramientas de desarrollo de software 30

1.11 Arquitectura del sistema. 37

1.12 Patrones de diseño. 38

1.13 Fundamentación de la selección de la tecnología, metodología y herramienta de desarrollo. 40

1.14 Conclusiones parciales. 41

Capítulo 2: Características y diseño del sistema 41

2.1 Introducción del capítulo. 41

2.2 Modelo de negocio. 42

2.3 Especificación de requisitos 42

2.4 Métricas de validación de requisitos. 45

2.5 Configuración del sistema. 47

2.6 Modelo del sistema. 48

2.7 Patrones de casos de uso utilizados. 51

2.8 Descripción de casos de uso. 52

2.9 Modelo de diseño 57

2.10 Descripción de la arquitectura del sistema. 59

2.11 Diagrama de clases del diseño 63

2.12 Conclusiones parciales. 64
Capítulo 3: Implementación y prueba....65
3.1 Introducción del capítulo. 65
3.2 Modelo de implementación. 65
3.3 Modelo de despliegue. 66
3.4 Medidas de seguridad del sistema. 67
3.5 Métricas de software. 68
3.6 Modelo de pruebas. 74
3.7 Conclusiones parciales. 81
Conclusiones 82
Recomendaciones 82
Bibliografía 83

Tabla de ilustraciones

Ilustración 1 Interfaz de autenticación para cada una de los subsistemas. 22
Ilustración 2 Interfaz configuración de la Mesa Electoral. 22
Ilustración 3 Interfaz configuración Urna Electrónica. 23
Ilustración 4 Interfaz principal del subsistema Mesa Electoral. 24
Ilustración 5 Interfaz parte votación. 24
Ilustración 6 Interfaz principal del subsistema urna electoral. 25
Ilustración 7 Interfaz para adicionar un nuevo candidato. 25
Ilustración 8 Interfaz principal del subsistema consolidación. 26
Ilustración 9 Ciclo de vida de la metodología RUP. 33
Ilustración 10 Diagrama de flujo de procesos del subsistema urna electrónica. 42
Ilustración 11 Diagrama de paquetes. 47
Ilustración 12 Diagrama de casos de usos del subsistema Configuración. 49
Ilustración 13 Diagrama de casos de usos del subsistema Mesa Electoral. 50
Ilustración 14 Diagrama de casos de usos del subsistema Urna Electrónica. 50
Ilustración 15 Diagrama de casos de usos del subsistema Consolidación. 51
Ilustración 16 Patrón Fachada. 58
Ilustración 17 Patrón Singleton. 58
Ilustración 18 Sistema de archivos. 60
Ilustración 19 Estructura de los paquetes por cada capa. 60
Ilustración 20 Capa Presentación. 61

Ilustración 21 Capa Lógica de Negocio.....61
Ilustración 22 Capa Acceso a Datos. 62
Ilustración 23 Capa Común.....62
Ilustración 24 Diagrama de clases del subsistema urna electrónica. 64
Ilustración 25 Diagrama de componentes. 66
Ilustración 26 Modelo de despliegue.....67
Ilustración 27 Incidencia de los resultados de la evaluación de la métrica TOC en el atributo Responsabilidad.....69
Ilustración 28 Incidencia de los resultados de la evaluación de la métrica TOC en el atributo Complejidad.....70
Ilustración 29 Incidencia de los resultados de la evaluación de la métrica TOC en el atributo Reutilización. 70
Ilustración 30 Representación de la cantidad de relaciones de uso en cada clase. 71
Ilustración 31 Incidencia de los resultados de la evaluación de la métrica RC en el atributo Acoplamiento. 72
Ilustración 32 Incidencia de los resultados de la evaluación de la métrica RC en el atributo Complejidad de Mantenimiento.....72
Ilustración 33 . Incidencia de los resultados de la evaluación de la métrica RC en el atributo Reutilización. 73
Ilustración 34 Incidencia de los resultados de la evaluación de la métrica RC en el atributo Cantidad de Pruebas. 73
Ilustración 35 Primera iteración de prueba unidad sobre la clase Control. 76
Ilustración 36 Segunda iteración de prueba unidad sobre la clase Control. 76
Ilustración 37 Pruebas de unidad realizada a una porción de las clases del sistema. 77

Introducción

Ante los constantes retos que afronta la sociedad en su informatización, existen implicaciones jurídicas que se deben tener presentes a la hora de implementar un sistema de voto electrónico o informático. Con el debido conocimiento de estas, se podrá garantizar la calidad requerida para este tipo de software, así como el cumplimiento de los términos legales y constitucionales que protegen el principio electoral.

La tendencia de los últimos años en los procesos electorales ha sido utilizar medios electrónicos para automatizar y hacer más eficientes los diferentes procesos de una elección. Aun cuando esta automatización se ha presentado de manera gradual, el propósito final es utilizar estos medios para el registro de votantes, autenticación de los mismos, emisión del voto, y desde luego, para el escrutinio y publicación de resultados.

La mayoría de los países en el mundo ha considerado el uso del voto electrónico. De ellos, una buena parte ha realizado pruebas y algunos ya lo utilizan de forma vinculante. En varios estados de Europa se han implementado diversos esquemas con sus respectivas pruebas y en otros lugares, como Estados Unidos, el empleo del voto electrónico está ampliamente desarrollado. Asimismo, está siendo considerado en buena parte de los países de América Central y del Sur.[1]

Esta forma de votación presenta ciertas amenazas por lo que hay que realizar un buen trabajo para no permitir que se materialicen convirtiéndose en vulnerabilidades del sistema. Entre estas se encuentra el deficiente diseño que realizan algunos desarrolladores en cuanto a mecanismos criptográficos, ya que un aspecto esencial en la seguridad de los sistemas de voto electrónico es la criptografía utilizada. Un diseño inapropiado de este mecanismo podría causar un riesgo importante a la integridad de la elección. El diseño comprende el algoritmo utilizado, la longitud de las claves, así como el medio de almacenamiento de las mismas. Por otra parte se tiene que controlar el hecho de que se pueda votar más de una vez. El sistema debe prevenir que un elector vote más de una vez en la misma elección. De otro modo, existiría una discrepancia entre el número de votantes que participaron y el número de votos

[1] Tomado de http://www.monografias.com/trabajos79/voto-electronico/voto-electronico2.shtml Autor: Brian Meza Vásquez el cual tiene 11 publicaciones en internet, abogado y escribano, original de Tacna, Perú. [Consultado 20 de noviembre de 2012].

recibidos. Esta situación desde luego ocasionaría un resultado de la elección no legítimo.

Los argumentos a favor de esta nueva modalidad de voto son la reducción del número de miembros de mesa, la mayor facilidad y precisión en el conteo, la agilidad del proceso, el aumento del número de votantes por mesa, la rapidez en la obtención de los resultados y la eliminación de los votos nulos. El uso de estas máquinas elimina muchos de los errores humanos así como las oportunidades para manipular el proceso y la consolidación de resultados.

En un proceso de elección, la precisión en los resultados es un requisito esencial, y la modernización de los sistemas de votación ha permitido que esto se materialice. Tradicionalmente, el escrutinio de los votos se ha llevado a cabo de forma manual, lo que tiene como consecuencia un retraso no deseado en la publicación de resultados y sobretodo una alta probabilidad de cometer errores. Hasta la fecha, el sistema tradicional de votación en Cuba se ha encargado de asegurar el cumplimiento de los principios indispensables para la realización de las elecciones, propiciando la libertad de voto, el secreto del voto, la no modificación de la intención expresada en el voto y la ausencia de intimidación durante esta operación. Estos principios no se ven socavados por la utilización de medios electrónicos para el proceso de votación, ya que el voto electrónico debe ser tan libre, secreto, confiable y seguro como los sistemas de votación que no impliquen el uso de medios electrónicos. A pesar del avance que ha tenido Cuba en el campo de la informática, esta ha sido una de las esferas donde no se ha hecho ningún aporte en relación al voto electrónico, debido a que el país no está preparado económicamente para asumir esta tarea e implementarlo en todo el territorio.

El sistema tradicional de votación requiere gran cantidad de recursos relacionados con la impresión de boletas, el control del personal que accede a las urnas, el conteo manual de los votos, esto obstaculiza el monitoreo del progreso de las votaciones además que implica retrasos a la hora de dar a conocer los resultados. Para organizar y conducir una elección se tiene que diseñar, adquirir, almacenar y distribuir material; contratar, capacitar y desplegar personal; diseñar, imprimir, distribuir, contar y reportar votos y esto hace de las elecciones un proceso complejo y multifacético. La Universidad de las Ciencias Informáticas no está exenta de ello y los procesos que comprende el sistema eleccionario, además de engorrosos, involucran gran cantidad de personas para el manejo de información como el currículo de los candidatos seleccionados en las facultades y el listado de electores. Se ha hecho obligatoria la

distribución de varias copias de los padrones[2] electorales por diferentes áreas de la universidad para dar a conocer a los candidatos, por lo que cada vez es mayor la demanda de recursos materiales para la impresión de las boletas y la confección de las urnas. El proceso de escrutinio se convierte en horas interminables, dificultando el conocimiento de los resultados en menor tiempo, además de requerir mucho personal para llevarlo a cabo, posibilitando la introducción de errores.

Debido a la importancia del desarrollo de procesos eleccionarios y la necesidad del ahorro de recursos materiales y humanos en la UCI, se plantea como **Problema a resolver:** ¿Cómo mejorar el sistema eleccionario en la Universidad de las Ciencias Informáticas de forma que contribuya al ahorro de recursos materiales y humanos?

Para el desarrollo de la investigación se plantea como **Objeto de estudio:** Proceso de desarrollo de Software de Gestión Gubernamental. Este tiene como **Campo de acción:** Proceso de desarrollo de software en Procesos Electorales. Por lo que el **Objetivo general** de la investigación es**:** Desarrollar un sistema informático que contribuya al ahorro de recursos materiales y humanos en el Sistema Eleccionario en la Universidad de las Ciencias Informáticas. Y la **Idea a defender** es que: Si se desarrolla un sistema para informatizar las elecciones en la UCI, contribuirá al ahorro de recursos materiales y humanos.

Se tienen como **Objetivos específicos:**

- Definir el marco teórico-referencial de la investigación.
- Desarrollar la propuesta de solución.
- Validar los resultados obtenidos.

Como **Tareas de la investigación:**

- Estudio del estado del arte sobre el desarrollo de los Sistemas Electorales.
- Selección de las herramientas, tecnologías y metodologías para el desarrollo de la aplicación de escritorio.
- Definición de requisitos con el cliente.
- Definición de la arquitectura a utilizar para el desarrollo de la aplicación de escritorio.
- Realización del análisis y diseño de la aplicación.
- Realización de la implementación de la aplicación.

[2] Padrón es un término con origen en el latín *patrōnus* que permite nombrar a un listado o nómina. El concepto suele utilizarse para hacer referencia al registro en el que se encuentran inscritos los ciudadanos habilitados para participar en elecciones. Tomado de http://definicion.de/padron/.

- Realización de las pruebas de validación de la aplicación.

Como **Posibles resultados:**

- Especificación de los requisitos.
- Modelo de arquitectura del sistema.
- Diseño de casos de pruebas.
- Sistema informático que contribuya al ahorro de recursos materiales y humanos en el sistema eleccionario de la UCI.

Los métodos científicos teóricos que se emplean para darle solución a las tareas de la investigación son:

- **Histórico - lógico:** Evidenciado en el análisis de cómo han venido comportándose históricamente los sistemas de votación electrónica, sus condiciones de uso y aplicación, características, fortalezas, entre otros aspectos.
- **Analítico - sintético:** Se utiliza en el análisis y síntesis de la documentación utilizada como bibliografía para la fundamentación teórica de la investigación.
- **Modelación:** Empleada para obtener una visión del objeto que se estudia y de lo que se quiere lograr con la implementación del sistema. Se refleja a través de diagramas.

Los métodos científicos empíricos que se emplean para darle solución a las tareas de la investigación son:

- **Observación:** Utilizada para identificar algunas características en el proceso de votación en la UCI como es la forma de realización, las personas que intervienen y quiénes lo dirigen.
- **Entrevista:** Realizada a los clientes, con el fin de identificar los procesos a automatizar y los requisitos que debe cumplir el sistema.

El desarrollo de este trabajo está organizado en tres capítulos, que a continuación se describen:

Capítulo 1. Fundamentación teórica: Se realiza un estudio del estado del arte de los sistemas de votación electrónica más utilizados, con motivo de poder seleccionar el adecuado para darle solución a la problemática planteada. Se abordan los principales conceptos asociados al voto o sufragio y se exponen las tendencias, tecnologías, metodologías, herramientas de desarrollo y lenguajes de modelado con los que se puede desarrollar el sistema, determinando finalmente los adecuados para obtener un producto de calidad.

Capítulo 2. Características y diseño del sistema: Se determina la configuración del sistema a desarrollar y se especifican los requisitos funcionales y no funcionales. Se confecciona el modelo de sistema y se realiza la descripción de los casos de uso. Se define la arquitectura del sistema y se realiza el diagrama de paquetes. Además, se hace una propuesta del sistema a desarrollar.

Capítulo 3. Implementación y Prueba: Se muestra el diagrama de despliegue y el de componentes. Se definen y realizan pruebas de caja negra y caja blanca, así como las métricas de validación de software.

Capítulo 1: Fundamentación Teórica

1.1 Introducción del capítulo.

En este capítulo se presentan algunos conceptos básicos relacionados con las votaciones, así como la forma en que la tecnología ha intervenido para tratar de llevarlas a cabo de una manera más eficiente. Se exponen los requisitos indispensables que debe cumplir un sistema electoral, se analizan ciertas implicaciones de carácter informático a tener en cuenta para concebir el sistema y se selecciona el sistema de votación electrónica a implementar. Además, se realiza el estudio de las tecnologías, metodologías y herramientas de desarrollo de software con el objetivo de determinar las que serán utilizadas para la implementación de la aplicación.

1.2 Antecedentes tecnológicos.

Debido al auge que ha tomado la utilización del voto electrónico en los últimos años, puede que parezca reciente su uso, así como la intención de emplear las tecnologías de la información en los procesos eleccionarios. Al principio estas necesidades fueron satisfechas utilizando la tecnología disponible, que incluía imprentas y artículos de escritura como plumas, y luego máquinas de escribir. Sin embargo la mayoría de los procesos eran manuales, aunque sigue siendo de esta forma en muchos lugares. Las listas electorales eran escritas a mano y conservadas en libros o tarjetas. Las papeletas eran distribuidas, marcadas y contadas manualmente así como el cálculo de

los resultados, que luego eran comunicados de manera impresa o a través de tableros.[3]

En realidad, una de las primeras manifestaciones en el desarrollo de dispositivos para emitir algún tipo de votación, se remonta al siglo XIX con el registro electro – gráfico de votos, invento concebido por Thomas Alva Edison. Este dispositivo permitía votar instantáneamente presionando un par de interruptores (si o no). Desafortunadamente en aquella época se sostuvo que tal invención no era necesaria. A esta le sucedió la cabina automática de Jacob H. Myers, máquinas de palanca, que en 1892 permitieron que el voto fuese automatizado. Su funcionamiento radicaba en que cada candidato era identificado por el elector con una etiqueta, esta era asignada a una palanca y el elector después de aislarse detrás de una cortina tiraba de una palanca hacia abajo, de acuerdo con su preferencia electoral. La palanca luego de ser tirada volvía de forma automática a su posición anterior, contando el voto emitido y sin permitir que el elector votara en más de una ocasión, ya que estaba preparada para eso. En Gambia se utilizó otro sistema de votación mecánica, se colocaban canicas o pelotitas en una máquina para indicar los votos por los candidatos preferidos. La máquina calculaba el número de canicas o pelotitas asignadas a cada candidato y así determinaba el resultado de la votación.
(Flores & Téllez Valdés, 2010)
Seguidamente aparecieron los sistemas a base de tarjetas perforadas y sus primeras aplicaciones en el ámbito electoral se dieron en 1964. Después, los sistemas de escaneo óptico donde los electores poseían una boleta de papel con los nombres de los candidatos, marcaban sus preferencias con un cuadrado u óvalo para luego introducirla en la máquina, que escaneaba las marcas y las interpretaba como sufragios emitidos. En este sistema si el elector utilizaba un marcador diferente al autorizado, podía causar distorsión en la lectura que realizaba el escáner y en caso de que la boleta fuese doblada, este la tomaba como nula. Surgió en la década de los 80 el televoto, o voto por teléfono, que tuvo sus primeras aplicaciones en Canadá.

Con el desarrollo de las tecnologías de información se crearon máquinas de votación de grabación electrónica directa, ordenadores que permiten emitir el sufragio a través de botones o pantallas táctiles. El principio de funcionamiento de estas máquinas

[3] Tomado de http://www.monografias.com/trabajos79/voto-electronico/voto-electronico2.shtml Autor: Brian Meza Vásquez el cual tiene 11 publicaciones en internet, es abogado y escribano, original de Tacna, Perú. [Consultado 20 de noviembre de 2012].

reside en grabar electrónicamente los votos, bajo elementos de criptografía en dispositivos informáticos de almacenamiento. En Bélgica, por ejemplo, la información de la votación es escrita tanto en un disco duro como en una tarjeta inteligente que se le expide al elector. Después de votar, el elector coloca la tarjeta utilizada en una urna. La tarjeta inteligente puede ser utilizada como respaldo si la copia en disco duro falla o como una forma de auditar la información registrada en el disco duro.[4]

Luego aparece la televisión digital interactiva donde los canales de televisión digital operan muy parecido a internet aunque esta posibilidad es bastante limitada y además, surge después de 1985 la tecnología SMS[5] (Short Message Service) presentando como mayor desventaja el costo por el envío del mensaje. (The Electoral Knowledge Network, 1998-2012)

En Cuba no se pueden utilizar muchas de estas técnicas debido a que es un país bloqueado, que no está preparado económicamente para asumir el empleo de estas nuevas tecnologías en toda la extensión del territorio.

La mitad del siglo XIX atestiguó el inicio de la revolución tecnológica que ha continuado hasta el presente y a medida que la tecnología se desarrolla, los organismos aplican distintas innovaciones a los procesos electorales[6]. La mayoría de los países en el mundo ha considerado el uso del voto electrónico, cada uno con sus particularidades, lo que ha hecho posible la existencia de más de una definición para este término, entre ellas se encuentran:

Voto (del latín votum): Es la expresión de una preferencia ante una opción. Dicha expresión puede pronunciarse de manera pública o secreta, según el caso. El término también se utiliza para nombrar a la papeleta, boleta u otro objeto con que se expresa dicha preferencia o al parecer que se explica ante una asamblea. (Definicion.de, 2008-2012)

Voto: Es sinónimo de sufragio, sobre todo cuando está vinculado al sistema electoral que se encarga de determinar la provisión de los cargos públicos. El sufragio, en este sentido, es un derecho constitucional y político que tiene dos dimensiones: el sufragio activo donde todas las personas tienen derecho a emitir su voto para elegir

[4] Tomado de http://aceproject.org/main/espanol/et/et60.htm [Consultado 19/1/2012]. Sistemas de Votación Mecánicos y Electrónicos.

[5] Traducido al español como servicio de mensajes cortos.

[6] Tomado de: ACE Project Escrutinio de Votos Uso de Tecnología, World Book 1999, CD-ROM, IBM y Federal Electoral Commission internet site, http://www.fec.gov/elections.html.

representantes y el sufragio pasivo que establece el derecho a presentarse como candidato para representar al resto de la comunidad. (Definicion.de, 2008-2012)

Voto electrónico: Conjunto de instituciones y procedimientos plasmados en disposiciones jurídico – electorales que regulan las acciones de organización, preparación, recepción de la votación, escrutinio, cómputo y transmisión de los resultados electorales sustentadas en las tecnologías de la información. (Flores & Téllez Valdés, 2010)

Voto electrónico: Con voto electrónico no nos referimos a la utilización de medios informáticos en los procesos de votación, lo cual, hoy en día, no supone ninguna novedad ni avance, ya que en estos la totalización de los resultados se realiza normalmente de forma electrónica, si no a la introducción de dispositivos electrónicos en el momento en que el ciudadano emite su voto, es decir, a la informatización de la papeleta o a su sustitución por un sistema electrónico más completo.
(Observatorio Regional de la Sociedad de la Información de Castilla y León (ORSI), 2012)

Voto electrónico: Aplicación, total o parcial, de dispositivos y sistemas de tecnología de la información y telecomunicaciones a todo el proceso electoral. Puede incluir la emisión misma del voto en una urna electrónica, con o sin impresión inmediata de una boleta en papel para control del ciudadano o de la autoridad, el registro y verificación de la identidad del elector, el recuento en la mesa o el global consolidado, la transmisión de resultados, u otras actividades. (Facultad de Ciencias Sociales de Buenos Aires, 2004)

Finalmente, después de haber analizado algunas definiciones de voto y voto electrónico, se considera como más acertada la definición tomada de la Facultad de Ciencias Sociales de Buenos Aires por considerarse más descriptiva, ya que plantea que el voto electrónico se puede ver como la aplicación total o parcial de dispositivos electrónicos al proceso de votación, en lo que se está completamente de acuerdo y no se refleja así en la definición tomada del ORSI, donde se plantea que esto no supone nada novedoso porque solo se automatiza la parte de la emisión del voto, y no ciertamente debe ser de esta forma puesto a que en este caso se van a automatizar todos los procesos referentes al registro y autenticación de los electores, emisión del voto, escrutinio y publicación de resultados, siendo este, el conjunto de todas las actividades que se llevan a cabo para la realización de las votaciones. Por esta razón, se entiende como voto electrónico a aquel que se diferencia del método tradicional por la utilización de componentes de hardware y software, que permiten automatizar los

diferentes procesos de una elección y que se realiza por medio de algún dispositivo electrónico, en una urna electrónica o una computadora personal.

1.3 Requisitos legales del sufragio.

Algunos de los requisitos contemplados en las leyes de mayor jerarquía se encargan de asegurar que el proceso de votación sea lo más transparente posible y que se realice de forma adecuada, dando la posibilidad del derecho al voto a todas las personas, aun cuando presenten ciertas discapacidades para ejercerlo, y establecen las pautas a seguir para que se lleve a cabo un proceso lejos del fraude y la corrupción. Entre estos requisitos se encuentran:

Universal: Establece que los sistemas deben estar habilitados para que todos los ciudadanos que cumplan con un conjunto de condiciones como la edad, nacionalidad y período de residencia en una determinada jurisdicción, puedan ejercer el voto y solamente ellos.

Igual: Todos los ciudadanos que componen el universo de una elección deben poder votar solo una vez y todos los votos tienen el mismo valor: un ciudadano, un voto.

Secreto: Debe asegurarse que la identidad de los ciudadanos no pueda ser vinculada, de ninguna forma, al voto que emitió. Debe garantizarse que ni siquiera el mismo elector tiene posibilidades de demostrar que votó de determinada manera.

Personal: El voto debe ser emitido en forma personal por el ciudadano. Salvo el caso particular de los ciudadanos con alguna discapacidad física especial, ya sea sensorial o intelectual, que pueden ejercer el voto asistido y son responsables de decidir si necesitan o no ser asistidos por alguien de su confianza a la hora de votar. Esta persona debe ser mayor de edad y no importa su sexo; debe comunicarlo al presidente de la mesa verbalmente o por escrito, y luego el secretario de mesa constatará el hecho precisando la identidad del votante[7] y su asistente. Esta normativa garantiza el ejercicio del derecho a sufragio y otorga facilidades para que los discapacitados ejerzan su voto en forma autónoma y participación política[8].

Los requisitos antes expuestos deben cumplirse tanto por los sistemas de votación tradicional como por los que utilicen medios electrónicos. En el caso de los sistemas informáticos que ejecutan el voto electrónico, deben contar con todos los requisitos del

[7] Es cualquier persona que tiene derecho a participar en un proceso de elección emitiendo un voto.

[8] Tomado de http://www.ciudadaccesible.cl/index.php?option=com_content&view=article&id=145:ley-sobre-voto-asistido-para-personas-con-discapacidad&catid=50:leyes&Itemid=82.

sistema electoral tradicional, pero además, deben ser como mínimo, flexibles, auditables y convenientes.

Flexible: Debe adaptarse a distintos tipos de elecciones. Existen para esto dos alternativas de solución; la primera es construir un sistema totalmente parametrizable, de modo que sin modificar el código, pueda utilizarse para distintas elecciones. La segunda es diseñar un sistema sencillo para cada elección. La desventaja es que, aunque sencillo, el software deberá ser auditado en cada oportunidad. En la UCI se debe implementar la primera opción, que es construir un sistema configurable que permita ser utilizado para las elecciones a nivel de facultad y de universidad.

Auditable: Deben poder auditarse la totalidad de los niveles de software y no solo el correspondiente al nivel de las aplicaciones de emisión del voto, contabilización de votos y agregación de resultados parciales. Para el sistema de voto electrónico a desarrollar, se llevará a cabo una auditoría previa a la elección, con el objetivo de asegurar que todos los elementos que se usarán en los diferentes procesos funcionen de acuerdo a las especificaciones. Se realizará un análisis exhaustivo de la arquitectura y funcionalidad del sistema de votación con el fin de determinar su seguridad. Se verificará la integridad de los componentes físicos y lógicos que se utilizarán en la elección. Además, se realizará una auditoría posterior a la elección para verificar el correcto funcionamiento de todas las fases de la elección una vez que esta haya finalizado. (Instituto de Investigaciones Juridicas de la UNAM, 2012)

Conveniente: El sistema debe facilitar su uso, aun a aquellos que no están habituados al empleo de herramientas computacionales y su interfaz no debe desvirtuar la voluntad del elector. Los electores de la UCI no presentan problemas en cuanto al uso de herramientas computacionales ya que están acostumbrados a trabajar en este medio, lo que facilita aun más el proceso. En cuanto a la interfaz del sistema, es algo en lo que el equipo de trabajo debe esforzarse para proporcionar un ambiente amigable y usable, evitando desviar la voluntad de los electores.

El sistema a desarrollar contará con todos los requisitos del sistema tradicional y además, con los que debe cumplir un sistema que emplee medios electrónicos, para de esta forma lograr un producto más completo, ya que para la seguridad no es suficiente el empleo de las técnicas y herramientas actuales si el software no se ha creado adecuadamente; es decir, si la base no se ha aplicado correctamente todo añadido posterior no producirá los beneficios esperados.

1.4 Criptografía[9] en el voto electrónico.

El proceso de votación enfrenta ciertas dificultades a la hora de utilizar las tecnologías electrónicas e informáticas, ya que no es fácil definir las medidas apropiadas para que los votos emitidos con dichas tecnologías no se relacionen con los rastros dejados por los electores en el sistema. Hoy en día, existen varias propuestas para los sistemas de voto electrónico, todas ellas basadas en las técnicas criptográficas.

Con el paso del tiempo se han desarrollado protocolos criptográficos aplicables al voto electrónico y los requisitos de seguridad que se tratan de satisfacer por medio de la criptografía son: privacidad de los votos, autenticación de los votantes e integridad de los elementos de la elección.

Para el Sistema de Votación Electrónica en la UCI se empleará el algoritmo criptográfico RSA (Rivest, Shamir y Adleman). Es un sistema criptográfico de clave pública desarrollado en 1978. Es el primer y más utilizado algoritmo de este tipo y válido tanto para cifrar como para firmar digitalmente.
La seguridad de este algoritmo radica en el problema de la factorización de números enteros. Los mensajes enviados se representan mediante números, y el funcionamiento se basa en el producto, conocido, de dos números primos grandes elegidos al azar y mantenidos en secreto. Actualmente estos primos son del orden de 10^{200}, y se prevé que su tamaño aumente con el aumento de la capacidad de cálculo de los ordenadores. (Chaum, febrero1981)

Como en todo sistema de clave pública, cada usuario posee dos claves de cifrado[10]: una pública y otra privada. Cuando se quiere enviar un mensaje, el emisor busca la clave pública del receptor, cifra su mensaje con esa clave, y una vez que el mensaje cifrado llega al receptor, este se ocupa de descifrarlo usando su clave privada. Se cree que RSA será seguro mientras no se conozcan formas rápidas de descomponer un número grande en producto de primos. La computación cuántica podría proveer de una solución a este problema de factorización.

Los sistemas de cifrado de clave pública o sistemas de cifrado asimétricos se inventaron con el fin de evitar por completo el problema del intercambio de claves de los sistemas de cifrado simétricos. Con las claves públicas no es necesario que el remitente y el destinatario se pongan de acuerdo en la clave a emplear. Todo lo que se

[9] Ciencia que permite que determinada información sea ininteligible o incomprensible para usuarios no autorizados.

[10] Es el proceso de convertir el texto plano en una serie de datos ilegibles, denominado texto cifrado o criptograma. Texto plano es la información original que debe resguardarse.

requiere es que, antes de iniciar la comunicación secreta, el remitente consiga una copia de la clave pública del destinatario. Es más, esa misma clave pública puede ser usada por cualquiera que desee comunicarse con su propietario. Por tanto, se necesitarán sólo n pares de claves por cada n personas que deseen comunicarse entre sí.[11]

1.5 Sistemas de votación electrónica.

- **Sistemas de registro electrónico directo (DRE por sus siglas en inglés):** Estos sistemas de votación utilizan medios digitales para la selección del voto, graban los votos por medio de una boleta de votación en forma de pantalla provista de componentes mecánicos o eléctrico-ópticos que pueden ser activados por el votante, puede ser por medio de botones o pantalla táctil incluidos en el terminal de votación. Están constituidos por máquinas especialmente fabricadas para la votación, que no se encuentran conectadas con otras máquinas. El registro del voto se almacena localmente en formato electrónico. Una variante de los sistemas DRE son los que imprimen la papeleta para que sea depositada en una urna, teniendo de esa manera un registro electrónico y otro impreso. Estos terminales suelen contar con una interfaz de red, por lo que los resultados generados localmente pueden enviarse a un servidor central por medio de una red de comunicación. Otra opción es almacenar una copia de los resultados locales en un medio de almacenamiento removible y entonces enviar, dicho registro, a un centro de escrutinio para la consolidación de los resultados. (The Electoral Knowledge Network, 1998-2012)

Los sistemas DRE son aquellos que más se corresponden con el imaginario popular de las urnas electrónicas. Representan, además, el modelo preferido de la mayoría de las empresas que participan de este mercado. Muchos de los proveedores señalan como una ventaja del sistema el hecho de que permite independizar del papel a la elección. Por lo general recomiendan no usar la opción ofrecida por algunos modelos de máquinas DRE, de usar impresoras para generar una cinta de auditoria, argumentando que desnaturaliza el voto electrónico, sino que se basan enteramente en los registros presentes en su memoria. Estos pueden configurarse de tal modo que permitan al usuario corregir sus opciones y hasta votar en blanco, pero no permiten invalidar el voto ni cometer errores clásicos que resultan en la anulación del voto. Los

[11] Tomado de http://eprints.ucm.es/10023/1/RSumoza-Proyecto-Fin-Master.pdf. E-Prints Complutense. Biblioteca Universidad Complutense Madrid 2012.

mismos previenen a los votantes de errores involuntarios, ya que el terminal de votación conduce al votante paso a paso hasta que un voto válido es registrado, pero existen muchísimos trucos y accesorios adicionales que pueden cambiar las bases de datos que se almacenan en sus memorias que a su vez pueden alterar resultados en los votos.

- **Sistema de voto electrónico en papel:** Es un sistema electoral basado en papel, en el cual los votos emitidos se cuentan manualmente. Primeramente surgieron sistemas en los cuales se podían marcar a mano, tarjetas o láminas de papel, que eran contadas electrónicamente. Estos incluían votación mediante máquinas de votar o tarjetas perforadas, sistemas de votación de escaneo óptico, sistemas de marcado y escaneo óptico y más tarde sistemas de votación con lápiz óptico. Recientemente, estos sistemas pueden incluir un marcador electrónico de boletas que permite a los votantes seleccionar usando una máquina, normalmente una pantalla sensible para digitación similar a un DRE. (About.com,Inventors, 2012)
- **Sistemas de votación en red:** Algunos o todos los datos del proceso electoral son transmitidos sobre una red de comunicaciones. La red es generalmente pública como Internet, o privada. Estos son parecidos a los sistemas DRE pero tienen capacidad para transmitir los resultados de la votación ya sea en línea, en lotes o al final de la jornada electoral. Estos sistemas tienen dos alternativas: Sistema de votación en red asistido; donde el elector tiene que asistir a un centro de votación previamente determinado, se identifica ante el administrador y este le asigna una computadora para votar y Sistema de votación en red no asistido; donde el elector no tiene que desplazarse a un lugar de votación sino que puede votar desde cualquier lugar donde tenga acceso a Internet. (Soldevilla, 2005-2012)

A partir de la realización del análisis de estos sistemas de votación, se tomaron los elementos necesarios para modelar el sistema que se va a utilizar en la UCI para las votaciones de la FEU.

Para acceder a cada subsistema se requiere autenticación previa.

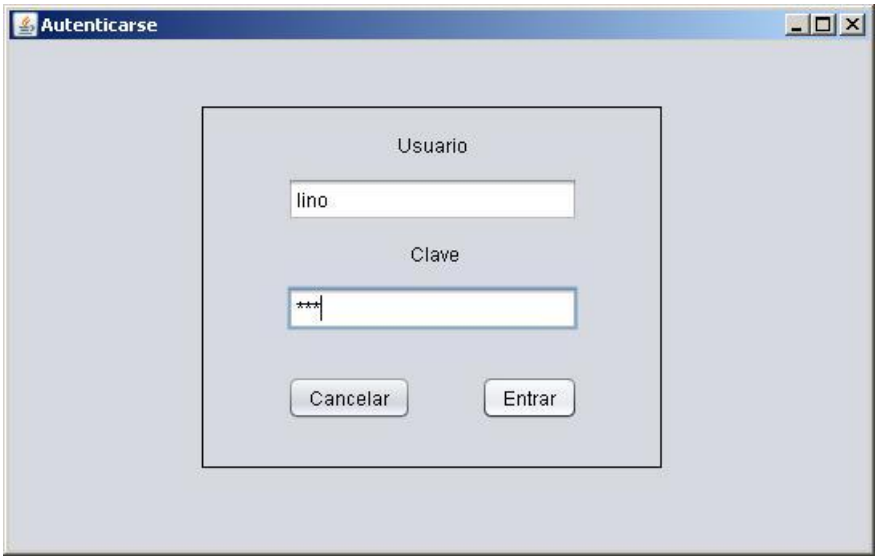


Ilustración 1 Interfaz de autenticación para cada una de los subsistemas.

Subsistema Configuración

Se ejecuta el subsistema de configuración para así poder crear los archivos necesarios para el funcionamiento de los demás subsistemas, cuando el usuario selecciona la opción de adicionar una nueva mesa electoral, se muestra una interfaz con los datos necesarios e imprescindibles para que este funcione, tales como listado de electores, listado de urnas de las cuales va a tomar responsabilidad, usuario, contraseña, hora de apertura, hora de cierre, en adición se generará una llave con la cual se va a poder descifrar todos estos datos una vez en el subsistema de mesa electoral, a continuación una imagen de esta interfaz.

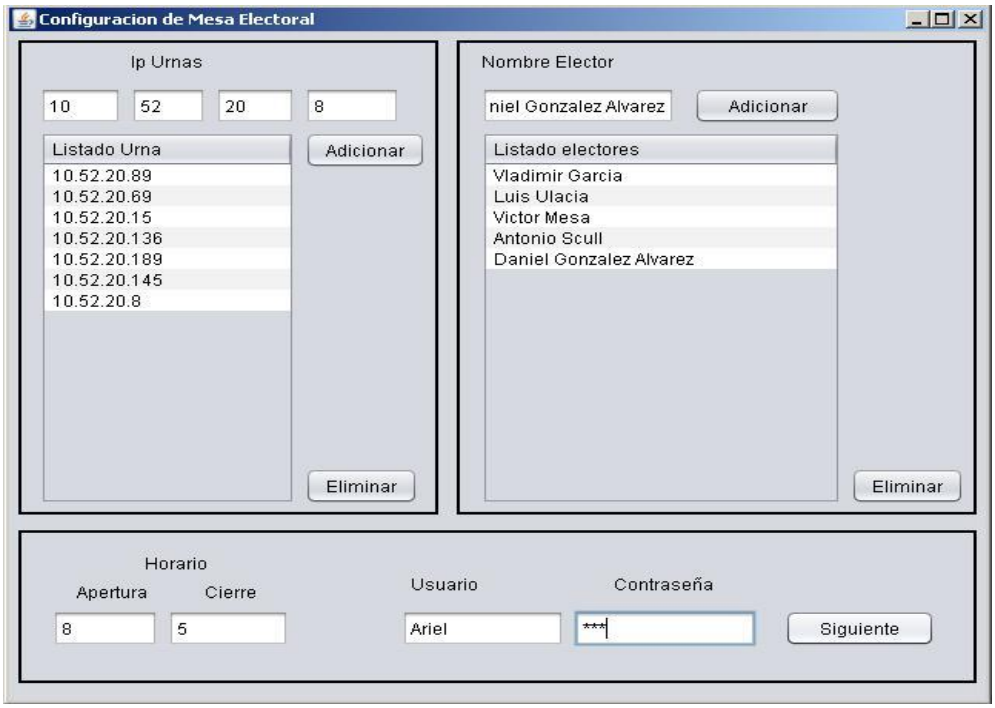


Ilustración 2 Interfaz configuración de la Mesa Electoral.

También se establece una configuración para las urnas, una vez creada la mesa se confecciona un archivo que va a ser el encargado de proveer al subsistema urna electoral de usuario, contraseña, listado candidatos, hora apertura, hora cierre, la llave va a ser compartida por el subsistema Mesa Electoral y este último.

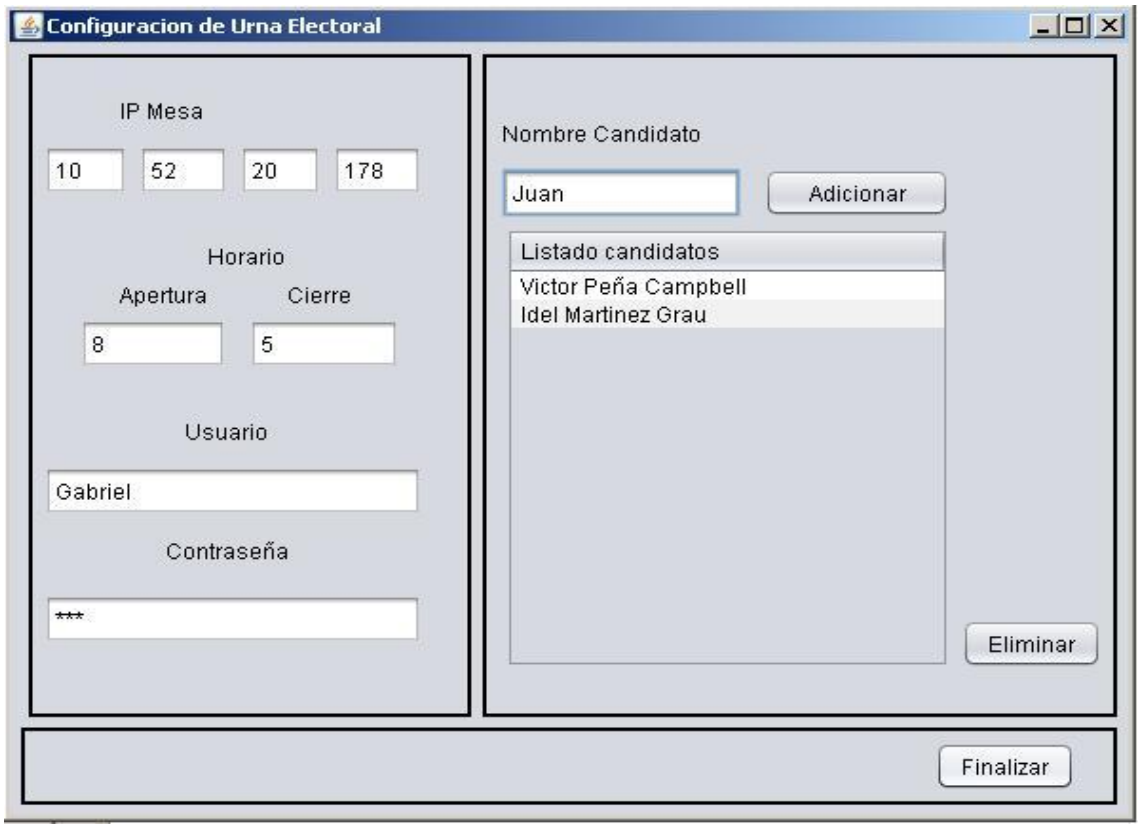


Ilustración 3 Interfaz configuración Urna Electrónica.

Subsistema Mesa Electoral

Una vez que el archivo de configuración de mesa sea llevado a su destino, se podrá trabajar en este subsistema. En esta interfaz se muestran los nombres de los electores que deben votar y las urnas relacionadas. Debido al hecho de tener un gran listado de electores, para dar una mayor facilidad a la hora de realizar la búsqueda en el sistema, se implementó un algoritmo que busca la sub - secuencia escrita en el campo de texto, en todos los posibles electores, mostrando solamente aquellos que la contengan.

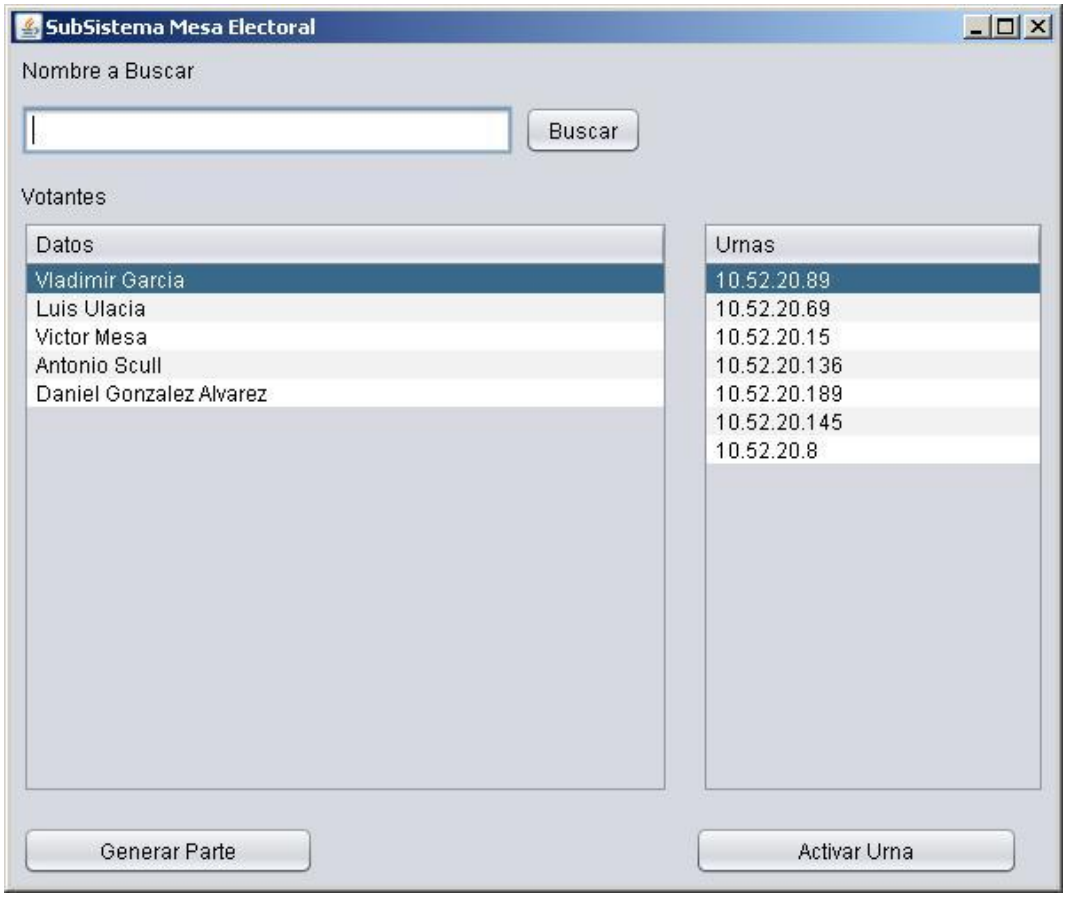


Ilustración 4 Interfaz principal del subsistema Mesa Electoral.

En cualquier momento el responsable de la mesa podrá crear un parte para así tener conocimiento de las personas que ya votaron y las que faltan por hacerlo.

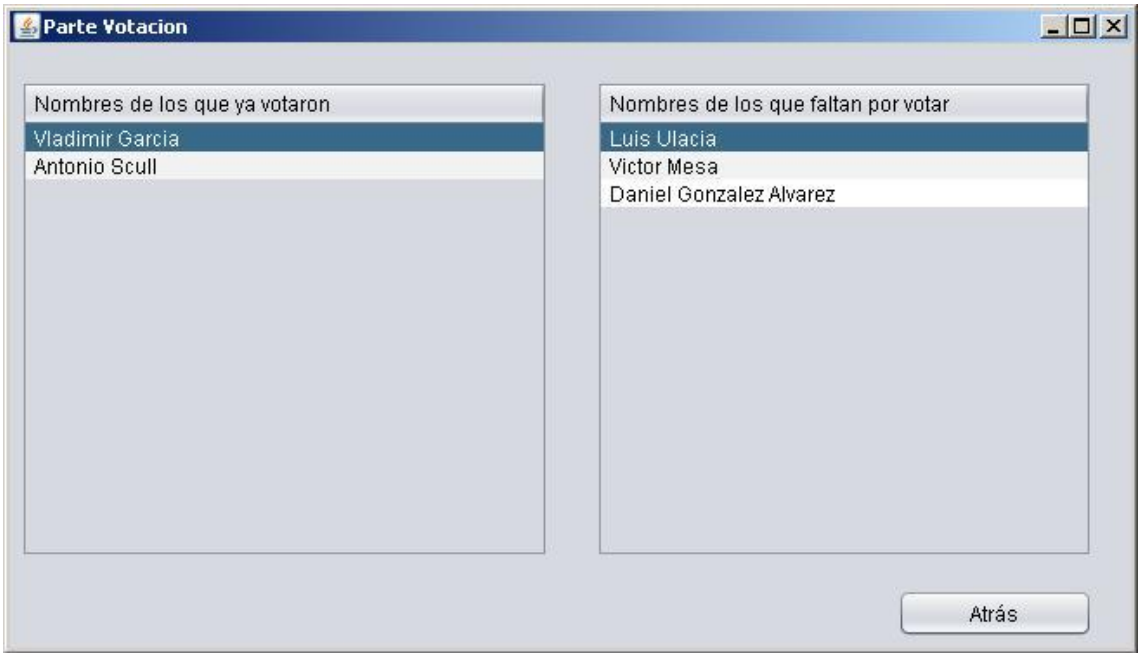


Ilustración 5 Interfaz parte votación.

Subsistema Urna Electoral

Se carga el archivo de configuración del subsistema, el cual quedará listo para ser usado, se requiere una autenticación previa del responsable del local para que el sistema quede en espera de activación, luego de recibir esta, el sistema permitirá al elector, seleccionar como máximo 5 candidatos y un único presidente, además el elector podrá adicionar 2 nuevos candidatos.

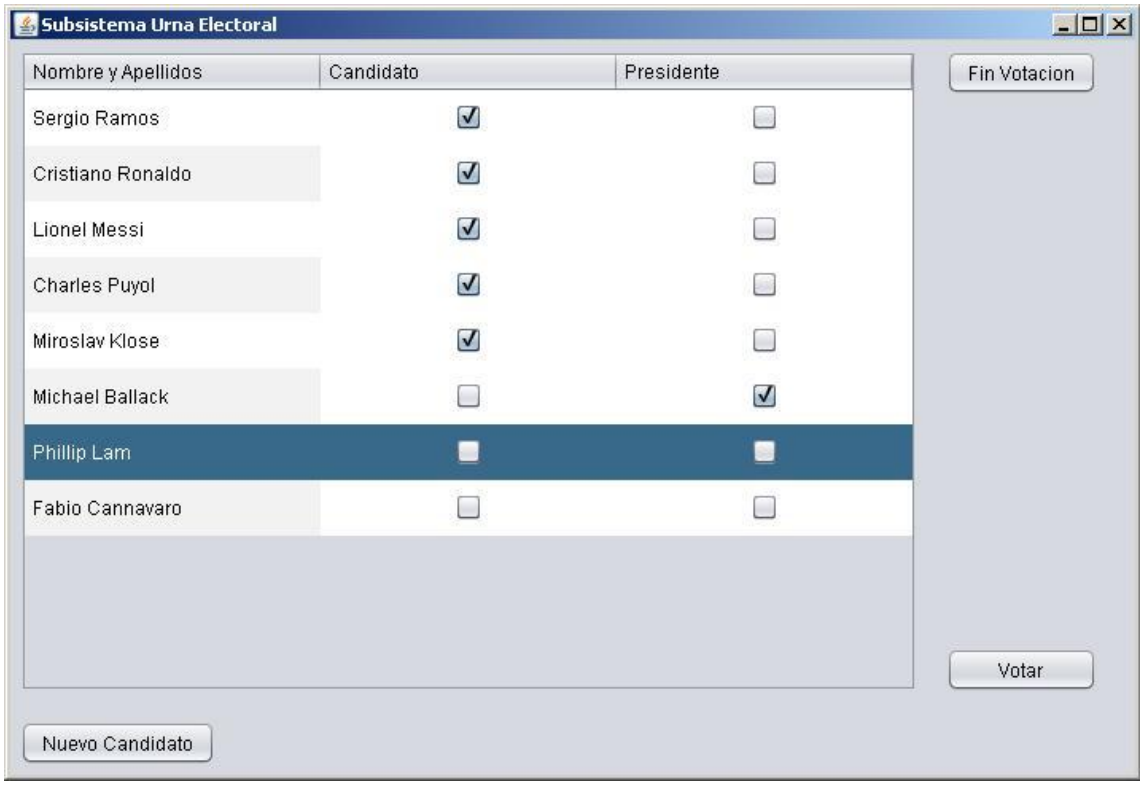


Ilustración 6 Interfaz principal del subsistema urna electoral.

Una vez seleccionada la opción adicionar nuevo candidato, la aplicación muestra una pequeña interfaz con un campo de texto, donde se podrá teclear el nombre del nuevo candidato.

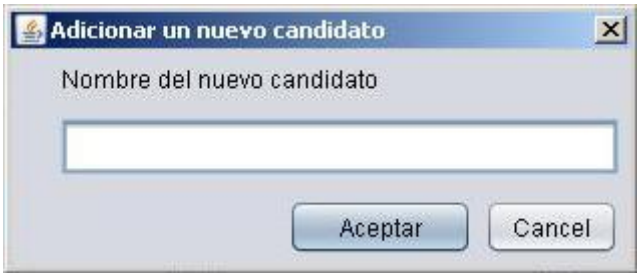


Ilustración 7 Interfaz para adicionar un nuevo candidato.

Si por esta vía ya han sido adicionados a la boleta 2 nuevos candidatos, el sistema muestra la notificación: “No se pueden añadir más candidatos”.

Una vez concluido el período de tiempo asignado para las elecciones, el subsistema no permitirá realizar ninguna otra votación y realizará el proceso de escrutinio, del cual se obtendrá como resultado un archivo que va a contener la cantidad de votos en esa

urna, así como la cantidad de votos por candidatos y presidente, este se lleva al subsistema de consolidación el cual va a ser el encargado de dar un resultado al proceso eleccionario.

Subsistema Consolidación

Es el encargado de dar un resultado al proceso eleccionario, para su funcionamiento también tiene que cargar un fichero de configuración que va a estar conformado por usuario, contraseña, listado de urnas que deben entregarle y el archivo resultado del escrutinio para realizar su trabajo.

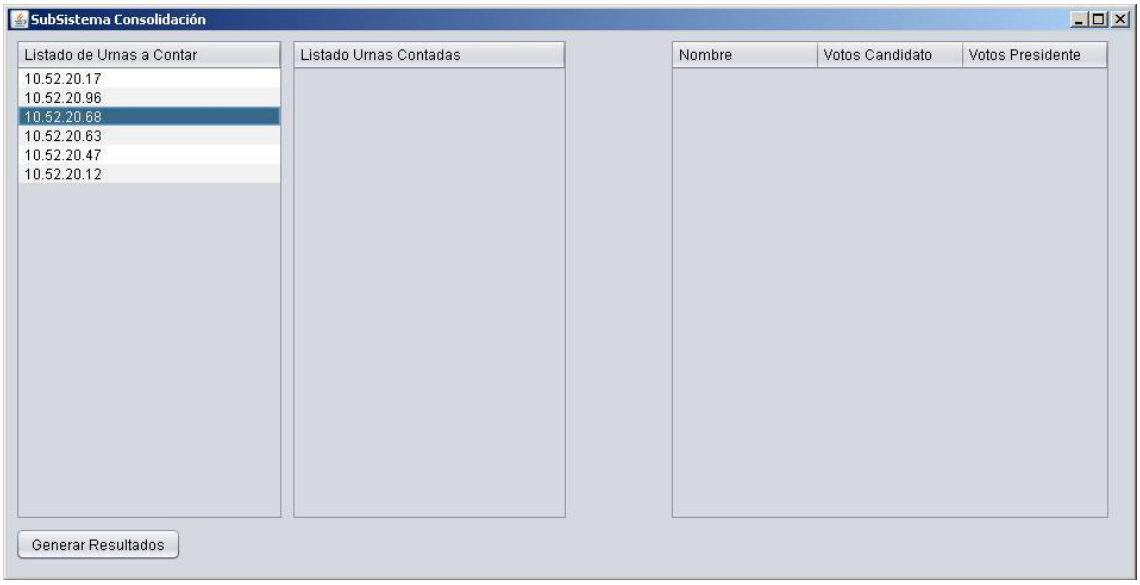


Ilustración 8 Interfaz principal del subsistema consolidación.

En la interfaz anterior, se muestra como queda el listado de urnas que el subsistema consolidación debe contar. Luego de seleccionar la opción generar resultados, el sistema recopila los datos de las urnas existentes, mostrando el resultado (en la tabla de la derecha) en el siguiente formato, nombre del candidato, seguido por la cantidad de votos por candidatos y finalmente los votos por presidente, estos datos se mostrarán organizados descendentemente de forma tal que encima quede el candidato con más votos, en las tablas de la izquierda se mostrarán las urnas que faltan por contar y las que ya han sido contadas.

Secuencia para realizar el voto:

- El elector llega al local seleccionado para realizar las elecciones.
- Se dirige hacia la mesa electoral.
- Muestra el solapín para así comprobar que es quien dice ser.
- Además se verifica si le toca votar por ese local.
- De ser así, se le activa una urna para que pueda efectuar la votación.

- Una vez que este vota, la urna queda bloqueada en espera de una próxima activación.
- El elector se retira.

1.6 Implicaciones de Carácter Informático.

El software electoral se define como el conjunto de instrucciones usadas directa o indirectamente en ordenadores, con el fin de obtener un resultado primordialmente autorizado en actividades relacionadas con el desarrollo de un proceso electoral. (Flores & Téllez Valdés, 2010)

La experiencia internacional sobre el desarrollo de software electoral indica que es recomendable estructurarlo a partir de módulos, donde no se permita su modificación. El módulo del software que procesa el conteo de la votación recibida debe ser escrito en un lenguaje de alto nivel y el software no debe residir en un dispositivo, excepto el software de base relacionado con el arranque del sistema o control de periféricos. El software electoral debe ser instalado en ocasión de cada proceso electoral, en un horario determinado y siendo supervisado.[12]

Con la implementación de estas medidas se garantiza un software de calidad para estos procesos. Estas constituyen una guía para obtener un producto completo, que cuente con los principios del voto tradicional y los del voto electrónico para mejorar el proceso de votación haciéndolo más fácil para las personas. Al utilizarlas en la implementación de software electoral, se maximizan las posibilidades de obtener un producto seguro reduciendo la posibilidad de que este pueda ser vulnerable o modificado.

1.7 Fortalezas del voto electrónico.

- Produce un impacto significativo sobre los materiales, actividades y procedimientos electorales debido a la eliminación de las boletas de votación, del padrón de electores impreso, del acta electoral y sus copias, reducción del número de mesas y del número de miembros de mesa, eliminación de los votos por error y disminución del tiempo de entrega de resultados.
- La votación se hace más sencilla y rápida para los electores debido a que pueden efectuar el voto con tan solo un clic, dejando atrás la tarea de utilizar papel y lápiz para la selección de los candidatos, corriendo el riesgo de

[12] (Instituto de Investigaciones Jurídicas de la UNAM – Serie Doctrina Jurídica, Núm. 550: Voto Electrónico, Derecho y Otras Implicaciones por Rodolfo Romero Flores y Julio Alejandro Téllez Valdés. Véase http://www.juridícas.unam.mx)

equivocarse y echar a perder la boleta que después de utilizada debe ser doblada y puesta en la urna correspondiente. Además, se evita la falta de boletas.

- Reduce la posibilidad de fraude en las mesas de votación ya que el escrutinio no se realiza de forma manual y además se evita la introducción de errores.
- Garantiza la privacidad ya que puede implementarse de modo que sea imposible asociar el voto con el votante.
- Contribuye a un rápido y exacto conteo de votos.
- Proporciona flexibilidad para modificaciones de última hora y posibilita la realización de elecciones con más frecuencia.

1.8 Desventajas del voto electrónico

El mayor peligro para los sistemas de voto electrónico es la posibilidad de injerencia externa en ellos y que la misma pueda pasar desapercibida, afectando los resultados de la votación. Por otra parte está el daño del hardware o equipo de red. El hardware incluye los terminales de votación, por ejemplo los servidores, ordenadores personales y por su parte, el equipo de red incluye los elementos que forman el o los canales de comunicación que se utilizan en los procesos de la elección. Tanto el hardware como el equipo de red pueden sufrir daños provocados, accidentales o incluso ambientales y pueden ocasionar que la elección no se lleve a cabo con normalidad o incluso que pueda ser interrumpida.

1.9 Sistemas informáticos.

Estos basan la parte fundamental de su procesamiento en el empleo de la computación. Son un conjunto de funciones interrelacionadas, hardware, software y recurso humano. Un sistema informático emplea dispositivos que se usan para programar y almacenar programas y datos. (McGraw-Hill Interamericana de España,SL, 2012)

Aplicaciones Web: Es una página web especial, que tiene una base de datos asociada y que permite una mayor interacción del usuario. Son fáciles de usar, no requieren de conocimientos avanzados de computación y se pueden realizar tareas sencillas sin necesidad de descargar ni instalar ningún programa. (GNU Consultores, 2011)

Habitualmente ofrecen menos funcionalidades que las aplicaciones de escritorio y esto se debe a que las funcionalidades que se pueden realizar desde un navegador son más limitadas que las que se pueden realizar desde el sistema operativo. Como principal desventaja presenta que la disponibilidad depende de un tercero, el

proveedor de la conexión a internet o el que provee el enlace entre el servidor de la aplicación y el cliente. Para ejecutar una aplicación web es necesario contar con un navegador web compatible y/o descargar los plug-ins necesarios, acciones las cuales requieren un nivel de conocimiento mayor al de un usuario de nivel básico (al cuál está orientada la aplicación), lo cuál puede representar una barrera para aquellas personas no habituadas a las herramientas computacionales

Aplicaciones de escritorio: Por aplicaciones de escritorio se entiende toda aplicación que ha sido desarrollada para ser ejecutada en una plataforma específica, ya sea Windows, Linux o Mac. El desarrollo sobre una plataforma, normalmente, implica que la aplicación "no" pueda ser ejecutada en otras. (GNU Consultores, 2011)

Una aplicación de escritorio es aquella que está instalada en el ordenador del usuario, que es ejecutada directamente por el sistema operativo y cuyo rendimiento depende de diversas configuraciones de hardware como memoria RAM, disco duro o memoria de video. Habitualmente su ejecución no requiere comunicación con el exterior, sino que se realiza de forma local. Esto repercute en mayor velocidad de procesamiento, y por tanto en mayor capacidad a la hora de programar herramientas más complicadas o funcionales. Suelen ser más robustas y estables que las aplicaciones web y en cuanto a rendimiento, el tiempo de respuesta es más rápido. También pueden ser muy seguras, dependiendo del desarrollador. Tienen sus propias ventajas sobre las aplicaciones web en cuanto a velocidad, capacidad de alcanzar el sistema de archivos, trabajo en segundo plano y notificaciones.

Presentan la desventaja de ser dependientes del sistema operativo que utilice el ordenador y sus capacidades, como video y memoria. Su acceso se limita al ordenador donde están instaladas y requieren instalación y actualización personalizada además de tener requerimientos especiales de software y librerías.

Para dar solución a la problemática existente en la Universidad de las Ciencias Informáticas para llevar a cabo las votaciones de la FEU, se decidió implementar una aplicación de escritorio debido a que estas pueden ser más robustas, su tiempo de respuesta es más rápido y se puede hacer todo lo que el sistema operativo permita. Las aplicaciones de escritorio mantienen un contacto permanente entre los procesos internos del programa y lo que sucede en la interfaz de usuario. Es por esto que no requieren del paradigma de páginas de la web y tienden a ofrecer una experiencia de usuario más fluida entre una acción y otra. Brindan mayor capacidad visual, menor tiempo de respuesta y mayor personalización. Al tratarse la solución de una pequeña aplicación de escritorio solo es necesario instalarla en todos los equipos que se desee

y podrá ser ejecutada en cualquier momento, sin depender de factores externos para funcionar correctamente. Los datos pueden ser respaldados en el propio disco duro del equipo y se pueden hacer copias de estos con mucha facilidad. La información es totalmente privada y del uso del usuario de la maquina donde esta se encuentra instalada. (GNU Consultores, 2011.

1.10 Tecnologías, Metodologías y herramientas de desarrollo de software

Tecnologías.

- **C#:** Fue creado por el danés Anders Hejlsberg. Se pretendió que incorporase las ventajas o mejoras que tiene el lenguaje Java, por eso su sintaxis es muy similar. Así se consiguió que tuviese las ventajas del C, del C++, pero además la productividad que posee el lenguaje Java. Es un lenguaje orientado a objetos y a componentes. Se ahorra tiempo en la programación ya que tiene una librería de clases muy completa y bien diseñada. (La Revista Informática.com)

Es un lenguaje orientado a objetos, simple y con seguridad en el tratamiento de tipos. Permite a los programadores de aplicaciones empresariales crear una gran variedad de aplicaciones y proporciona la capacidad de generar componentes de sistema duraderos en virtud de su gran robustez, gracias a la recolección de elementos no utilizados, liberación de memoria y a la seguridad en el tratamiento de tipos. En este lenguaje la seguridad es implementada por medio de mecanismos de confianza intrínsecos del código y presenta plena compatibilidad con conceptos de metadatos extensibles. (Microsoft, 2012)

C# soporta todas las características propias del paradigma de programación orientada a objetos: encapsulación, herencia y polimorfismo y se caracteriza por su sencillez debido a que elimina muchos elementos que otros lenguajes incluyen y que son innecesarios en .NET. Por ejemplo:

- ✓ El código es auto contenido, lo que significa que no necesita de ficheros adicionales al propio, tales como ficheros de cabecera.
- ✓ El tamaño de los tipos de datos básicos es fijo e independiente del compilador, sistema operativo o máquina para quienes se compile, lo que facilita la portabilidad del código. (Clikear.com)

C# proporciona un amplio conjunto de operadores, símbolos que especifican qué operaciones realizar en una expresión. Predefine los operadores aritméticos y lógicos más usuales, así como otros diversos. Además, el usuario puede sobrecargar muchos

operadores, con lo que cambiará su significado durante su aplicación a un tipo definido por el usuario. (Visual C#, 2012), (Scribd, 2012)

- **Java:** Gracias a su versatilidad y eficiencia ha logrado ser un recurso inestimable porque permite desarrollar software en una plataforma y ejecutarlo en prácticamente cualquier otra. Permite combinar aplicaciones o servicios que usan el lenguaje Java para crear servicios o aplicaciones totalmente personalizados y desarrollar potentes y eficientes aplicaciones para prácticamente cualquier tipo de dispositivo digital. La Plataforma Java se compone de un amplio abanico de tecnologías, cada una de las cuales ofrece una parte del entorno de ejecución en tiempo real. Para el desarrollo de aplicaciones utiliza un conjunto de herramientas conocidas como JDK, Java Development Kit, o herramientas de desarrollo para Java. Un programa destinado a la plataforma Java necesita dos componentes en el sistema donde se va a ejecutar: una máquina virtual de Java y un conjunto de bibliotecas para proporcionar los servicios que pueda necesitar la aplicación. Los lenguajes de programación se encuentran fuera del ámbito de lo que es una plataforma, aunque el lenguaje de programación Java es uno de los componentes fundamentales de la plataforma Java. El propio lenguaje y el entorno en tiempo de ejecución suelen considerarse una única entidad. (James Gosling, A brief history of the Green project. Java.net)

Metodologías.

Las metodologías de desarrollo de software representan un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información. Tienen como objetivo presentar un conjunto de técnicas tradicionales y modernas de modelado de sistemas que permitan desarrollar software de calidad.

- **XP[13]:** Es el más destacado de los procesos ágiles de desarrollo de software. Se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad. Surge como respuesta y posible solución a los problemas derivados del cambio en los requisitos. En la mayoría de los casos se plantea como una metodología a emplear en los proyectos con riesgos de requisitos muy cambiantes. Los defensores de XP consideran que los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del desarrollo de software y creen que ser

[13] eXtreme Programming por sus siglas en inglés. Traducido al español como programación extrema. Véase: http://www.extremeprogramming.org/.

capaces de adaptarse a estos cambios en cualquier punto de la vida del proyecto es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos. En su ciclo de vida se realizan pruebas unitarias continuas, frecuentemente repetidas y automatizadas, incluyendo pruebas de regresión. Emplea la refactorización del código, es decir, se reescriben ciertas partes del código para aumentar su legibilidad pero sin modificar su comportamiento.

Las características fundamentales de XP son[14]:

- **Pruebas unitarias continuas:** frecuentemente repetidas y automatizadas, incluyendo pruebas de regresión. Se aconseja escribir el código de la prueba antes de la codificación.
- **Corrección de todos los errores:** se realiza antes de añadir nuevas funcionalidades. Hacer entregas frecuentes.
- **Refactorización del código:** reescribir ciertas partes del código para aumentar su legibilidad y mantenibilidad pero sin modificar su comportamiento. Las pruebas han de garantizar que en la refactorización no se ha introducido ningún fallo.
- **Propiedad del código compartida:** en vez de dividir la responsabilidad en el desarrollo de cada módulo en grupos de trabajo distintos, este método promueve el que todo el personal pueda corregir y extender cualquier parte del proyecto. Las frecuentes pruebas de regresión garantizan que los posibles errores serán detectados. (extreme Programming explained, 2009)

XP es muy parecido a un rompecabezas debido a que hay muchas piezas pequeñas. Individualmente, las piezas no tienen sentido, pero cuando se combinan, se puede ver un panorama completo. (Extreme Programming, 1999-2009)

- **RUP[15]:** Se caracteriza por seguir un ciclo de vida iterativo e incremental y propone que cada fase se desarrolle en iteraciones. Cada iteración engloba actividades de todos los flujos de trabajo, aunque desarrolla algunos más que otros. Cada iteración tiene que proponerse un incremento en el proceso de desarrollo pero todas deben aportar al principal resultado de la fase en la que

[14] Tomado del sitio extreme programming explained. Véase: http://extremeprogramming.host56.com.

[15] Rational Unified Process por sus siglas en inglés. Traducido al español como Proceso Unificado de Rational.

se desarrolla. Es centrado en la arquitectura y asume que no existe un modelo único que cubra todos los aspectos del sistema, por tal motivo existen múltiples modelos y vistas que definen la arquitectura de software de un sistema. Es guiado por casos de uso porque se utilizan para capturar los requisitos funcionales y para definir los contenidos de las iteraciones. La idea es que cada iteración tome un conjunto de casos de uso o escenarios y desarrolle todo el camino a través de las distintas disciplinas. (ibm.com, 2012)

La siguiente figura muestra la arquitectura global de RUP y sus dos dimensiones:

- El eje horizontal representa el tiempo y muestra los aspectos del ciclo vital del proceso a medida que se desarrolla.
- El eje vertical representa las disciplinas que agrupan de forma lógica las actividades por naturaleza.

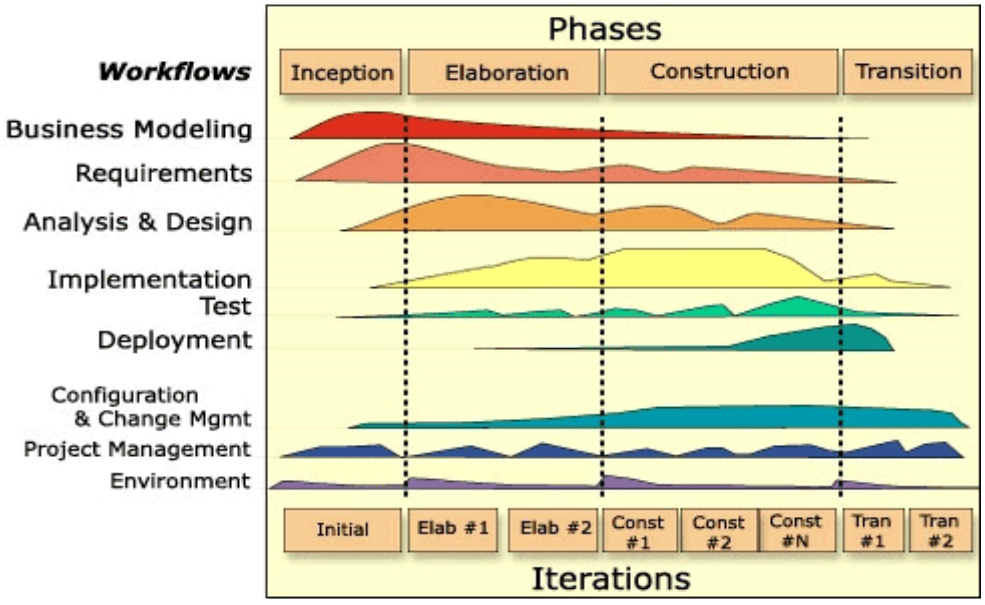


Ilustración 9 Ciclo de vida de la metodología RUP.

RUP consta de ciclos que puede repetir durante la vida útil de larga duración de un sistema. Un ciclo se compone de cuatro fases: Concepción, Elaboración, Construcción y Transición. Cada ciclo se concluye con un comunicado, también hay versiones dentro de un ciclo. Las cuatro fases de una iteración son:

Creación del fondo de fase: Durante la fase inicial de la idea central se desarrolla en una visión del producto. En esta fase, revisar, discutir y entender el modelo de negocio del proyecto. La fase de inicio se establece la viabilidad del producto y delimita el alcance del proyecto.

Elaboración de fase: Durante la fase de elaboración la mayoría de los casos de uso se especifican en detalle y la arquitectura del sistema está diseñada. Se identifican los riesgos significativos y se prepara el programa, el personal y el perfil de costo para todo el proyecto.

Fase de construcción: Durante la fase de construcción el producto se mueve desde la línea de base arquitectónica a un sistema completo, suficiente para la transición a la comunidad de usuarios. La línea de base arquitectónica crece para convertirse en el sistema completo, como el diseño se refina en el código.

La transición de fase: En la fase de transición el objetivo es asegurar que los requisitos se han cumplido a satisfacción de las partes interesadas. Esta fase se suele iniciar con una versión beta de la aplicación. La fase de transición termina con un post-mortem dedicado al aprendizaje y el registro de las lecciones para futuros ciclos.
(Skill Resource (Habilidad de Recursos de Software), 2005-2006)

RUP es una plataforma flexible que ayuda brindando guías consistentes y personalizadas de procesos para todo el equipo de proyecto. Describe cómo utilizar de forma efectiva reglas de negocio y procedimientos comerciales probados en el desarrollo de software, conocidos como “mejores prácticas”. Es una guía de cómo utilizar de manera efectiva UML[16]. Provee fácil acceso a una base de conocimiento con guías, plantillas y herramientas para todas las actividades críticas de desarrollo. Reduce el riesgo de no sacar el producto en el calendario previsto y acelera el ritmo de desarrollo.

Lenguajes de modelado.

Un modelado se puede ver como un conjunto de recursos y actividades interrelacionados que transforman elementos de entrada en elementos de salida. Los recursos pueden incluir personal, finanzas, instalaciones, equipos, técnicas y métodos.

- **BPMN[17]:** Esta notación gráfica estandarizada permite el modelado de procesos de negocio en un formato de grafos de flujo para crear modelos de operaciones de procesos de negocio. Su creación tuvo como base la recopilación de experiencias de estándares como UML. Está planeada para dar soporte, únicamente, a aquellos procesos que sean aplicables a procesos

[16] Unified Modeling Language por sus siglas en inglés. Traducido al español como Lenguaje Unificado de Modelado.

[17] Business Process Management Notation por sus siglas en inglés. Traducido al español como Notación para el Modelado de Procesos de Negocio.

de negocios y esto significa que cualquier otro tipo de modelado realizado por una organización con fines distintos de los del negocio no estará en el ámbito de BPMN. (Object Management Group , 1997-2012)**,** (bpm.com, 2012)

- **UML:** Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. Está consolidado como el lenguaje estándar en el análisis y diseño de sistemas de cómputo. Mediante él es posible establecer la serie de requisitos y estructuras necesarias para plasmar un sistema de software previo al proceso intensivo de escribir código. Ofrece un estándar para describir un "modelo" del sistema, incluyendo aspectos conceptuales tales como procesos de negocio, funciones del sistema y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables. Aunque es un lenguaje, posee más características visuales que programáticas, las cuales facilitan a integrantes de un equipo multidisciplinario participar e intercomunicarse fácilmente, siendo estos integrantes: los analistas, diseñadores, especialistas de área y desde luego los programadores. Un modelo UML puede ser usado en todas las fases de desarrollo de software, lo que facilita su entendimiento y la comunicación. (Stevens, Pooley, & Wesley, 2002)

Herramientas CASE [18].

- **Rational Rose:** Es una herramienta de diseño unificada, producida y comercializada por Rational Software Corporation, ahora propiedad de IBM. Entre sus principales características presenta la integración con otras herramientas de desarrollo de Rational, capacidad de análisis de calidad de código, control por separado de componentes modelo y modelado UML como medio para facilitar la captura de la semántica de dominio y la intención arquitectura / diseño, para trabajar en diseños de base de datos, con capacidad de representar la integración de los datos y los requisitos de aplicación a través de diseños lógicos y físicos. Su principal desventaja reside en que es una herramienta compatible solamente con sistemas operativos de Microsoft, aunque se integra con varios entornos de desarrollo. (rationalrose.com, 2012)

[18] Computer Aided Software Engineering por sus siglas en inglés. Traducido al español como Ingeniería de Software Asistida por Computadora.

Rational constituye un conjunto de herramientas operativas que utiliza UML como medio para facilitar la captura de la semántica de dominio y la intención arquitectura / diseño. (Computer Science and Engineering, 2012), (IBM, 2012)

- **Visual Paradigm:** Es una herramienta de modelado basada en la notación UML profesional que soporta el ciclo de vida completo del proceso de desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todo tipo de diagramas, código inverso, generar código desde diagramas y generar documentación. (Pressman R. , 2002)

Visual Paradigm constituye una poderosa herramienta de software respaldada bajo licencia gratuita y comercial, de probada utilidad para los analistas. Propicia un conjunto de ayudas para el desarrollo de programas informáticos. Permite la sincronización entre el código fuente y el modelo en tiempo real y brinda soporte para toda la notación UML. Fue diseñado para una amplia gama de usuarios interesados en la construcción de sistemas de software de forma fiable a través de la utilización de un enfoque orientado a objetos. Se caracteriza por el diseño centrado en casos de uso y enfocado al negocio, lo que permite generar un software de mayor calidad. Usa un lenguaje estándar común a todo el equipo de trabajo facilitando la comunicación con modelo y código sincronizados en todo el ciclo de desarrollo, generación de código para Java y exportación como HTML siendo fácil de instalar y actualizar.

Entornos de desarrollo.

- **Microsoft Visual Studio:** Es un IDE[19] para sistemas operativos Windows. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J# y Visual Basic .NET, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros. Permite a los desarrolladores crear aplicaciones de escritorio, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET a partir de la versión .NET 2002. Así se pueden crear aplicaciones que se intercomuniquen entre estaciones de trabajo, páginas web y dispositivos móviles. (Visual Studio, 2012)

19 Integrated Development Environment por sus siglas en inglés. Traducido al español como Entorno de Desarrollo Integrado.

- **NetBeans:** Es un producto libre y gratuito sin restricciones de uso, hecho principalmente para el lenguaje de programación Java. Es un proyecto de gran éxito con una gran base de usuarios y una comunidad en constante crecimiento con cerca de 100 socios en todo el mundo. Esta plataforma permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las APIs[20] de NetBeans y un archivo especial que lo identifica. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en esta plataforma pueden ser extendidas fácilmente por otros desarrolladores de software. Ofrece servicios comunes a las aplicaciones de escritorio, permitiéndole al desarrollador enfocarse en la lógica específica de su aplicación. (Netbeans, 2012)

La plataforma NetBeans proporciona una arquitectura de aplicación fiable y flexible. No añade muchos gastos para su aplicación y se puede ahorrar enorme cantidad de tiempo y trabajo. (DZone, 1997-2012)

Entre las características de la plataforma están:

1. Administración de las interfaces de usuario.
2. Administración de las configuraciones del usuario.
3. Administración del almacenamiento.
4. Administración de ventanas.
5. Framework basado en asistentes (diálogo paso a paso).

1.11 Arquitectura del sistema.

Según lEEE STD 1471-2000, la Arquitectura de Software es la organización fundamental de un sistema encarnado en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución.

Esta proporciona una visión global del sistema que se quiere implementar, permite analizar la efectividad del diseño y reducir los riesgos asociados a la construcción del software. Cada uno de los subsistemas representados en el diagrama de paquetes va a estar estructurado según el paquete Subsistema.

20 Application Programming Interface por sus siglas en inglés. Traducido al español como Interfaz de Programación de Aplicaciones.

- **Arquitectura en N capas:** La arquitectura en capas soporta un diseño basado en niveles de abstracción crecientes y proporciona una alta reutilización y modularidad del sistema. Esta facilita la localización de errores y mejora el soporte del sistema, además, ayuda a controlar y encapsular aplicaciones complejas. El mantenimiento y las mejoras de la solución son fáciles debido al bajo acoplamiento entre las capas, la alta cohesión de las capas y la habilidad de cambiar su implementación sin cambiar las interfaces. Esta arquitectura presenta como principal desventaja que puede ser difícil definir qué componentes ubicar en cada una de las capas.

1.12 Patrones de diseño.

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Son una solución a un problema de diseño y para que una solución sea considerada un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Otra es que debe ser reutilizable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias.

Los patrones de diseño pretenden:

- Proporcionar catálogos de elementos reusables en el diseño de sistemas software.
- Evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente.
- Formalizar un vocabulario común entre diseñadores.
- Estandarizar el modo en que se realiza el diseño.
- Facilitar el aprendizaje de las nuevas generaciones de diseñadores condensando conocimiento ya existente.

Asimismo, no pretenden:

- Imponer ciertas alternativas de diseño frente a otras.
- Eliminar la creatividad inherente al proceso de diseño.

No es obligatorio utilizar los patrones, solo es aconsejable en el caso de tener el mismo problema o similar que soluciona el patrón, siempre teniendo en cuenta que en un caso particular puede no ser aplicable.

Patrones GRASP[21] (General Responsability Asignment Software Paterns).

- **Experto:** Expresa simplemente la "intuición" de que los objetos hacen cosas relacionadas con la información que poseen.
- **Creador:** Su propósito fundamental es definir quién es la clase encargada de crear las instancias de otra clase.
- **Bajo acoplamiento:** Evita las dependencias entre las clases y ofrece una mayor reutilización, las responsabilidades se asignan de modo que se mantenga el bajo acoplamiento.
- **Alta cohesión:** Establece que se asignen las responsabilidades de modo que se mantenga una alta cohesión, se logra que las responsabilidades colaboren para producir un comportamiento bien definido. Una clase está en alta cohesión cuando el objeto tiene delimitadas sus responsabilidades, es decir, no se le asignan responsabilidades que no estén limitadas dentro de sus funciones. Las responsabilidades de un objeto se traducen después en métodos que realizan acciones.
- **Controlador:** Encargado de asignar a una clase la responsabilidad de administrar un evento del sistema. (unGrid - Universidad Nacional de Colombia, 2012), (GRASP, 2012)

Patrones GOF.

Gang of Four es el nombre con el que se conoce comúnmente a los autores del libro Design Patterns (ISBN 0-201-63361-2), referencia en el campo del diseño orientado a objetos. Según su propósito estos se clasifican en: creacionales, estructurales y de comportamiento. (UML y Patrones. Introducción al análisis y diseño orientado a objetos.)

Patrones creacionales.

Estos se encargan de la creación de los objetos de las clases, dentro de este grupo se encuentran: ***abstract factory*** (fábrica abstracta), ***builder*** (constructor virtual), ***factory method*** (método de fabricación), ***prototype*** (prototipo) y el ***singleton*** (instancia única).

Patrones estructurales.

Se basan en las relaciones entre clases, las combinan y forman estructuras mayores. Tratan de conseguir que los cambios en los requisitos de la aplicación no ocasionen cambios en las relaciones entre los objetos. En este grupo se encuentran los siguientes patrones: ***adapter*** (adaptador), ***bridge*** (puente), ***composite*** (objeto

[21] Traducido al español como Patrones de Software para la Asignación General de Responsabilidad.

compuesto), ***decorator*** (envoltorio), ***facade*** (fachada), ***flyweight*** (peso ligero) y el patrón ***proxy***.

Patrones de comportamiento.
Plantean la interacción y cooperación entre las clases y objetos, en este caso son: chain of responsibility (cadena de responsabilidad), command (orden), interpreter (intérprete), iterator (iterador), mediator (mediador), memento (recuerdo), observer (observador), state (estado) y template method (método plantilla).

1.13 Fundamentación de la selección de la tecnología, metodología y herramienta de desarrollo.

Para el desarrollo del Sistema de Votación Electrónica para la UCI, se realizó el estudio de las tecnologías, metodologías y herramientas de desarrollo de software. Se determinó utilizar como tecnología la plataforma Java debido a que es un producto de código abierto y multiplataforma que permite desarrollar potentes aplicaciones así como darle a las mismas un alto nivel de seguridad. Otro motivo por el cual fue elegida, se debe a su versatilidad y eficiencia, por lo que ha logrado convertirse en un recurso inestimable. Esta tecnología ha sido mejorada, ampliada y probada por una comunidad especializada de más de 6,5 millones de desarrolladores, la mayor y más activa del mundo. NetBeans fue seleccionado como IDE para el desarrollo del sistema porque la UCI se encuentra en medio de la migración al software libre y este cuenta con las condiciones necesarias para poder desarrollar la aplicación. Permite que las aplicaciones sean desarrolladas a partir de un conjunto de módulos y en este caso el sistema de votación también está estructurado por módulos, es decir, trabajar con NetBeans sería la mejor opción ya que este IDE constituye una herramienta para programadores, pensada para escribir, compilar, depurar y ejecutar programas.

El producto a obtener estará sometido a constante transformación debido al surgimiento de las nuevas tecnologías y métodos de seguridad, lo que conlleva a tener una amplia documentación, la cual se puede generar fácilmente a través de RUP. Es una metodología de desarrollo de software basada en componentes e interfaces bien definidas, y junto con UML constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. Se utilizará debido a que en todos los proyectos se debe minimizar el riesgo, garantizar la predictibilidad de los resultados y entregar software de calidad superior a tiempo, además de ser la metodología que dominan los desarrolladores. También, se selecciona por la característica de generar gran cantidad de documentación, y como el sistema a implementar es de alta criticidad fundamentalmente, entonces esta se hace necesaria para apoyar el entendimiento de los procesos y los casos de uso. El equipo

de desarrollo al emplear esta metodología sigue el consejo dado por Pressman en la quinta edición del libro Ingeniería de Software: Un enfoque práctico, el cual plantea: “Trabaja muy duro para entender lo que tienes que hacer antes de empezar. No serías capaz de desarrollar cada detalle; por más que sepas, toma el menor riesgo.” Junto a esta metodología se utilizará UML para modelar los procesos de negocio porque cuenta con varios tipos de diagramas que muestran diferentes aspectos de las entidades representadas. Como herramienta de modelado se empleará Visual Paradigm, ya que soporta UML, es multiplataforma y soporta el ciclo de vida completo del desarrollo de software, facilitando la tarea de dibujar todo tipo de diagramas. Además, se realizó el estudio del modelo de arquitectura en capas y de los patrones de diseño para posteriormente definir cuales serán empleados en la solución.

1.14 Conclusiones parciales.

En este capítulo, después de haber realizado una investigación teórica para respaldar la labor a desarrollar, puntualizando los conceptos necesarios referentes al campo de acción de manera detallada, se realiza un análisis donde se seleccionan las metodologías, herramientas y lenguajes de modelado atendiendo a los requisitos del sistema. De ahí la elección de UML como lenguaje de modelado y RUP como metodología de desarrollo. Además, se designó Visual Paradigm como herramienta CASE porque soporta todo el ciclo de desarrollo de un proyecto, Java como lenguaje de programación que unido a Netbeans como entorno de desarrollo de software son los que más se ajustan a las necesidades del sistema. Se realizó el estudio de la arquitectura y los patrones de diseño para su posterior selección de acuerdo a los requerimientos del sistema.

Capítulo 2: Características y diseño del sistema

2.1 Introducción del capítulo.

Para el buen funcionamiento del sistema, antes de comenzar a desarrollarlo es necesario comprender el entorno de trabajo donde se va a aplicar, realizar entrevistas a los clientes del negocio con el fin de conocer cómo se realizan los procesos. Luego, se especifican los requisitos funcionales y no funcionales y se determina la configuración del sistema, se realiza el modelado de los casos de uso y se elabora el diagrama de componentes a partir de la arquitectura seleccionada. Son precisamente

tales elementos los que se desarrollan en este capítulo, determinando así la propuesta del sistema a implementar.

2.2 Modelo de negocio.

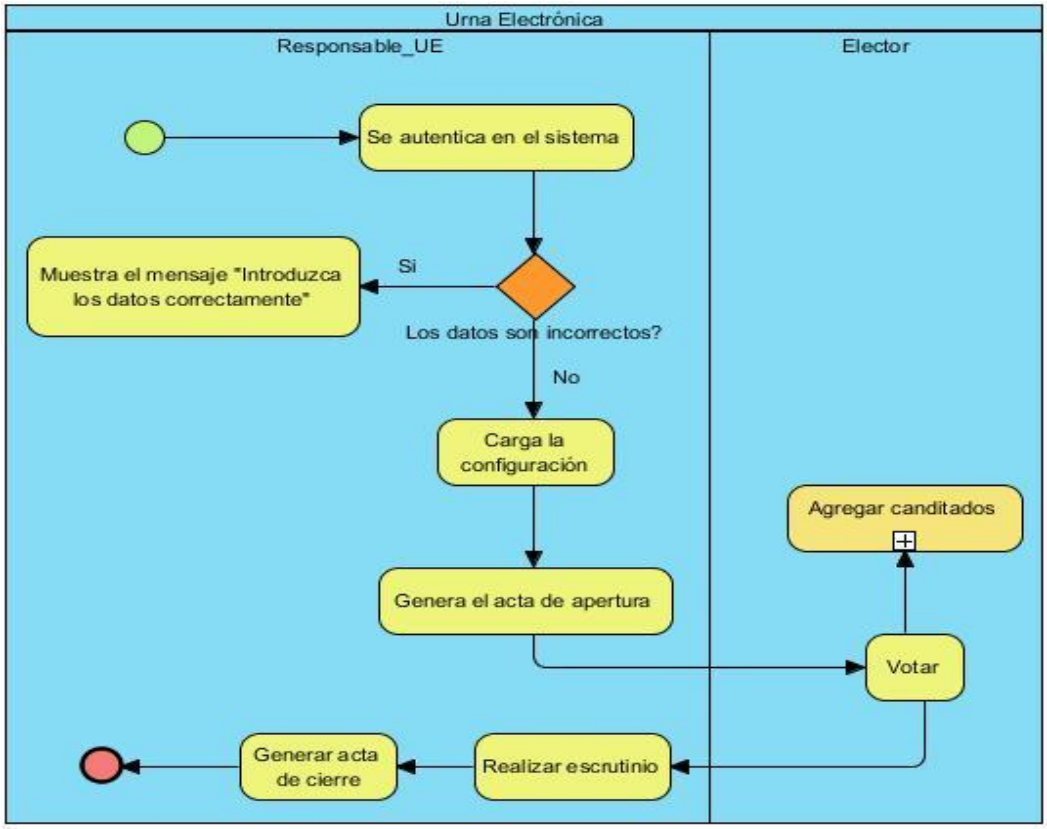


Ilustración 10 Diagrama de flujo de procesos del subsistema urna electrónica.

En la imagen anterior se muestra el procedimiento para realizar el proceso de votación, conformado por las principales actividades para realizarlo.

2.3 Especificación de requisitos.

IEEE (Standard Glossary of Software Engineering Terminology)[22] define un requisito como:

1. Condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo.
2. Condición o capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente.
3. Una representación en forma de documento de una condición o capacidad como las expresadas en 1 o en 2.

22 Es la asociación técnica y profesional, sin fines de lucro, más grande del mundo formada por profesionales de todas las disciplinas de la ingeniería.

Requisitos funcionales.

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir sin alterar la funcionalidad del producto, esto quiere decir que se mantienen invariables sin importar con qué propiedades o cualidades se relacionan. Su principal tarea consiste en la generación de especificaciones correctas que describan con claridad, sin ambigüedades, en forma consistente y compacta, el comportamiento del sistema. (Pressman, 2002)

Para el desarrollo del sistema de votación electrónica para la Universidad de las Ciencias Informáticas, de acuerdo con las actividades a realizar, se identificaron 27 requisitos funcionales (RF). La captura de requisitos se llevó a cabo a través de encuestas, revisión de documentos y entrevistas realizadas a los clientes para entender cuales actividades debía efectuar el sistema.

RF1 Autenticar usuario
RF2 Buscar persona
RF3 Tomar asistencia
RF4 Cargar configuración
RF5 Cargar listado de urnas
RF6 Activar urna para votar
RF7 Activar urna para autenticarse
RF8 Cargar listado de electores
RF9 Cargar listado de candidatos
RF10 Votar
RF11 Agregar candidatos
RF12 Realizar escrutinio
RF13 Generar acta de apertura
RF14 Generar acta de cierre
RF15 Cargar resultados intermedios
RF16 Generar resultados
RF17 Generar parte del por ciento de boletas anuladas
RF18 Configurar boleta
RF19 Configurar datos de la mesa electoral
RF20 Configurar urna
RF21 Adicionar roles
RF22 Eliminar roles
RF23 Modificar roles
RF24 Listar roles

RF25 Generar parte de la cantidad de urnas contadas

RF26 Generar parte de la cantidad de urnas por contar

RF27 Configurar Consolidación de Resultados

Requisitos no funcionales.

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Definiendo a las propiedades como características que hacen al producto atractivo, usable, rápido o confiable. (Pressman, 2002)

Se identificaron 16 requisitos no funcionales (RNF):

RNF Usabilidad

1. Interfaz sencilla, fácil de utilizar, incluso para usuarios con poca experiencia en el uso de computadoras.

RNF Rendimiento

2. El sistema debe cargar la configuración en un máximo de tiempo de 3 segundos.
3. Memoria RAM 512 Mb como mínimo.

RNF Seguridad

RNF Integridad

4. El sistema debe guardar los datos correspondientes a las votaciones y debe proporcionarlos cuando sean requeridos.

RNF Confidencialidad

5. El sistema solo puede ser manipulado por el personal autorizado.

RNF Disponibilidad

6. El sistema debe estar disponible solo el día en que se realizan las elecciones.

RNF Apariencia o interfaz externa

7. El sistema debe presentar una interfaz con colores amigables, para no desvirtuar la atención de los usuarios.

RNF Ambiente

8. Las funcionalidades del sistema deben encontrarse agrupadas de acuerdo a sus características y datos sobre los que operan.
9. Los errores que pueda lanzar el sistema deben ser presentados con mensajes personalizados.
10. Los mensajes deben identificarse según su tipo o función.
11. El sistema debe funcionar correctamente para todos los tamaños de ventanas.

12. Los textos que aparezcan deben estar normalizados con la misma tipografía.

RNF Hardware

13. Las computadoras destinadas para el uso de la aplicación deben contar con un microprocesador Pentium 4 o superior (2 GHz), memoria RAM 512, mouse, teclado, monitor y tarjeta de red.

RNF Software

14. Las computadoras destinadas para el uso de la aplicación deben tener Windows 2000/XP sp1 sp2 sp3, Windows 7 o Linux (Ubuntu, Knoppix) y la máquina virtual Java.

RNF Requisitos de Licencia

15. Utilización de la licencia GNU/GPL (Licencia Pública General de GNU).

RNF Requisitos de Soporte

16. Se permitirá la adición de nuevas funcionalidades y modificación de las ya existentes por el personal destinado a dar soporte a la aplicación.

2.4 Métricas de validación de requisitos.

El objetivo del flujo de trabajo requerimientos es establecer y mantener un acuerdo con los clientes acerca de lo que el software debe hacer.

Según el estándar IEEE 830, 1998 se considera que una especificación es de calidad si cumple con las siguientes características:

1. Especificación correcta: Un conjunto de requisitos es correcto sólo si todos los requisitos contenidos representan algo que es requerido para la construcción del sistema y no hay errores que afecten el diseño.
2. Especificación no ambigua: Un requisito es no ambiguo si y sólo si puede estar sujeto a una única interpretación.
3. Especificación completa: Si y sólo si, describe todos los requisitos relevantes para el usuario, incluyendo requisitos asociados con funcionalidad, desempeño, restricciones de diseño, atributos o interfaces externas.
4. Especificación consistente: si y sólo si no hay ningún subconjunto de requisitos descrito dentro de ella que está en conflicto con cualquier otro.
5. Especificación verificable: Se considera que una especificación es verificable si lo son cada uno de los requisitos constituyentes. Se considera que un requisito individual es verificable si existe un proceso acotado (en plazo y presupuesto) que permita determinar que el sistema construido satisface lo descrito en el propio requisito. La descripción detallada y prueba de los requisitos una vez implementados ayudan considerablemente en su verificación.

6. Especificación modificable: Si su estructura es tal que permite realizar cambios sobre los requisitos que contiene de forma sencilla, completa y consistente, manteniendo la estructura inicial del conjunto.
7. Especificación trazable: Una especificación es trazable si el origen de cada requisito individual está claro y existe algún mecanismo que permita seguir el impacto de dicho requisito a lo largo del resto de las actividades del ciclo productivo.

Métrica Estabilidad de los requisitos.

Es necesario lograr una estabilidad en los requisitos para el correcto funcionamiento de los demás flujos de trabajo. El objetivo de esta métrica es medir la estabilidad de los requisitos para asegurar su adecuación antes de pasar al próximo flujo de trabajo. Se considera que los requisitos son estables cuando no existen adiciones o supresiones en ellos que impliquen modificaciones en las funcionalidades principales de la aplicación. La estabilidad de los requisitos se calcula como[23]:

$$ETR = \left[\frac{RT - RM}{RT}\right] \times 100$$

Donde:

- ETR: valor de la estabilidad de los requerimientos.
- RT: total de requerimientos definidos.
- RM: número de requerimientos modificados, que se obtienen como la sumatoria de los requerimientos insertados, modificados y eliminados.

Esta métrica ofrece valores entre 0 y 100. El mejor valor de ETR es el más cercano a 100 porque mostrará que no se están realizando cambios sobre los requisitos, son estables y por tanto es confiable trabajar el análisis y diseño sobre ellos.

Calculando el ETR para el total de requisitos definidos que fueron 38, de ellos 22 funcionales, se tiene:

RM= 2

RT= 36

Por lo que ETR= 94, demostrando que se puede trabajar sobre estos requisitos en la siguientes fases debido a que son estables en un buen nivel de aceptación.

[23] Tomado de Revista vinculando. Autores del trabajo: Universidad de las Ciencias Informáticas, dalcantara@uci.cu Y Universidad de las Ciencias Informáticas, ehernandezl@uci.cu. http://vinculando.org/articulos/sociedad_america_latina/propuesta_guia_de_medidas_para_evaluacion_sistemas_informacion.html. [consultado 9 de junio de 2012]

2.5 Configuración del sistema.

Debido al creciente interés por mejorar la eficiencia en los sistemas electorales, en los últimos años se han estado considerando distintas vías para automatizar los diferentes procesos implicados en una elección. Estas mejoras han dado como resultado un amplio rango de propuestas de esquemas aplicados a los diferentes procesos de una elección. La mayoría de esas propuestas se han enfocado principalmente en la fase de votación de una elección, sin embargo, existen otros procesos que pueden llevarse a cabo utilizando medios electrónicos, como el registro de votantes.

Para darle solución al problema planteado, se ha decidido implementar un Sistema de Votación Electrónica que contará con cuatro módulos. La siguiente figura muestra la representación en paquetes de los diferentes subsistemas, así como sus relaciones de dependencia.

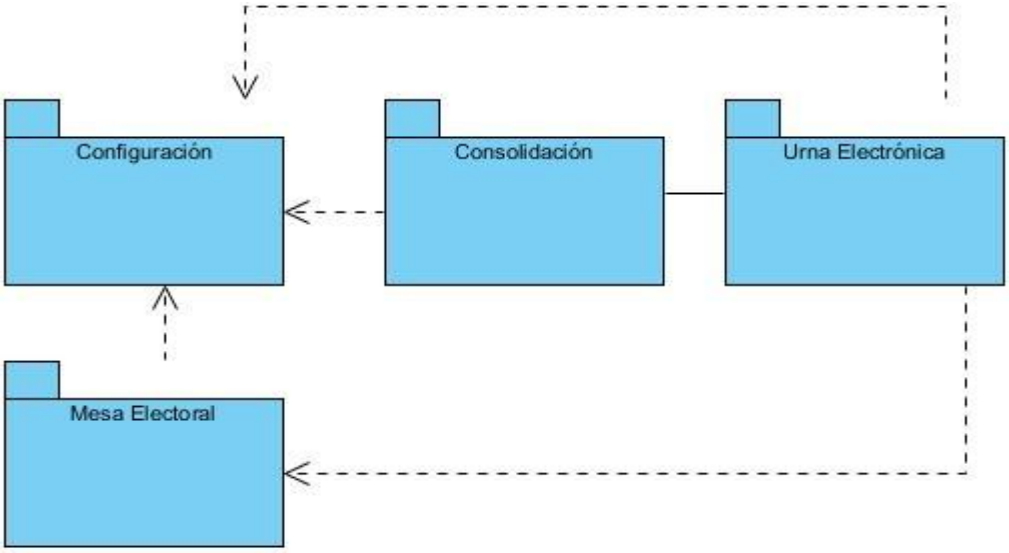


Ilustración 11 Diagrama de paquetes.

El módulo de **Configuración** que permitirá gestionar las boletas, los permisos para acceder al sistema, roles, listado de electores, candidatos, además de configurar las mesas electorales, las direcciones Protocolo de Internet (IP por sus siglas en inglés) de las urnas y también la configuración de llaves para cifrar los ficheros y las contraseñas. En este se definen los parámetros de la elección, preparación técnica y configuración de la elección. Todo esto genera una serie de archivos que serán usados por los subsistemas de consolidación, urna electrónica y mesa electoral.

El módulo **Mesa Electoral** será el encargado de gestionar el listado de electores y activar las urnas correspondientes a través de la configuración propuesta, así como generar el parte de las personas que han votado y las que faltan por votar. Es el encargado del registro de los datos de los electores a fin de constituir el censo electoral. Este recibe los datos necesarios para su funcionamiento a través de un

archivo emitido del subsistema de configuración, además de activar la urna para así poder llevar a cabo el voto.

El módulo **Urna Electrónica** generará el acta de apertura y cierre, permitirá agregar candidatos a la boleta, realizar la votación y enviar el voto. Se encargará de realizar el escrutinio y generará un fichero encriptado con los resultados de cada urna electrónica**,** los que serán enviados al módulo de consolidación de resultados. Este recibe los datos necesarios para su funcionamiento a través de un archivo emitido del subsistema de configuración, para su funcionamiento hace falta activación previa del subsistema mesa electoral y es el encargado de crear archivos con los resultados parciales de la votación, los que serán transportados hacia el subsistema de consolidación.

El módulo **Consolidación de resultados** se encargará de cargar los resultados intermedios a través de la recolección de los votos, generar reportes del por ciento de votos emitidos por cada urna para los diferentes candidatos, así como el por ciento de boletas anuladas, además de determinar y publicar los resultados. Recibe ficheros necesarios para su funcionamiento, uno para cargar los datos necesarios para su funcionamiento básico y otro para dar los resultados de las elecciones, de los subsistemas configuración y urna electrónica correspondientemente.

2.6 Modelo del sistema.

Se identificaron 6 actores del sistema con sus respectivas descripciones.

Actor	Descripción
Usuario	Se autentica en el sistema para realizar ciertas acciones de acuerdo a los niveles de acceso que posea.
Elector	Encargado de votar, teniendo la posibilidad de agregar candidatos a la boleta.
Administrador de configuración	Encargado de realizar la configuración de los diferentes subsistemas. Responsable de gestionar las boletas, permisos de acceso, roles, listado de electores, listado de candidatos, además de configurar las direcciones Protocolo de Internet (IP) de las urnas y también la configuración de llaves para cifrar los ficheros y las contraseñas.

Responsable de mesa electoral	Es el encargado del registro de los electores a fin de constituir el censo electoral, llevando el control de la cantidad de personas que han votado y las que no han votado, así como activar las urnas para que se lleve a cabo el proceso de votación.
Responsable de urna electrónica	Se autentica en el sistema para velar porque estén correctos los datos de las urnas electrónicas como el acta de apertura y cierre de las votaciones, que reportan la fecha y hora de inicio y fin de las elecciones y la cantidad de votos emitidos.
Responsable de consolidación	Encargado de cargar los resultados intermedios a través de la recolección de los votos, generar reportes del por ciento de urnas que han sido contadas y las que no han sido contadas. Es el responsable de publicar los resultados luego del escrutinio.

Diagramas de casos de uso del sistema.

Los casos de uso son una técnica de captura de requisitos que representan las funcionalidades del sistema, definidas por los requisitos funcionales. Estos guían el diseño, la implementación y prueba. A continuación se muestra el diagrama de casos de uso del sistema para cada subsistema.

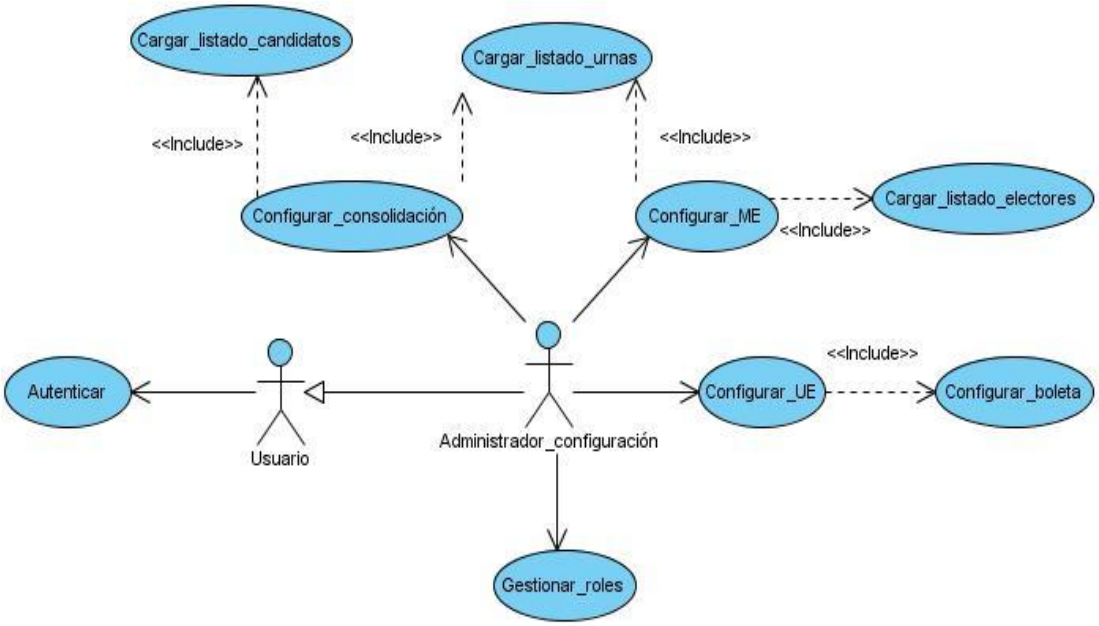


Ilustración 12 Diagrama de casos de usos del subsistema Configuración.

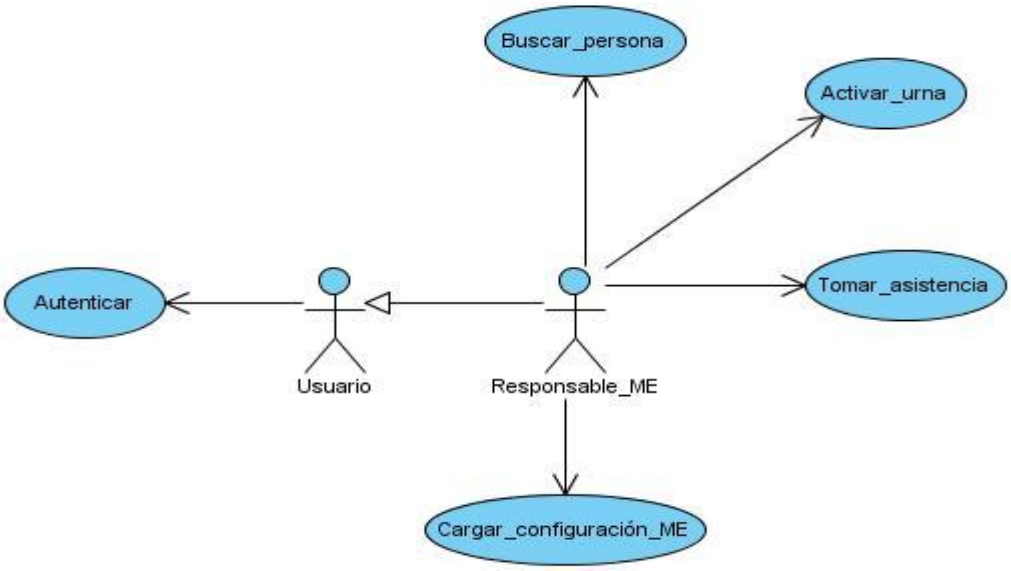


Ilustración 13 Diagrama de casos de usos del subsistema Mesa Electoral.

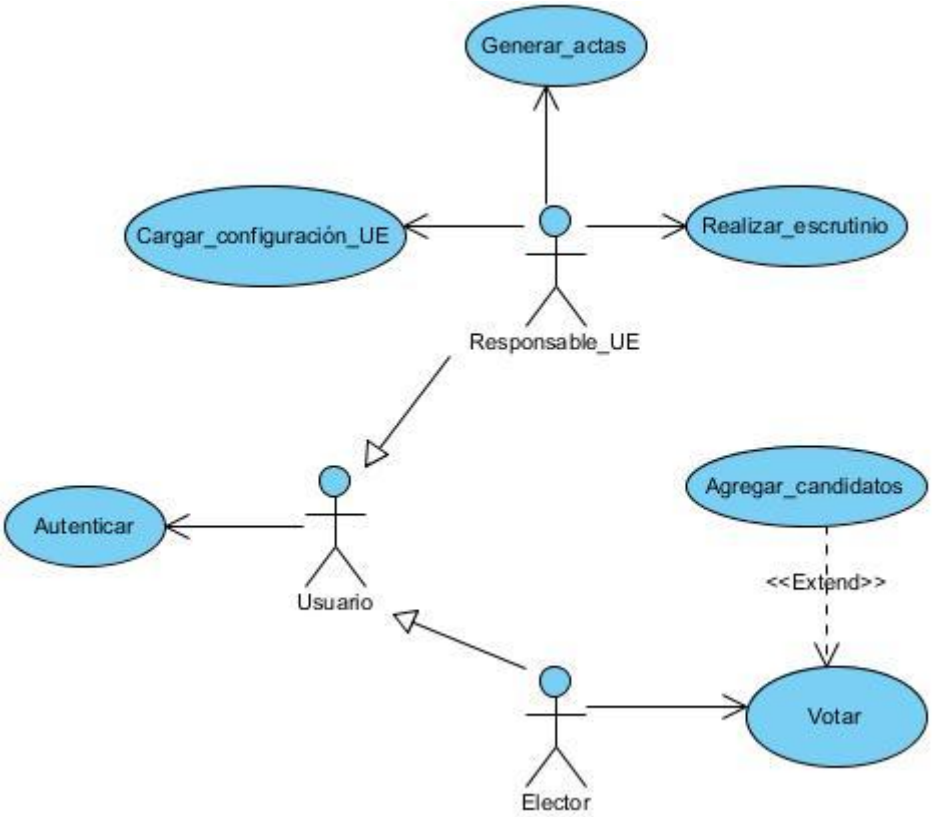


Ilustración 14 Diagrama de casos de usos del subsistema Urna Electrónica.

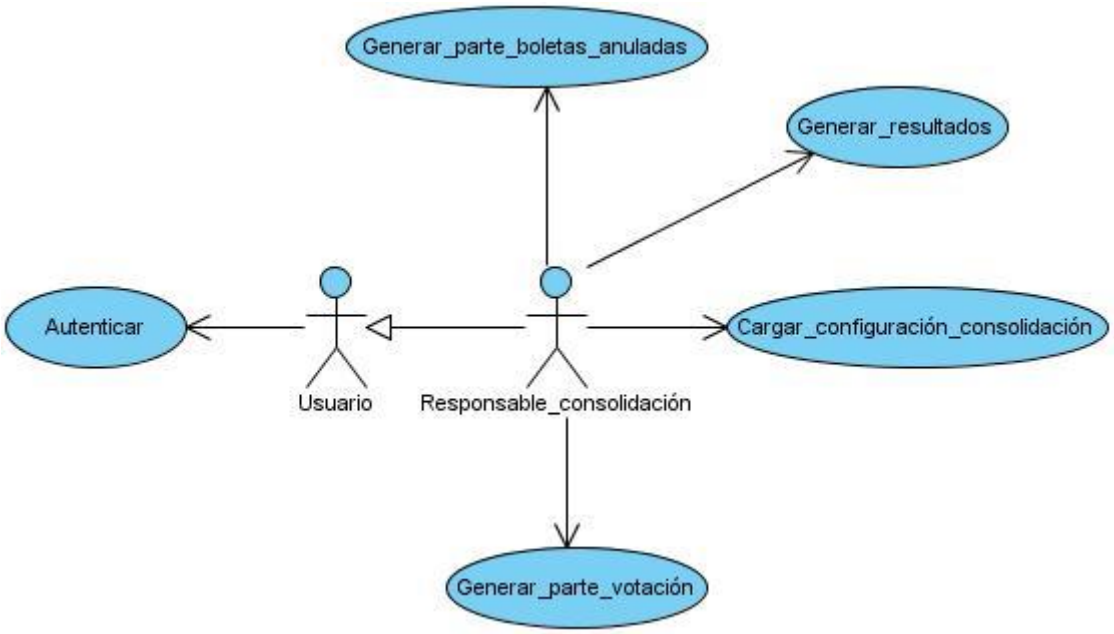


Ilustración 15 Diagrama de casos de usos del subsistema Consolidación.

2.7 Patrones de casos de uso utilizados.

Patrón: Pareja problema / solución con un nombre, que codifica (estandariza) buenos principios y sugerencias relacionadas frecuentemente con la asignación de responsabilidades.

Múltiples actores – Rol Común: Varios actores desempeñan el mismo rol en un determinado caso de uso y este rol es representado a la vez por otro actor, que contiene de forma hereditaria a los actores que comparten el rol. En el sistema se identificó este patrón para el caso de uso **Autenticar**, debido a la necesidad de que varios actores interactuaran con el sistema, ya sea como responsable de consolidación, elector, administrador de configuración u otro actor. Es por ello que heredan del actor **Usuario** para poder autenticarse en el sistema.

CRUD[24] (Crear, Leer, Actualizar y Eliminar): Presenta un caso de uso llamado "información CRUD" utilizado para gestionar la información contenida en él, es por ello que puede leerla, eliminarla, actualizarla y crearla. Este patrón es utilizado en el sistema por el **Administrador_configuración** para gestionar los roles de acceso a la aplicación.

Concordancia – Reusabilidad: Se presenta cuando es obligatorio realizar un caso de uso para que se cumpla otro. Este se manifiesta claramente en el caso donde es obligatorio realizar el caso de uso **Cargar_listado_urnas** (en el subsistema Configuración) para poder llevar a cabo los casos de uso **Configurar_consolidación** y **Configurar_ME.**

[24] (Creating, Reading, Updating and Deleting) por sus siglas en inglés.

Concordancia – Adición: Se presenta cuando para realizar un caso de uso se hace opcional la realización de otro. Este se evidencia en el caso de uso **Votar**, donde puede ser opcional o no realizar el caso de uso **Agregar_candidatos** (subsistema Urna_Electrónica).

2.8 Descripción de casos de uso.

CU1 Autenticar.

Objetivo	Acceso al sistema.
Actores	Usuario
Resumen	El caso de uso se inicia cuando se desea acceder al sistema. El usuario introduce la contraseña, el sistema comprueba que sean válidos y permite el acceso de acuerdo al rol que este ocupe.
Complejidad	Media
Prioridad	Crítico
Precondiciones	El sistema debe estar funcionando correctamente.
Postcondiciones	Acceso al sistema.

Flujo de eventos		
Flujo básico: Autenticar		
	Actor	**Sistema**
1.	El caso de uso inicia cuando el usuario desea autenticarse en el sistema.	2. Valida que los datos introducidos son correctos.
		3. Verifica que los campos obligatorios no están vacíos.
		4. Permite el acceso. • Terminando así el caso de uso.
Flujos alternos		
Flujo alterno al paso 2.		
	Actor	**Sistema**
		2.a Muestra el mensaje “Introduzca los datos correctamente”. 2.b Volver al paso 1 del flujo básico.
Flujo alterno al paso 3.		
		3.a Muestra el mensaje” No puede dejar los campos obligatorios vacíos.”

		• Volver al paso 1 del flujo básico.

CU2 Cargar configuración de ME.

Objetivo	Se realiza para poder llevar a cabo todas las acciones en la mesa electoral ya que estas son provistas por la configuración que se debe cargar.
Actores	Responsable_ME
Resumen	El caso de uso se inicia cuando el Responsable_ME desea cargar la configuración para poder realizar diferentes actividades. Al cargar la configuración propuesta se obtiene el listado de electores y el listado de urnas que serán activadas por la mesa electoral.
Complejidad	Simple
Prioridad	Crítico
Precondiciones	El usuario debe estar autenticado en el sistema.
Postcondiciones	Cargada la configuración del sistema.
Flujo de eventos	
Flujo básico: Cargar configuración de Mesa Electoral	

	Actor	**Sistema**
1.	El caso de uso inicia cuando el Responsable_ME desea cargar la configuración propuesta para la mesa electoral.	2. Lee el fichero con los datos para la mesa electoral.
		3. Descifra los datos.
		5. Muestra el fichero de datos que contiene el listado de electores y listado de urnas. • Terminando así el caso de uso.

CU3 Generar parte de votación.

Objetivo	Genera un listado con la cantidad de urnas que han sido contadas y las que faltan por contar, así como la cantidad de boletas anuladas.
Actores	Responsable_Consolidación
Resumen	El caso de uso se inicia cuando el Responsable_Consolidación desea reportar la cantidad de urnas que han sido contadas y las que faltan por contar. El sistema muestra un listado con estos datos quedando generado el parte de votación.
Complejidad	Simple
Prioridad	Secundario
Precondiciones	El actor debe estar autenticado en el sistema.
Postcondiciones	Generado el parte de personas que han votado, las que faltan por votar y el por ciento de boletas anuladas.

Flujo de eventos

Flujo básico: Generar parte de votación

	Actor	**Sistema**
1.	El caso de uso inicia cuando el Responsable_Consolidación desea generar el parte de votación.	2. Muestra el informe de la votación: • Urnas que faltan por contar. • Urnas que han sido contadas. • Terminando así el caso de uso.

CU4 Votar.

Objetivo	Demostrar su afinidad con uno o varios candidatos.
Actores	Elector
Resumen	El caso de uso se inicia cuando el Elector decide votar por determinados candidatos que van a ocupar un puesto como miembro o presidente.
Complejidad	Simple

Prioridad	Crítico	
Precondiciones	El usuario debe estar autenticado en el sistema.	
Postcondiciones	Voto emitido.	
Flujo de eventos		
Flujo básico: Votar		
	Actor	**Sistema**
1.	El caso de uso inicia cuando el Elector desea votar por determinados candidatos. • Por miembro • Por presidente	2. Muestra la boleta con los candidatos.
3.	Selecciona los candidatos y les asigna un determinado cargo que puede ser: • Miembro • Presidente	
4.	Efectúa el voto.	5. Crea la boleta.
		6. Guarda la boleta. • Terminando así el caso de uso.
Relaciones	**CU Extendidos**	**Agregar candidato**: Paso 3 del Flujo básico. Agregar candidato en el CU Votar
Flujos alternos		
Flujo alterno al paso 3.		
Actor		**Sistema**
		3.a Muestra el mensaje" Debe seleccionar solo la cantidad establecida." • Volver al paso 3 del flujo básico.

	3.b Muestra el mensaje “Debe seleccionar un presidente.” • Volver al paso 3 del flujo básico.

CU5 Configurar Urna Electrónica.

Objetivo	Definir las condiciones para el trabajo en el subsistema urna electrónica.
Actores	Administrador_configuración
Resumen	El caso de uso se inicia cuando el Administrador_configuración desea configurar la urna electrónica. Para esto carga la boleta y los datos de la mesa electoral.
Complejidad	Compleja
Prioridad	Crítico
Precondiciones	El usuario debe estar autenticado en el sistema.
Postcondiciones	Queda configurada la urna electrónica.

Flujo de eventos

Flujo básico: Configurar Urna Electrónica

	Actor	**Sistema**
1.	El caso de uso inicia cuando el Administrador_configuración desea configurar la urna electrónica.	2. Carga un fichero con el listado de candidatos.
		3. Crea un objeto donde guarda la información contenida en el fichero.
4.	Entra el IP de la mesa electoral que va a activar la urna correspondiente.	5. Guarda estos datos en un objeto.
6.	Ingresa el usuario y la contraseña.	7. Guarda los datos en el objeto.
		8. Cifra la información.
		9. Guarda la información en un

		archivo. • Terminando así el caso de uso.
Relaciones	**CU Incluidos**	Configurar Urna Electrónica. Ver CU 17 Configurar_boleta

La complejidad de los casos de uso se determinó de acuerdo al número de transacciones o eventos realizados. Una transacción es representada por la interacción entre el actor y el sistema, es decir, la acción del actor y las posibles respuestas del sistema. Ivar Jacobson, inventor del caso de uso, describe una transacción de caso de uso como un "viaje de ida y vuelta" que va desde el usuario hasta el sistema para luego volver al usuario.[25] (IBM - developerWoks, 2012)

2.9 Modelo de diseño.

El modelo de diseño es un modelo de objetos que describe la realización de los casos de uso del sistema y sirve como abstracción del modelo de implementación y del código fuente. Es utilizado como entrada fundamental para las actividades de los flujos de trabajo de implementación y pruebas, puede ser calificado como un artefacto integral ya que abarca todas las clases del diseño, subsistemas, paquetes, colaboraciones y las relaciones entre ellos.

Patrones GOF empleados en la solución.

Un patrón de diseño es una descripción de clases y objetos que se comunican entre sí, adaptada para resolver un problema general de diseño en un contexto particular[26].

En el desarrollo de la aplicación se utilizó el patrón estructural Facade o Fachada ya que brinda una interfaz unificada que representa a todo un subsistema, de ahí que la solución cada subsistema o capa tiene su propia fachada, reduciendo las dependencias entre los subsistemas. La idea principal es la de ocultar todo lo posible la complejidad de un subsistema, el conjunto de clases o componentes que lo forman, de forma que solo se ofrezca un punto de entrada al sistema encubierto por la fachada. También se empleó el Singleton.

[25] Mohagheghi, Parastoo, Bente Anda et Reidar Conradi, "Effort estimation of Use Cases for incremental large-scale software development",International Conference on Software Engineering (ICSE), 2005, pp. 303-31.

[26] Tomado de Universidad de Valladolid. Departamento de Informática. Félix Prieto 2008.

- Fachada: Provee de una interfaz unificada simple para acceder a una interfaz o grupo de interfaces de un subsistema.

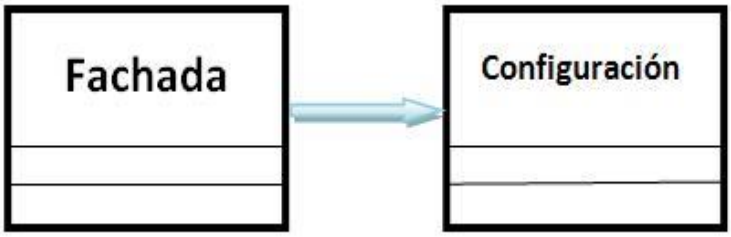


Ilustración 16 Patrón Fachada.

Ejemplo de implementación:

```
public CBoleta(LinkedList<String> e, LinkedList<String> c, String p) {
        for(int i = 0 ; i < e.size(); i++)
       {          candidatos.add(e.get(i).getBytes());       }
          for(int i = 0 ; i < c.size(); i++)
       {          candidatos.add(c.get(i).getBytes());       }
       presidente = p.getBytes();
   }
```

- Singleton: Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia.

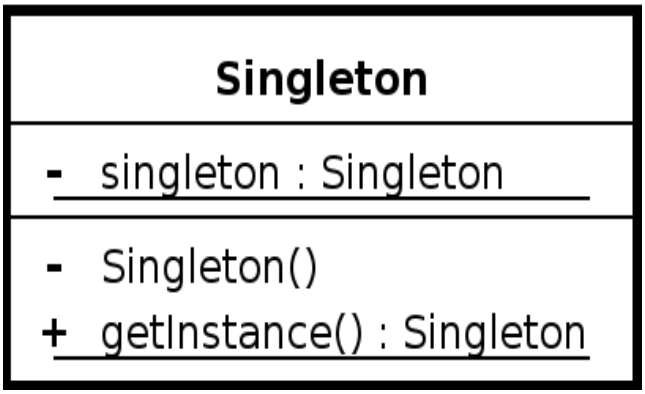


Ilustración 17 Patrón Singleton.

Ejemplo de implementación:

```
public class Conexion {
   private static Conexion INSTANCE = new Conexion();
   // El constructor privado no permite que se genere un constructor por defecto
   // (con mismo modificador de acceso que la definición de la clase)
   private Conexion() {}
   public static Conexion getInstance() {
```

```
        return INSTANCE;
    }
}
```

2.10 Descripción de la arquitectura del sistema.

Para el sistema a desarrollar se utilizará Arquitectura en Capas. Este descompone el sistema en un conjunto de capas independientes y ordenadas jerárquicamente, de modo que cada capa proporciona servicios a la capa inmediata superior y se sirve de las prestaciones que le brinda la inmediata inferior. Con su uso proporciona una amplia reutilización ya que brinda la posibilidad de valerse de un mismo nivel en varias aplicaciones, admite optimizaciones, refinamientos y estandarización.

La arquitectura en n capas implementa el patrón arquitectónico Modelo-Vista-Controlador (MVC), dicho patrón divide la aplicación en tres capas o niveles de abstracción las cuales son: Modelo, Vista y Controlador. Se puede decir que la principal característica de la aplicación de MVC es aislar la vista del modelo.

Flujo de control del patrón MVC:

1. El usuario realiza una acción en la interfaz.
2. El controlador trata el evento de entrada.
    - Previamente se ha registrado.
3. El controlador notifica al modelo la acción del usuario, lo que puede implicar un cambio del estado del modelo.
4. Se genera una nueva vista. La vista toma los datos del Modelo.
    - El modelo no tiene conocimiento directo de la vista.
6. La interfaz de usuario espera otra interacción del usuario, que comenzará otro nuevo ciclo.

En la siguiente imagen se muestra la estructura de los subsistemas usando este patrón. El paquete DAO tendrá los componentes del modelo, la Lógica de Negocio será la encargada del control y la Presentación será la vista.

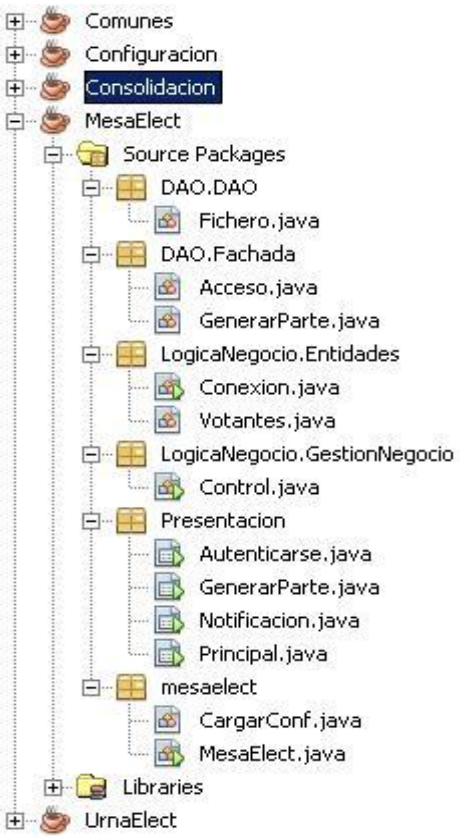


Ilustración 18 Sistema de archivos.

La figura 17 muestra la representación arquitectónica a utilizar para el Sistema de Votación Electrónica para la Universidad de las Ciencias Informáticas.

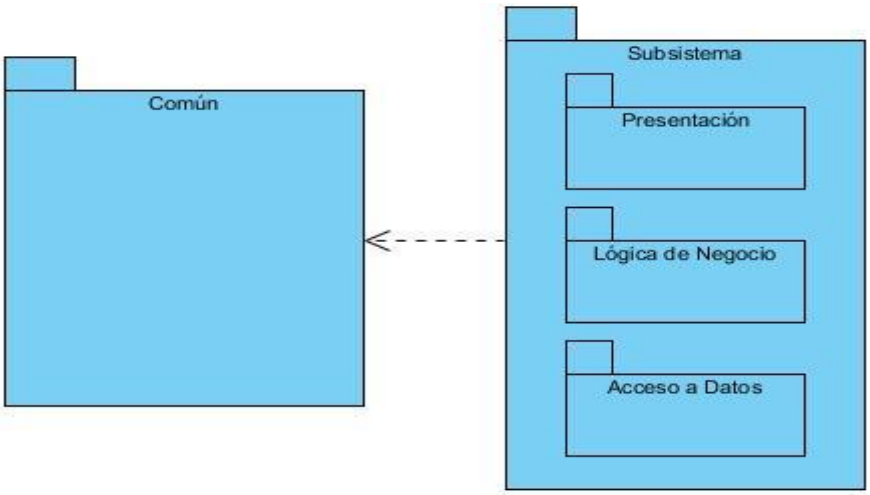


Ilustración 19 Estructura de los paquetes por cada capa.

Descripción de la Vista de Arquitectura

- **Capa Presentación:** Esta capa se encarga del manejo de las interfaces necesarias para que los usuarios interactúen con el sistema.

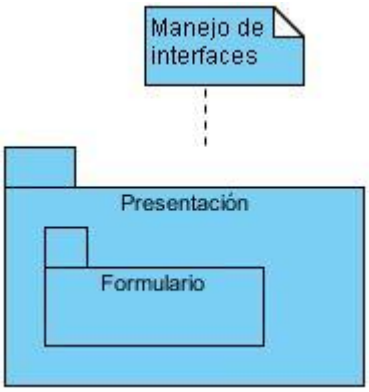


Ilustración 20 Capa Presentación.

- ✓ Paquete Formulario: Es el paquete que contiene las clases de los formularios que el usuario usará para comunicarse con el sistema.

- **Capa Lógica de Negocio:** Esta capa encapsula toda la lógica del negocio. Está compuesta por clases encargadas de recibir las peticiones de la capa de presentación y procesar la lógica de negocio relacionada con cada una de las entidades del negocio.

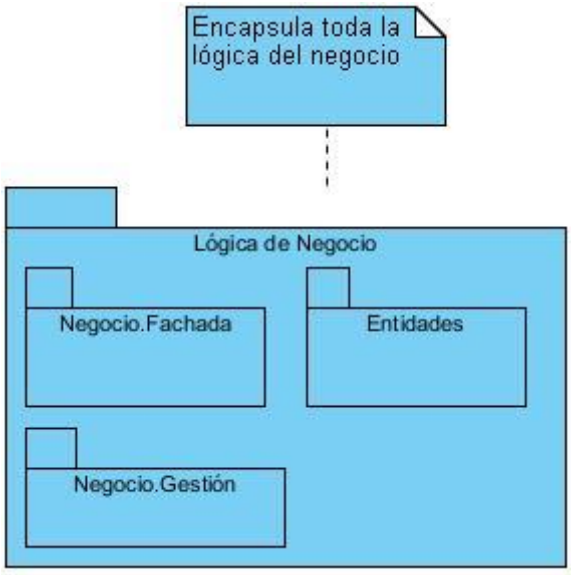


Ilustración 21 Capa Lógica de Negocio.

- ✓ Paquete Negocio. Fachada: Este paquete contiene las interfaces e implementaciones de los servicios de las fachadas, que proporcionan una interfaz unificada de alto nivel para un conjunto de clases gestoras del subsistema.
- ✓ Paquete Negocio. Gestión: Este paquete contiene las interfaces e implementaciones las clases gestoras que dan solución a los requisitos funcionales, haciendo uso de las clases entidades.

- ✓ Paquete Entidades: Este paquete contiene la implementación de todas las clases entidades que modelan el domino del subsistema.

- **Capa Acceso a Datos:** Esta capa se encarga de las consultas y persistencia de los datos que maneja el sistema.

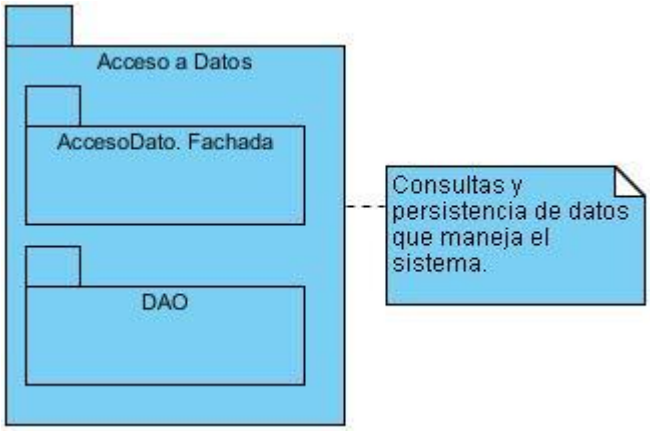


Ilustración 22 Capa Acceso a Datos.

- ✓ Paquete AccesoDato. Fachada: Este paquete contiene las interfaces e implementaciones de los servicios de las fachadas, que proporcionan una interfaz unificada de alto nivel para un conjunto de clases encargadas del acceso a los datos del subsistema.
- ✓ Paquete DAO: Este paquete contiene las clases encargadas del acceso a los datos que el sistema necesita y que se encuentran guardados en ficheros.

- **Capa Común:** Esta capa se encarga de contener una serie de paquetes que son utilizados en las capas anteriores como complemento a las funciones que ellas realizan.

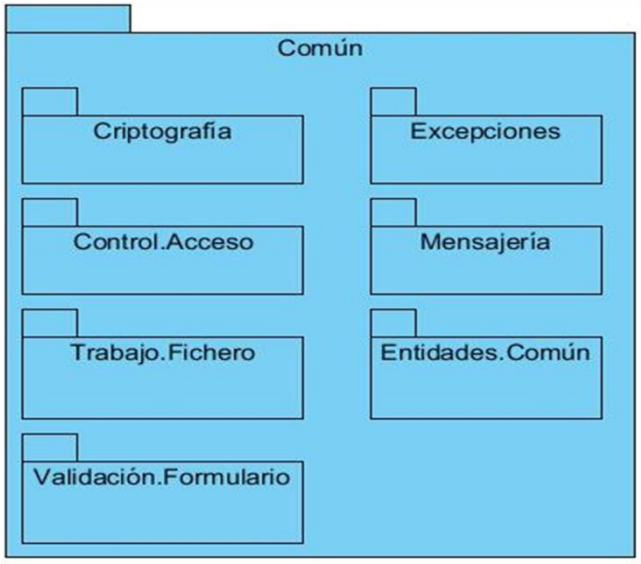


Ilustración 23 Capa Común.

- ✓ **Paquete Criptografía:** Este paquete contiene las interfaces e implementaciones de todas las clases encargadas de proporcionar los métodos que permiten codificar los datos.
- ✓ **Paquete Control Acceso:** Este paquete contiene las interfaces e implementaciones de todas las clases encargadas de gestionar el acceso al sistema por parte de los usuarios.
- ✓ **Paquete Trabajo Fichero:** Este paquete contiene las interfaces e implementaciones de todas las clases encargadas de proporcionar los métodos que permiten guardar y recuperar los datos de los ficheros.
- ✓ **Paquete Validación Formulario:** Este paquete contiene las interfaces e implementaciones de todas las clases encargadas de validar los datos entrados por los formularios.
- ✓ **Paquete Excepciones:** Este paquete contiene las interfaces e implementaciones de todas las clases encargadas de brindar un sencillo y adecuado trabajo con las excepciones del sistema.
- ✓ **Paquete Mensajería:** Este paquete contiene las interfaces e implementaciones de todas las clases encargadas de brindar un servicio de mensajería al sistema.
- ✓ **Paquete Entidades Común:** Este paquete contiene la implementación de todas las clases entidades que modelan el domino del sistema y son utilizadas en distintos subsistemas.

2.11 Diagrama de clases del diseño

Los diagramas de clases de la fase del diseño son una representación estática de los elementos de la solución del sistema, mostrando la especificación para las clases software de una aplicación, presentando como notación de los elementos que lo forman (clases e Interfaces) y las relaciones que existen entre los mismos (asociaciones).

Las clases se representan mediante una caja subdividida en tres partes, en la parte superior se muestra el nombre, en el medio los atributos y debajo las operaciones.

A continuación se muestra el diagrama de clases del subsistema urna electrónica.

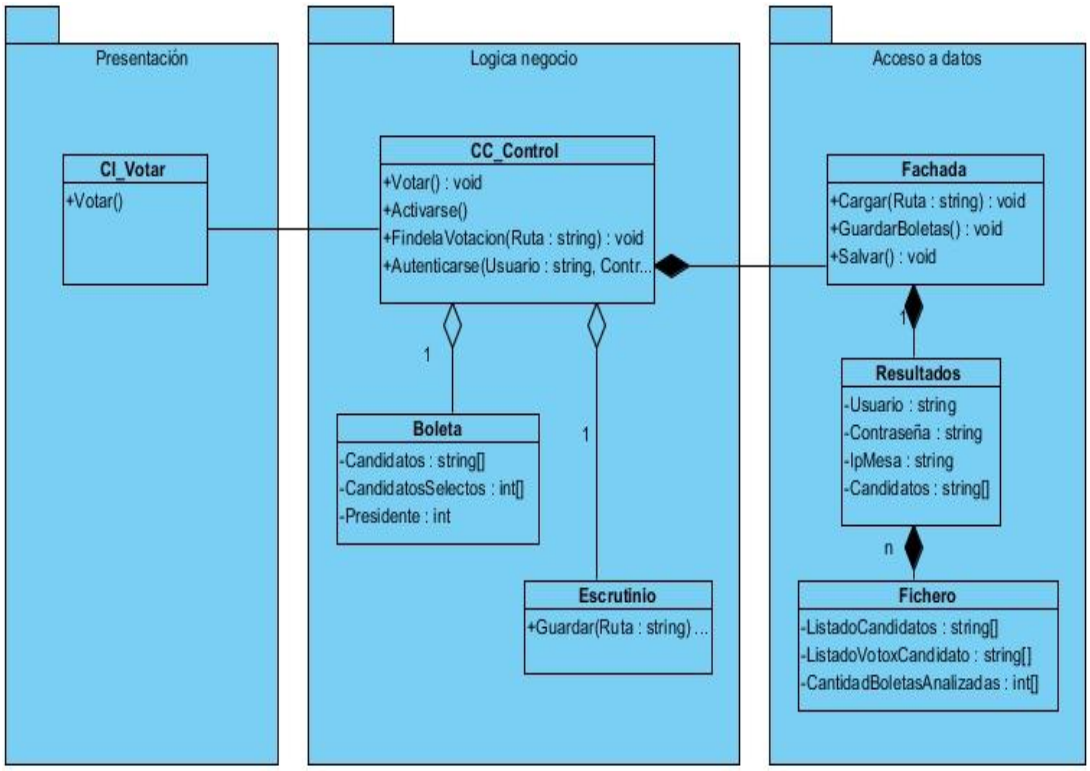


Ilustración 24 Diagrama de clases del subsistema urna electrónica.

2.12 Conclusiones parciales.

En este capítulo se identificaron los requisitos funcionales y no funcionales como parte del flujo de trabajo de requerimientos, se obtuvo la especificación de requisitos como artefacto fundamental para poder avanzar en la solución del sistema. Estos se obtuvieron a partir de entrevistas con el cliente y de igual forma fueron validados a partir de una métrica que arrojó un resultado satisfactorio, ya que al no existir una gran variación en los requisitos ni un elevado número de peticiones de cambios por parte del cliente, se pudo determinar que los mismos son estables con un valor de 94 en un rango de 1-100 para un total de 38 requisitos de los cuales solo 2 fueron modificados luego de su captura. Además, se realizó la descripción de los casos de uso con el fin de lograr el entendimiento de las actividades a realizar por cada actor, así como conocer la respuesta del sistema ante determinada situación. Se definió una arquitectura para el sistema de acuerdo a las características del mismo, permitiendo agrupar los casos de uso de acuerdo a su funcionamiento y a través de esta se seleccionaron los patrones de diseño a utilizar en la implementación de la solución. Se le dio cumplimiento a la fase de análisis y diseño ya que los requisitos fueron transformados en un diseño que a su vez quedó adaptado al entorno de implementación.

Capítulo 3: Implementación y prueba

3.1 Introducción del capítulo.

La implementación se empieza con el resultado del análisis y diseño para construir el sistema en términos de componentes. Los objetivos de la implementación son: definir la organización del código en términos de subsistemas de implementación organizados en capas, implementar los elementos de diseño en términos de elementos de implementación, hacer pruebas de unidad a los componentes desarrollados e integrar los resultados producidos por implementadores individuales o equipos en un sistema ejecutable. Estas actividades tendrán lugar en el presente capítulo y además se validará la solución a partir de métricas, pruebas de caja negra y caja blanca.

Las pruebas al software se realizan con el objetivo de encontrar y documentar los defectos en la calidad del software, aconsejar en base a la calidad determinada, validar y probar las hipótesis hechas en el diseño y la especificación de requisitos mediante una demostración concreta, validar que el producto trabaja de acuerdo a lo que fue diseñado y validar que los requisitos están correctamente implementados. (Pressman, 2005)

3.2 Modelo de implementación.

Describe la implementación de los elementos del diseño en términos de componentes como ficheros de código fuente y ejecutables. Detalla la organización de los componentes de acuerdo a los mecanismos de estructuración, el lenguaje o lenguajes de programación utilizados, así como por sus dependencias. (Stevens, Pooley, & Wesley, 2002)

Un componente se corresponde con una o varias clases, interfaces o colaboraciones. La característica básica de un componente es que debe definir una abstracción precisa con una interfaz bien definida, y permitiendo reemplazar fácilmente los componentes más viejos con otros más nuevos y compatibles.

No es necesario realizar un diagrama de componentes que incluya todos los componentes del sistema, ya que este puede ser dividido en varios diagramas para llegar a un mejor entendimiento. Para el Sistema de Votación Electrónica a desarrollar se procederá a realizar un diagrama de componentes para cada subsistema de implementación.

Diagrama de componentes.

Un diagrama de componentes muestra las organizaciones y dependencias lógicas entre componentes de software, sean estos componentes de código fuente, binarios o ejecutables, por este motivo en muchos aspectos se puede considerar que un diagrama de componentes es un diagrama de clases a gran escala. Estos se representan como un grafo de componentes software unidos por medio de relaciones de dependencia. Puede mostrar también que un componente de software contiene una interfaz, es decir, la soporta. (Dueñas, 2009)

La siguiente figura representa el diagrama de componentes, organizado de acuerdo a la arquitectura seleccionada.

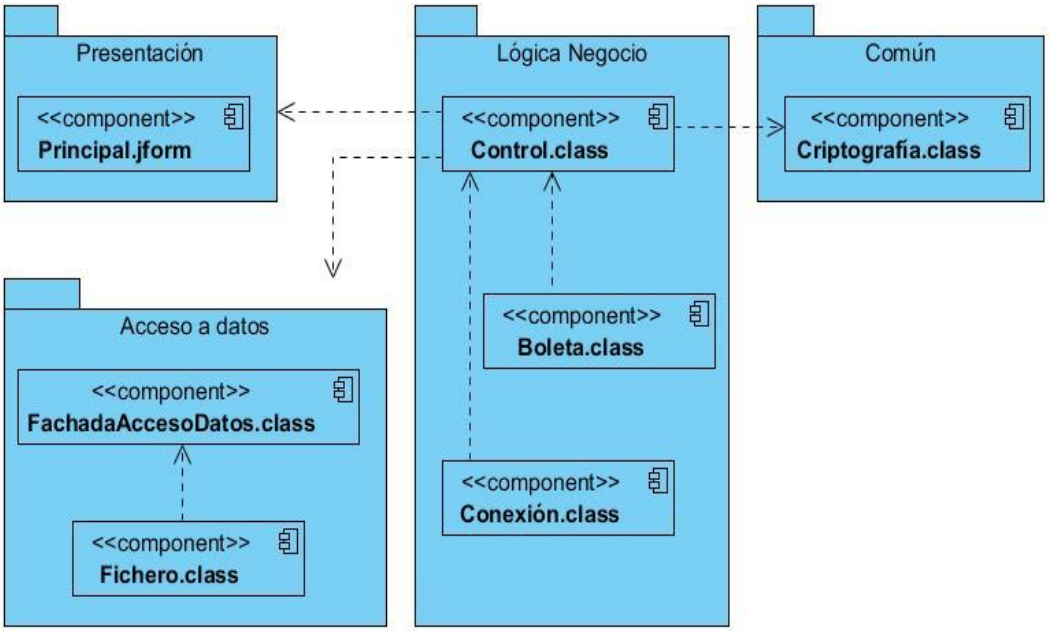


Ilustración 25 Diagrama de componentes.

3.3 Modelo de despliegue.

Un diagrama de despliegue es un grafo de nodos unidos por conexiones de comunicación. Este muestra las relaciones físicas entre los componentes de hardware y software en el sistema final, es decir, la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes de software, procesos y objetos que se ejecutan en ellos. Pueden mostrar también qué interfaces implementan y qué objetos contienen. (Dueñas, 2009)

Los diagramas de despliegue son fundamentalmente diagramas de clases que se ocupan de modelar los nodos de un sistema. Esta vista cubre principalmente la distribución, entrega e instalación de las partes que configuran un sistema físico.

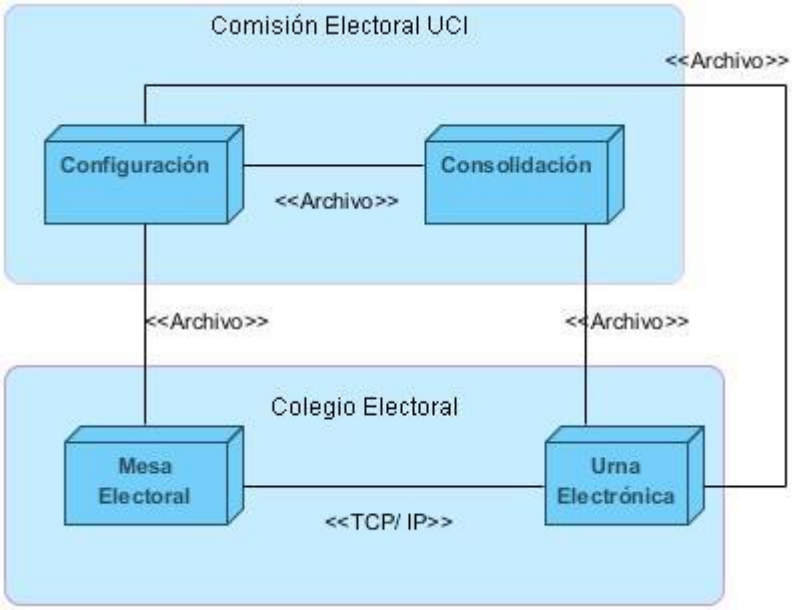


Ilustración 26 Modelo de despliegue.

3.4 Medidas de seguridad del sistema.

- Elección en un local, a cargo del responsable de la mesa en cuestión, con acceso restringido.
- Aplicación de escritorio, para una mayor fiabilidad, debido a la no transferencia de datos por la red.
- Requiere autenticarse para poder acceder a los datos del sistema, así se valida el acceso de personal autorizado para el uso de los subsistemas.
- Cifrado de los archivos antes de ser almacenados, brinda mayor seguridad y otorga un alto grado de confiabilidad en la información.
- Uso del Algoritmo Asimétrico RSA.

Sobre RSA

RSA es un sistema criptográfico de clave pública. Es el primer y más utilizado algoritmo de este tipo y válido tanto para cifrar como para firmar digitalmente.

La seguridad de este algoritmo radica en el problema de la factorización de números enteros. Los mensajes enviados se representan mediante números, y el funcionamiento se basa en el producto, conocido, de dos números primos grandes elegidos al azar y mantenidos en secreto.

Es este algoritmo el usado para operaciones que requieren conexiones seguras**,** tales como transacciones electrónicas e incluso accesos a webmail.

Usando RSA

"*algoritmo / modo / relleno"*

RSA/ECB/PKCS5Padding

Para usar el algoritmo RSA.

Modo ECB (ElectronicCode Book). Divide los mensajes en bloques y cada uno de ellos es cifrado por separado utilizando la misma clave K.

Padding PKCS5Padding. Relleno para el bloque de byte.

- Se comprueban los IP para tener mayor control de la seguridad, la urna electoral es activada a través de una conexión TCP/IP, la urna antes de activarse comprueba que los datos recibidos le llegan desde la mesa electoral correcta.
- Una vez los subsistemas sean ejecutados estos no permitirán cambios en los archivos de configuración asociados a ellos.

3.5 Métricas de software.

Las métricas son la maduración de una disciplina, que, según Pressman van a ayudar a la (1) evaluación de los modelos de análisis y de diseño, (2) en donde proporcionarán una indicación de la complejidad de diseños procedimentales y de código fuente, y (3) ayudaran en el diseño de pruebas más efectivas. Es por eso que propone un proceso de medición, el cual se puede caracterizar por cinco actividades: (desarrolloweb.com, 2012)

(1) Formulación: La obtención de medidas y métricas del software apropiadas para la representación de software en cuestión.
(2) Colección: El mecanismo empleado para acumular datos necesarios para obtener las métricas formuladas.
(3) Análisis: El cálculo de las métricas y la aplicación de herramientas matemáticas.
(4) Interpretación: La evaluación de los resultados de las métricas en un esfuerzo por conseguir una visión interna de la calidad de la representación.
(5) Realimentación: Recomendaciones obtenidas de la interpretación de métricas técnicas trasmitidas al equipo de software.

Métrica Tamaño operacional de clase (TOC).

- **TOC:** Está dado por el número de métodos asignados a una clase.
- **Responsabilidad:** Un aumento del TOC implica un aumento de la responsabilidad asignada a la clase.
- **Complejidad de implementación:** Un aumento del TOC implica un aumento de la complejidad de implementación de la clase.
- **Reutilización:** Un aumento del TOC implica una disminución del grado de reutilización de la clase.

Los valores grandes para la métrica TOC indican que la clase debe tener bastante responsabilidad, lo que propicia un bajo nivel de reutilización de la clase y complicará la implementación y las pruebas. En general, operaciones y atributos heredados o públicos deben ser ponderados con mayor importancia cuando se determina el tamaño de clase, pues las operaciones y atributos privados, permiten la especialización.

También se pueden calcular los promedios para el número de atributos y operaciones de cada clase. Cuanto más pequeño sea el valor promedio para el tamaño, será más viable para que las clases dentro del sistema puedan reutilizarse. Durante la realización de este trabajo se decidió aplicar la métrica para el tamaño de clases en función de sus operaciones. Luego de aplicar dicha métrica a las clases de la solución la misma arrojó los siguientes resultados:

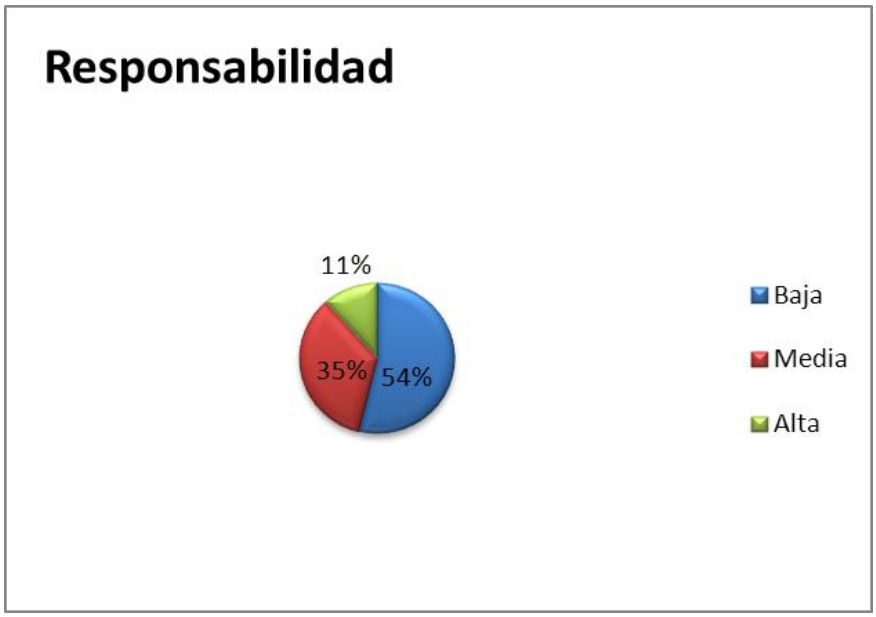


Ilustración 27 Incidencia de los resultados de la evaluación de la métrica TOC en el atributo Responsabilidad.

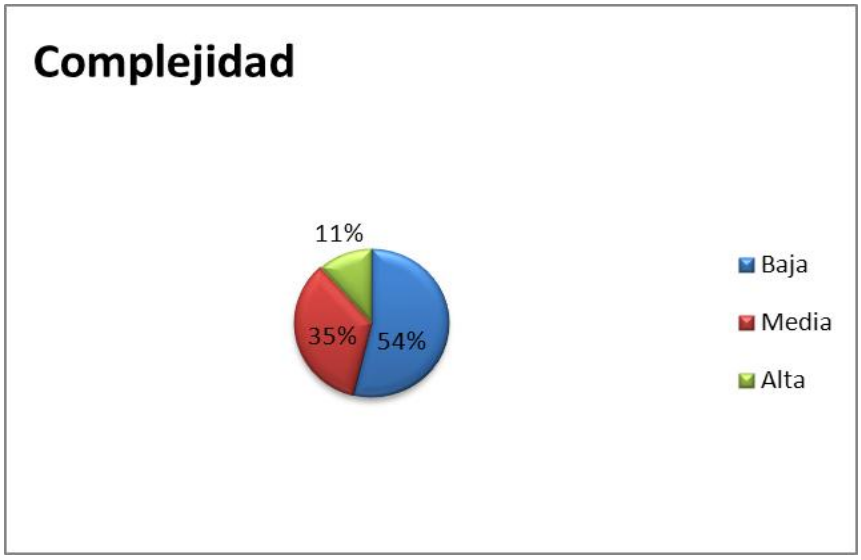


Ilustración 28 Incidencia de los resultados de la evaluación de la métrica TOC en el atributo Complejidad.

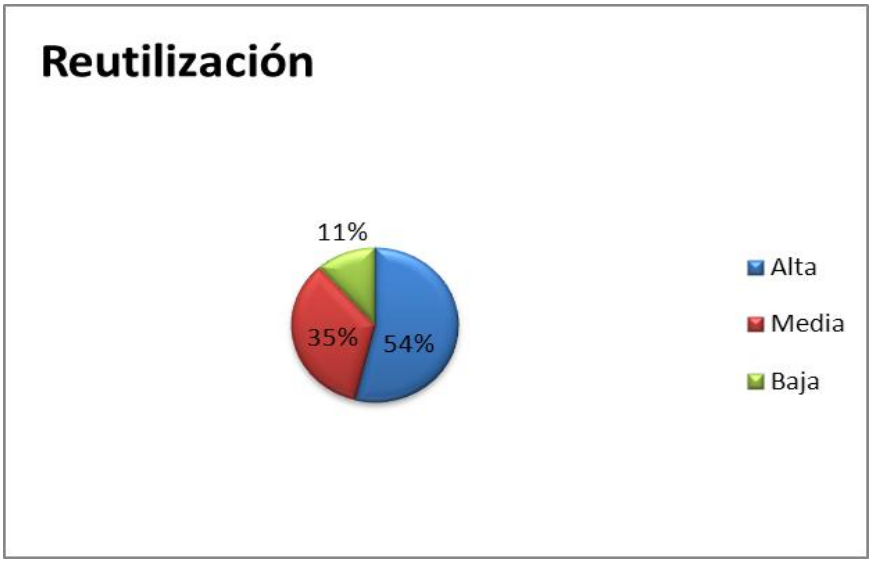


Ilustración 29 Incidencia de los resultados de la evaluación de la métrica TOC en el atributo Reutilización.

Haciendo un análisis de los resultados obtenidos en la evaluación del instrumento de medición de la métrica TOC, se puede concluir que el diseño de los diferentes sub-sistemas tienen una buena calidad teniendo en cuenta que el 87,5 % de las clases incluidas en este sistema posee menos cantidad de operaciones que la mitad del valor máximo registrado en las mediciones que fue de 21 procedimientos. Esto influye de manera positiva en el hecho de que predomine una responsabilidad baja de las clases en un 54%(ilustración 27). Además el 54% de las clases poseen evaluaciones positivas en los atributos de calidad complejidad de implementación (ilustración 28) y reutilización (ilustración 29), lo cual garantiza una solidez en el diseño del sistema

puesto que habrá gran reutilización de las clases y su implementación no resulta complicada.

Métrica relación entre clases (RC).

- **RC:** Esta dado por el número de relaciones de uso de una clase con otra.
- **Acoplamiento:** Un aumento del RC implica un aumento del Acoplamiento de la clase.
- **Complejidad de mantenimiento:** Un aumento del RC implica un aumento de la complejidad del mantenimiento de la clase.
- **Reutilización:** Un aumento del RC implica una disminución en el grado de reutilización de la clase.
- **Cantidad de pruebas:** Un aumento del RC implica un aumento de la cantidad de pruebas de unidad necesarias para probar una clase.

El resultado de la aplicación de la métrica **RC** está dado por el número de relaciones de uso de una clase con otras. La aplicación de dicha métrica permite evaluar atributos como el Acoplamiento, la Complejidad de mantenimiento, la Reutilización y la Cantidad de pruebas. De forma tal que mientras mayor sea las relaciones de uso entre clases mayor será el Acoplamiento, la Complejidad de Mantenimiento y la Cantidad de pruebas, mientras que su Reutilización disminuye. Después de aplicar dicha métrica se obtuvieron los siguientes resultados:

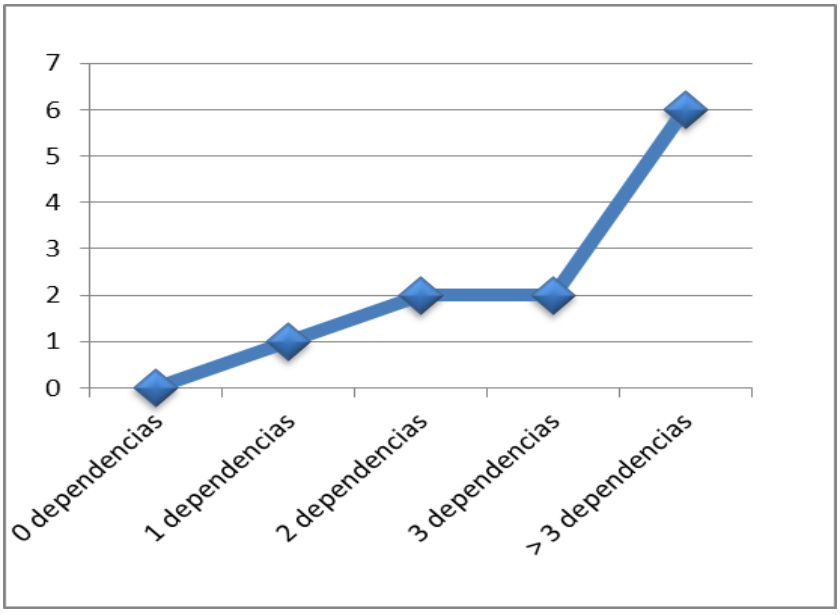


Ilustración 30 Representación de la cantidad de relaciones de uso en cada clase.

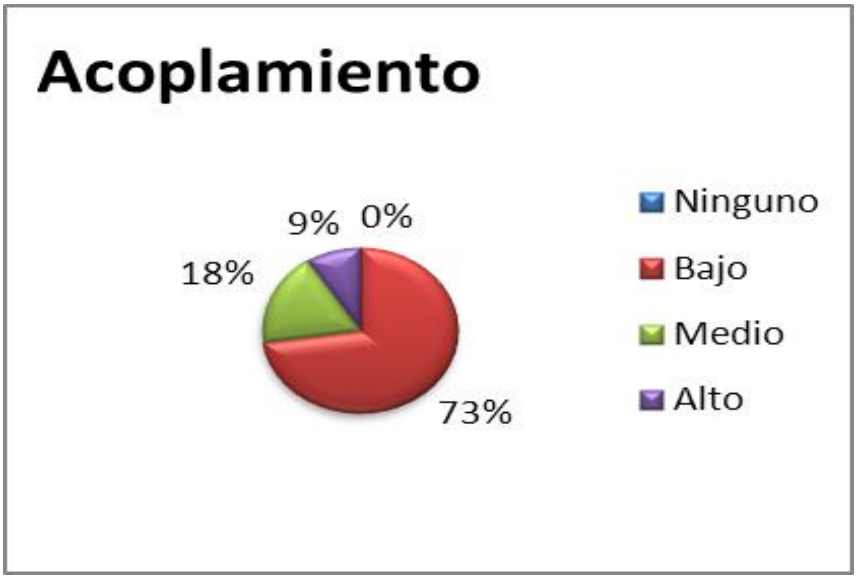


Ilustración 31 Incidencia de los resultados de la evaluación de la métrica RC en el atributo Acoplamiento.

Ilustración 32 Incidencia de los resultados de la evaluación de la métrica RC en el atributo Complejidad de Mantenimiento.

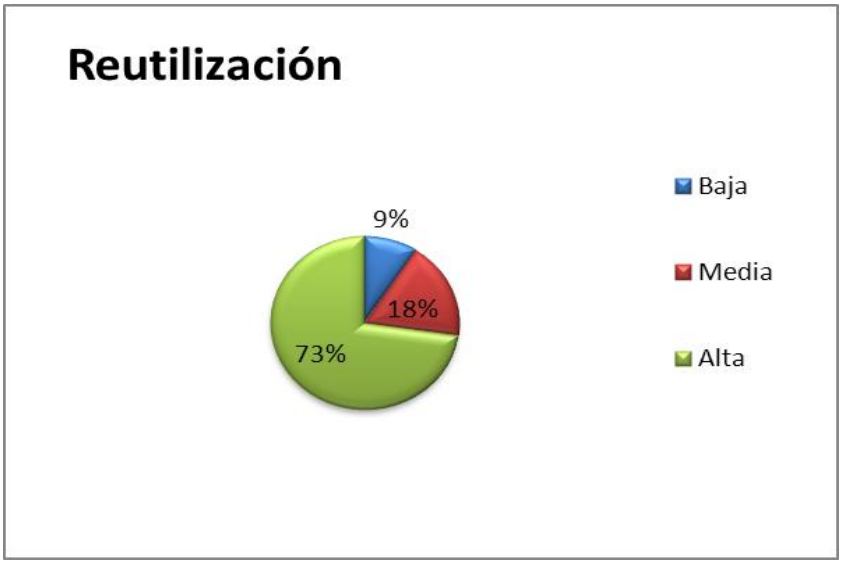


Ilustración 33 . Incidencia de los resultados de la evaluación de la métrica RC en el atributo Reutilización.

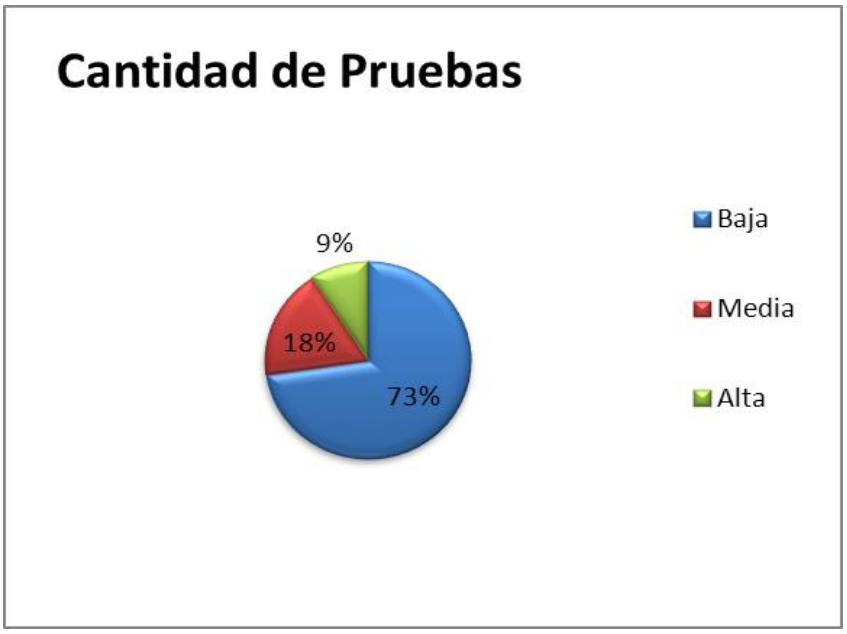


Ilustración 34 Incidencia de los resultados de la evaluación de la métrica RC en el atributo Cantidad de Pruebas.

Haciendo un análisis de los resultados obtenidos en la evaluación del instrumento de medición de la métrica RC, se puede concluir que el diseño diferentes sub-sistemas tiene una buena calidad teniendo en cuenta que el 80% de las clases incluidas en este subsistema posee 3 o menos dependencias de otras clases. Esto favorece el hecho que al ocurrir un cambio de alguna de las clases, la afectación en las restantes sea de poca importancia. El 73% de las clases posee bajos índices en cuanto a Acoplamiento (ilustración 31), este es un factor es muy importante porque posibilita que se realicen modificaciones en el sistema teniendo poco impacto en el mismo y que entre las clases exista un bajo acoplamiento. Por su parte los atributos de calidad Complejidad

de mantenimiento (ilustración 32), Cantidad de pruebas (ilustración 34) y Reutilización (ilustración 33) se comportan satisfactoriamente en un 73% de las clases, teniendo 5 o menos dependencias de otras clases, fomentando el rehúso de las mismas y facilitando el mantenimiento del sistema así como las pruebas necesarias para la comprobación del software.

Estas métricas brindan la posibilidad medir de forma cuantitativa la calidad de los atributos internos del software, permitiendo al ingeniero evaluar la calidad durante el desarrollo del sistema.

3.6 Modelo de pruebas.

El proceso de pruebas de software no va orientado a demostrar la ausencia de fallos en el software, sino más bien a todo lo contrario: encontrar cuantos fallos existan, por escondidos que se encuentren o difícilmente reproducibles que sean. Por tanto, una definición acertada acerca de lo que son las pruebas de software podría enunciarse de la siguiente forma:

“Pruebas de software es el proceso de ejecutar un programa con la intención de encontrar fallos." (Tuya & Universidad de Oviedo, 2008)

Pruebas de caja blanca o pruebas estructurales.

Se denomina caja blanca a un tipo de prueba de software que se realiza sobre las funciones internas de un módulo. Esta consiste específicamente en diseñar los casos de prueba atendiendo al comportamiento interno y la estructura del programa, examinándose la lógica interna sin considerar los aspectos de rendimiento.

Algunas técnicas de prueba de caja blanca son:

- **Prueba de condición:** Es un método de diseño de casos de prueba que ejercita las condiciones lógicas contenidas en el módulo de un programa.
- **Prueba de flujo de datos:** Se selecciona caminos de prueba de un programa de acuerdo con la ubicación de las definiciones y los usos de las variables del programa.
- **Prueba de bucles:** Es una técnica de prueba de caja blanca que se centra exclusivamente en la validez de las construcciones de bucles.
- **Prueba del camino básico:** Esta técnica permite obtener una medida de la complejidad lógica de un diseño y usar la misma como guía para la definición de un conjunto básico. La idea es derivar casos de prueba a partir de un conjunto dado de caminos independientes por los cuales puede circula el flujo de control. Para obtener dicho conjunto de caminos independientes se construye el grafo de flujo asociado y se calcula su complejidad ciclomática

- **Pruebas unitarias:** Estas tienen como objetivo verificar la funcionalidad y estructura de cada componente individualmente una vez que ha sido codificado.

La prueba de unidad centra el proceso de verificación en la menor unidad del diseño del software: el componente software o módulo. (Pressman R. , 2002)

Este es un proceso para probar los subprogramas, las subrutinas, los procedimientos individuales o las clases en un programa. Es decir, es mejor probar primero los bloques más pequeños del programa, que inicialmente probar el software en su totalidad. Las motivaciones para hacer esto son tres. Primera, las pruebas de unidad son una manera de manejar los elementos de prueba combinados, puesto que se centra la atención inicialmente en unidades más pequeñas del programa. En segundo lugar, la prueba de una unidad facilita la tarea de eliminar errores (el proceso de establecer claramente y de corregir un error descubierto), puesto que, cuando se encuentra un error, se sabe que existe en un módulo particular. Finalmente, las pruebas de unidad introducen paralelismo en el proceso de pruebas del software presentándose la oportunidad de probar los múltiples módulos simultáneamente.

Se necesita dos tipos de información al diseñar los casos de prueba para una prueba de unidad: la especificación para el módulo y el código fuente del módulo. La especificación define típicamente los parámetros de entrada y de salida del módulo y su función.

Para la realización de estas pruebas se uso el Junit 4.0, el cual es una potente herramienta que se asocia al Netbeans para permitir la ejecución de este proceso.

Se tomaron algunas clases de diferentes módulos, las cuales fueron sometidas a estas pruebas.

A continuación se muestran las pruebas realizadas al subsistema Mesa Electoral, específicamente a la clase Control, que cuenta con 7 métodos.

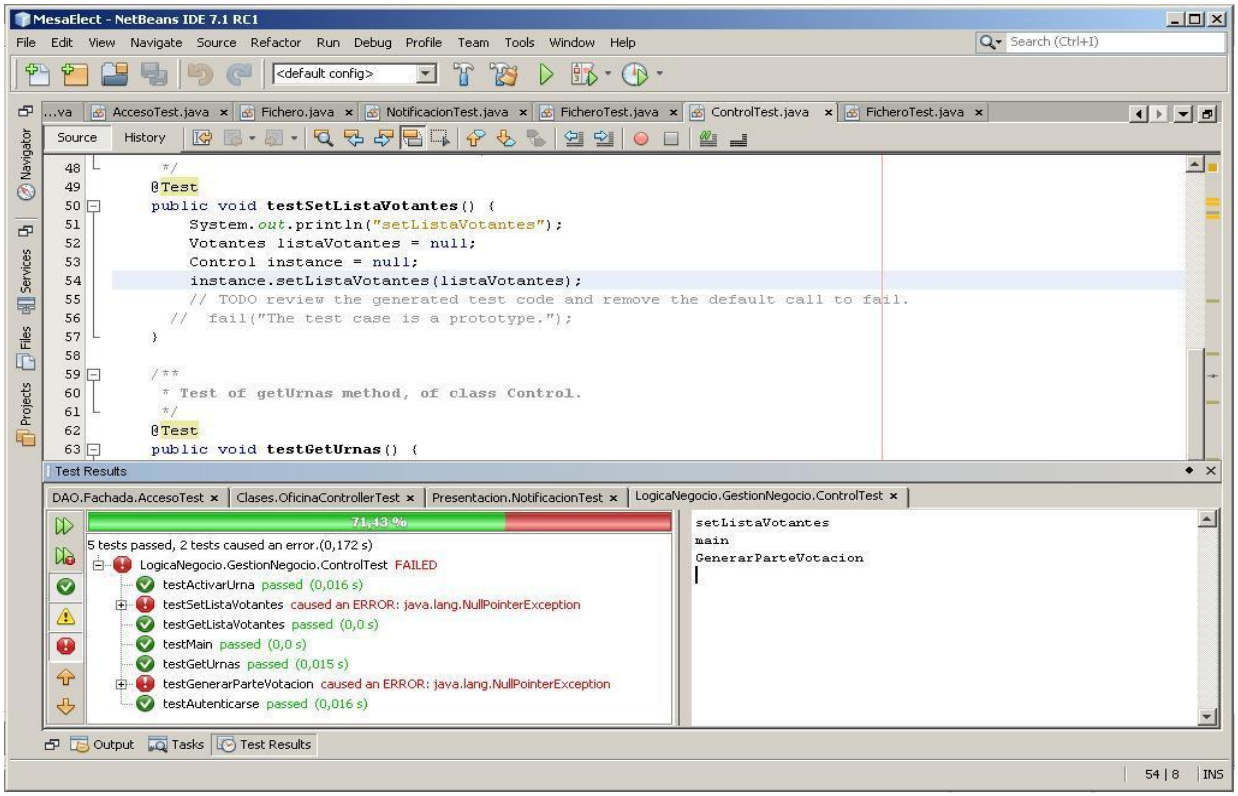


Ilustración 35 Primera iteración de prueba unidad sobre la clase Control.[27]

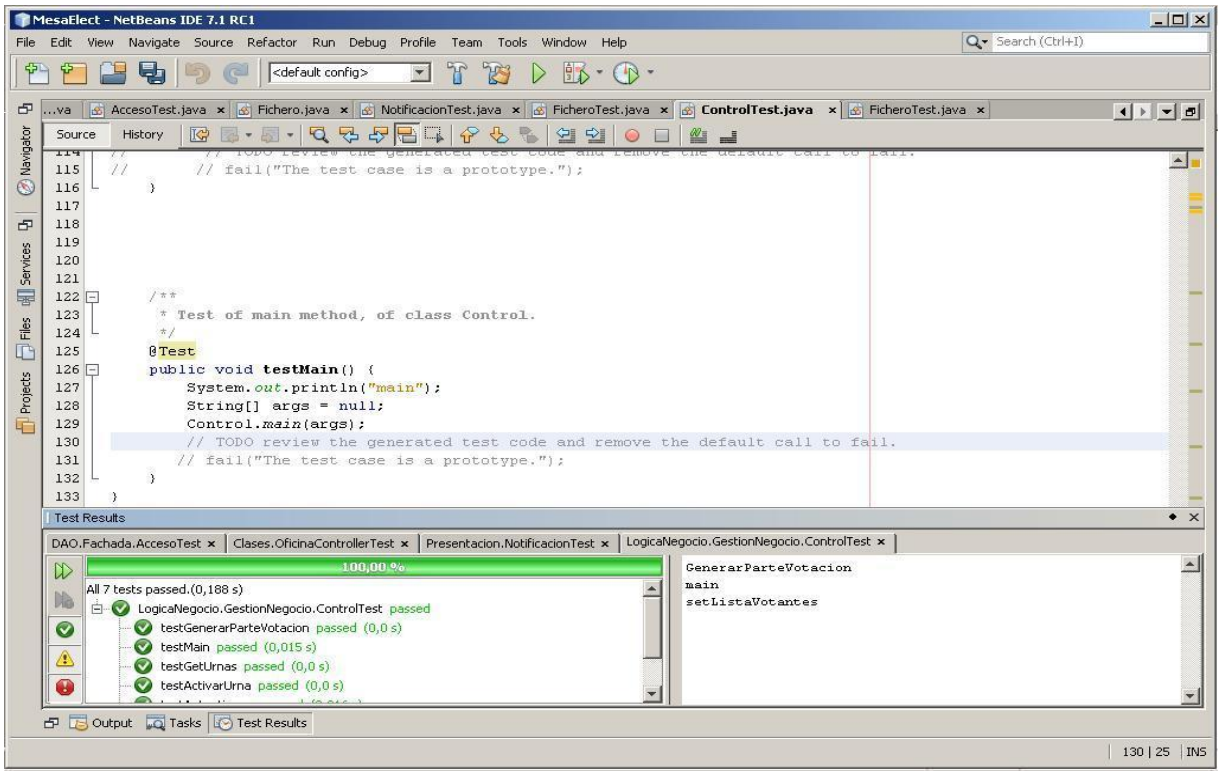


Ilustración 36 Segunda iteración de prueba unidad sobre la clase Control.

[27] En la figura se muestra la primera iteración realizada sobre la clase Control, la cual muestra la existencia de errores en 2 de sus métodos, esto podría ser por parámetros incorrectos o porque el resultado no es el esperado.

En esta otra iteración se mitigaron todos los errores existentes quedando comprobado el buen funcionamiento de la clase Control.

Los resultados obtenidos a través de la realización de los métodos de prueba expuestos con anterioridad como parte de las pruebas internas realizadas a los subsistemas, fueron satisfactorios desde el punto de vista interno y funcional, atendiendo al correcto comportamiento del mismo ante diferentes situaciones. Esto se evidencia en la última iteración donde queda comprobado que las funciones de esta clase están al 100%. Esta prueba da buen grado de confiabilidad debido a que en ella se comprueba el valor de los datos así como el flujo de cada procedimiento.

A continuación se muestra el resultado de las pruebas realizadas a otras clases del sistema.

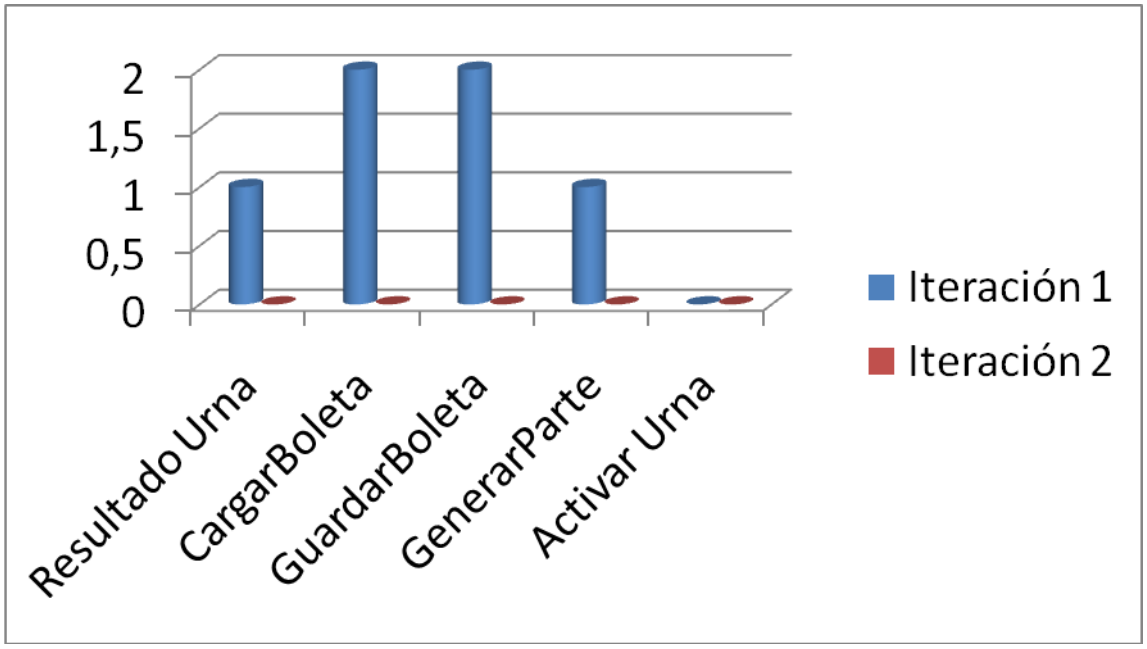


Ilustración 37 Pruebas de unidad realizada a una porción de las clases del sistema.

En la imagen mostrada, se evidencian 2 iteraciones de pruebas donde algunas clases poseían 2 errores como máximo y en la última iteración todos estos fueron erradicados.

Pruebas de caja negra.

Las pruebas de caja negra también conocidas como pruebas funcionales, pruebas de caja opaca, pruebas de entrada/salida o pruebas inducidas por los datos, son las que no toman en cuenta el código, el que lo prueba no sabe cómo está estructurado por dentro el programa o bien no necesita saber nada de programación, solo debe saber cuáles pueden ser las posibles entradas sin necesidad de entender cómo se deben

obtener las salidas, donde se trata de encontrar errores en la interfaz mientras se está usando, el cómo luce, se maneja, etc. (Tuya & Universidad de Oviedo, 2008)

Para realizar estas pruebas se necesitan las entradas y los parámetros, no se tiene exactamente un modelo para realizarlo pero un ejemplo seria en el que si una entrada es booleana, pues solo puede ser verdadero o falso.

Algunas ventajas de este tipo de pruebas radican en que la prueba termina siendo imparcial por que el que diseñó el software y el que lo prueba son independientes de el, y el probador no necesita conocimientos de programación. Además, las pruebas se realizan desde un punto de vista de usuario y no como programador, tomando en cuenta todas las funciones de cómo luce y su usabilidad por así decirlo. Estas pruebas tienen la desventaja de que se puede tener gran cantidad de datos pero al final se termina encontrando un error interno donde menos se espera.

Casos de prueba.

Un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces. (Pressman S. R., 2005)

Los casos de prueba de la caja negra pretenden demostrar que las funciones del sistema son operativas; que las entradas son aceptadas de forma adecuada y que las salidas correspondientes son las correctas. Los errores esperados durante la realización de las pruebas de caja negra al sistema informático se agrupan en las siguientes categorías:

- Funciones incorrectas o ausentes.
- Errores de interfaz.

En la prueba de la caja negra, los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta.

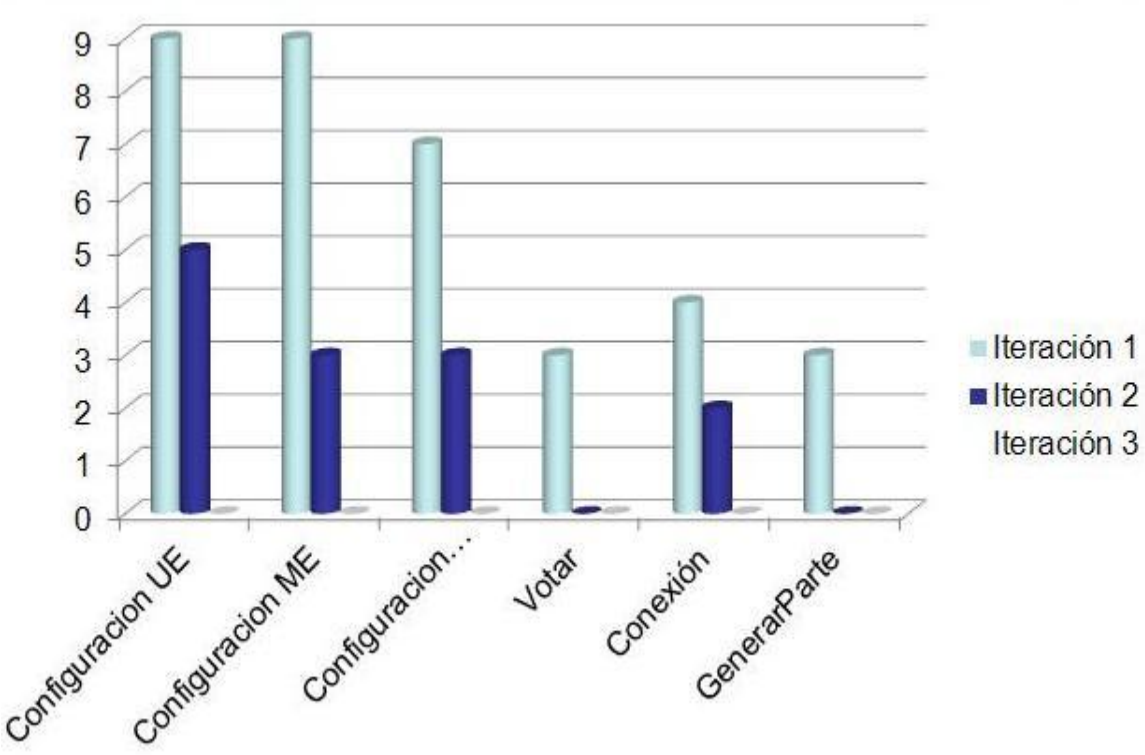
Nombre del requisito	**Descripción general**	**Escenarios de pruebas**	**Flujo del escenario**
Configurar Urna	El sistema debe permitir introducir los datos correspondientes para el funcionamiento de una urna	EP 1.1: Configurar urnas introduciendo datos válidos.	– Se introducen los datos de la configuración de la urna correctamente. – Se presiona el botón **Aceptar**. – Se muestra un mensaje de información. – Se presiona el botón

			Aceptar.
		EP 1.2: Configurar urna introduciendo datos válidos presionando el botón **Aceptar.**	– Se introducen los datos de la configuración de la urna correctamente. – Se presiona el botón **Siguiente**. – Se muestra un mensaje de información. – Se presiona el botón **Aceptar**.
		EP 1.3: Configurar urna introduciendo datos inválidos.	– Se introducen los datos inválidos de la configuración de la urna. – Se presiona el botón **Aceptar**. – Se muestra un mensaje informando del error.
		EP 1.4: Configurar urna dejando campos vacíos.	– Se introducen los datos dejando algún campo en blanco. – Se presiona el botón **Aceptar**. – Se muestra un mensaje informando del error.
		EP 1.5: Cancelar.	– Se introducen o no los datos de la notificación de configuración de la urna Se presiona el botón **Cancelar**.

No	Nombre de campo	Tipo	Válido	Inválido	Inválido

1	Ip urnas	Campo de texto	Números	Letras y caracteres especiales	Vacío
2	Hora Apertura	Campo de texto	Números	Letras y caracteres especiales	Vacío.
3	Hora Cierre	Campo de texto	Números	Letras y caracteres especiales	Vacío.
4	Candidatos	Campo de texto	Letras	Caracteres especiales y números	Vacío.

Se realizaron 3 iteraciones las cuales arrojaron los siguientes resultados:



En la primera iteración se detectaron varios errores ortográficos y los resultados esperados fueron incorrectos, pero disminuyeron en la segunda iteración y fueron completamente erradicados en la tercera y última iteración.

Pruebas de Aceptación.

Estas pruebas las realiza el cliente. Son básicamente pruebas funcionales, sobre el sistema completo y buscan una cobertura de la especificación de requisitos. Estas pruebas no se realizan durante el desarrollo, pues sería impresentable de cara al cliente, si no, una vez pasadas todas las pruebas de integración por parte del desarrollador.

La experiencia muestra que aún después del más cuidadoso proceso de pruebas por parte del desarrollador, quedan una serie de errores que sólo aparecen cuando el cliente se pone a usarlo.

Muchos desarrolladores ejercitan unas técnicas denominadas "pruebas alfa" y "pruebas beta". Las pruebas alfa consisten en invitar al cliente a que venga al entorno de desarrollo a probar el sistema. Se trabaja en un entorno controlado y el cliente siempre tiene un experto a mano para ayudarle a usar el sistema y para analizar los resultados.

3.7 Conclusiones parciales.

Durante la fase de construcción el producto se mueve desde la línea de base arquitectónica a un sistema completo, suficiente para la transición a la comunidad de usuarios. La línea de base arquitectónica que crece para convertirse en el sistema completo como el diseño, se refina en el código.

En este capítulo se comprobó el funcionamiento del sistema a través de las pruebas de unidad las cuales arrojaron resultados satisfactorios, aplicación de métricas de software y pruebas de interfaz. Además, se realizó el modelo de implementación donde se confeccionó el diagrama de despliegue y el de componentes. En resumen, se muestra como se estructuró el sistema para poder llevar a cabo las elecciones. Este fue probado obteniendo resultados satisfactorios. Además, se expuso una serie de medidas de seguridad que fueron implementadas para la no alteración y corrupción de los datos, garantizando así un resultado fiable en el proceso eleccionario, razón muy cuestionada en este tipo de proceso por su relevancia y repercusión futura.

Conclusiones

Como resultado de la investigación realizada se dio cumplimiento a los objetivos propuestos, por lo que se pueden plantear las siguientes conclusiones:

1. A través del estudio del estado del arte de los sistemas de votación electrónica, se pudo escoger el adecuado para ser utilizado en la UCI en las votaciones de la FEU.
2. Se realizó una adecuada selección de las tecnologías, metodologías y herramientas a partir de las necesidades existentes.
3. Se realizó el diseño e implementación de la aplicación donde se definió la arquitectura y se obtuvo una versión funcional del producto.
4. Se le realizaron diferentes pruebas de validación al sistema para comprobar que cumplía con los requisitos propuestos.

Recomendaciones

Que se le de uso al sistema desarrollado, en aras de contribuir al ahorro de los recursos materiales y humanos empleados en el proceso de votación.

Que se perfeccione el sistema de manera que pueda ser utilizado en cualquier institución.

Darle seguimiento y soporte al producto en términos de seguridad, ya que la implementada hasta el momento pudiera ser arcaica en años posteriores, por lo que se debe trabajar en la mejora continua del producto.

Bibliografía

1. *About.com,Inventors.* (2012). Recuperado el 17 de mayo de 2012, de http://inventors.about.com/library/weekly/aa111300b.htm
2. *ACE Project Escrutinio de Votos Uso de Tecnología,World Book 1999, CD-ROM,internet, IBM y Federal Electoral Commission*. (1999). Recuperado el 19 de noviembre de 2011, de http://www.fec.gov/elections.html
3. *aceproject.org / Sistemas de Votación Mecánicos y Electrónicos.* (2012). Recuperado el 19 de enero de 2012, de http://aceproject.org/main/espanol/et/et60.htm
4. *Biblioteca Universidad Complutense Madrid*. (2012). Recuperado el 19 de mayo de 2012, de http://eprints.ucm.es/10023/1/RSumoza-Proyecto-Fin-Master.pdf.
5. *bpm.com.* (2012). Recuperado el 1 de abril de 2012, de http://www.bpm.com/
6. Chaum, D. (febrero1981). *Untraceable electronic mail, return addresses, and digital pseudonyms.* Communications of the ACM.
7. *ciudadaccesible.cl / Voto asistido para personas con discapacidad*. (2012). Recuperado el 13 de febrero de 2012, de http://www.ciudadaccesible.cl/index.php?option=com_content&view=article&id=145:ley
8. *Clikear.com.* (s.f.). Recuperado el 28 de febrero de 2012, de 2012: http://www.clikear.com/manuales/csharp/c10.aspx
9. *Computer Science and Engineering.* (2012). Recuperado el 20 de febrero de 2012, de http://www.cse.sc.edu/~jimdavis/Tools/rational_rose.htm
10. *Definicion.de*. (s.f.). Recuperado el 29 de 01 de 2012, de Definicion.de: http://definicion.de
11. *Definicion.de*. (2008-2012). Recuperado el 29 de 01 de 2012, de Definicion.de: http://deficicion.de
12. *desarrolloweb.com*. (2012). Recuperado el 14 de junio de 2012, de http://www.desarrolloweb.com/articulos-copyleft/articulo-metricas-de-software.html
13. *dofactory.com.* (2001-2012). Recuperado el 20 de mayo de 2012, de http://www.dofactory.com/Patterns/Patterns.aspx
14. Dueñas, C. P. (2009). *El modelo dinámico y de implementación.* Recuperado el 11 de junio de 2012,

http://www.elai.upm.es:8009/spain/Asignaturas/InfoInd/apuntesAOOD/cap5UMLDinamicoImpl.pdf

15. *DZone.* (1997-2012). Recuperado el 14 de abril de 2012, de http://netbeans.dzone.com/
16. *Extreme Programming.* (1999-2009). Recuperado el 26 de enero de 2012, de http://www.extremeprogramming.org/
17. *extreme Programming explained.* (2009). Recuperado el 4 de febrero de 2012, de http://extremeprogramming.host56.com
18. *Facultad de Ciencias Sociales de Buenos Aires*. (1 de 11 de 2004). Recuperado el 10 de 01 de 2010, de http://www.caminandoutopias.org.ar/contenidos/notas/tecnologias/0038.php
19. Flores, R. R., & Téllez Valdés, J. A. (2010). *Biblioteca Jurídica Virtual.* Recuperado el 9 de enero de 2012, de Biblioteca Jurídica Virtual: http://biblio.juridicas.unam.mx/libros/6/2801/7.pdf
20. *GNU Consultores*. (2011). Recuperado el 12 de 01 de 2012, de GNU Consultores: http://www.gnuconsultores.com/es/ingenieria/desarrollo/escritorio
21. GRASP. (2012). *GRASP.* Recuperado el 21 de mayo de 2012, de http://grasp.org/
22. *Grupo soluciones GSInnova*. (s.f.). Recuperado el 13 de 02 de 2012, de GSInnova: http://www.rational.com.ar/herramientas/roseenterprise.html
23. *IBM - developerWoks*. (2012). Recuperado el 9 de junio de 2012, de IBM - developerWoks: http://www.ibm.com/developerworks/ssa/rational/library/edge/09/mar09/collaris_dekker/index.html
24. *IBM.* (2012). Recuperado el 5 de abril de 2012, de http://www.ibm.com/software/awdtools/developer/rose/
25. *ibm.com.* (2012). Recuperado el 1 de abril de 2012, de http://www.ibm.com/software/awdtools/rup/
26. *Instituto de Investigaciones Juridicas - UNAM*
27. *Instituto de Investigaciones Juridicas - UNAM.* (27 de 08 de 2000). Recuperado el 9 de 01 de 2012, de Instituto de Investigaciones Juridicas - UNAM: www.bibliojuridica.org/libros/6/2801/7.pdf
28. *Instituto de Investigaciones Juridicas de la UNAM*
29. *La Revista Informática.com*. (s.f.). Recuperado el 25 de marzo de 2012, de 2006: http://www.larevistainformatica.com/C1.htm

30. *McGraw-Hill Interamericana de España,SL.* (2012). Recuperado el 16 de marzo de 2012, de http://www.mcgraw-hill.es/bcv/guide/capitulo/8448169204.pdf
31. Microsoft. (2012). *msdn*. Recuperado el 20 de enero de 2012, de msdn: http://msdn.microsoft.com/es-es/default.aspx
32. *Netbeans.* (2012). Recuperado el 15 de marzo de 2012, de http://www.netbeans.org/
33. *Netbeans.* (2012). Recuperado el 25 de marzo de 2012, de http://platform.netbeans.org/
34. *Object Management Group .* (1997-2012). Recuperado el 5 de febrero de 2012, de http://www.bpmn.org/
35. *Observatorio Regional de la Sociedad de la Información de Castilla y León (ORSI).* (2012). Recuperado el 20 de mayo de 2012, de http://www.orsi.jcyl.es
36. Övergaard, & Palmkvist, G. e. (s.f.). *ibm.com / "Use Cases: Patterns and Blueprints"*. Recuperado el 9 de junio de 2012, de http://www.ibm.com/developerworks/ssa/rational/library/edge/09/mar09/collaris_dekker/index.html
37. Pressman, R. (2002). *Ingeniería de Software: Un Enfoque Práctico. Quinta edición, ISBN: 8448132149.* S.l.: McGraw-Hill Companies.
38. Pressman, S. R. (2005). *Ingeniería del Software. Un enfoque prático. Ciudad de .* Ciudad de la Habana.
39. *rationalrose.com.* (2012). Recuperado el 3 de febrero de 2012, de http://www.rationalrose.com/
40. *Scribd*. (2012). Recuperado el 15 de abril de 2012, de http://es.scribd.com/doc/7411856/Caracteristicas-de-C
41. *Skill Resource (Habilidad de Recursos de Software).* (2005-2006). Recuperado el 13 de febrero de 2012, de http://www.skillresource.com/methodology/development-methodology/rational-unified-process.php
42. Soldevilla, F. T. (2005-2012). *OFICINA NACIONAL DE PROCESOS ELECTORALES(ONPE).* Recuperado el 8 de abril de 2012, de Autoridad máxima en la organización y ejecución de los procesos electorales: http://www.web.onpe.gob.pe/modEscaparate/caratulas/tuesta2.pdf
43. Stevens, P., Pooley, R., & Wesley, A. (2002). *Utilización de UML en Ingeniería del Software con Objetos y Componentes.*
44. *The Electoral Knowledge Network.* (1998-2012). Recuperado el 19 de enero de 2012, de The Electoral Knowledge Network: http://aceproject.org/main/espanol/et/et60.htm

45. *The Electoral Knowledge Network.* (1998-2012). Recuperado el 11 de febrero de 2012, de http://aceproject.org/main/espanol/et/et71.htm
46. Tuya, J., & Universidad de Oviedo, D. d. (2008). *RePrIS - Red para la Promoción y Mejora de las Pruebas en Ingeniería del Software*. Recuperado el 10 de junio de 2012, de Pruebas del Software: Descubrir Errores y Más...: http://in2test.lsi.uniovi.es/repris/actividades/TestingUPO20080421.pdf
47. *UML y Patrones. Introducción al análisis y diseño orientado a objetos.*
48. *unGrid - Universidad Nacional de Colombia.* (2012). Recuperado el 5 de mayo de 2012, de http://ungrid.unal.edu.co/seminar/GraspPatterns.htm
49. Valdés, R. R. (2012). *Instituto de Investigaciones Jurídicas de la UNAM – Serie Doctrina Jurídica, Núm. 550: Voto Electrónico, Derecho y Otras Implicaciones*. Recuperado el 23 de noviembre de 2012, de http://www.juridícas.unam.mx
50. *Visual C#*. (2012). Recuperado el 6 de mayo de 2012, de http://msdn.microsoft.com/es-es/vcsharp/aa336809.aspx
51. *Visual Studio.* (2012). Recuperado el 17 de enero de 2012, de http://www.microsoft.com/visualstudio/en-us