UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

# EXTENSIÓN Y MEJORA DE UN SISTEMA DE VOTACIÓN ELECTRÓNICA PARA HACERLO MÁS ROBUSTO, UNIVERSALMENTE VERIFICABLE, FÁCILMENTE USABLE Y PRÁCTICO

**MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL EN COMPUTACIÓN**

**RODRIGO ANTONIO PORRAS FLORES**

PROFESOR GUÍA:
ALEJANDRO HEVIA ANGULO

MIEMBROS DE LA COMISIÓN:
TOMAS BARROS ARANCIBIA
RODRIGO PAREDES MORALEDA

SANTIAGO DE CHILE
ABRIL 2012

# Resumen

La votación electrónica tiene potencialmente múltiples ventajas en comparación con otros sistemas de votación. Puede reducir drásticamente el tiempo del recuento de votos así como el número de errores en dicho proceso debidas a fallas humanas o por anotación deficiente de las papeletas. También permite a los votantes emitir su voto en cualquier recinto habilitado sin restricciones geográficas, disminuir la posibilidad de fraudes por autoridades maliciosas o comprometidas (por medio de administración distribuida de la elección, en particular, del recuento de votos), y obtener *verificación universal* de la elección. Esta última característica permite garantizar un proceso completamente transparente ante cualquier observador interno o externo, quien puede verificar - en forma *online* - la realización correcta de cada una de las etapas del proceso de votación.

En general, los sistemas de votación electrónica propuestos en los últimos años se clasifican en dos tipos: implementaciones simples que carecen de muchas de las características anteriores pero cuya ejecución y/o administración es muy fácil y práctica, y sistemas que satisfacen las características anteriores (basados en esquemas teóricos que requieren sofisticadas técnicas criptográficas) pero cuya usabilidad tanto para los votantes como los administradores de la elección es frecuentemente muy limitada. Peor aún, gran parte de los sistemas públicamente disponibles a la fecha carecen de las técnicas y los modelos adecuados en términos de diseño de software, con implementaciones complejas, infactibles de mantener y extender. En particular, en Chile no se han visto experiencias con sistemas de votación públicamente disponibles que satisfagan todas las propiedades mencionadas anteriormente y, al mismo tiempo, sean fáciles de utilizar y administrar para personas sin mayor experticia en el área.

El objetivo de este trabajo es extender y mejorar un prototipo de votación electrónica existente en el Departamento de Ciencias de la Computación de la U. de Chile, el cual implementa un esquema criptográfico eficiente y universalmente verificable. El propósito del trabajo es mejorar la implementación del sistema de tal forma que éste no sólo sea más robusto y tolerante a fallas, sino que también sea más fácil de utilizar y administrar, todo ello a un bajo costo. También, el trabajo finaliza un tema pendiente en la implementación existente: el desarrollo de una herramienta para permitir a todo votante u observador externo verificar en forma *online* el correcto desarrollo del proceso electoral.

Como resultado este trabajo logró cumplir los objetivos propuestos, obteniendo una versión mejorada del sistema de votación electrónica la cual es utilizable en la práctica. En particular, tolera fallas comunes en la práctica, como fallas de componentes físicos o conexiones de red, e intentos de ataques de seguridad al sistema. A pesar de estas fallas, el sistema logra seguir funcionando, o bien, tener la capacidad de recuperarse, conservando la integridad y privacidad de los datos. Por otro lado, el sistema se implementó utilizando una arquitectura de software que lo hace extensible y mantenible. Además, se mejoró la usabilidad para quienes administran la elección, puesto que se simplificó el uso, administración y configuración del sistema.

# Agradecimientos

*Agradezco a mis padres y familiares por su incondicional apoyo y cariño brindado. En especial durante este largo proceso.*
*Agradezco a mi profesor guía por su constante apoyo, tiempo y conocimiento, los que me han ayudado a realizar este difícil pero entretenido trabajo.*
*Agradezco también a todos aquellos que me dieron consejos y sugerencias sobre el desarrollo de este trabajo.*

# Índice general

# Capítulo 1

# Introducción

## 1.1. Contexto

A lo largo de la historia las sociedades se han organizado de diversas maneras. Sin embargo, en los últimos dos siglos se ha producido un cambio generalizado en la forma de organización y de gobierno. En efecto, muchos países han adoptado un modelo de gobierno democrático, incluso en los últimos años se ha visto cómo diversos pueblos con gobiernos autoritarios se han levantado contra estos en favor de establecer una nueva sociedad democrática.

En un régimen democrático el poder reside en la totalidad de sus miembros. Una característica fundamental de la democracia es que sus gobiernos derivan su legitimidad de las elecciones, que son la principal forma que tiene la sociedad para ejercer el poder de manera libre e igualitaria.

Dada la importancia de las elecciones es absolutamente necesario contar con un sistema de votación confiable y seguro. Un sistema de este tipo debe cumplir una serie de requisitos esenciales:

- El voto debe ser secreto, tanto al momento de emitir el voto, como al realizar el conteo de los votos. (Si el voto es efectivamente secreto, la coerción[1] del votante no es posible).
- Un votante sólo puede votar la cantidad de veces establecida legalmente. Normalmente esto es un voto por carrera, aunque pueden haber excepciones.
- Un votante sólo puede emitir voto en las suscripciones o carreras en las que legalmente puede votar.
- Un votante no puede vender su voto.
- Todas las papeletas de votación deberían ser contadas sin errores.
- No deben haber obstáculos para que un votante pueda emitir su voto.
- El sistema de votación utilizado debe estar operable durante todo el proceso de votación.

[1] Coerción: presión ejercida sobre alguien para forzar o condicionar su comportamiento, usualmente mediante amenazas.

### 1.1.1. Términos útiles en el contexto de votaciones

**Votación:** Acción y resultado de votar.

**Elección:** Votación que se hace para designar un candidato a un cargo, una elección puede agrupar una o más competencias (o carreras).

**Carrera:** Carrera (electoral) es una competencia entre varios candidatos para ser escogidos para un determinado cargo político o social. También puede representar una propuesta o moción.

**Candidato:** Persona que se postula a ser elegida para algún cargo o responsabilidad en unas elecciones.

**Voto:** Es el acto por el cual un individuo expresa apoyo o preferencia por cierta moción, propuesta, candidato, o selección de candidatos durante una votación.

**Papeleta:** Una papeleta electoral o boleta electoral, llamada también hoja de votación, es aquella con la que los electores emiten su voto en una elección; la papeleta puede ser una hoja de papel que se deposita en una urna o puede ser un documento digital que registra el voto.

**Votante:** El votante (o elector) es una persona con derecho a voto en una elección o consulta y que emite su voto válidamente en la misma.

**Vocal de Mesa:** Es el encargado de instalar la mesa de votación, entregan los recursos necesarios a los votantes, cuidan del orden y cumplimiento del proceso electoral, y resuelven consultas de los votantes.

**Mesa de votación:** Es el local donde un votante puede hacer su votación, generalmente contiene una urna donde se almacena la papeleta de votación. Hay un vocal de mesa o encargado de ayudar al votante y velar por el cumplimiento del proceso.

**Cámara secreta de votación:** Lugar físico de votación privado en el que un votante puede emitir su voto.

### 1.1.2. El proceso electoral

El proceso electoral es el mecanismo mediante el cuál los votantes pueden participar y ejercer su rol de ciudadanos. Todo proceso electoral puede ser dividido en etapas bien definidas:

**Registro:** Paso previo a la elección donde se definen y asignan los votantes que pueden votar, y las carreras en las que pueden participar. Esta etapa puede requerir la participación de los votantes o puede ser automática.

**Validación:** Durante la elección, el votante es autenticado y autorizado a votar.

**Votación:** Los votantes manifiestan sus preferencias, emitiendo su voto.

**Escrutinio o conteo:** Cómputo de los votos al final del proceso de elección.

### 1.1.3. Tecnologías existentes para realizar elecciones

Hoy en día existen al menos cinco tecnologías de votación en uso:

- Voto Australiano.
- Máquinas de votación con palancas.
- Sistemas de votación con tarjetas perforadas.
- Sistemas de votación de escaneo óptico.
- Sistemas de votación electrónica de grabación directa.

A continuación se describe cada una de estas tecnologías:

**Voto Australiano.** Este sistema de voto secreto en papel es el más utilizado hoy en día, el voto es registrado en una papeleta de papel pre-impresa con las posibles opciones de selección. Su nombre se debe a que se usó por primera vez en Australia en el año 1858. Es un sistema sencillo de implementar, pero su debilidad está en el proceso de recuento de los votos, el cual es ineficiente. Un problema, de carácter social, habitual en el proceso de emisión del voto se produce por errores del votante al marcar sus preferencias en la papeleta.

**Máquinas de votación con palancas.** (*Lever voting machines*), son máquinas de votación mecánicas, en las cuales el votante puede marcar sus preferencias directamente sin usar una papeleta de papel como intermediaria. La máquina realiza el recuento del voto inmediatamente después que éste ha sido registrado. Se usó por primera vez el año 1892 en Nueva York, Estados Unidos. Las debilidades de estos sistemas son: no mantener respaldo de los votos ingresados, si el mecanismo falla no es posible recuperar los votos ingresados. Estas máquinas son muy complejas, con muchas piezas mecánicas que pueden fallar en cualquier momento, comprometiendo gravemente la correctitud del sistema.

**Sistemas de votación con tarjetas perforadas.** En este sistema se utilizan tarjetas que se perforan para registrar los preferencias de los votantes, pueden realizarse las marcas usando sistemas mecánicos o asistidos por software. Se usaron por primera vez en 1964 en Georgia, Estados Unidos. Este sistema no puede garantizar que todas las tarjetas son perforadas correctamente, y el votante carece de un mecanismo para validar que la impresión realizada en la tarjeta refleje correctamente su intención de voto.

**Sistemas de votación de escaneo óptico.** (*Optical mark-sense voting*), en estos sistemas se utiliza una papeleta impresa en papel con las opciones de selección, el votante debe marcar sus preferencias siguiendo alguna regla definida. El recuento de votos se hace mediante el escaneo óptico de cada una de las papeletas emitidas, los votos son interpretados siguiendo las mismas reglas definidas para marcar las preferencias, y procesados utilizando software. Este sistema se usó por primera vez en la década de 1970 en Estados Unidos. Este sistema ha demostrado no ser demasiado efectivo al escanear e interpretar la intención de voto de los votantes, debido principalemente a un problema de carácter social, ya que los votantes no siempre siguen exactamente las reglas definidas para marcar la papeleta.

**Sistemas de votación electrónica de grabación directa.** (DRE o *Direct recording electronic voting*), estos sistemas permiten al votante emitir su voto directamente en una máquina de votación mediante un teclado de botones o a través de una pantalla táctil. Una vez que el voto ha sido emitido por el votante éste es respaldado en un formato digital. La primera generación de estos sistemas se utilizó en 1986, mientras que la segunda generación, utilizando pantalla táctil, se comenzó a utilizar en los 1990's. La interfaz de usuario de estos sistemas es la más flexible y capaz de reducir la cantidad de votos interpretados correctamente, sin embargo, debido a su naturaleza de software es necesario confiar en el correcto funcionamiento del sistema, ya que no deja un respaldo físico del voto. Para resolver estos problemas se puede incluir una técnica de verificación por registro en papel (o *voter-verified paper audit trail*), el que permite hacer un recuento manual de los votos. Otro mecanismo para tener confianza en el sistema es mediante el uso de esquemas criptográficos de votación que sean verificables.

A pesar de que existen variadas tecnologías, ninguna de ellas es realmente antigua. La primera gran revolución en el proceso de votación comenzó con el uso de la *papeleta de votación australiana*, la cuál se utilizó por primera vez en Australia el año 1858, a partir de entonces este sistema se ha utilizado en todo el mundo con bastante éxito y se sigue utilizando hasta el día de hoy dada su simpleza. Otro cambio importante se dió en Estados Unidos en 1892, donde se utilizaron por primera vez las máquinas de votación con palancas (*lever voting machines*), iniciando el uso de máquinas de votación mecánicas. En 1986 se comenzó a utilizar sistemas de votación por computador, sin embargo a pesar de sus ventajas teóricas, aún no se ha sido implementado de forma masiva tal sistema de votación de una manera realmente satisfactoria. Esto debido a las grandes dificultades que conlleva la implementación de un esquema de votación seguro y confiable.

### 1.1.4. La votación electrónica

Los sistemas de votación electrónica pueden clasificarse en al menos dos tipos, con distintas ventajas y limitaciones: *e-voting* (presencial) o *i-voting* (votación remota por Internet).

**E-voting, votación presencial.** Es el esquema mayormente aceptado por permitir un mayor grado de seguridad que su contraparte remota. Se puede implementar de dos maneras: *polling station e-voting* y *kiosk e-voting stations*. *Polling station e-voting* consiste en un esquema donde se tienen lugares de votación especialmente habilitados, lo que permite tener un alto control de la seguridad de los lugares de votación por parte de las autoridades, dado que cada votante debe autenticarse físicamente antes de acceder a la cámara secreta de votación, lo que lo hace menos vulnerable ante ataques. El esquema *kiosk e-voting stations* consiste en tener estaciones de votación públicas, sin estar en un ambiente controlado y asegurado (similar a la utilización de cajeros automáticos), en este esquema es más difícil evitar la coerción de los votantes, debido a la falta de control en las estaciones.

**I-voting, votación remota por Internet.** Este esquema consiste en emitir los votos de manera remota, sin necesidad de identificarse físicamente para sufragar, votando a través de un navegador web. Este esquema despierta mucho interés en las autoridades y en la gente en general, ya que haría mucho más cómodo y simple el trámite de emitir un voto sin tener que concurrir a un lugar predeterminado para votación; sin embargo, con este enfoque es imposible evitar la coerción de los votantes, además no es posible evitar que el software (como el navegador web) esté infectado por

*malware*[2] (o software malicioso) que podría robar la información del votante o modificar el voto antes de que este sea enviado y almacenado. Por esos motivos hay consenso en que un esquema de votación a través de Internet no es adecuado para una votación de gran escala, donde puede haber una alta motivación política, económica o social por modificar o robar los datos de la votación. Quienes promueven la utilización de un esquema de tipo *i-voting* proponen que un sistema de este tipo puede ser adecuado para votaciones de escala menor donde las probabilidades de coerción son más bajas. Puede ser al menos cuestionable el justificar el uso de un sistema de votación con menor nivel de seguridad en votaciones de menor escala, sólo porque estas votaciones tienen una probabilidad menor de ser afectadas por este tipo de problemas de seguridad.

### 1.1.5. Criptografía y votación electrónica

Un sistema de votación electrónica requiere cumplir con variados requerimientos de seguridad, como por ejemplo: asegurar la privacidad de los votos, evitar que los votos sean alterables, garantizar que el recuento de los votos sea realizado sin errores, que ningún voto válidamente emitido sea ignorado durante el conteo de los votos, y que el proceso de votación sea auditable o verificable por agentes externos a la votación o por los mismos votantes. La criptografía se ha utilizado durante los últimos años para satisfacer este tipo de requerimientos en muy diversas áreas de la computación, por lo que para satisfacer los requisitos de un sistema de votación electrónica, se hace indispensable la utilización de diversas herramientos y protocolos criptográficos ya conocidos. Por ejemplo, algunas de estas herramientas son: criptografía simétrica y criptografía de clave pública, encriptación y desencriptación de datos, firma digital y verificación, hashing y huellas digitales, y esquemas de compartición de secretos. Estas técnicas y herramientas son fuertemente estudiadas y analizadas, teniendo hoy en día esquemas y protocolos altamente seguros en la práctica para satisfacer los requerimientos de seguridad antes mencionados.

A pesar de los avances en criptografía, el problema que plantea la implementación de un sistema de votación electrónica no está absolutamente resuelto en términos prácticos, aún no existe una solución perfecta. En un sistema de votación, a diferencia de otros sistemas que también requieren seguridad, el requisito de seguridad no es negociable, en otras palabras: no se puede hacer un análisis de costo y beneficio para, por ejemplo, tolerar un porcentaje de pérdida de votos o votos alterados. Lo anterior ocurre porque todos los votos son igualmente importantes, y un solo voto puede hacer la diferencia entre ganar o perder una elección. Esta característica particular de un sistema de votación lo hace un problema difícil y costoso de implementar en la práctica. También es importante notar que en todo sistema, la seguridad se alcanza con una combinación de diversas herramientas y protocolos, las que pueden ser combinadas de distinta forma, pudiendo llegar incluso a un sistema inseguro si no se combinan de una manera correcta. La seguridad en un sistema complejo es difícil de lograr ya que ésta se comporta como una cadena, donde la seguridad total no es mayor que la seguridad del eslabón más débil de la cadena.

Actualmente existen cuatro modelos criptográficos base que se proponen como protocolos para votación electrónica, estos modelos son:

- Modelo de firmas ciegas.

[2]Malware: (del inglés *malicious software*) término que agrupa a todo tipo de software malicioso o malintencionado que tiene por objetivo espiar un computador, robar información, dañar el computador o realizar acciones no autorizadas por el usuario.

- Modelo mix-net.
- Modelo de Benaloh.
- Modelo de encriptación homomórfica.

A continuación se describen estos modelos, de manera muy simplificada:

**Modelo de firmas ciegas.** En este modelo criptográfico los votos son encriptados y luego firmados por un *validador* que no conoce el contenido. Luego el votante envía la encriptación y la firma a otra autoridad y los votos son publicados en un *bulletin board*. Finalmente, cada votante debe verificar su voto publicado y enviar anónimamente su clave de desencriptación a la autoridad. Este modelo requiere la participación del votante para llevar a cabo el recuento de los votos.

**Modelo Mix-net.** Una *mix-net* es una red compuesta de muchos servidores que se utiliza como alternativa criptográfica a un canal anónimo. En este modelo los votos son primero encriptados, luego estos ingresan a la red de servidores o *mix-net*, la que se encarga de producir múltiples permutaciones a los votos encriptados con el fin de lograr que estos sean anónimos (se pierda la relación entre el autor del voto y el voto en sí mismo). Finalmente, los votos encriptados (ya permutados por la *mix-net*) son desencriptados y publicados.

**Modelo de Benaloh.** Utiliza un *esquema de compartición de secretos homomórfico*, donde cada voto es dividido en trozos, los que son distribuidos entre una cantidad $n$ de autoridades de votación. Cada autoridad encripta su parte del voto y lo publica. Al final de la elección todas las autoridades juntan los trozos de los votos, combinándolos para obtener el total. Bajo este modelo se requiere que a lo menos una cantidad $t$ de las autoridades sea honesta, de tal manera que: $n/2+1 \leq t \leq n$.

**Modelo de encriptación homomórfica.** Usa una propiedad algebraica aplicable a ciertos conjuntos numéricos, donde se tiene que el *producto* de los votos encriptados es igual a la encriptación de la *suma* de los votos no encriptados, lo que permite hacer el conteo final de los votos sin necesidad de desencriptarlos individualmente, esto hace que los votos sean totalmente anónimos. Este modelo originalmente se propuso sólo para elecciones del tipo sí/no, pero puede ser extendido para esquemas de votación de múltiples candidatos y múltiples selecciones. El modelo de encriptación homomórfica es uno de los preferidos para la implementación de votación electrónica por su seguridad, dado que permite que el sistema sea *universalmente verificable*.

### 1.1.6. Votación electrónica en la práctica

A pesar de que un esquema criptográfico correcto es la base para construir un buen sistema de votación, el llevar la implementación de un sistema de votación a la práctica exige que el sistema sea capaz de manejar y resolver una serie de problemas a los que puede estar expuesta una votación de gran escala. Estos problemas por sí solos son complejos y difíciles de resolver. Entre ellos están los siguientes:

- El sistema debe ser altamente disponible y tolerante a fallas debidas a agentes externos, ataques informáticos o incluso debido a errores propios del sistema. Para esto el sistema debe recuperarse ante las fallas y volver al estado en que se encontraba antes, sin pérdida de la información o de votos emitidos por los votantes.

- El sistema debe ser auditable o verificable tanto en su implementación como en su funcionamiento para dar confianza a los usuarios.
- La implementación del sistema debe ser mantenible lo que exige una arquitectura de software bien diseñada que permita reducir el costo y tiempo de encontrar y corregir errores. Esto incluye también la capacidad del sistema para ser modificable y extendible.
- La comunicación entre los componentes del sistema debe hacerse mediante canales de comunicación seguros, sin pérdida de información.
- El sistema debe ser fácilmente administrable y configurable de manera de no dejar espacio para que se cometan errores humanos ni intencionales durante su uso.
- El proceso de instalación y administración del sistema debe ser lo más automatizado posible limitando la necesidad de interacción humana en su funcionamiento.
- La correctitud y seguridad del sistema no se puede alcanzar de forma confiable haciendo secreta su implementación, esto se conoce como *seguridad por oscuridad*, es mucho más confiable alcanzar la seguridad y correctitud del sistema completo mediante un buen diseño y una correcta implementación. Para esto la implementación del sistema debe estar abierta al escrutinio e inspección pública en busca de errores o vulnerabilidades de seguridad.
- El sistema debe usar formatos de datos estándares y abiertos que puedan ser revisados externamente al sistema antes, durante o después del proceso de elección.
- El sistema debería ser ejecutado en un entorno de red protegido que limite el acceso de usuarios no autorizados, limitando la posibilidad de ataques de denegación de servicios (DoS[3]) o la suplantación de algún cliente o servidor.
- Se deberían certificar los paquetes de software ejecutables que se usarán en una elección y asegurar que el software no sea modificado o reemplazado en las estaciones de votación.
- La administración de la elección no debe depender de una sola autoridad. Es mejor que sea distribuida entre varias autoridades u organismos independientes, así se restringe la posibilidad de manipular una elección ya que sería necesaria la colusión[4] de la mayoría de estas autoridades.
- El sistema debe presentar una interfaz de usuario simple y fácil de aprender que no induzca a errores y que no sea un impedimento para usuarios no acostumbrados a la tecnología. Se debe reducir el impacto que puede generar la brecha digital (ver *digital divide*) [5] entre la población al usar la tecnología de votación que puede inhabilitar de forma implícita a personas para ejercer su derecho a votar si éstas tienen menor acceso o conocimiento de tecnologías de la información y las comunicaciones.

[3]DoS: (de la sigla en inglés *Denial of Service*) en seguridad informática, ataque de denegación de servicios es un ataque a computadores o una red causando que un recurso o servicio sea inaccesible a los usuarios legítimos.

[4]Colusión: pacto ilícito para dañar a un tercero. En el caso de una elección puede haber colusión entre dos o más autoridades para modificar los resultados de una elección a su conveniencia o para perjudicar a un tercero.

[5]*Digital divide* (o brecha digital): se refiere a las desigualdades entre grupos de gente con respecto al acceso, uso y conocimiento de tecnologías de la información y las comunicaciones.

Estas condiciones son necesarias en la práctica para la implantación de un sistema de votación electrónica, las que en conjunto son difíciles de alcanzar, haciendo que sea altamente costoso. Por lo tanto, uno de los objetivos de esta memoria es proponer una implementación que cumpla con buena parte de estos requisitos, y que a la vez pueda ser implantada a un bajo costo y con un mínimo de conocimientos técnicos, de manera de permitir que quien quiera usar un sistema de votación de este tipo, pueda hacerlo sin impedimentos y sin sacrificar ni transar aspectos de seguridad y calidad.

### 1.1.7. Ventajas de la votación electrónica

Si la implantación de un sistema de votación electrónica resulta un desafío complejo y costoso, ¿por qué invertir tanto esfuerzo en implementarlo? Los sistemas de papeletas impresas de votación y conteo manual, y los sistemas mecánicos de votación están expuestos a diversos problemas, los que pueden ser reducidos o eliminados usando un sistema de votación electrónico. Los sistemas mecánicos tienen altas tasas de errores difíciles de detectar y son de un costo muy alto. Además hoy en día hay una tendencia a la disminución del porcentaje de la población que participa en las elecciones, ya sea porque no tienen interés, no confían lo suficiente en el sistema o porque encuentran muy tedioso el proceso de ir a sufragar.

Entre los beneficios que tiene un sistema de votación electrónica verificable se distinguen:

- El proceso de conteo de los votos se puede llegar a reducir a unos pocos minutos.
- Se reduce la posibilidad de votos mal contados debido a errores humanos.
- Se puede tener certeza de que un voto emitido sea considerado en el resultado final.
- Se reduce la posibilidad de que un votante con una intención clara de voto se equivoque (por ejemplo si el votante marca mal su preferencia en papel).
- Se puedan dar múltiples interfaces de usuario para personas con discapacidades auditivas o visuales.
- Se puede eliminar la restricción (desde el punto de vista tecnológico) de imponer un lugar geográfico fijo donde votar (un votante podría emitir su voto en cualquier estación habilitada para sufragar incluso si está vacacionando), lo que podría incentivar la participación de la gente en las elecciones.
- Es posible realizar una elección exitosa incluso tolerando colusión entre algunas de las autoridades encargadas de la elección (por ejemplo, debido a corrupción).
- Se puede reducir la coerción de votantes, al ser posible que un votante sufrague repetidas veces. Además el votante no debe recibir ningún recibo que indique por quién votó (esto es importante para que no haya venta de votos).
- Se puede verificar el proceso de votación desde principio a fin.
- Los votos una vez emitidos son anónimos, públicos y verificables.
- El conteo de votos puede ser realizado por terceros sin restricción (haciendo el conteo verificable por el público).

- Se puede argumentar que podría aumentar la participación de la gente en las elecciones de carácter voluntario, si se usa una tecnología más moderna, visualmente atractiva, fácil de usar y que entregue un alto grado de confianza.
- Economías de escala: si se realizan elecciones de forma frecuente se puede ver una disminución en el mediano o largo plazo al implantar un sistema de votación electrónica, ya que el costo de inversión inicial sólo se realiza la primera vez que se obtiene el sistema. En usos posteriores del sistema se ahorra el costo de obtención y aprendizaje del sistema.

En los sistemas de votación electrónica existe un concepto adicional muy importante y necesario para dar confianza a los usuarios y credibilidad al sistema y al proceso electoral, la "verificación universal". Este concepto implica que cualquier votante, autoridad u observador ajeno al proceso electoral puede auditar o verificar que el proceso se está llevando a cabo correctamente, y que cualquier intento de alteración del proceso será detectado. Que el sistema sea "universalmente verificable" es una cualidad necesaria para la transparencia del proceso, independientemente de la implementación del sistema o de las autoridades encargadas de la administración del proceso.

## 1.2. Motivación

La implementación de sistemas de votación electrónica no ha sido fácil de lograr, a pesar de que hay varios esquemas teóricos que se proponen para su implementación. Estos sistemas no son fáciles de implementar, debido a que para llevar un sistema de este tipo a la práctica hay una serie de dificultades prácticas que el modelo teórico no resuelve directamente. Por lo tanto se requiere integrar distintas técnicas y herramientas que se utilizan en aplicaciones de tipo comercial y empresarial, y que pueden ser aprovechadas para una exitosa implantación del sistema. De todos los sistemas de votación electrónica que se han implementado en distintas partes del mundo, pocos se han logrado implementar de forma satisfactoria, principalmente porque no implementan los esquemas criptográficos adecuados que permitan hacerlo verificable, porque tienen fallas de seguridad, porque pueden ser manipulados por una entidad, porque no pueden ser verificables debido a sus licencias de software privativo. Otro factor a considerar es que los costos de tales sistemas son muy elevados, debido a la infraestructura requerida para su uso, por ejemplo, se utilizan en muchos lugares las máquinas de votación especialmente construidas para este fin, las cuales usualmente tienen un costo de compra y de administración muy elevado.

Hay varios motivos que hacen razonable el pensar que se puede implementar un sistema de votación electrónica confiable y seguro, entre ellos: actualmente el acceso a Internet está prácticamente al alcance de todos, el hardware ha bajado drásticamente de precio y hay servicios de software de calidad y a bajo costo (*Cloud Computing* [6]). Estos recursos disponibles pueden ser aprovechados para la implantación de un sistema de votación confiable y a un costo más bajo de usar, poniendo al sistema al alcance de instituciones grandes o pequeñas.

En el planteamiento de este trabajo se intenta mostrar una forma de implementación de un sistema de votación electrónica verificable, que permita aprovechar todas las ventajas de un sistema de estas características en comparación con las otras tecnologías, y que resuelva gran parte de los problemas anteriormente señalados, utilizando un esquema criptográfico seguro, utilizando

[6]Cloud Computing (o computación en la nube): es un paradigma informático que permite ofrecer servicios de computación a través de Internet. Hay tres categorías: *software as a service* (SaaS), *Platform as a service* (PaaS), e *infrastructure as a service* (IaaS)

software de código abierto, utilizando esquemas de datos y seguridad distribuidos, creando un sistema de software robusto y tolerante a fallas. Además de estas características se desea implementar un sistema de bajo costo, que aprovecha hardware disponible de uso general, también se pueden aprovechar servicios comerciales (*Cloud Computing*) que permiten reducir los costos de instalación y administración del sistema. Todas estas características podrían permitir que el sistema propuesto sea implantado con poco esfuerzo por instituciones de pequeño o gran tamaño. Otro punto clave es que se pueden entregar herramientas a los usuarios u observadores que les permitan por sí mismos auditar de forma independiente el correcto funcionamiento del sistema, y la integridad de sus votos. Así se puede establecer una base de confianza en el sistema, condición primordial para la credibilidad de la elección y la mantención de una sana democracia.

## 1.3. Situación previa

Previamente al desarrollo de este trabajo de título existe un prototipo de sistema de votación electrónica (evoting), que ha sido desarrollado por el CLCERT de la Universidad de Chile a partir de la memoria de Pamela Cordero [3]. Este sistema ha sido utilizado en elecciones estudiantiles y académicas del Departamento de Ciencias de la Computación de la Universidad de Chile.

Este sistema de votación es de tipo presencial en ambiente controlado (*polling station e-voting*): el registro se hace por entidades autorizadas, la verificación de identidad puede ser física o electrónica, los procesos de votación y escrutinio se hacen de manera electrónica, la votación se lleva a cabo en recintos electorales especialmente habilitados y asegurados, y el voto puede ser almacenado en un dispositivo o enviado a una central por medio de una conección remota segura. El sistema utiliza un esquema criptográfico eficiente para múltiples candidatos, y *universalmente verificable*.

El sistema tiene una implementación poco robusta y poco mantenible, y carece de una arquitectura de software bien diseñada. Además el sistema no está preparado para ser tolerante a fallas ni disponible frente a eventos de desconexión. Es muy difícil de configurar e instalar. Funcionalmente el sistema no tiene una herramienta que permita hacer uso de la propiedad de ser *universalmente verificable*.

Éste es el sistema original. Tomando este sistema como base se construirá una nueva versión mejorada que cubra aspectos prácticos tratando de llevar el sistema a un estado más maduro y confiable.

## 1.4. Objetivos

### 1.4.1. Objetivo general

Extender y mejorar un prototipo existente de sistema de votación electrónica presencial, eficiente y teóricamentente universalmente verificable, basado en un modelo criptográfico homomórfico. En particular, construir a partir de este sistema un sistema de votación electrónica que pueda ser utilizado en elecciones donde la corrupción y la coerción son una preocupación importante, que sea robusto, tolerante a fallas, de fácil uso, configuración y administración. Esto incluye agregar al sistema herramientas para hacerlo *universalmente verificable* por los votantes u observadores del proceso en la práctica.

### 1.4.2. Objetivos específicos

- Hacer una reingeniería de la persistencia de datos del prototipo de votación existente, reemplazando la persistencia de datos en archivos planos por la utilización de una base datos relacional. De esta forma mejora la robustez del sistema y consistencia de los datos.
- Extender el prototipo existente, agregando un esquema de memoria *cache*[7] de votos en las estaciones de votación, permitiendo de esta forma emitir un voto bajo condiciones de desconexión entre las estaciones de votación y los servidores. Si no hay conectividad, el voto es almacenado localmente, y es enviado al servidor una vez que se ha recuperado la conectividad. En caso de que sea imposible reestablecer la conectividad entre la cámara y el servidor, los votos (respaldados en un medio físico portable) son llevados al servidor manualmente resguardando la seguridad física de los votos en el camino.
- Implementar una herramienta para la configuración de una nueva elección, la cual permita configurar las carreras, candidatos y votantes. Debe ser simple y fácil de usar.
- Implementar una herramienta externa de Verificación de la elección, que permita monitorear el proceso completo de votación, comprobar la existencia e integridad de los votos, comprobar el conteo de votos y comprobar las etapas de los protocolos utilizados por las *Autoridades* de votación. Esta herramienta utiliza los algoritmos criptográficos ya existentes (ya implementados) en las *Autoridades* del sistema original, re-adecuándolos para ser utilizados en el proceso de verificación externa.
- Implementar esquemas de recuperación y restauración de datos ante fallas en los servidores o en las estaciones de votación, sin pérdida de información.
- Extender la herramienta de instalación del sistema de votación, simplificando el proceso de instanciación de los servidores en máquinas (computadores) utilizando *live-cd's*. Opcionalmente se estudia la posibilidad de instalar el sistema utilizando servicios de servidores *cloud computing* comerciales.
- Extender el prototipo existente, agregando *logging*[8] de datos, para generar un reporte detallado de las acciones de cada módulo del sistema.

## 1.5. Descripción general de la solución

Para satisfacer los objetivos expuestos, en este trabajo se propone hacer una re-ingeniería del sistema, estableciendo una arquitectura más robusta, que separe adecuadamente las responsabilidades y modularice el sistema en componentes, haciéndolo más extensible y mantenible. Se propone hacer *refactoring* de código de las componentes existentes, eliminando redundancia de código y utilizando patrones de diseño conocidos. Para esto es importante contar con un conjunto de *tests* unitarios, de integración y de sistema.

Adicionalmente se implementa un componente para verificar el proceso de votación, y otra para configurar una elección, con el fin de conseguir un sistema fácil de usar y de administrar.

[7] Memoria cache: es una memoria para almacenamiento de datos de forma temporal, que permite un acceso posterior rápido.

[8] Computer data logging: es un proceso de registro de eventos ocurridos en programas informáticos, que puede ser usado para entender el comportamiento del sistema o diagnosticar problemas.

También se construye un sistema operativo personalizado minimal el despliegue del sistema, con herramientas para simplificar la instalación y configuración del sistema.

## 1.6. Resultados de la solución implementada

La implementación de la solución propuesta se validó bajo una serie de escenarios para poner a prueba la correctitud del sistema, correcto funcionamiento de los esquemas de restauración, correctitud de la verificación del proceso, correcto uso de la memoria caché, los que se cumplieron satisfactoriamente. La implementación del sistema se validó mediante el montaje de elecciones de prueba, tanto en escenarios virtualizados como en un escenario real. Como resultado de este trabajo se obtuvo una implementación más robusta, confiable y segura, que se orienta a un uso práctico, por ser más fácil de usar y a un bajo costo. Estás mejoras al sistema original se acercan un poco más a cumplir las características deseables de un sistema de votación electrónica.

# Capítulo 2

# Antecedentes

## 2.1. Otros sistemas de votación electrónica existentes

Se estudiaron tres sistemas de votación electrónica distintos, todos con características particulares, con objetivos, tecnologías y técnicas criptográficas distintas entre sí. Todos estos sistemas tienen en común el ser proyectos recientes, que utilizan y aprovechan los avances e investigaciones hechas en el área. Estos proyectos son interesantes, pero a su vez se ha podido constatar que tienen algunas deficiencias, así como también ventajas, en comparación con el sistema propuesto en este trabajo. Esto se debe principalmente al hecho de que ésta es un área de la computación reciente, por las dificultades técnicas de llevar un esquema de votación electrónica a la práctica y porque es difícil lograr el perfecto equilibrio entre la seguridad, usabilidad, rendimiento y costo del sistema.

Los sistemas estudiados son los siguientes: Helios [5], VoteBox [6] y Prêt à Voter [7].

### 2.1.1. Helios

Helios es un sistema de votación de tipo *i-voting*, es decir que los votantes pueden votar por Internet sin necesidad de ir a un local de votación. Este sistema es “abiertamente auditable", es decir, que el comportamiento y los resultados del sistema pueden ser verificados por cualquier persona o entidad. El sistema ha sido diseñado para ser utilizado en elecciones donde se necesita confianza, voto secreto, pero donde la coerción de los votantes no es una preocupación importante.

El sistema está construido en Python, utiliza una interfaz Web implementada con JavaScript, jQuery, y utiliza JSON como estructura de datos.

Se utiliza el protocolo de votación de Sako-Kilian [11], el cual utiliza *mixnets* para asegurar el anonimato de los votos, utilizando El-Gamal como esquema de encriptación.

#### Cualidades

Este sistema se presenta como una alternativa segura para realizar elecciones *on-line*, que puede ser utilizado por comunidades de Internet, grupos sociales o centros estudiantiles. Tiene la clara ventaja de permitir que los votantes puedan sufragar cómodamente desde cualquier lugar, en poco tiempo, especialmente útil en organizaciones o grupos que están muy distribuidos geográficamente como para votar todos en un mismo local de votación.

El sistema es públicamente auditable. Para esto proveen dos programas, un programa para verificar un único voto, verificando su integridad y validez, mediante revisión de *checksums*. Otro

programa permite verificar las etapas del proceso de combinación, desencriptación y conteo de los votos.

En el lado del cliente, se utiliza una técnica llamada *Single-Page Web Application*, esto se refiere a que en el lado del cliente (*browser*) toda la aplicación se carga una sola vez, sin realizar solicitudes a través de la red hasta terminar el proceso de preparación del voto.

#### Desventajas

Este sistema no fue diseñado para votación presencial sino que para realizar votación remota (por Internet), por ende carece de una arquitectura del tipo mesa vs cámara. Además no hay posibilidad de evitar la coerción. Esto es una deficiencia común a todos los sistemas de votación no presencial. Por esta razón el sistema no puede ser utilizado en elecciones donde haya un alto riesgo de coerción, como en elecciones políticas o donde hay intereses económicos importantes.

Debido a la implementación del sistema, éste tiene la restricción de funcionar sólo en el navegador Web *Mozilla Firefox*.

Los esquemas implementados no toleran participantes (servidores) corruptos, a diferencia del sistema propuesto en este trabajo donde se detecta y se excluye al servidor malicioso de los protocolos. Ésto permite continuar con el proceso sin detenerse hasta el final.

No se encontró información acerca acerca de la implementación de esquemas de tolerancia a fallas y la capacidad de restauración automatizada del sistema.

### 2.1.2. VoteBox

VoteBox es un sistema de votación electrónica presencial que tiene por objetivo dar garantía de ser verificable de extremo a extremo, permitiendo a observadores independientes auditar la ejecución del sistema. El sistema está diseñado de manera de minimizar el volumen de la implementación. Otras propiedades del sistema son: resistencia a pérdida de datos, evidenciar intentos de fraude y servir como una herramienta para estudiar el comportamiento humano frente a sistemas de votación, bajo distintos escenarios.

Votebox emplea un esquema de encriptación homomófica para encriptar y hacer el conteo de los votos.

#### Cualidades

La interfaz de usuario del sistema utiliza interfaces gráficas pre-renderizadas. Esta idea fue tomada del proyecto pVote [12]. Con esta técnica se crean las interfaces tal como se verán en la pantalla antes de iniciar la votación .Durante la votación estas son usadas como imágenes, sin la necesidad de utilizar librerías de interfaces gráficas sofisticadas, ni lógicas complejas. Con esto se reduce el tamaño del sistema que debe ser auditado.

El *auditorio* registra los eventos del sistema que ocurren en cada una de las estaciones de votación. Usa *secure audit logging*, que permite detectar fallas o comportamientos inesperados. Cada *log* es firmado y enviado a cada una de las estaciones del sistema dentro de la red, teniendo replicación de los *logs* del sistema en todas las estaciones.

El sistema tiene una consola de supervisión, que permite coordinar los eventos durante la elección como abrir o cerrar estaciones de votación, también genera reportes en tiempo real del estado de cada máquina en el centro de votación.

El sistema alcanza verificabilidad extremo a extremo, es decir que el votante puede probar que su voto refleja su intención de voto y que su voto es contado correctamente. Para esto la estación de votación permite al votante retar al sistema para probar si su voto ya generado corresponde a lo que el votante escogió. Esta prueba funciona ya que el sistema no sabe cuando un voto puede ser retado; así que si la estación es maliciosa podría ser detectada.

Se hizo una variante del sistema con el fin de estudiar factores humanos en torno al uso del sistema, que recoge información acerca de todas las actividades que realiza un votante. Intencionalmente se incluye comportamiento malicioso para estudiar si estos son detectados por los usuarios.

**Desventajas**

El sistema no implementa *proofs* (o pruebas de verificacion de la correctitud de los votos) para verificar la validez de los votos, sin vulnerar el anonimato de los votos. Por ejemplo, para asegurar que el votante ha hecho una sola selección, o que no ha votado múltiples veces por el mismo candidato. Para esto se pueden utilizar Zero Knowledge Proofs [13], como lo hace el sistema utilizado en este trabajo de título.

VoteBox no implementa un esquema que tolere autoridades maliciosas (que intenten modificar el resultado). Esto puede lograrse mediante la utilización de servidores de autoridades distribuidos que comparten una clave mediante compartición de secretos distribuidos [14, 15].

### 2.1.3. Prêt à Voter

Prêt à Voter es un sistema de votación electrónica creado en el año 2004. El 2006 se creó una versión mejorada del sistema, ésta es la versión analizada en este trabajo.

Este sistema, es un sistema de votación de escaneo óptico, que utiliza un modelo criptográfico de *mixnets*. El sistema cuenta con una familia de variantes que cubren escenarios distintos de votación: selección de un candidato, selección de múltiples candidatos y selección de candidatos usando *ranking*. El funcionamiento del sistema se divide en cuatro fases:

- Generación de la papeleta: Se genera papeletas impresas con dos partes, al lado izquierdo está la lista de candidatos (en orden aleatorio), al lado derecho se encuentran los cuadros de selección y un código que representa la lista de candidatos (de manera encriptada). El orden aleatorio de los candidatos más la posterior eliminación de esa parte de la papeleta provee la privacidad del voto. La otra parte de la papeleta puede es escaneada por el sistema y el votante puede quedarse con ella. El código en la papeleta le permite posteriormente al votante verificar su voto en el sistema.

- Captura del voto: Una vez que el votante selecciona sus preferencias, éste corta la papeleta. El sistema escanea el lado derecho de la papeleta (voto encriptado) y lo publica en un *bulletin board* de forma pública. El votante se queda con el lado derecho como recibo.

- Procesamiento de los votos: El sistema utiliza el modelo de *mixnet*, donde los *mix-servers* se encargan de mezclar y lograr el anonimato de los votos. Terminado este proceso se desencriptan los votos (ya anónimos) y se publican en el *bulletin board*.

- Auditoría: El sistema puede ser verificado de dos formas: en la primera cada votante puede verificar que su voto sea válido y sea correcto, comparando su recibo con el voto publicado. En la segunda manera los auditores del proceso pueden verificar que los votos publicado

han sido correctamente desencriptados y escrutados. Dado que los votos desencriptados son públicos, cualquier auditor pueda realizar el conteo para comprobar el resultado.

**Ventajas**

Las ventajas del sistema se agrupan en las siguientes categorías:

- Integridad: El sistema permite asegurar que el voto representa la intención del votante y asegurar que los votos son contados correctamente.
- Verificabilidad: Cada puede obtener un número aleatorio de papeletas y escoger uno para votar. Los otros los puede usar para retar al sistema. Se puede verificar que los *mixnets* actúen honestamente.
- Anonimato: El sistema provee privacidad del votante.
- Robustez: Tiene un alto nivel de robustez frente a autoridades que fallan o que actúan deshonestamente.
- Usabilidad: Los votantes no necesitan conocimiento o habilidades especiales para votar.

**Desventajas**

Entre las desventajas del sistema se cuentan:

- Ataque de recibo descartado: Si un votante descarta su recibo, un atacante podría alterar su voto sin que sea acusado.
- Ataque de cadena de voto: Si se imprimen todos las papeletas antes del proceso, un adversario podría robar papeletas en blanco y utilizarlas para ejercer coerción sobre una cantidad de votantes.
- Ataque de aleatorización: Adversarios podrían forzar a votantes a votar de forma aleatorio exigiéndoles que siempre marquen una misma posición en la papeleta
- El sistema deposita su confianza en un *bulletin board*, que puede ser un único punto de fallo.

## 2.2. Metodologías de desarrollo

### 2.2.1. Re-ingeniería de software

La esencia de la re-ingeniería de software [19] es mejorar y transformar software existente, de manera que pueda ser entendido, controlado y usado. La necesidad de la re-ingeniería de software ha aumentado debido la necesidad de extender sistemas legados cuya arquitectura está obsoleta en términos de su estabilidad, adaptabilidad, capacidad de evolucionar y de soportar requerimientos cambiantes. Esta práctica es importante para recuperar y reutilizar software existente, mejorando la calidad y robustez del sistema, manteniendo controlados los costos de mantenimiento y evolución posteriores.

Usualmente los requerimientos, diseño y documentación no están disponibles o están muy desactualizados, por lo que una de las mayores dificultades de la re-ingeniería de software es entender el sistema existente.

### Objetivos de la re-ingeniería

Un problema usual de los sistemas de software es la falta de un buen diseño estructural y organización de código, lo que produce que hacer cambios o introducir mejoras sea difícil y costoso. El reto de la re-ingeniería de software es tomar un sistema existente y agregar buenos métodos de desarrollo de software y propiedades, generando un nuevo sistema que mantiene las antiguas funcionalidades mientras se aplican nuevas tecnologías. Hay cuatro objetivos generales para cualquier proceso de re-ingeniería de software:

- Preparación para mejoras funcionales.
- Mejorar la mantenibilidad del sistema.
- Migración.
- Aumentar la confiabilidad del sistema.

### Modelo general para re-ingeniería

Todo sistema de software puede ser separado en cuatro capas de abstracción:

- **Conceptualización**: Características funcionales del sistema.
- **Requerimientos**: Descripción detallada de las funcionalidades.
- **Diseño**: Arquitectura, componentes, algoritmos y estructuras de datos.
- **Implementación**: Codificación del sistema en un lenguaje entendible por la máquina.

El proceso de re-ingeniería se aplica a cada una de estás capas de abstracción del sistema, donde por cada capa se debe hacer lo siguiente:

- Conceptualización - Re-pensar el sistema.
- Requerimientos - Re-especificar el sistema.
- Diseño - Re-diseñar el sistema.
- Implementación - Re-implementar el sistema.

### Enfoques para aplicar la re-ingeniería

**Enfoque *Big Bang*:** Se reemplaza el antiguo sistema de una sola vez, no se requieren interfaces entre antiguos y nuevos componentes, no se mantienen ni operan estados intermedios del proceso. Puede ser difícil y costoso en sistemas grandes.

**Enfoque Incremental:** Se reemplazan componentes del sistema antiguo de forma incremental, teniendo nuevas versiones del sistema en cada incremento. Este proceso puede no ser adecuado en casos donde se requiere cambiar toda la estructura del sistema, y requiere una cuidadosa gestión de la configuración en cada fase.

**Enfoque Evolucionario:** Se reemplazan secciones del sistema antiguo, identificando las secciones basándose en las funcionalidades y no en la estructura del sistema. El resultado es un sistema de diseño modular donde cada componente tiene un alcance reducido.

### 2.2.2. Refactorización de código

*Code Refactoring* (o Refactorización de código) es una técnica disciplinaria para restructurar un cuerpo de código existente, alterando su estructura interna sin cambiar su comportamiento externo. Se aplican una serie de pequeñas transformaciones que preservan el comportamiento. Cada transformación (o *refactoring*) es pequeña, pero una secuencia de transformaciones puede producir una reestructuración significativa. Esta técnica mantiene el sistema completamente funcional después de cada pequeña refactorización, reduciendo las posibilidades de que un sistema sea seriamente roto durante la reestructuración.

El uso de está práctica permite mejorar los atributos *no funcionales* de un sistema, sin modificar sus funcionalidades. Los beneficios de utilizar esta práctica se clasifican en dos categorías:

**Mantenibilidad:** Se reduce la complejidad al mejorar la mantenibilidad de código, es más fácil corregir errores al aumentar la legibilidad del código. Esto puede ser alcanzado mediante la reducción de grandes rutinas monolíticas en un conjunto de métodos concisos, bien nombrados y con un propósito bien definido. También sirve reorganizar las clases y métodos de forma más apropiada, eliminando la duplicación de código, así como agregando comentarios útiles y eliminando los comentarios que no agregan valor.

**Extensibilidad:** Es más fácil extender las capacidades y funcionalidades del sistema si este usa **patrones de diseño** reconocibles, proveyendo flexibilidad donde antes no existía.

Para una adecuada aplicación de refactorización es importante contar con un conjunto sólido de *test unitarios* que comprueban que el comportamiento después de la refactorización es correcto. Distintas metodologías ágiles de desarrollo de software consideran el *Refactoring* como una parte integral del ciclo de desarrollo de software, por ejemplo *Extreme Programming*.

#### Técnicas de refactorización

*Técnicas que permiten mayor abstracción:*

- Campo encapsulado: Forzar el acceso a los atributos de un objeto mediante métodos *getter* y *setter*.
- Generalizar tipos: Crear tipos más generalizados que permiten mayor compartición de código.
- Reemplazar tipo: Crear tipos que utilizan una interfaz común, y escoger entre ellos mediante el patrón de diseño *State* o *Strategy*.
- Reemplazar los condicionales múltiples por polimorfismo.

*Técnicas para dividir código en piezas lógicas:*

- Extraer método: Extraer parte de un método grande en un nuevo método o función.
- Extraer clases: Mover parte del código de una clase existente a una nueva clase.

*Técnicas para mejorar nombres y ubicación de código:*

- Mover método o campo: Mover un método o campo a una clase o archivo más apropiado.

- Renombrar método o campo: Cambiar el nombre por uno nuevo que revele de mejor manera su propósito.
- *Pull up*: En *OOP*, mover a una superclase.
- *Push down*: En *OOP*, mover a una subclase.

## 2.3. Evaluación de tecnologías para la implementación

Se ha investigado acerca de distintas tecnologías y posibles soluciones existentes que permitan resolver los problemas planteados por los objetivos definidos en el proyecto. Entre estos se han analizado tecnologías de caché de datos, se han revisado distintos formatos estándares de almacenamiento de datos, y se han analizado alternativas de comunicación entre sistemas distribuidos.

### 2.3.1. Representación y persistencia de datos

#### Formato de datos estándar

Se requiere de un formato estándar de datos para guardar información de configuración de la elección que no será modificada durante la elección: como candidatos, votantes y carreras. Se analizaron formatos de archivos para almacenar datos estructurados: XML, YAML, JSON.

- **XML**: Formato basado en *marcas* extensible que admite validación de formato mediante uso de esquemas.
- **YAML**: Formato para serialización de datos, de fácil legibilidad para humanos, y de tamaño más comprimido que XML, pero no muy difundido.
- **JSON**: Formato para serialización de datos simple, similar a YAML, de menor tamaño que XML, pero no hay un soporte oficial en Java.

Se ha escogido XML por ser legible por humanos, por estar altamente soportado en Java y por ser validable mediante esquemas usando tecnologías como DTD o XML Schema. En este caso no es importante el rendimiento o tamaño de almacenamiento de los datos.

#### Bases de datos para almacenamiento de datos persistente en servidor de mensajes

El sistema de votación requiere de un mecanismo de almacenamiento persistente de datos robusto, para almacenar los votos emitidos y los mensajes usados por las autoridades de votación para la implementación de los protocolos distribuidos.

Se consideraron sólo sistemas de administración de bases de datos relacionales que funcionen de forma embebida, que sean livianos o que no requieren servidores adicionales, y que almacenen la base de datos en un archivo de forma local. La razón para escoger este tipo de base de datos es la de simplificar la administración de la base de datos, y entregar una solución pre-construida que requiera mínima administración y configuración. Adicionalmente todos los accesos a la BD se hacen localmente desde el mismo servidor de mensajes, por lo tanto no es necesario tener un servidor de BD accesible desde distintos equipos. Los sistemas de bases de datos analizados son: SQLite, JavaDB/Derby y HSqlDB.

- **SQLite**: BD embebible, utiliza un fichero para guardar los datos, soporta transacciones, tiene alto rendimiento, admite accesos concurrentes a la BD y es tolerante a fallas. Es poco configurable.
- **JavaDB**: BD con implementación nativa en framework de Java, soporta transacciones y acceso de usuarios concurrente a los datos. Tiene menor rendimiento que SQLite y sólo puede ser utilizada como servidor de BD.
- **HSqlDB**: BD con implementación nativa en Java, permite usar archivo como base de datos, soporte de usuarios concurrentes. Tiene menor rendimiento que SQLite.

La base de datos está en el *BulletinBoard* o servidor de mensajes, pero las datos son enviados desde otros servidores remotamente, esto hace importante que la base de datos sea eficiente, reduciendo los tiempos de comunicación remota al servidor. Las operaciones sobre los datos son muy simples, pero se necesita que el motor de base de datos maneje accesos concurrentes a los datos de manera robusta. Se escogió SQLite, ya que cumple estos requerimientos, es activamente desarrollado y está bastante probado.

#### Caching de datos

Se requiere el uso de *caché* de datos para el envío de datos al servidor del sistema. Para asegurar la comunicación de datos bajo condiciones de desconexión, manteniendo un caché persistente mientras no haya una conexión disponible. Esto es necesario para mantener la disponibilidad de los locales o estaciones de votación.

Se encontraron dos tecnologías de caché disponibles para Java de fácil uso, y que permiten persistir los datos: OSCache y EHCache.

- **OSCache**: Librería de caché bastante usada, de muy fácil uso, pero descontinuado.
- **EHCache**: Librería de caché robusto y muy difundido, con más opciones de configuración que el anterior y de fácil uso.

Se hicieron pruebas de concepto con EHCache para probar el comportamiento y la configuración de la librería logrando resultados exitosos, por lo que se escogió como sistema de caché, debido a su uso difundido y activo desarrollo.

### 2.3.2. Comunicación entre sistemas y seguridad

#### Tecnologías de comunicación de sistemas distribuidos

Se analizaron distintas tecnologías de comunicación en sistemas distribuidos, que permiten establecer una comunicación confiable entre los componentes del sistema. Las tecnologías analizadas son: Java RMI, Java RMI sobre HTTP, SOAP Web Services y RESTful Web Services. Más información sobre diferencias entre Java RMI y WebServices en [17, 16].

- **Java RMI**: (*Java Remote Method Invocation*) API de Java que implementa invocación de métodos remotos orientada a objetos, basado en RPC [1] (*Remote Procedure Call*). Sus ventajas son: fácil integración entre aplicaciones Java, uso de objetos remotos similar a objetos

[1] RPC (*Remote Procedure Call*): es un esquema de comunicación entre procesos, que permite ejecutar rutinas de programas en otro espacio de memoria u otro computador.

locales, protocolo de alto rendimiento y bajo tráfico. Sus desventajas son: alto acoplamiento entre cliente/servidor, difícil integración con otras tecnologías, limitaciones en protocolo de comunicación, debido a uso de puertos propios y sin HTTP. Alternativamente existe la posibilidad de implementar RMI sobre HTTP, con la adición de un servidor de aplicaciones, como Apache Tomcat, esto reduce limitaciones de comunicación (por uso de estándar HTTP). Sin embargo, el uso de RMI sobre HTTP reduce la velocidad de la comunicación entre cliente y servidor, siendo de un orden de diez veces más lento que RMI puro.

- **SOAP Web-services**: Es un protocolo de servicios Web basado en SOAP[2]. En vez de usar objetos remotos, este protocolo permite publicar servicios, sin estados. Sus ventajas son: utilización de protocolo HTTP, envío de datos usando XML, lo que lo hace fácilmente interoperable con distintas tecnologías; también es posible agregarle seguridad mediante el protocolo WS-Security. Sus desventajas son: menor rendimiento comparado con RMI, más complejo de configurar y requiere la utilización de un servidor de aplicaciones adicional.

- **RESTful Web-services**: Este protocolo es similar a SOAP, pero a difierencia de SOAP, se utilizan mensajes HTTP (get, post, put) para la transferencia y consulta de datos. Sus ventajas son: bajo acoplamiento entre cliente/servidor, integrable entre distintas tecnologías, comunicación segura con HTTPS. Sus deventajas son: menor rendimiento que RMI, más complejo de implementar que SOAP o RMI.

Dadas estas opciones se ha escogido seguir utilizando RMI (no reemplazarlo por otra tecnología). Esto debido a que el rendimiento de la comunicación entre los componentes del sistema es una cualidad importante, lo cual permite disminuir el tiempo de votación de los votantes, junto con el tiempo de ejecución de los protocolos criptográficos utilizados entre los servidores del sistema. El sistema no requiere integrar distintas tecnologías. Las limitaciones de comunicación no son un problema, ya que el sistema debe desplegarse sobre una red de servidores dedicados que deben ser configurados especialmente para este fin. La seguridad en la comunicación entre los componentes (privacidad e integridad de datos) se puede lograr, por ejemplo, con el uso de una red VPN[3] (*virtual private network*).

### Seguridad de la comunicación entre los módulos del sistema

Se estudiaron distintas maneras de agregar seguridad en los canales de comunicación entre los componentes del sistema, es decir, asegurar integridad y privacidad en la transferencia de los datos. Se analizó agregar a Java RMI la utilización de puertos SSL, usar RMI sobre HTTP(S), o el uso de una red VPN.

- **Java RMI + SSL**[4]: Esto permite encriptar los datos que se envian por RMI, y autenticar a los clientes. Para esto es necesario modificar la implementación de servidor y clientes, utilizando puertos SSL en las peticiones RMI.

- **Java RMI sobre HTTP(S)**: Se utiliza un servidor de aplicaciones para publicar los servicios RMI, la configuración de seguridad (HTTPS) es independiente de la implementación y recae sobre el servidor de aplicaciones.

[2]SOAP (*Simple Object Access Protocol*): protocolo de publicación de servicios simple, que implementa el concepto de RPC.

[3]VPN (*Virtual Private Network*): es una tecnología de red que permite una extensión de la red local sobre una red pública o no controlada, como por ejemplo Internet.

[4]SSL (*Secure Socket Layer*): protocolo criptográfico que provee comunicación segura sobre Internet.

- **Red VPN**: Es posible crear una red VPN sobre la cual se montan tanto los servidores como los clientes, de esta forma se puede dar seguridad a los datos enviados, y se logra que los equipos no estén disponibles en Internet, sino que sólo dentro de la red privada. Esta opción es totalmente independiente de la implementación del sistema. En ciertos casos donde pudieran haber *firewalls* que bloqueen el tráfico no TCP entre los clientes y el servidor Web, sería necesario configurar el servidor VPN sobre TCP a través del puerto 80.

De entre estas opciones se ha escogido utilizar una red VPN, ya que es la opción menos acoplada con el resto del sistema, haciendo innecesario modificar el software actual y evitando la necesidad de agregar un servidor de aplicaciones al BulletinBoard.

### 2.3.3. Registro de eventos del sistema

Un sistema de votación electrónica es un tipo de aplicación crítica que debe tener un cuidadoso manejo de los errores o excepciones, junto con recopilar detalladamente los eventos importantes que ocurran durante el funcionamiento del sistema, tanto de los eventos o errores esperados como de los no esperado. Existen al menos dos frameworks de logging muy utilizados en Java: Log4J y Java Logging API. Ambos *frameworks* tienen similares características, siendo los dos configurables usando archivos de configuración fuera de compilación, pueden usarse incluso para enviar mensajes de log mediante email o RSS[5]. En el sistema se utilizará Java Logging API por estar incluido en el runtime de Java, reduciendo el uso de librerías externas al sistema.

### 2.3.4. Uso de sistemas operativos preconfigurados

#### Creación de sistemas operativos personalizados

Se pueden crear sistemas operativos personalizados, los cuales se construyen a partir de otro sistema existente y se seleccionan manualmente todos los programas o paquetes que serán instalados en el sistema. Con este proceso se puede obtener un sistema operativo minimal que contiene solo lo necesario para la ejecución del sistema. También es posible construir sistemas *live-cd* (o *live disc*), estos pueden cargarse directamente sobre la memoria RAM de un computador, sin necesidad de instalar el sistema operativo de manera persistente en el disco duro. Estos sistemas pueden ser un disco CD, un disco DVD, o una memoria *flash USB*. Para la construcción del *live-cd* se analizaron las siguientes herramientas:

- **LiveBuild**: antes llamado *live-helper*, es una herramienta de Debian Linux que permite construir imágenes personalizadas de basadas en Debian, de forma sencilla mediante una interfaz de línea de comandos.
- **Suse Studio**: es un servicio de Novell que mediante una interfaz Web permite construir y personalizar un sistema basado en Suse Linux.
- **Ubuntu from scratch**: es un proceso para construir una imagen personalizada de Ubuntu Linux a partir de cero, es un proceso completamente manual, difícil de aplicar.

[5]RSS (*Rich Site Summary*): es un formato de notificación de cambios en el contenido Web de un sitio.

#### Uso de sistemas operativos en servidores virtuales privados

Se estudió la factibilidad de utilización de servidores VPS (servidores virtuales privados en *cloud computing*) para instalar los servidores de votación mediante *live-cd's*, junto con pruebas de comunicación entre distintos servidores con RMI.

En el mercado existen varias empresas internacionales que dan servicios *cloud computing* de VPS a bajo precio, con alta disponibilidad y con esquemas de respaldo y persistencia de datos, entre los más conocidos están: Amazon Elastic Computing Cloud (EC2), RackSpace y ElasticHosts. Otras empresas que proveen estos servicios son: GoGrid, SoftLayer y Serverlove.

Se hicieron pruebas con el servicio de ElasticHosts, el cuál permite probar su sistema por una semana sin costo. Este servicio permite escoger entre varios sistemas operativos que tienen precargados, desde Windows hasta distintas distribuciones Linux. Se usaron servidores Ubuntu Server 2010, y se probó la comunicación con RMI, lo cuál funcionó sin problemas, siendo de fácil administración (mediante terminal SSH).

Entre todos los servicios analizados, sólo algunos permiten utilizar una distribución de Linux propia (subir una imagen propia sistema operativo). Los servicios encontrados que lo permiten son: ServerLove y ElasticHosts.

El uso de estos servidores podría reducir los costos de instalación, despliegue y mantención de una elección por parte de una institución que no tiene los recursos de hardware disponibles para instalar el sistema. Lo que haría que el sistema sea accesible para una mayor cantidad de personas o instituciones.

### 2.3.5. Dispositivos para votación

#### Utilización de unidad de memoria *flash* USB

Es importante tomar resguardos a la hora de utilizar *pendrives* para transportar datos del votante, ya que se podría intentar atacar el sistema insertando código auto-ejecutable en el dispositivo. Para mitigar éste y otros riesgos, las estaciones de votación deben cumplir las siguientes condiciones:

- Los *pendrives* deben ser montados sin permisos de ejecución.
- No se debe permitir que el votante tenga acceso a los recursos del sistema, estos recursos son: la unidad *flash*, los datos de configuración del sistema y los datos guardados del sistema. El votante sólo debe tener acceso a los recursos a través de la interfaz de usuario de la cámara de votación.
- En el proceso de montaje de la unidad *flash* no debe auto-ejecutarse ningún tipo de archivo.

#### *Tokens* de seguridad

Se estudió la factibilidad de utilización de tokens criptográficos de seguridad de tipo USB, son dispositivos con memoria y capacidad de procesamiento propios, que pueden generar claves, firmar o encriptar documentos. Estos se conectan a un computador usando un puerto USB.

Se probó con dispositivos eToken PRO, de la marca Alladin. Alladin provee de un SDK[6] para la utilización de los tokens para Windows y Linux. Una restricción de nuestro sistema es la utilización de Linux como plataforma.

[6]SDK (*Software Development Kit*): es tipicamente un conjunto de herramientas de desarrollo que permite la creación de aplicaciones

Las pruebas consideraban poder utilizar estos tokens en Linux, pero las pruebas no fueron exitosas, debido a que el SDK provisto por Alladin para Linux es de pago. Tampoco es posible utilizarlos con herramientas *open source* disponibles, ya que tienen ciertos problemas de compatibilidad con estos dispositivos.

Estos *tokens* no podrán ser utilizados en el sistema, pero no se descarta que se pudieran utilizar en el futuro otros dispositivos que sí funcionen correctamente en linux o que sean más accesibles.

# Capítulo 3

# Situación previa

Como ya se ha presentado, el objetivo de este trabajo de título es extender y mejorar el prototipo de sistema de votación electrónica existente (Evoting) que ha sido desarrollado por el CLCERT de la Universidad de Chile a partir de la memoria de Pamela Cordero[3]; la cual ha sido utilizado en elecciones estudiantiles y académicas del Departamento de Ciencias de la Computación de la Universidad de Chile. En este capítulo se describe el sistema existente y su estado actual de desarrollo.

El sistema de votación original es de tipo presencial en ambiente controlado (*polling station e-voting*): el registro se hace por entidades autorizadas, la verificación de identidad puede ser física o electrónica, los procesos de votación y escrutinio se hacen de manera electrónica, la elección se lleva a cabo en recintos electorales especialmente habilitados y asegurados, y el voto es enviado al servidor central por medio de una conección remota segura.

## 3.1. Características del sistema

- Esquema de votación electrónica presencial, de tipo *polling place*, utilizando estaciones de votación protegidas. Esto es importante ya que permite reducir los riesgos de seguridad y evita la coerción.
- Interfaz de usuario amigable y simple de usar, minimizando la posibilidad de errores al emitir un voto. Estas interfaces han sido probadas públicamente.
- Utiliza un modelo de encriptación homomórfica, implementando de forma eficiente los procesamientos criptográficos, para múltiples candidatos.
- La papeleta de votación (voto encriptado) es público.
- Garantiza que los votos sean anónimos y verificables.
- Garantiza que el recuento de los votos sea realizado sin errores, contando la totalidad de los votos válidamente emitidos, sin contar votos que fueran alterados o reemplazados.
- El sistema permite distribuir el proceso de votación entre tres autoridades (servidores). El distribuir la votación entre múltiples autoridades permite que la votación sea resistente ante intentos de manipulación de los datos y colusión de autoridades. Se necesita que al menos una cantidad mayor a la mitad de las autoridades no sea corrupta para preservar la validez e integridad de la elección.

## 3.2. Tipos de usuarios y casos de uso del sistema

En el cuadro 3.1 se listan todos los tipos de usuarios del sistema.

| Actor | Descripción |
|---|---|
| Votante | Es el usuario primario del sistema. Puede votar en las carreras en las que se encuentre habilitado. |
| Observador Web | Actor primario del sistema. Puede ser un votante o cualquier otra persona. Puede revisar el estado de un voto y consultar el resultado de la elección. |
| Vocal de Mesa (o Vocal) | Es un actor secundario, encargado de la identificación física del votante y de utilizar el módulo de la Mesa. |
| Administrador de BulletinBoard | Actor secundario, encargado de la ejecución del servidor BulletinBoard. |
| Administrador de Autoridad | Actor secundario, encargado de la ejecución del módulo Autoridad. |

Cuadro 3.1: Lista de tipos de usuarios del sistema

### 3.2.1. Casos de uso

En la Figura 3.1 se presenta un diagrama de casos de uso, donde se muestran los casos de uso (o funcionalidades) más relevantes del sistema. A continuación se describen los casos de uso:

**Emitir voto:** El *votante* puede emitir su voto seleccionando sus preferencias en una pantalla.

**Autorizar votante:** El *vocal de mesa* autentica al votante y lo autoriza a votar, el sistema provee la funcionalidad de verificar si el votante está habilitado para votar.

**Desautorizar votante:** El *vocal de mesa* registra en el sistema el término de la votación de un votante.

**Consultar estado de un voto:** El *observador Web* puede consultar por el estado de su voto en el sistema, y verificar que el voto sea válido.

**Consultar resultados elección:** El *observador Web* puede consultar los resultados finales de la elección una vez que estos han sido publicados en el sitio Web.

**Generar parámetros para elección:** El *administrador de BulletinBoard* inicia el proceso de generación de parámetros aleatorios que serán utilizados por los protocolos de votación.

**Iniciar servidor:** El *administrador de BulletinBoard* inicia el servidor de votación, con esto puede comenzar la elección.

**Obtener resultados:** El *administrador de BulletinBoard* inicia el proceso de obtención de los resultados finales de la elección y publica estos resultados en el servidor Web.

**Generar claves del sistema:** El *administrador de Autoridad* inicia el proceso de generación de claves distribuida del sistema en una Autoridad.

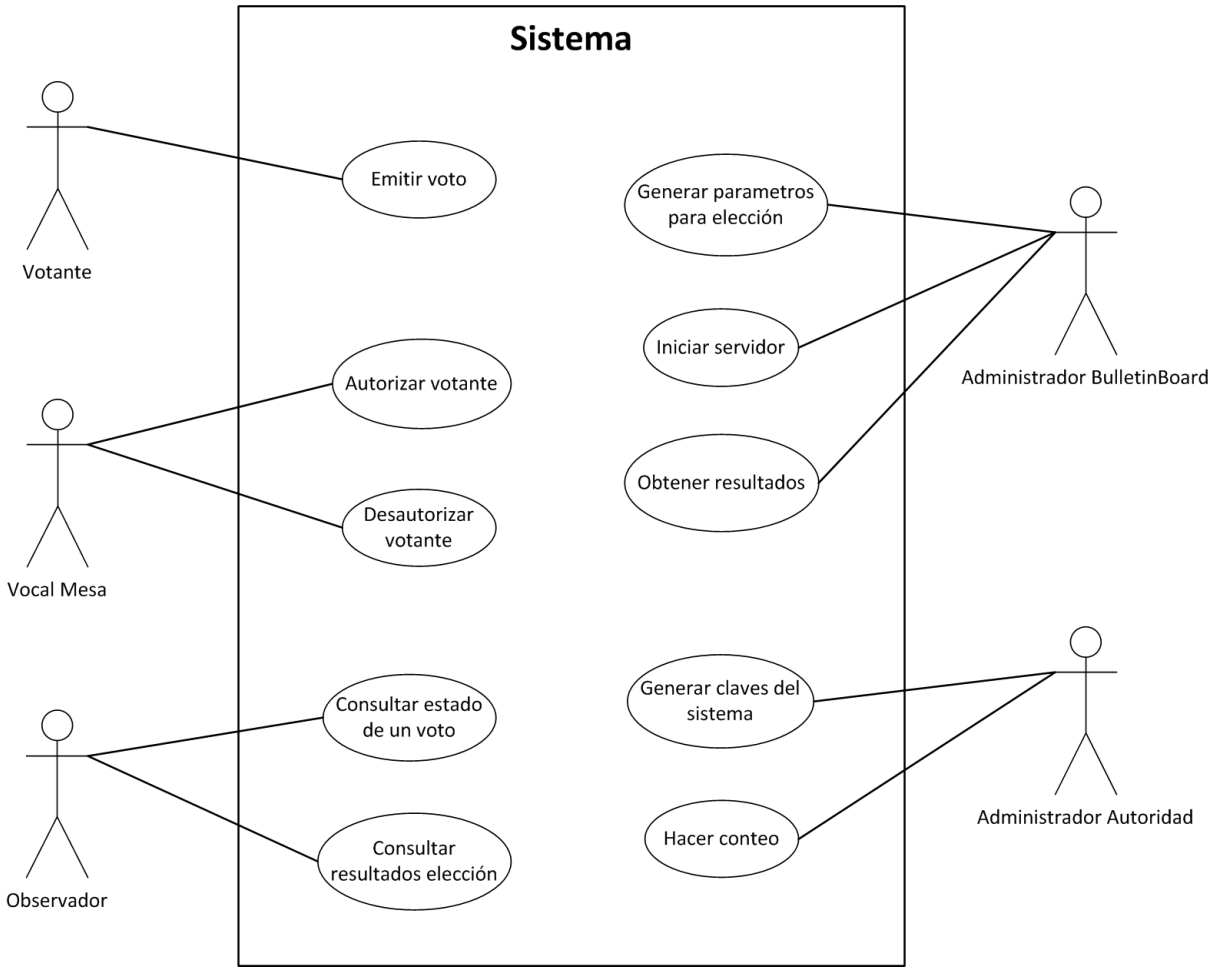


Figura 3.1: Casos de uso de la implementación previa del sistema

**Hacer conteo:** El *administrador de Autoridad* inicia el proceso de conteo distribuido en una Autoridad, se cuentan todos los votos encriptados válidos.

## 3.3. Componentes del sistema original

El prototipo de votación está compuesto de los siguientes módulos o componentes de software desde el punto de vista lógico:

**BulletinBoard:** Servidor o repositorio central de votos emitidos.

**Autoridades de Conteo:** Tres servidores independientes que realizan el escrutinio de los votos (de forma distribuida).

**Sitio Web de despliegue de resultados:** Sitio Web donde se puede consultar el estado de los votos y los resultados de la elección.

**Mesas de Votación:** Módulo de identificación de los votantes. Se da autorización al votante para acceder a la Cámara de Votación.

**Cámaras de Votación:** Cámara secreta donde el votante puede emitir su voto.

El sistema utiliza como estrategia de comunicación el patrón de repositorio de mensajes (patrón *repository*). En este patrón los datos son depositados en un servidor central (el BulletinBoard) y cada componente obtiene o deposita datos cuando es necesario, de acuerdo al protocolo del sistema. Bajo este esquema, cada componente es un *cliente*, donde el *servidor* es el BulletinBoard. Un caso distinto es el módulo del *Observador Web*. En este caso los datos son copiados desde el BulletinBoard al servidor Web público, utilizando una conección unidireccional. Es decir, el servidor Web no puede conectarse al BulletinBoard. En la Figura 3.2 se muestra la arquitectura física del sistema con los detalles de despliegue del sistema.

## 3.4. Fases del proceso de elección

El protocolo utilizado por el sistema de votación para realizar una elección se divide en cuatro fases:

### 3.4.1. Fase 1: Inicialización

- Se generan parámetros aleatorios iniciales (en el BulletinBoard) que son utilizados a lo largo del proceso.
- Las Autoridades de Conteo generan un par de claves pública y privada de manera distribuida. La clave pública es publicada en el BulletinBoard.

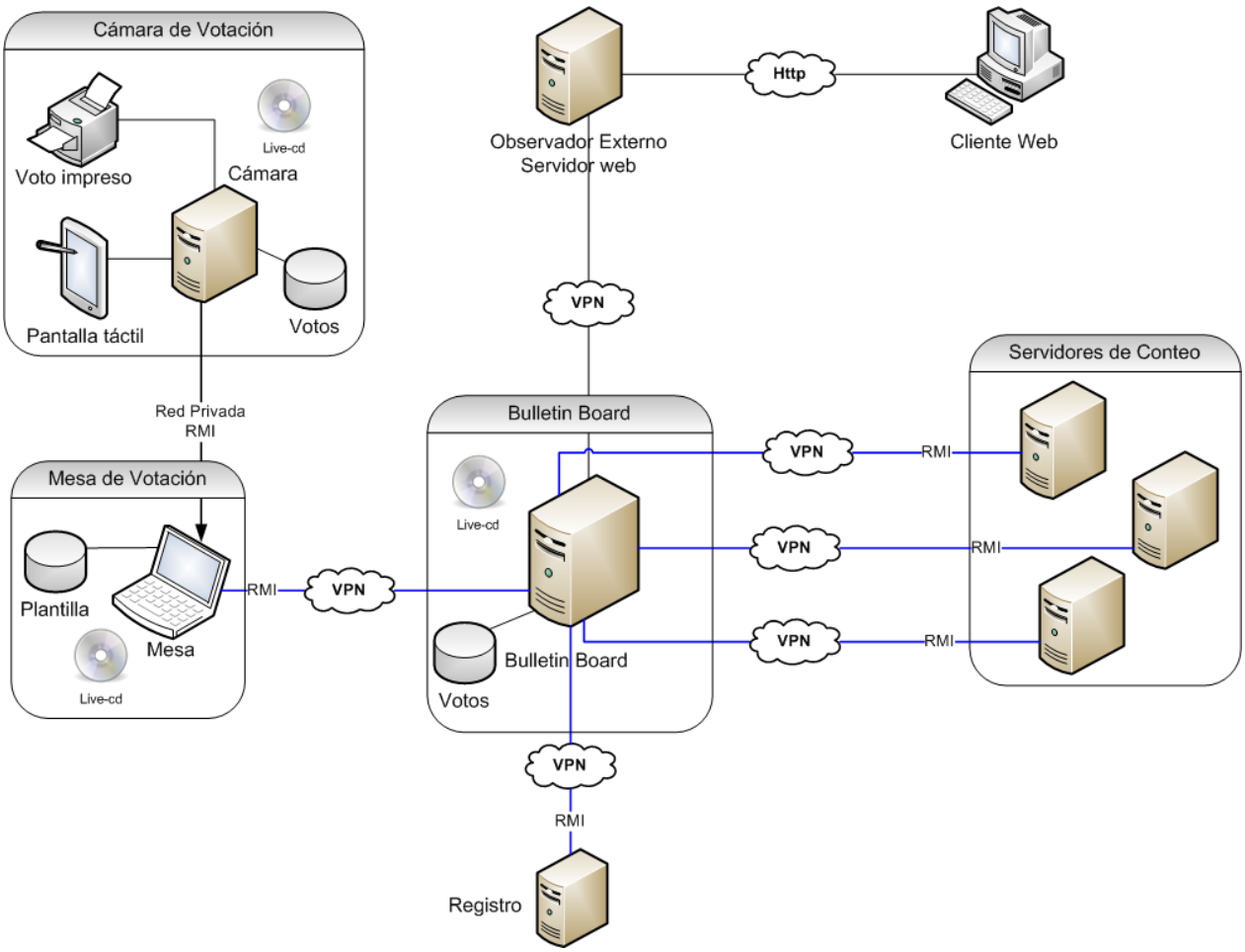


Figura 3.2: Arquitectura física del sistema

### 3.4.2. Fase 2: Generación y depósito de papeleta de votación

- El votante se identifica en la *Mesa de Votación*, se generan el par de claves pública y privada del votante. La clave pública del votante es publicada en el BulletinBoard.
- El votante ingresa a la *Cámara de Votación*, se emite el voto con las preferencias del votante, el voto es encriptado con la clave pública generada por las autoridades, y se genera un "checksum" de la correctitud del voto (o *proof* ZKP[1]). Estos valores se firman usando la clave privada del votante que en conjunto forman la *papeleta de votación*. La papeleta es publicada en el BulletinBoard.
- Aparte de la papeleta digital enviada al BulletinBoard, se utiliza la técnica *paper trail*, para lo cual se imprime una versión física de la papeleta, la que es validada por el votante e ingresada en una urna. Las papeletas impresas son un respaldo de los votos, que puede ser utilizada para un recuento en caso de ocurrir una falla irrecuperable del sistema. Este proceso de *paper trail* además permite facilitar una transición cultural desde sistemas en papel hacia un sistema de votación completamente digital.

[1]ZKP (*Zero Knowledge Proof*): El detalle matemático del esquema ZKP utilizado e implementado por el sistema está descrito en [3], página 26.

### 3.4.3. Fase 3: Verificación de la papeleta de votación

- Las autoridades confirman la validez del voto verificando su firma y *proof* ZKP. El resultado de la verificación es publicado en el BulletinBoard.

### 3.4.4. Fase 4: Escrutinio

- Las autoridades, de acuerdo a la propiedad homomórfica del sistema, calculan el producto de los votos encriptados, luego este resultado es desencriptado obteniendo la suma de los votos (sólo se cuentan votos verificados válidos por al menos dos autoridades). Ningún voto es desencriptado individualmente, manteniendo así la privacidad del voto.
- Las autoridades desencriptan el total de votos de manera distribuida (desencriptación por capas). Sólo se requieren dos autoridades para realizar la desencriptación, y éstas deben generar *proofs* para garantizar la validez de las desencriptaciones parciales.

## 3.5. Protocolos criptográficos del sistema

Los protocolos utilizados por el sistema de votación utilizan técnicas criptográficas. En cada etapa de estos protocolos se generan *outputs* o mensajes que son publicados en el servidor BulletinBoard. Estos protocolos pueden ser verificados por un observador externo, o verificador, mediante el análisis de los *outputs* publicados. Los protocolos son los siguientes:

- Protocolo de generación distribuida de claves del sistema (protocolo DKG).
- Protocolo distribuido de recuento de los votos (protocolo Revisión)

A continuación se describen estos protocolos, pero sin entrar en los detalles matemáticos y criptográficos de éstos, ya que está fuera del alcance de este trabajo analizar o modificar es esquema criptográfico utilizado por el sistema:

### 3.5.1. Protocolo de generación distribuida de claves del sistema (protocolo DKG)

Este protocolo consta de 7 etapas, que cada Autoridad ejecuta de forma sincronizada. En cada etapa cada Autoridad ejecuta ciertas operaciones o cálculos y publica sus resultados en el BulletinBoard, los cuáles van firmados por la Autoridad y acompañados de un *checksum* para conservar su integridad. También se incluyen *proofs* ZKP que permiten comprobar que se hicieron los cálculos correctamente.

Al comenzar una siguiente etapa, cada Autoridad solicita los resultados publicados por el resto de las autoridades en el paso anterior. Los resultados se validan y comparan entre sí.

Cada Autoridad puede realizar quejas sobre el resto si el *proof* o el *checksum* es inválido.

Cuando una Autoridad recibe al menos dos *quejas* válidas, entonces esta Autoridad es excluida del resto del proceso por ser considerada deshonesta.

El resultado de este protocolo es un par de claves criptográficas pública y privada. La clave pública es enviada al BulletinBoard, mientras que la clave privada está distribuida en partes. Cada Autoridad tiene una parte y no conoce las partes del resto. La clave pública generada es utilizada para la encriptación de las papeletas de votación.

El detalle matemático y la comprobación de este protocolo puede ser revisado en [3], página 32.

### 3.5.2. Protocolo distribuido de recuento de los votos (protocolo Revisión)

Este protocolo consta de cuatro etapas, las cuales son ejecutadas por cada Autoridad de forma sincronizada. En cada etapa cada Autoridad ejecuta ciertas operaciones o cálculos y publica sus resultados en el BulletinBoard, los cuáles van firmados por la Autoridad, y acompañados de un *checksum* para conservar su integridad. También se incluyen *proofs* ZKP que permiten comprobar que se hicieron los cálculos correctamente. Los resultados calculados por las *Autoridades* consideradas deshonestas son ignorados.

Al comenzar una siguiente etapa cada Autoridad solicita los resultados publicados por el resto de las autoridades en el paso anterior, se validan y comparan los resultados de las autoridades.

Para hacer el recuento de votos primero se analizan todas las papeletas publicadas y se toman en cuenta sólo las papeletas válidas (mediante revisión de su *checksum* y *proof* ZKP). El recuento de los votos se hace sin desencriptar cada voto, sino que éstos son multiplicados, luego cada Autoridad envía el resultado de la mutiplicación, después se desencripta sólo el resultado. Esto es posible gracias a la propiedad homomórfica del esquema de encriptación utilizado.

El proceso de desencriptación se realiza por capas, cada Autoridad desencripta parcialmente los resultados usando su propia parte de la clave privada. Finalmente el resultado de todas las desencriptaciones parciales es la desencriptación final.

Cada Autoridad publica los resultados finales en el BulletinBoard.

El detalle matemático y la comprobación de este protocolo puede ser revisado en [3], páginas 47 a 49.

## 3.6. Interfaces de usuario del sistema

### 3.6.1. Interfaz de la Cámara de votación

La interfaz de usuario del módulo de la *Cámara* está implementada utilizando Java SWT y fue diseñada para ser utilizada en pantallas táctiles (usando *stylus*). A continuación se muestran algunas pantallas de la aplicación.

En la Figura 3.3 se muestra la pantalla donde se le presenta al votante el listado de candidatos que se puede seleccionar.

En la Figura 3.4 se muestra una pantalla donde se resumen las selecciones realizadas por el votante mientras se imprimen las papeletas de votación en papel.

### 3.6.2. Interfaz de la Mesa de votación

La interfaz de la *Mesa* está implementada con Java SWT. Es una interfaz sencilla con tres opciones: Autorizar votante, devolver *token* y cerrar la *Mesa*. En la Figura 3.5 se muestra la pantalla principal de la aplicación.


Figura 3.3: GUI Cámara, pantalla de selección de candidatos

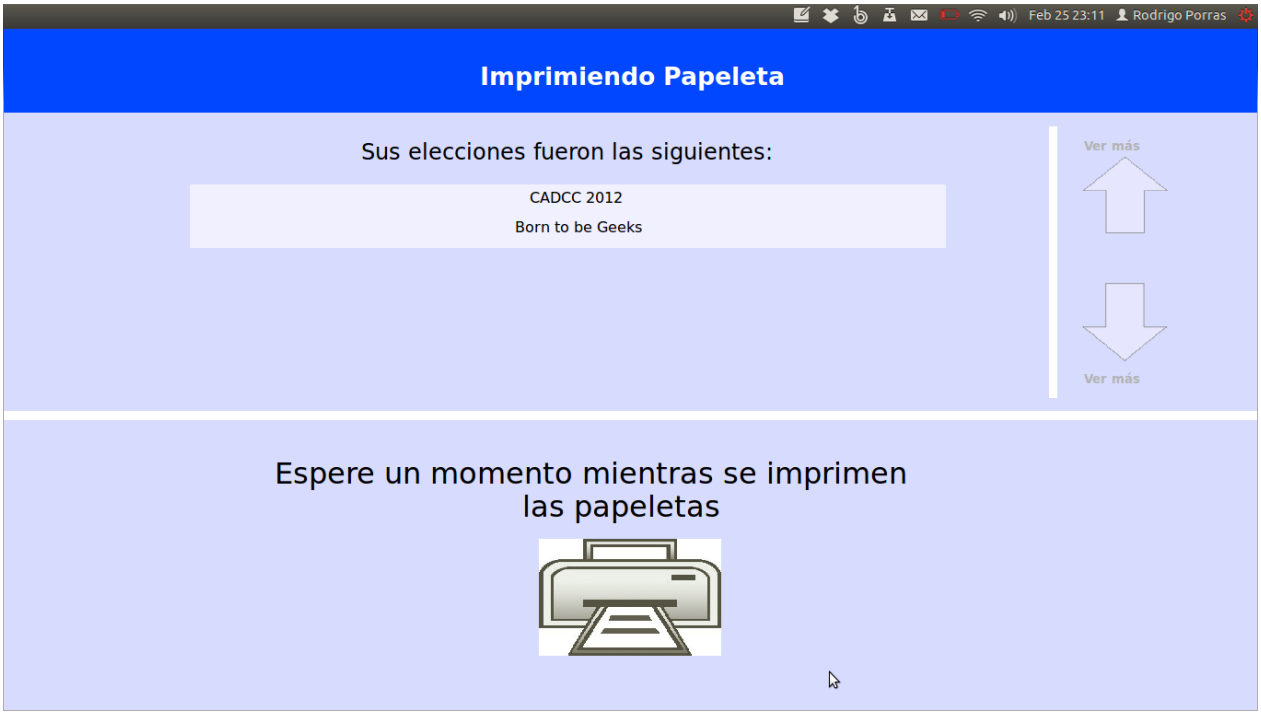


Figura 3.4: GUI Cámara, pantalla de impresión de papeletas

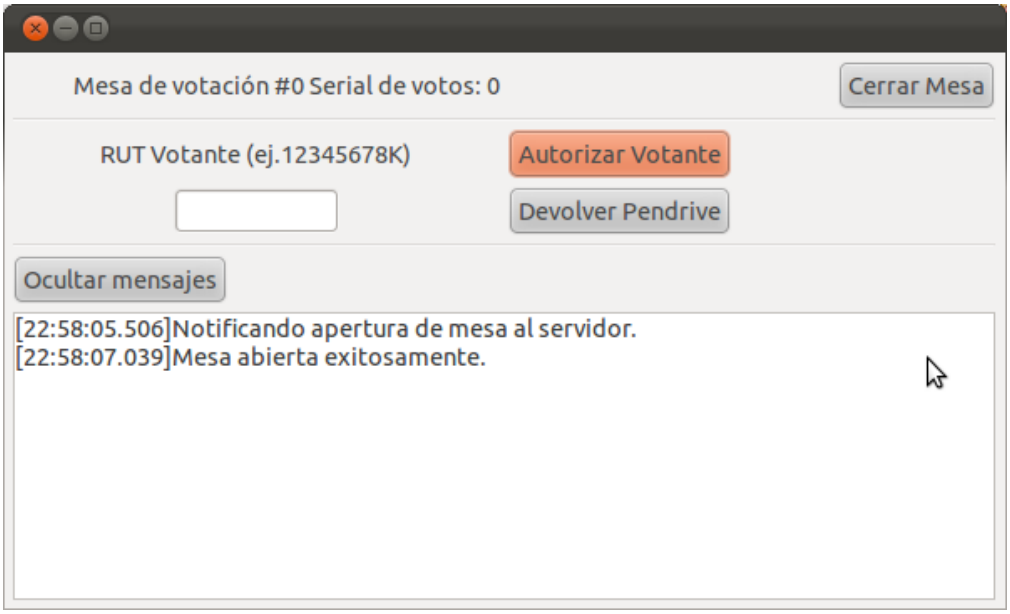


Figura 3.5: GUI Mesa, pantalla principal

### 3.6.3. Interfaz del sitio Web del Observador Web

La interfaz del sitio Web del *Observador Web* está implementada con PHP, en este sitio los votantes u observadores pueden consultar por el estado de un voto, y ver los resultados finales de la elección. En la Figura 3.6 se muestra la sección para consultar por el estado de un voto.

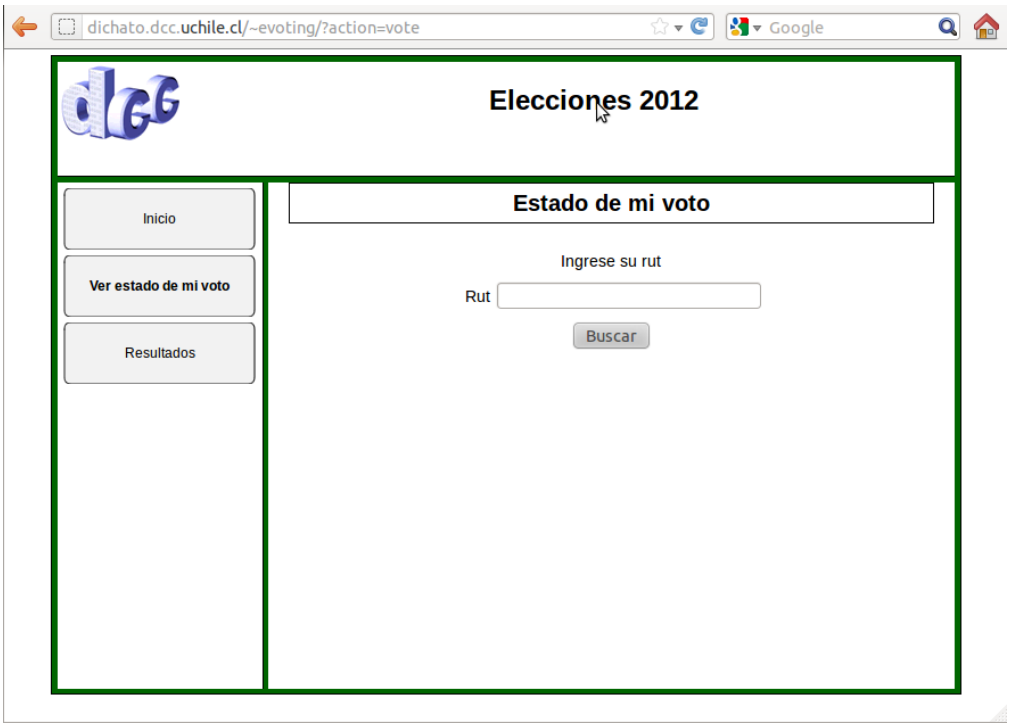


Figura 3.6: GUI Observador Web, página de consulta de estado de votos

# Capítulo 4

# Requerimientos del sistema

Actualmente el sistema o prototipo tiene una implementación poco robusta, poco mantenible y no tiene una arquitectura de software bien diseñada. Además el sistema no está preparado para ser tolerante a fallas ni disponible frente a eventos de desconexión. Es muy dificil de configurar e instalar. Funcionalmente el sistema no tiene una herramienta que permita hacer uso de la propiedad de ser *universalmente verificable*.

Sin las características descritas no es factible usar este sistema en votaciones reales, ya que al no estar preparado frente a escenarios adversos se podría inutilizar e invalidar el sistema y los datos ya procesados, perdiéndose el esfuerzo e inversión en utilizar el sistema. Además al no tener una herramienta que permita verificar el proceso, éste se hace menos transparente, por lo tanto no es capaz de generar la confianza suficiente en los usuarios.

## 4.1. Requerimientos funcionales para la solución

El sistema debe conservar las funcionalidades existentes. Además se deben implementar las siguientes funcionalidades nuevas:

| Cód. | Título | Descripción |
|---|---|---|
| RF01 | Configurador elecciones | Herramienta de configuración de una nueva elección, que reciba como entrada los datos de la elección en formato CSV, y que entregue como salida un documento XML de especificación de la elección consistente y sin errores. |
| RF02 | Verificación elección | Aplicación gráfica para verificar todas las etapas del proceso de votación, utilizando los datos públicos de la votación actualizados. |
| RF03 | Instanciación de servidores | Los servidores deben instanciarse de forma guíada, utilizando live-cd's preconfigurados. |
| RF04 | Instanciación de clientes | Los computadores clientes deben instanciarse de forma guíada, utilizando live-cd's preconfigurados. |
| RF05 | Restauración de servidores | En caso de falla de servidores, deben tener la opción de restaurarse, sin perdida de información, utilizando live-cd's preconfigurados. |
| RF06 | Restauración de clientes | En caso de falla de un computador cliente, deben tener la opción de restaurarse sin pérdida de información, utilizando live-cd's preconfigurados. |

## 4.2. Requerimientos no funcionales para la solución

Junto con mantener los atributos de calidad que el sistema previo ya tenía, se requieren los siguientes atributos de calidad en la nueva implementación del sistema:

| Cód. | Categoría | Descripción |
|---|---|---|
| RC01 | Confiabilidad | Acceso concurrente a datos. El almacenamiento de los datos en el servidor de datos debe ser confiable, manejando apropiadamente concurrencia de acceso a los datos. |
| RC02 | Confiabilidad | Manejo de excepciones. Pueden ocurrir tanto excepciones esperadas como no esperadas. Todas deben ser atrapadas y manejadas adecuadamente. |
| RC03 | Confiabilidad | Disponibilidad. El sistema de votación debe seguir funcionando en caso de no disponibilidad de los servidores de votación, utilizando caché local persistente. |
| RC04 | Confiabilidad | Recuperabilidad. En caso de falla en algún componente del sistema, como corte de electricidad, el sistema debe recuperarse, volviendo al estado en que se encontraba antes, sin pérdida de información, manteniendo el sistema consistente. |
| RC05 | Confiabilidad | Auditabilidad. El funcionamiento del sistema debe ser auditable. En caso de fallas éstas deben registrarse en logs de eventos, mostrando información que permita conocer el motivo de la falla. |
| RC06 | Seguridad | Confidencialidad de datos. La claves privadas utilizadas por los votantes deben ser eliminadas una vez que han sido utilizadas sin dejar copias de ellas. |
| RC07 | Seguridad | Integridad de datos. Los datos enviados al servidor de votación deben transmitirse utilizando canales de comunicación seguros, de manera de prevenir que los datos sean modificados en el camino por un tercero. |
| RC08 | Usabilidad | Facilidad de administración. El sistema debe ser fácilmente administrable, requiriendo una capacitación breve en su operación y teniendo herramientas que faciliten su configuración e instanciación. |
| RC09 | Usabilidad | Facilidad de uso. El sistema debe ser fácil de usar, con interfaces de usuario intuitivas, sin requerir manuales de usuario. |
| RC10 | Mantenibilidad | La implementación del sistema debe ser mantenible, utilizando arquitecturas y patrones de diseño estándares. Se deben separar adecuadamente las responsabilidades de las componentes. |

# Capítulo 5

# Descripción de la solución

En este capítulo se describe de forma integral la solución diseñada e implementada, que permite satisfacer los requerimientos propuestos. La solución consiste en la implementación de las nuevas funcionalidades deseadas, junto con una refactorización de las componentes fundamentales del sistema, de esta manera es posible alcanzar los atributos de calidad propuestos, mejorando la robustez y confiabilidad del sistema.

## 5.1. Estrategia de implementación

Los atributos de calidad claves del sistema se corresponden con los requerimientos no funcionales analizados anteriormente, por lo tanto la arquitectura e implementación propuesta tiene que satisfacer estas necesidades. Podemos resumir los atributos de calidad de la siguiente forma:

**Confiabilidad:** Acceso concurrente a datos, manejo de excepciones, disponibilidad, recuperabilidad y auditabilidad.

**Seguridad:** Confidencialidad de datos e integridad de datos.

**Usabilidad:** Facilidad de administración, facilidad de uso.

**Mantenibilidad:** Capacidad de mantener y modificar y extender el sistema.

Para satisfacer los atributos de calidad necesarios se utilizarán las siguientes estrategias de implementación de software de forma transversal al sistema:

- Utilizar un proceso de re-ingeniería de software, de forma de mejorar la arquitectura y los atributos de calidad del sistema, y a la vez manteniendo las características y funcionalidades del sistema ya existentes.
- Mantener y mejorar la arquitectura de software basada en componentes, que se comunican sólo mediante interfaces públicas. Además mejorar arquitectura interna de los componentes, separando las responsabilidades mediante capas de abstracción como el encapsulamiento del acceso a datos y el acceso a los servicios. Así se separan las responsabilidades haciendo el sistema más modular y mantenible, fomentando la reusabilidad y menor acoplamiento entre sí.

- Utilizar una metodología basada en metodologías ágiles como *Kanban*, con énfasis en la integración continua, validación continua mediante pruebas unitarias y de integración, utilización de técnicas de *refactoring* de software, para reducir la redundancia de código y aumentar la mantenibilidad del sistema.

## 5.2. Arquitectura del sistema

Para la descripción formal de la arquitectura de software se usará el modelo "4+1" vistas propuesto por [18]. Con este modelo se describe la arquitectura de un sistema desde distintas vistas, donde cada vista aborda aspectos relevantes para los distintos *stakeholders* del sistema, permitiendo separar aspectos funcionales de aspectos estáticos, dinámicos y físicos.

### 5.2.1. Vista de casos de uso

La vista de casos de uso presenta las funcionalidades del sistema desde el punto de vista de los usuarios y *stakeholders*.

En la descripción de la situación previa se describieron los actores y casos de uso del sistema. En este trabajo se agregarán nuevas funcionalidades al sistema junto con nuevos actores. Los nuevos actores del sistema se describen en la tabla 5.1.

| Actor | Descripción |
|---|---|
| TRICEL (Tribunal calificador de elecciones) | Es el responsable de todo el proceso de votación, en particular es el encargado de crear y configurar una nueva elección, siendo responsable de que los datos de la elección utilizados por el sistema sean correctos. |
| Verificador Externo | Usuario primario del sistema, pueda verificar o auditar el proceso completo de votación del sistema. Este usuario puede ser un votante o cualquier persona interesada en comprobar el correcto (y honesto) funcionamiento del proceso. |
| Administrador de Servidor | Usuario secundario, es el encargado de la instanciación y restauración de los servidores del sistema, este actor puede ser un *Administrador de BulletinBoard* o *Administrador de Autoridad*. |
| Administrador de Cliente | Usuario secundario, es el encargado de la instanciación y restauración de los terminales cliente del sistema, este actor podría ser el mismo *Vocal de Mesa*. |

Cuadro 5.1: Listado de nuevos actores del sistema

Los nuevos casos de uso relevantes del sistema que serán implementados en este trabajo son los siguientes:

- **Caso de Uso: Configurar nueva elección.** Ver cuadro 5.2.
- **Caso de Uso: Verificar elección.** Ver cuadro 5.3.

- **Caso de Uso: Instalar servidor.** Ver cuadro 5.4.
- **Caso de Uso: Restaurar servidor.** Ver cuadro 5.5.
- **Caso de Uso: Instalar cliente.** Ver cuadro 5.6.
- **Caso de Uso: Restaurar cliente.** Ver cuadro 5.7.

El diagrama de casos de uso, donde se describe la relación entre los nuevos actores y los casos de uso, se puede ver en la Figura 5.1.

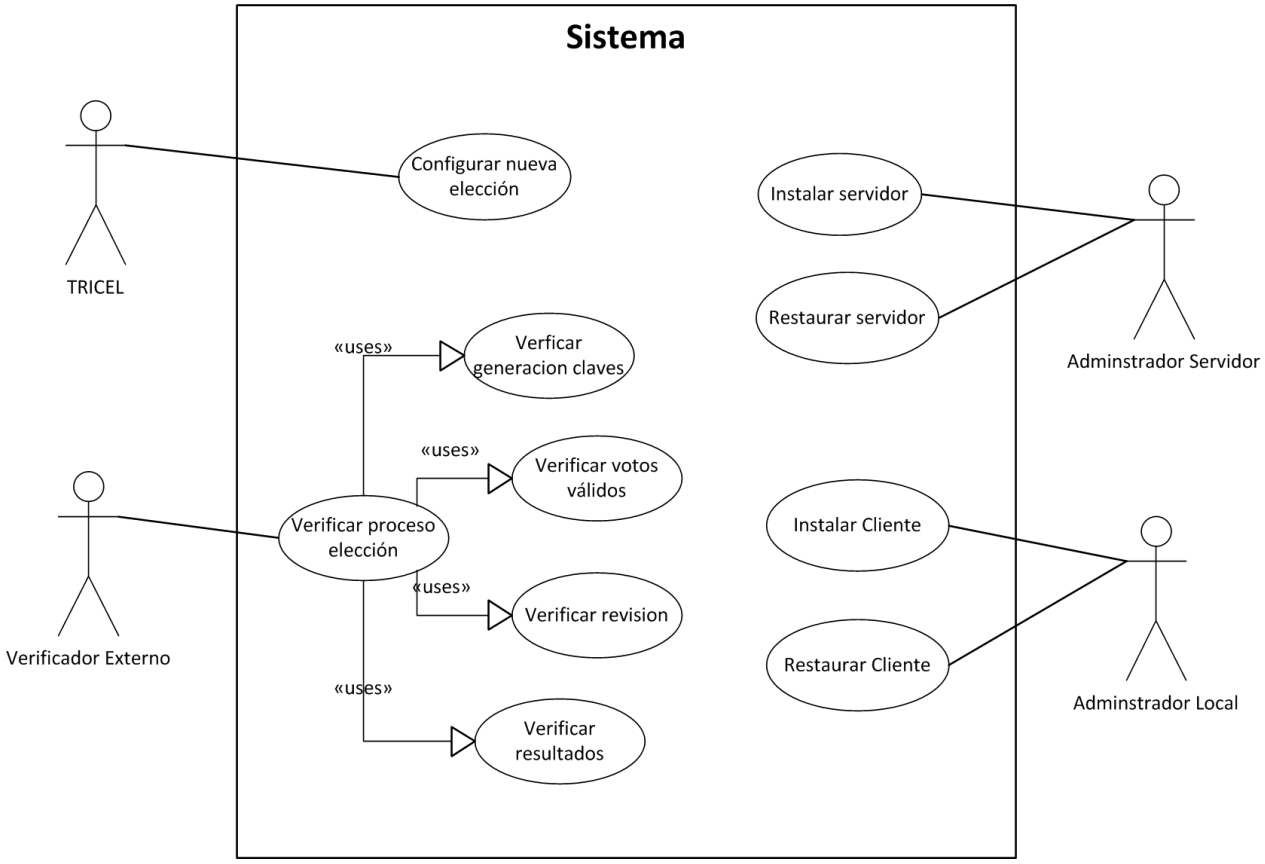


Figura 5.1: Nuevos casos de uso del sistema

| Caso de Uso: Configurar nueva elección | |
|---|---|
| **Identificador** | CU01: Configurar nueva elección |
| **Actor Principal** | TRICEL |
| **Descripción** | Permite configurar una nueva elección de forma confiable, a partir de datos creados por el usuario, con lo cuál se genera un documento oficial de configuración |
| **Pre-condiciones** | Usuario ha creado los archivos CSV usando la notación requerida por el programa |
| **Post-condiciones** | No tiene |

| Flujo Principal | | |
|---|---|---|
| **Pasos** | **Actor** | **Sistema** |
| 1 | Ingresa ruta de archivos CSV que contienen la información de la nueva elección | |
| 2 | | Procesa archivos CSV y valida datos de votación |
| 3 | | Genera archivo XML con definición formal de la nueva elección |

| Flujo Alternativo | | |
|---|---|---|
| **Pasos** | **Actor** | **Sistema** |
| 1 | Ingresa ruta de archivos CSV que contienen la información de la nueva elección | |
| 2 | | Los archivos CSV son inválidos, o los datos no son correctos. El programa termina e informa al usuario acerca del error encontrado |

Cuadro 5.2: Caso de Uso: Configurar nueva elección

| Caso de Uso: Verificar elección | |
|---|---|
| **Identificador** | CU02: Verificar elección |
| **Actor Principal** | Verificador Externo |
| **Descripción** | Permite verificar que el proceso de votación se realice correctamente, revisando los datos públicos que se han generado durante la elección, de esta forma se comprueba que los protocolos criptográficos de votación sean correctamente realizados, detectando si hubiera algún participante deshonesto. Al usuario se le muestra un reporte con el resultado de la verificación por cada etapa del proceso de votación (Inicialización, votos, conteo y resultados). |
| **Pre-condiciones** | El proceso de votación ya ha sido iniciado, está habilitado el servidor público con los datos generados durante la elección. |
| **Post-condiciones** | No hay |

| Flujo Principal | | |
|---|---|---|
| **Pasos** | **Actor** | **Sistema** |
| 1 | Da la orden de actualizar los datos del proceso | |
| 2 | | Se actualizan (descargan) los datos públicos del proceso desde el servidor web |
| 3 | Inicia el proceso de verificación | |
| 4 | | Se procesa la verificación de la etapa de inicialización del sistema |
| 5 | | Se verifican todos los votos almacenados en el sistema |
| 6 | | Se verifica la etapa de conteo de votos |
| 7 | | Se verifica (compara) el resultado oficial con el resultado obtenido en la verificación |
| 8 | | Se genera un reporte con los resultados de toda la verificación del proceso |

| Flujo Alternativo | | |
|---|---|---|
| **Pasos** | **Actor** | **Sistema** |
| 1 | Da la orden de actualizar los datos del proceso | |
| 2 | | Si no hay datos disponibles o falla la actualización se informa al usuario |

Cuadro 5.3: Caso de Uso: Verificar elección

| Caso de Uso: Instalar servidor | | |
|---|---|---|
| **Identificador** | CU03: Instalar servidor | |
| **Actor Principal** | Administrador de Servidor | |
| **Descripción** | Permite instalar (instanciar) un nuevo servidor del sistema (BulletinBoard o Autoridad) | |
| **Pre-condiciones** | La votación no ha comenzado, se ha Iniciado el live-cd del servidor | |
| **Post-condiciones** | | |
| **Flujo Principal** | | |
| **Pasos** | **Actor** | **Sistema** |
| 1 | Escoge instalar servidor | |
| 2 | | Se instala y configura el servidor (el usuario responde las preguntas realizadas durante el proceso) |
| **Flujo Alternativo** | | |
| **Pasos** | **Actor** | **Sistema** |
| 1 | Escoge instalar servidor | |
| 2 | | Si no es posible configurar el servidor se termina la instalación |

Cuadro 5.4: Caso de Uso: Instalar servidor

| Caso de Uso: Restaurar servidor | | |
|---|---|---|
| **Identificador** | CU04: Restaurar servidor | |
| **Actor Principal** | Administrador de Servidor | |
| **Descripción** | Permite restaurar (re-instanciar) un servidor del sistema ya instalado (BulletinBoard o Autoridad), que por algún motivo dejo de funcionar (por ej: corte de energía eléctrica) | |
| **Pre-condiciones** | El servidor ya fue instalado anteriormente, se ha Iniciado el live-cd del servidor | |
| **Post-condiciones** | | |
| **Flujo Principal** | | |
| **Pasos** | **Actor** | **Sistema** |
| 1 | Escoge restaurar servidor | |
| 2 | | Se restaura y reconfigura el servidor (el usuario responde las preguntas realizadas durante el proceso) |
| **Flujo Alternativo** | | |
| **Pasos** | **Actor** | **Sistema** |
| 1 | Escoge restaurar servidor | |
| 2 | | Si no es posible restaurar el servidor o no hay datos válidos de restauración se termina el proceso |

Cuadro 5.5: Caso de Uso: Restaurar servidor

| Caso de Uso: Instalar cliente | | |
|---|---|---|
| **Identificador** | CU05: Instalar cliente | |
| **Actor Principal** | Administrador de Cliente | |
| **Descripción** | Permite instalar (instanciar) un nuevo cliente del sistema (Cámara o Mesa de votación) | |
| **Pre-condiciones** | La votación no ha comenzado, se ha Iniciado el live-cd del cliente | |
| **Post-condiciones** | | |
| **Flujo Principal** | | |
| **Pasos** | **Actor** | **Sistema** |
| 1 | Escoge instalar cliente | |
| 2 | | Se instala y configura el cliente (el usuario responde las preguntas realizadas durante el proceso) |
| **Flujo Alternativo** | | |
| **Pasos** | **Actor** | **Sistema** |
| 1 | Escoge instalar cliente | |
| 2 | | Si no es posible configurar el cliente se termina la instalación |

Cuadro 5.6: Caso de Uso: Instalar cliente

| Caso de Uso: Restaurar cliente | | |
|---|---|---|
| **Identificador** | CU06: Restaurar cliente | |
| **Actor Principal** | Administrador de cliente | |
| **Descripción** | Permite restaurar (re-instanciar) un cliente del sistema ya instalado (Cámara o Mesa de votación), que por algún motivo dejo de funcionar (por ej: corte de energía eléctrica) | |
| **Pre-condiciones** | El servidor ya fue instalado anteriormente, se ha Iniciado el live-cd del cliente | |
| **Post-condiciones** | | |
| **Flujo Principal** | | |
| **Pasos** | **Actor** | **Sistema** |
| 1 | Escoge restaurar cliente | |
| 2 | | Se restaura y reconfigura el cliente (el usuario responde las preguntas realizadas durante el proceso) |
| **Flujo Alternativo** | | |
| **Pasos** | **Actor** | **Sistema** |
| 1 | Escoge restaurar cliente | |
| 2 | | Si no es posible restaurar el cliente o no hay datos válidos de restauración se termina el proceso |

Cuadro 5.7: Caso de Uso: Restaurar cliente

### 5.2.2. Vista lógica

En esta sección se describe la estructura del sistema, se muestran los subsistemas, las capas y se describe como estos artefactos colaboran entre sí.

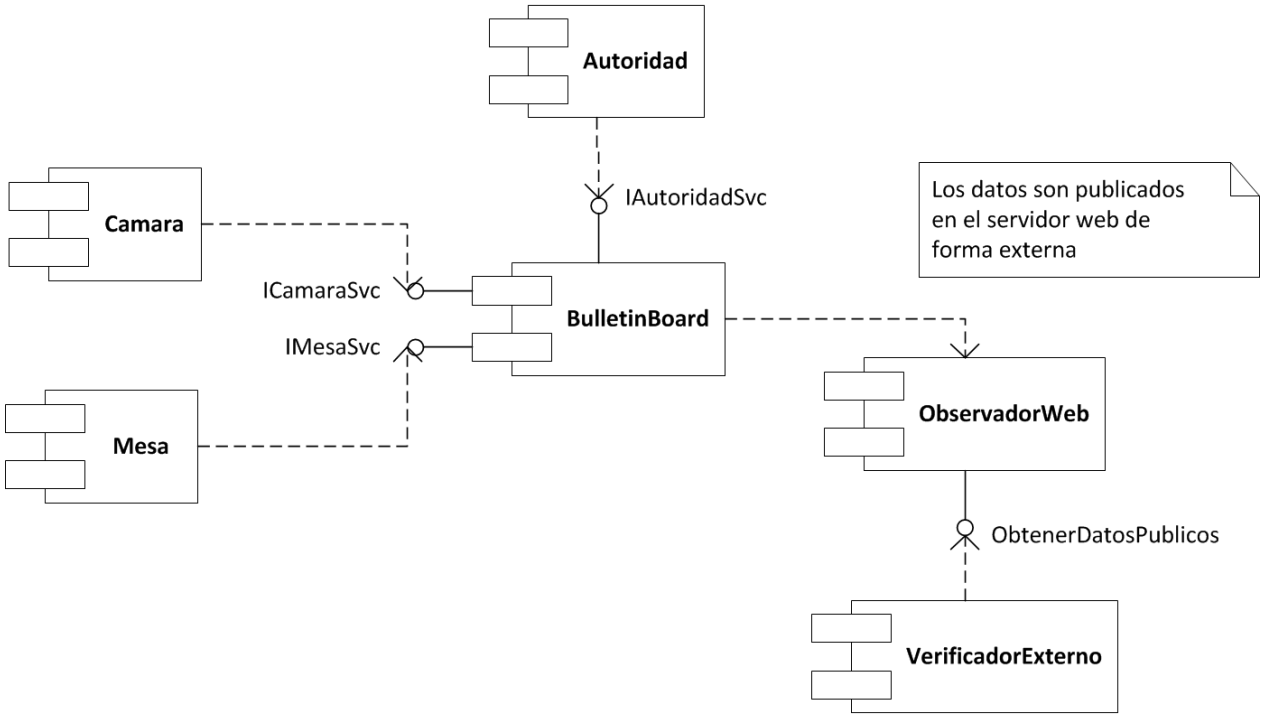


Figura 5.2: Diagrama UML de componentes del sistema

El sistema de votación electrónica se descompone en 5 subsistemas o módulos, donde se utiliza un modelo de comunicación cliente-servidor. El servidor se implementa usando el patrón arquitectónico de *repositorio* de mensajes (BulletinBoard), es decir que cada participante se encarga de publicar y consultar los mensajes desde el repositorio, sin recibir notificaciones. Para acceder al servidor de mensajes, cada módulo cuenta con una interfaz de servicios disponibles mediante una API. Los módulos que componen al sistema de votación son:

- BulletinBoard
- Autoridad
- Mesa
- Cámara
- Verificador Externo
- Configurador
- Observador Web

Como se puede ver en la Figura 5.2 el BulletinBoard tiene tres interfaces de servicio: IAutoridadSvc, ICamaraSvc e IMesaSvc, las cuales son accedidas por los módulos Autoridad, *Cámara* y *Mesa* respectivamente.

El módulo *Observador Web* consiste en un sitio web que muestra el estado y los resultados de la votación. Los datos de la votación son copiados desde el BulletinBoard en el servidor web. El módulo *Verificador Externo* consulta y descarga los datos desde el servidor del *Observador Web*.

El módulo *Configurador* no aparece en la Figura 5.2 ya que no interactúa directamente con el resto de los componentes. Este módulo se utiliza en la fase de creación de una nueva elección, antes de comenzar la ejecución del sistema de votación.

A continuación se descomponen y describen cada uno de los módulos del sistema.

### Módulo BulletinBoard

Este módulo es el servidor central del sistema, contiene el repositorio de mensajes, brinda servicios al resto de los componentes y gestiona los datos públicos generados durante la elección.

A este módulo se le aplica re-ingeniería de software, ya que la implementación anterior es poco mantenible, con exceso de redundancia de código y el almacenamiento de los datos es poco robusto. Como se puede ver en la Figura 5.3 la re-ingeniería consiste en separar las responsabilidades utilizando una arquitectura de capas de abstracción. Se reemplaza el sistema de almacenamiento de datos en archivos por un sistema de base de datos relacional, el acceso a los datos se realiza mediante el patrón de *Mapeo Objeto-Relacional* que permite gestionar los datos a través de objetos especializados. Las capas de abstracción son las siguientes:

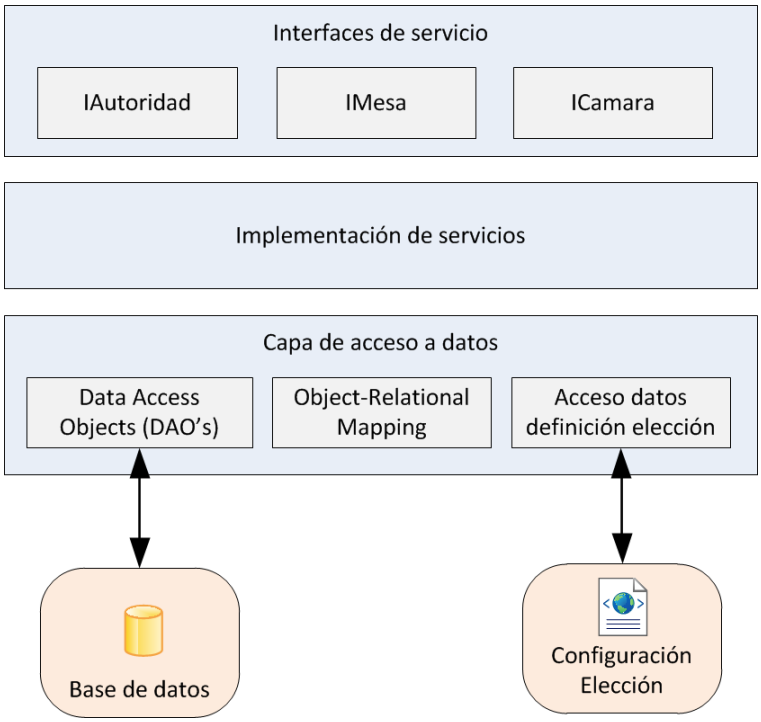


Figura 5.3: Arquitectura interna del módulo BulletinBoard

- **Capa de servicios**: Se definen los servicios (interfaces Java RMI) que el módulo brinda a otros componentes. Contiene las estructuras de datos y clases utilizadas en la transferencia de datos entre los componentes del sistema.

- **Capa lógica**: Se implementan los servicios definidos en la capa de servicios. Los datos son solicitados a la capa de acceso a datos, los datos son procesados o transformados para ser enviados.

- **Capa de acceso a datos**: Se encarga de gestionar el acceso físico a los datos, le entrega a la capa superior interfaces genéricas para acceder a los datos. Como estrategia de persistencia se utiliza el patrón arquitectónico *Mapeo Objeto-Relacional* (u *ORM*), que permite acceder a los datos en una base de datos relacional como si fueran objetos del lenguaje. Para esto se implementó un motor de *ORM* muy simple que contiene las operaciones básicas: *agregar entidad*, *obtener entidad*. También se incluye el patrón *Objetos de Acceso a Datos o DAO*, el que consiste en tener objetos especializados en la gestión de cada *entidad* del modelo de datos. En esta capa también se implementa el acceso a los datos de definición de la elección, en este caso los datos se leen desde un archivo XML y son entregados a la capa superior como objetos.

### Módulo Autoridad

El sistema de votación utiliza un protocolo criptográfico homomórfico, en el cuál los votos se encriptan utilizando criptografía asimétrica (clave pública y privada). El sistema de votación genera un par de claves pública y privada, con la clave pública se encriptan los votos, gracias a la encriptación homomórfica es posible hacer el recuento sin desencriptar los votos, y sólo desencriptando el resultado final (usando la clave privada del sistema de votación).

La clave privada del sistema es generada mediante el protocolo DKG (ver sección 3.5.1, página 30), que permite construir una clave de forma distribuida entre varios participantes, luego para utilizarla, los participantes deben utilizar otro protocolo para desencriptar también de forma distribuida. De esta forma se necesita más un participante o Autoridad para obtener el recuento de los votos.

Este módulo implementa los protocolos distribuidos DKG y Revisión, que se apoyan en el BulletinBoard para intercambiar mensajes y sincronizarse. Es relevante que se utilice un método para guardar estados consistentes de cada protocolo, es decir que si falla uno de los participantes, este se pueda recuperar y continuar desde el último punto consistente o *checkpoint*.

La arquitectura de este componente se pueda ver en la Figura 5.4. Se definen 3 capas:

- **Capa de presentación**: En esta capa se presenta una interfaz de usuario en línea de comando donde el usuario indica las operaciones a realizar.

- **Capa lógica**: Se implementan los protocolos criptográficos del sistema de votación, además de los esquemas de *checkpoint y restauración*.

- **Capa de acceso a datos**: Se controla el acceso a datos locales y se realizan las peticiones al servidor del sistema, para lo cuál se utiliza el patrón *Service Agent*, de esta forma se encapsula la responsabilidad de acceder a los servicios del servidor.

### Módulo Verificador-Externo

El sistema de votación tiene la cualidad de ser universalmente verificable, es decir que el desarrollo del proceso de votación puede ser auditado. En la práctica esto se traduce en que cada etapa del proceso genera datos públicos (que deben ser utilizados en el resto del proceso) los cuáles

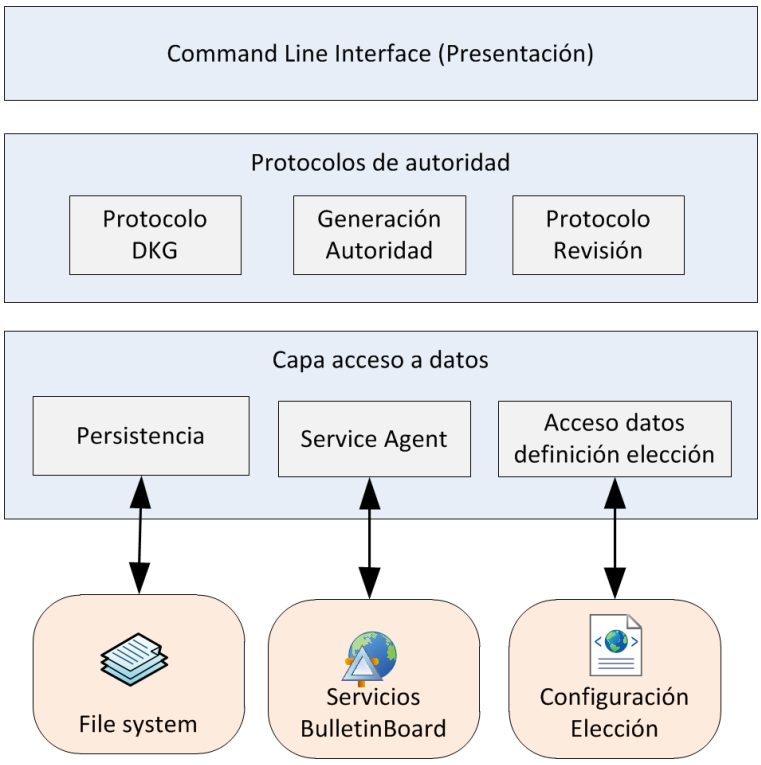


Figura 5.4: Arquitectura interna del módulo Autoridad

pueden ser consultados por cualquier votante u observador del proceso, el procedimiento para verificar cada etapa depende del protocolo específico que se está verificando, es decir que se requiere comprender exactamente el significado de cada mensaje y su relevancia en el proceso para poder verificarlo.

El módulo *Verificador-Externo* es una aplicación de escritorio que permite verificar cada una de las etapas realizadas durante el proceso de votación, desde la creación de las claves distribuidas (DKG) hasta el protocolo distribuido de Revisión pasando por la verificación de cada uno de los votos publicados. Mediante la interfaz gráfica, esta aplicación entrega un reporte detallado de la verificación realizada por cada fase. De esta forma el observador puede detectar si el recuento de los votos es válido, o detectar si una de las autoridades es deshonesta. También es posible saber cuáles son los votos válidos que fueron contados y a que votante pertenecen (sin violar la privacidad del voto).

En la Figura 5.5 se muestra la arquitectura del módulo. Se utilizó una arquitectura basada en capas de la siguiente manera:

- **Capa de presentación**: Interfaz gráfica de usuario.
- **Capa lógica**: Se implementa la verificación de los protocolos criptográficos del sistema de votación.
- **Capa de acceso a datos**: Se actualizan los datos de votación, se controla el acceso a datos locales.

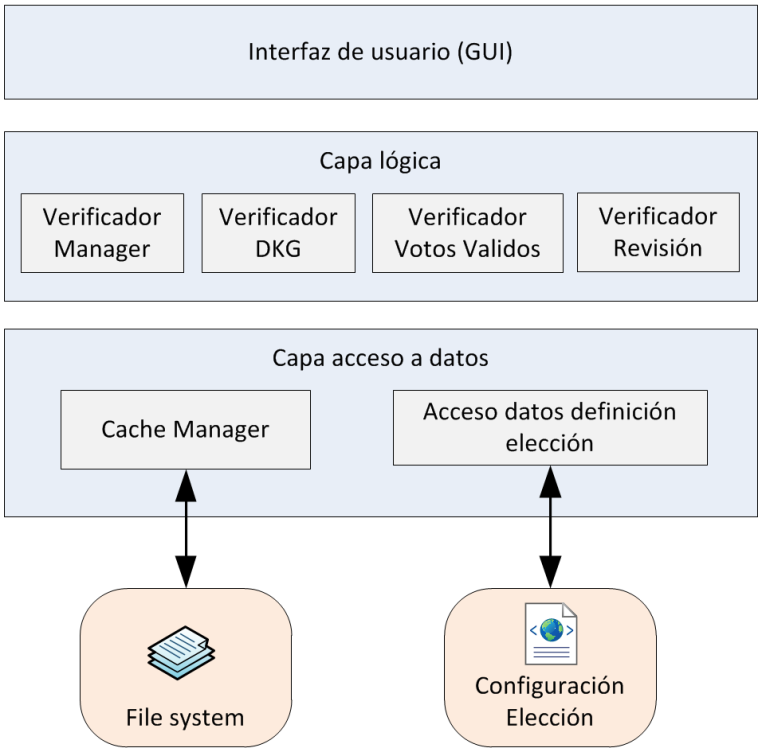


Figura 5.5: Arquitectura interna del módulo Verificador Externo

### Módulo Mesa

Módulo de apoyo para la autorización de los votantes, permite crear un par de claves pública/privada que habilitan al votante para emitir voto. La clave pública es publicada en el BulletinBoard mientras que la clave privada es eliminada de la *Mesa*. Se utiliza una memoria flash USB como token para transportar las claves del votante. Se mantiene un caché de datos persistente en caso de no haber conección con el servidor.

Este módulo define una arquitectura de capas como se muestra en la Figura 5.6.

- **Capa de presentación**: Contiene la implementación de la interfaz gráfica de usuario.
- **Capa lógica**: Contiene la lógica de autorización del votante y generación de claves.
- **Capa de acceso a datos**: Se controla el acceso a datos locales y se mantiene un caché local de datos y se realizan las peticiones al servidor del sistema, para lo cual se utiliza el patrón *Service Agent*, de esta forma se encapsula la responsabilidad de acceder a los servicios del servidor.

### Módulo Cámara

Este módulo es responsable de presentar los candidatos y carreras de la elección al votante, el usuario de forma interactiva escoge las opciones y emite el voto. Se utiliza una memoria flash USB que contiene las claves del votante que son utilizadas para firmar los votos. Se utiliza un caché de datos persistente en caso de no tener conección con el BulletinBoard. Antes de enviar el voto, se imprime en papel y el votante puede chequear que su selección coincida con el voto impreso.

Se utiliza una arquitectura de capas como se ve en la Figura 5.7.

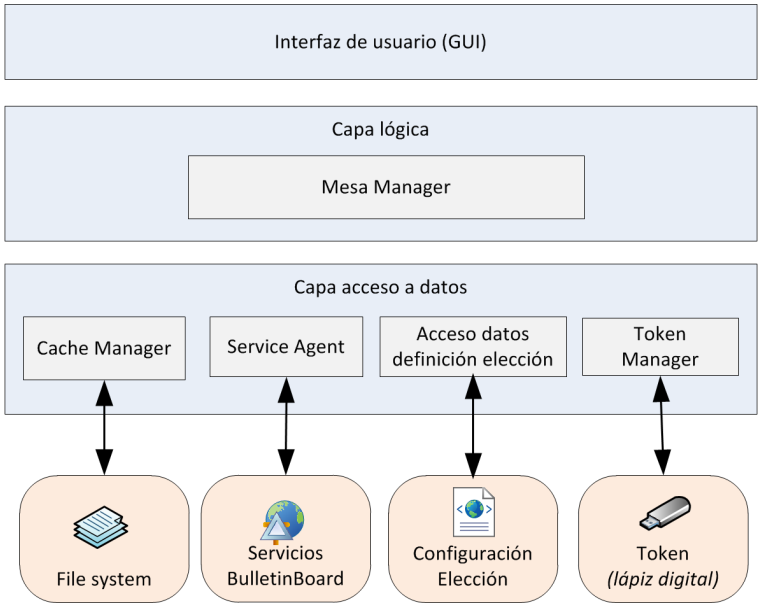


Figura 5.6: Arquitectura interna del módulo Mesa

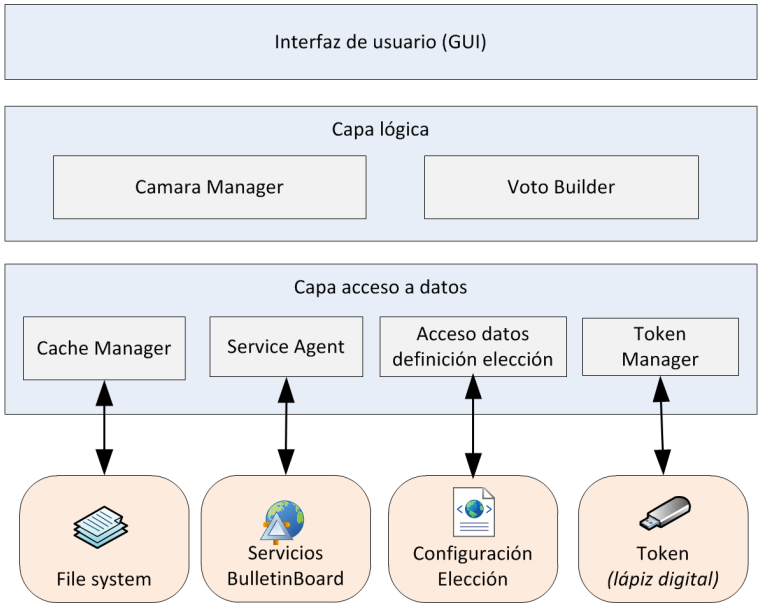


Figura 5.7: Arquitectura interna del módulo Cámara de Votación

- **Capa de presentación**: Contiene la implementación de la interfaz gráfica de usuario.
- **Capa lógica**: Contiene la lógica de generación del voto.
- **Capa de acceso a datos**: Se controla el acceso a datos locales y se mantiene un caché local de datos y se realizan las peticiones al servidor del sistema, para lo cuál se utiliza el patrón *Service Agent*, de esta forma se encapsula la responsabilidad de acceder a los servicios del servidor.

### Módulo Configurador de Elecciones

Este componente permite generar la configuración de una nueva elección a partir de archivos CSV fácilmente editables por un usuario de forma manual o en algún programa de planilla de cálculo, como *Microsoft Excel*.

Este componente ha sido diseñado usando el estilo arquitectónico *Pipeline*, de esta forma los datos de entrada son procesados en varias etapas hasta obtener el resultado final. Como se ve en la Figura 5.8, se cargan los datos CSV de carreras, candidatos y votantes mediante una interfaz de usuario en línea de comando. Posteriormente se verifica que los datos sean válidos y que cumplan las reglas de validación de una elección. Finalmente se genera un documento XML con la configuración de los datos. De esta forma se asegura que la configuración de la elección sea válida y fácil de consultar por todos los módulos del sistema.

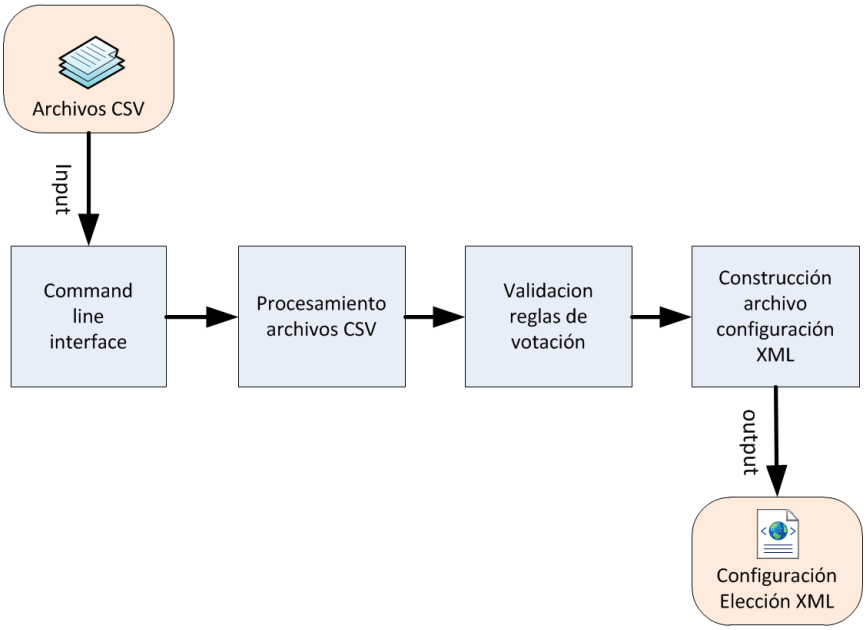


Figura 5.8: Arquitectura interna del módulo Configurador (estilo *Pipeline*)

### Módulo Observador Web

Este componente consiste en una página web de despliegue de resultados de una elección. La página también da la opción de verificar el estado de un voto emitido, validando que su papeleta exista y sea válida. La arquitectura de este componente se mantendrá tal cual como era anteriormente, aunque éste será modificado para implementar el acceso a datos almacenados en la base

de datos del sistema y en el archivo de definición de una elección (sólo se requieren operaciones de lectura a los datos). Cabe notar que la arquitectua actual no es la más adecuada para darle extensibilidad a su implementación, sin embargo está fuera del alcance de este trabajo mejorar este diseño.

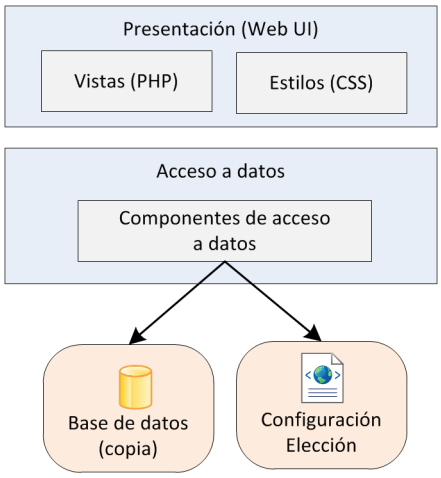


Figura 5.9: Arquitectura interna del módulo Observador Web

Se identifican dos capas lógicas en la estructura de este módulo, una capa que encapsula el acceso a los datos del sistema y una capa que implementa interfaces de usuario unidas a la lógica de procesamiento de los datos y generación de los resultados.

### 5.2.3. Vista de procesos

Esta vista arquitectónica describe como interactúan los componentes del sistema, se descomponen procesos en actividades y responsables. Permite tener una visión más clara acerca del funcionamiento del sistema. Se describirán los procesos relevantes del sistema revisando desde lo más general a lo más particular.

El procedimiento para utilizar el sistema de votación no sólo consiste en la ejecución del sistema, también hay una fase de creación y configuración de una nueva elección, además de una fase de instalación (o instanciación) de los servidores y componentes del sistema. En la Figura 5.10 se ven las fases más generales.

Cada una de estas fases es revisada con mayor detalle a continuación:

#### Creación de una nueva elección

En esta fase se crea y configuran los datos de definición de una nueva elección. El TRICEL o tribunal calificador de elecciones es responsable de esta fase, se generan los datos de la votación en formato CSV (o *Comma Separated Values*), los cuales pueden ser fácilmente creados en software tipo planilla de cálculo. Estos datos son validados por el módulo *Configurador de Elecciones* el cual genera un documento XML con la configuración final. Ver Figura 5.11.

Posteriormente se empaquetan los instaladores del sistema cargados con los datos de la elección. Finalmente se preparan los *tokens* y *pendrives* que se utilizarán en el proceso.

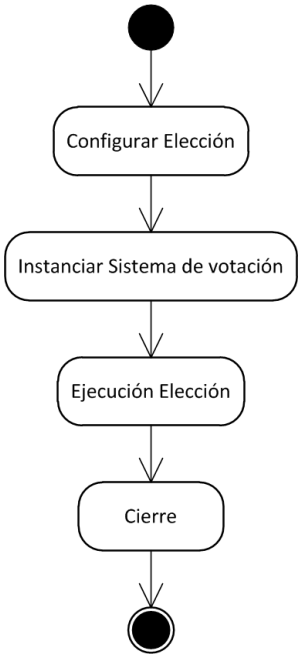


Figura 5.10: Diagrama UML de actividades general del sistema

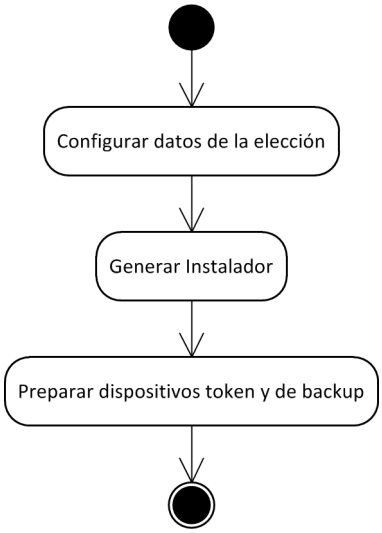


Figura 5.11: Diagrama UML de actividades de la configuración de una nueva elección

### Instanciación del sistema de votación

Esta fase contempla la instalación de los servidores y computadores cliente (*live-cd*), además de la configuración de despliegue del sistema (configuración de red).

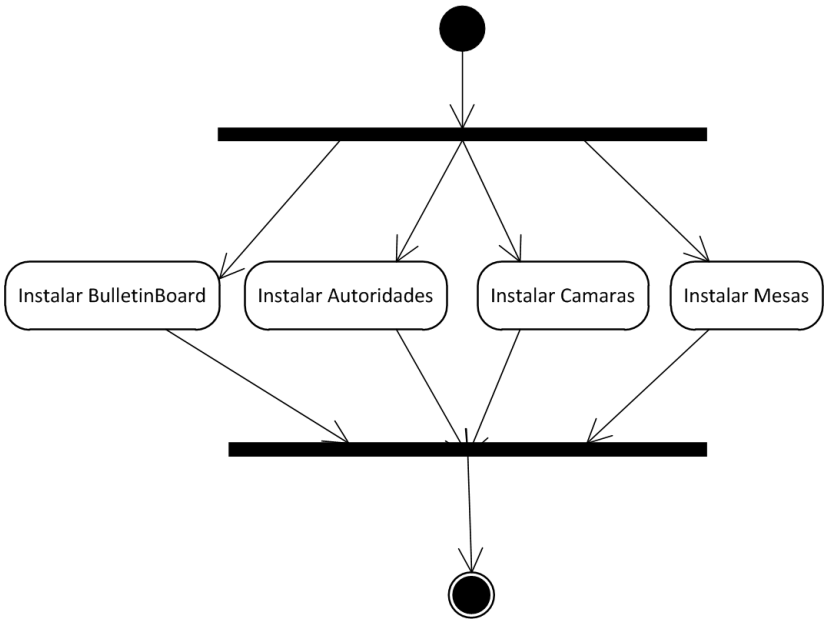


Figura 5.12: Diagrama UML de actividades del proceso de instanciación (despliegue) del sistema

Las etapas de esta fase son la instalación de cada nodo donde se ejecutará un módulo del servidor, esto quiere decir, la instalación del servidor del BulletinBoard, los servidores de Autoridad, los clientes de *Mesa* y los clientes de *Cámara*. Estas etapas pueden ser realizadas en paralelo, tal como se muestra en la Figura 5.12.

### Ejecución y verificación de la elección

La elección propiamente tal comprende varias etapas (ver Figura 5.13):

1. En el servidor de BulletinBoard se generan los parámetros aleatorios iniciales de la votación.
2. Se levanta el módulo servidor (BulletinBoard).
3. Se inicializan las autoridades, es decir, cada Autoridad crea su par de claves pública/privada y envía su clave pública al BulletinBoard.
4. A partir de este momento, se puede iniciar el módulo *Verificador Externo*, de esta forma de puede verificar de forma paralela a la ejecución del sistema.
5. Las autoridades ejecutan de forma sincronizada el protocolo DKG[1].
6. El verificador puede ejecutar la verificación del protocolo DKG revisando los datos generados por las autoridades durante el protocolo.
7. Comienza la *Votación*, los votantes pueden comenzar a emitir votos.

[1] Ver descripción de protocolo DKG en capítulo *Situación previa*, sección *Protocolos criptográficos del sistema*.

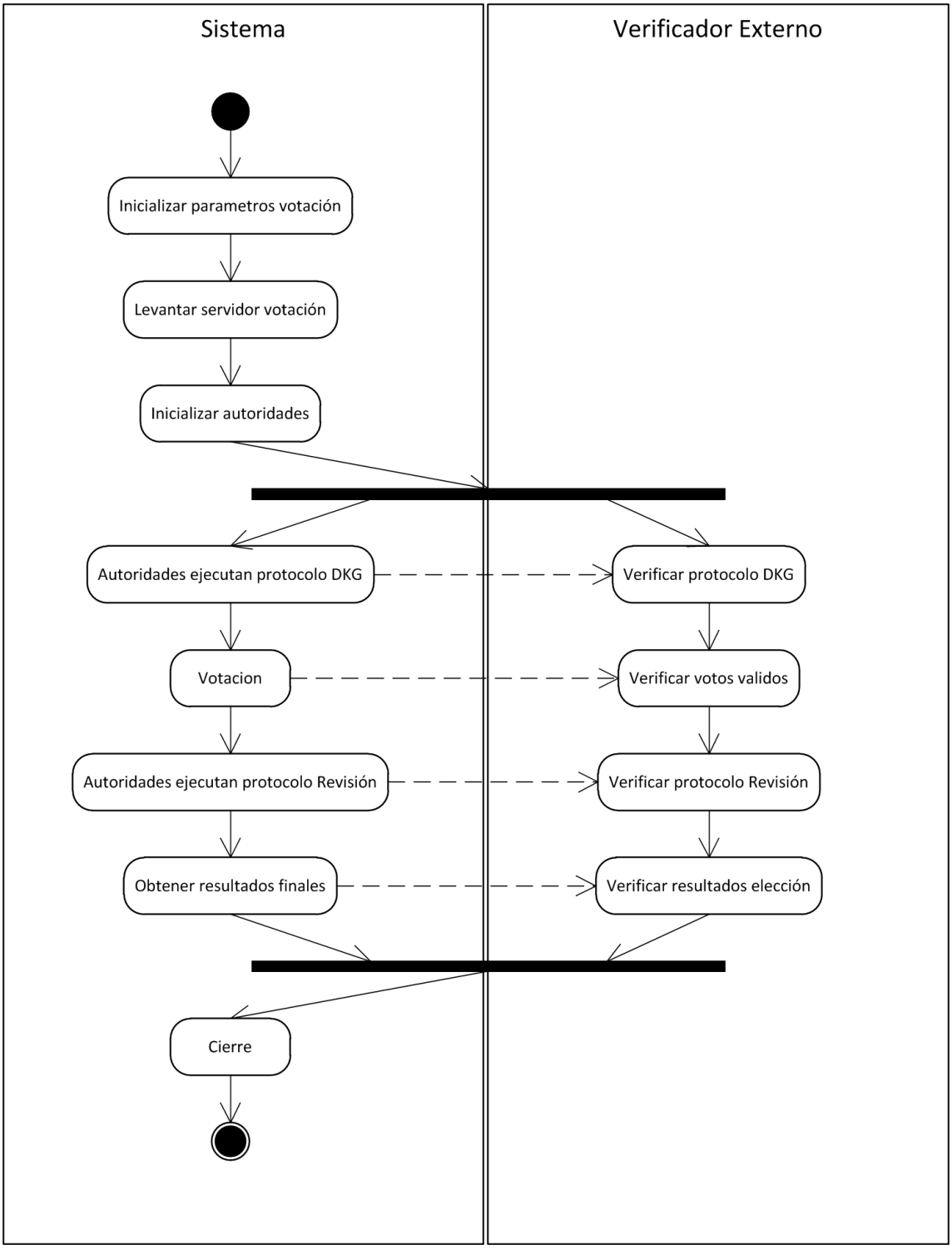


Figura 5.13: Diagrama UML de actividades de la ejecucion y verificación de una elección

8. El verificador puede verificar los votos generados durante la *votación*.
9. Cuando se termina la fase de votación, las autoridades ejecutan el *protocolo distribuido de Revisión* de forma síncrona.
10. El verificador puede ejecutar la verificación del protocolo de Revisión revisando los datos generados por las autoridades durante el protocolo.
11. El BulletinBoard publica los resultados finales de la elección (calculados en la etapa de Revisión).
12. El verificador compara los resultados finales publicados con los resultados calculados por el mismo *Verificador Externo*.
13. Cierre de la elección.

**Fase de votación**

Este proceso define las etapas necesarias para que un votante puede emitir y publicar su voto. En la Figura 5.14 se muestra un diagrama de actividades UML con las siguientes tareas:

1. El votante es autenticado físicamente por el *Vocal de Mesa*
2. Mediante el módulo de *Mesa* el *Vocal* compureba el estado del votante y lo autoriza a votar.
3. Mediante el módulo de *Mesa* se generan el par de claves pública/privada del votante las que son grabadas en el *token* (pendrive), luego la clave privada es eliminada del módulo *Mesa*
4. La clave pública es enviada al BulletinBoard (o se envía posteriormente en caso que el servidor no se encuentre disponible)
5. Se le entrega el *token* generado (*pendrive*) al votante para que ingrese a la *Cámara de votación*.
6. El votante ingresa a la *Cámara*, introduce el *token* y marca sus preferencias en la pantalla de votación.
7. Se imprime una copia en papel de los *papeletas* emitidos por el votante.
8. Si el votante considera que la *papeleta* impresa es incorrecta o decide volver comenzar puede hacerlo.
9. Si el votante acepta la impresión de la *papeleta* entonces la papeleta digital es enviada al BulletinBoard (o se envía posteriormente en caso que el servidor no se encuentre disponible.
10. El votante devuelve el *token* al vocal de mesa, se elimina el contenido del *token*.
11. Se actualiza el estado final del votante.

Cabe notar que el sistema permite que el votante vote varias veces, sin embargo se definió como política del sistema que solamente se considere válida la primera clave del votante que sea publicada en el BulletinBoard. El único voto válido es el primero que es publicado en el BulletinBoard cuya firma digital corresponde con la clave válida. Esta decisión puede ser modificada si es necesario en una nueva versión del sistema.

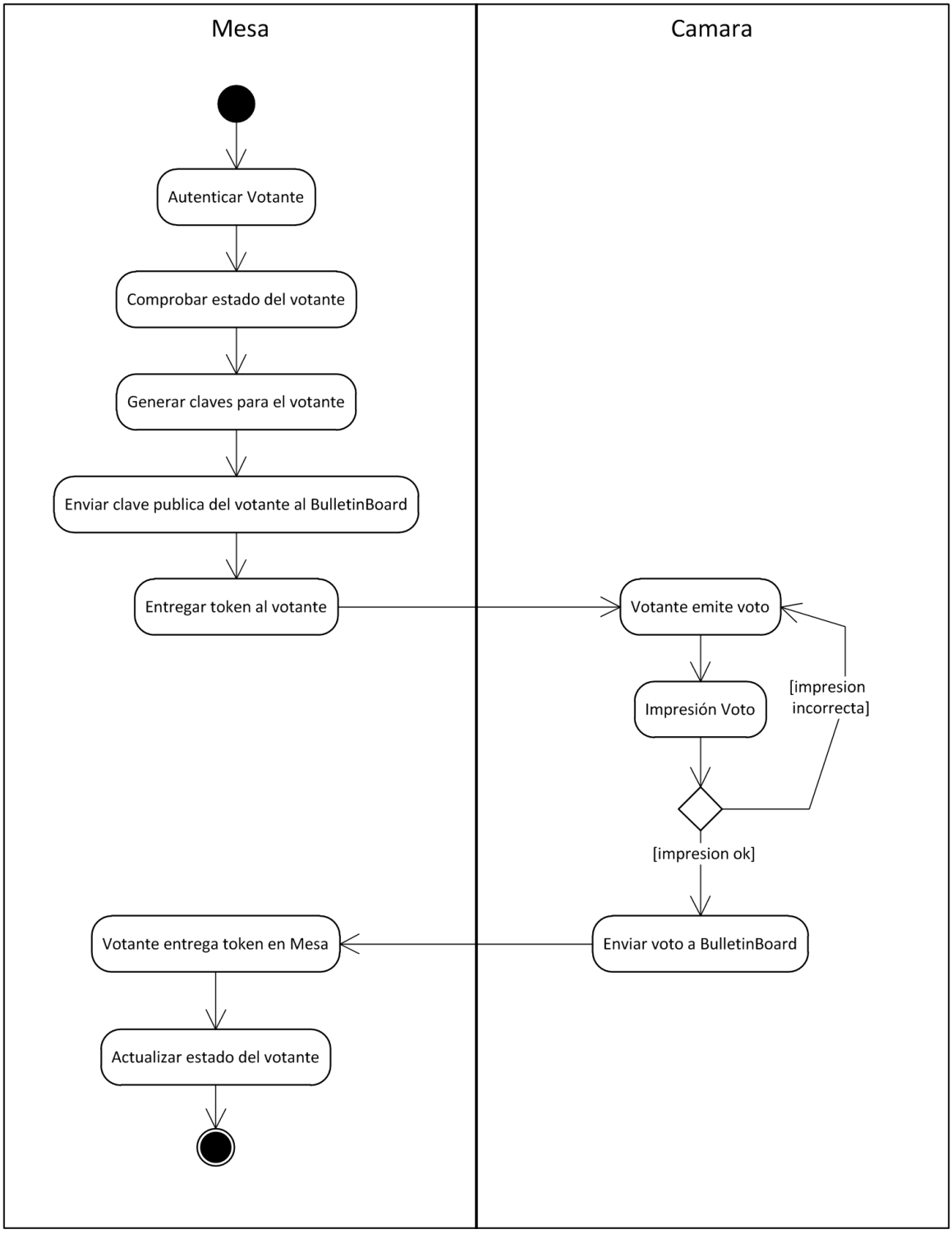


Figura 5.14: Diagrama UML de actividades para la emisión de votos

### 5.2.4. Vista de implementación

La vista de implementación describe la composición física o de implementación en términos de artefactos de software como paquetes, ejecutables, *frameworks* y datos.

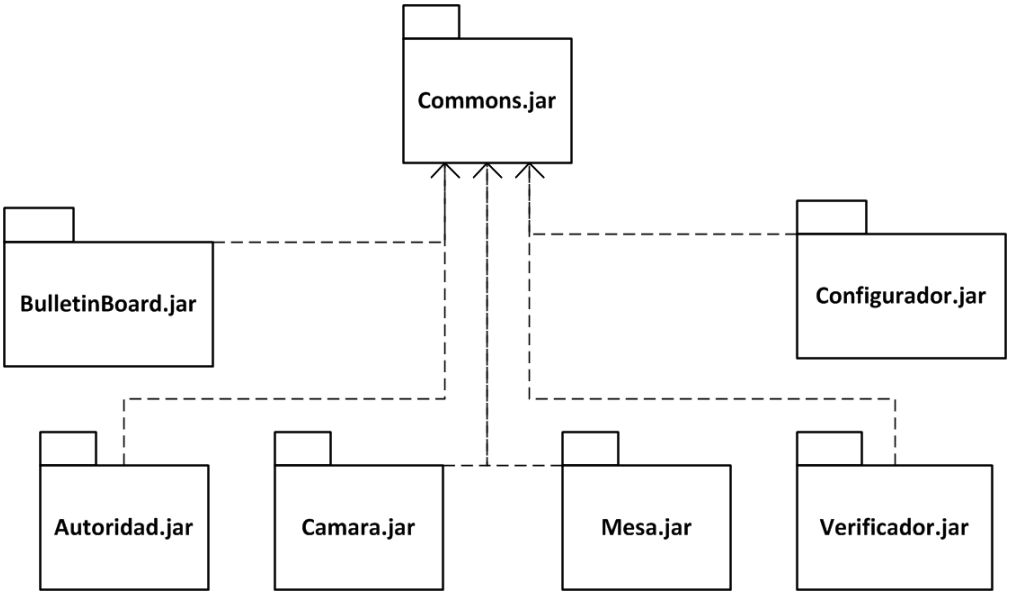


Figura 5.15: Diagrama de implementación, con las relaciones entre paquetes

La implementación del sistema se distribuye entre varios paquetes de Java (*jar*), los que pueden verse en la Figura 5.15.

#### Ejecutables e instaladores

Aparte de los paquetes Java se crearon *scripts* de *bash* ejecutables para Debian Linux, los que simplifican la ejecución del sistema. En el cuadro 5.9 se muestra el detalle de los ejecutables.

También se implementaron *scripts* de *Bash* para Debian Linux. Estos *scripts* sirven para configurar cada estación, configurar la red y preparar las memorias *flash* USB para respaldo de datos. Además se incluyen *scripts* que permiten instalar o restaurar cada módulo del sistema de votación. Utilizan una interfaz en línea de comandos que guía al usuario en el proceso de instalación o restauración.

#### Preparación de imagen de sistema operativo *live* minimal

Al utilizar un sistema operativo minimal, es decir que contiene sólo los paquetes de software estrictamente necesarios para utilizar el sistema de votación, se minimiza el riesgo de introducir software que puede contener vulnerabilidades que podrían ser utilizadas para corromper el sistema de votación. También es importante utilizar una versión de sistema operativo lo más estable y seguro posible, para esto se escogió la rama *stable* de Debian Linux (Debian 6), la que utiliza solo paquetes estrictamente probados y con actualizaciones de seguridad.

Al proveer el sistema con un sistema operativo personalizado que se carga directamente en memoria RAM se elimina el requisito de que los usuarios instalen manualmente el sistema operativo, ya que esto es riesgoso si el sistema no es correctamente instalado o si se instalan programas

| Paquete | Descripción | Dependencias |
|---|---|---|
| Commons.jar | Contiene las clases que son compartidas o utilizadas por distintos módulos del sistema, tales como las entidades de datos, las interfaces de servicio RMI y funcionalidades criptográficas comunes | bouncycastle-provider.jar<br>xstream.jar<br>simple-xml.jar |
| BulletinBoard.jar | Contiene el módulo del BulletinBoard | Commons.jar<br>sqlite-jdbc-xerial.jar |
| Autoridad.jar | Contiene el módulo Autoridad | Commons.jar |
| Verificador.jar | Contiene el módulo *VerificadorExterno* | Commons.jar<br>sqlite-jdbc-xerial.jar |
| Mesa.jar | Contiene el módulo *Mesa* | Commons.jar<br>swt.jar |
| Camara.jar | Contiene el módulo *Camara* | Commons.jar<br>swt.jar |
| Configurador.jar | Contiene el módulo *Configurador* | Commons.jar<br>Autoridad.jar<br>open-csv.jar<br>simple-xml.jar |

Cuadro 5.8: Paquetes Java del sistema

| Módulo | Ejecutable | Uso |
|---|---|---|
| BulletinBoard | bulletinboard.sh | -inicializar: Se crea la base de datos y parámetros iniciales.<br>-start-server: Levanta el servidor BulletinBoard. |
| Autoridad | autoridad.sh | -inicializar: Inicializa la autoridad y comienza protocolo DKG.<br>-revision: inicia protocolo de Revisión de votos. |
| Mesa | mesa.sh | Ejecuta el módulo de la Mesa de votación |
| Camara | camara.sh | Ejecuta el módulo de la Cámara de votación |

Cuadro 5.9: Archivos ejecutables del sistema

adicionales (e innecesarios) que aumenten el riesgo de introducir vulnerabilidades o incluso *malware* que podría atacar al sistema.

Se crearon dos sistemas operativos: uno para los servidores (sin interfaz gráfica) y otro para los clientes (con interfaz gráfica). Se utilizó la herramienta Debian *LiveBuild* [22]. En la siguiente tabla se resumen las características de cada sistema.

| **Tipo** | **Descripción** | **Paquetes incluidos** | **Tamaño aprox.** |
|---|---|---|---|
| Servidor | Live CD/USB<br>S.O. Debian 6 *stable*<br>32 bits | Java *Openjdk-6-jre*<br>Sin interfaz gráfica | 250MB |
| Cliente | Live CD/USB<br>S.O. Debian 6 *stable*<br>32 bits | Java *Openjdk-6-jre*<br>Administrador de ventanas *Openbox*<br>Administrador de impresoras *Cups-bsd* | 350MB |

Cuadro 5.10: Versiones del sistema operativo personalizado

En las figuras 5.16 y 5.17 se muestra el resultado de la creación de los *live-cd* del sistema para el servidor y para las estaciones de votación respectivamente.

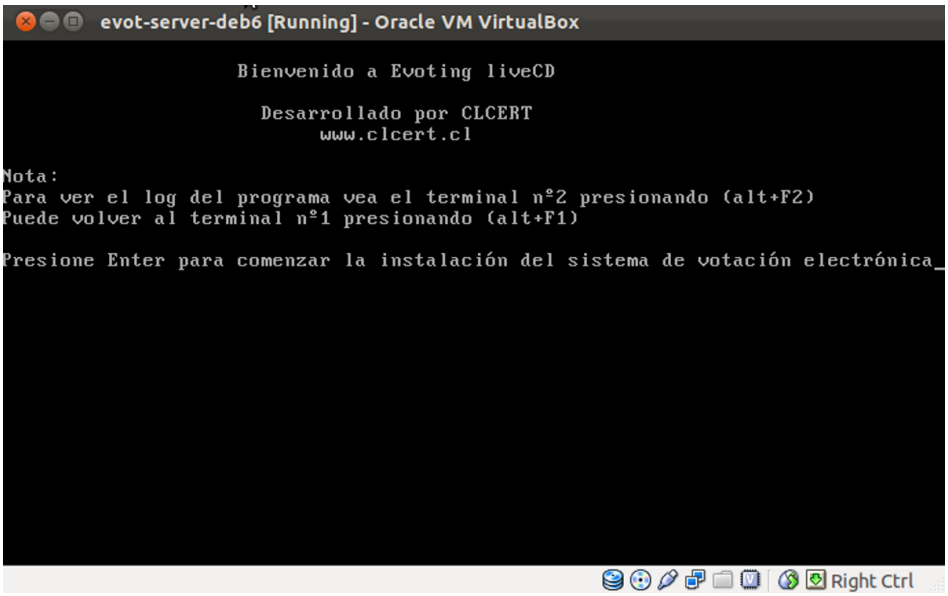


Figura 5.16: *Live-cd* para uso del servidor, recién cargado

### 5.2.5. Vista de despliegue

Esta sección describe la infraestructura y componentes sobre los que es instanciado y ejecutado el sistema.

Para simplificar el proceso de despliegue del sistema se ha creado un sistema operativo minimal basado en Debian Linux que puede ser iniciado en modo *live*, gracias a esto el sistema puede ser utilizado sobre computadores o servidores sin sistema operativo instalado ni disco duro interno.

El sistema *live-cd* construido tiene dos variantes:

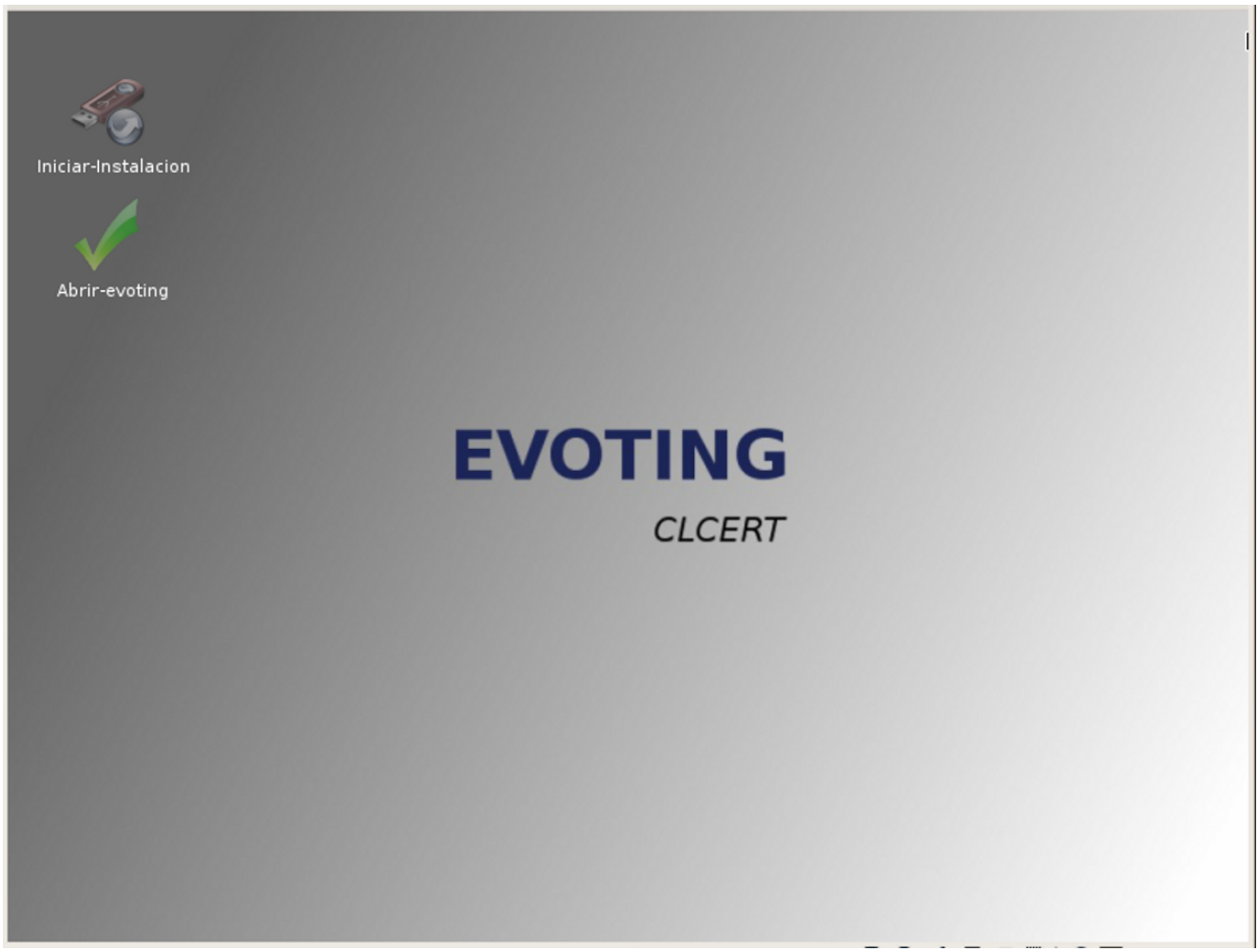


Figura 5.17: *Live-cd* para uso de los clientes (estaciones de votación)

- Servidor live-cd: S.O. minimal, sin interfaz gráfica de usuario, para BulletinBoard y autoridades.
- Cliente live-cd: S.O. minimal, con interfaz gráfica de usuario, Cámara y Mesa.

El sistema una vez que ha cargado comienza automáticamente la instalación del sistema de votación, utilizando los scripts de instalación que se crearon para el sistema. A cada módulo también se agregan scripts de ejecución que simplifican el inicio de los componentes.

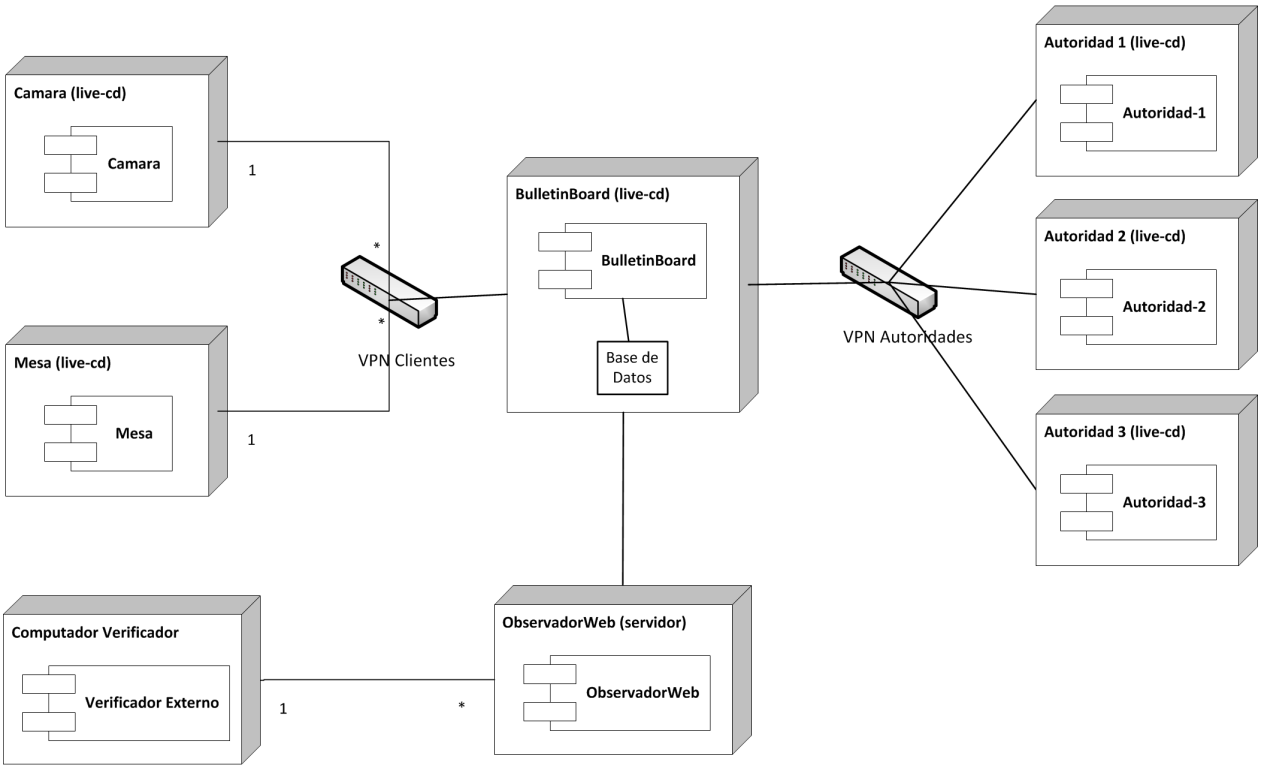


Figura 5.18: Diagrama físico del despliegue del sistema

### Infraestructura de Red

La infraestructura de red a utilizar es fundamental para lograr la privacidad e integridad de las comunicaciones del sistema. Se han creado instaladores del sistema que permiten dos maneras de desplegar el sistema: Usando una red privada o utilizar redes VPN.

- **Usando una red privada**. Este caso es el más simple de administrar, pero es poco práctico en entornos de votación reales donde los componentes están físicamente distantes. En este caso el sistema completo es desplegado en una red privada exclusiva para el sistema de votación. Cada componente del sistema puede conectarse a la red definiendo direcciones IP estáticas o mediante DHCP.
- **Utilizando redes VPN**. Este es el caso más general y el más recomendable para una votación real. Para este caso se requiere que exista una o más redes VPN, en caso de utilizar una red particionada, administradas independientemente al sistema de votación, de esta forma cada componente del sistema se conecta a la red VPN. Idealmente se espera que se utilice una

red particionada, es decir, que se utilice una red VPN para conectar a las autoridades con el BulletinBoard y que se utilice otra red VPN para conectar a los cliente de votación con el BulletinBoard. Incluso es posible ir más lejos y utilizar redes separadas para conectar cada autoridad y cada cliente con el servidor. El *script* por implementación sólo permite conectarse a servidores VPN Cisco, utilizando el software cliente *vpnc*. Sin embargo este instalador se podría extender para soportar otros tipos de servidores VPN como *OpenVPN*.

Se descarta el uso de una infraestructura de red pública para el despliegue del sistema debido a que por su diseño, la responsabilidad de establecer la seguridad de la comunicación se ha dejado de forma externa a la implementación del sistema. Está tema fue tratado en la sección 2.3.2 página 21.

Además de lo anterior es necesario que el BulletinBoard tenga conexión con el servidor web (Observador Web), esto se logra incluyendo en el BulletinBoard una interfaz de red con acceso a la red donde se encuentra el servidor web, el instalador del sistema configura el *firewall* para que solo permita conecciones salientes desde el BulletinBoard usando el puerto SSH a través de esta interfaz de red.

Cabe notar que es posible usar una variante de despliegue del sistema utilizando servidores privados virtuales (VPS), ésta es una tecnología *Cloud Computing* de tipo *Infraestructure as a Service*. Para este caso se puede generar un sistema operativo personalizado para el servidor de votación, de modo que pueda ser instalado en un VPS. una variante del sistema operativo que pueda ser instalada en un VPS. Con esta alternativa se gana en disponibilidad y confiabilidad del sistema, siendo más fácil de administrar ya que no requiere tanta infraestructura física instalada. Esta variante puede ser fácilmente soportada a partir del sistema ya implementado.

### 5.2.6. Vista de datos

El modelo de datos del sistema está dividido en dos partes: por un lado se encuentran los datos de definición de una elección y por otro lado se encuentran los datos públicos generados durante la elección, que son almacenados en el BulletinBoard.

Los datos de definición de una elección son datos estáticos (no modificables durante la elección) generados antes de comenzar una elección, para su generación se utiliza el módulo *Configurador*. Estos datos contienen el listado de candidados, carreras y votantes de la elección. En la Figura 5.19 se muestra el modelo de datos utilizado. Los datos de definición de la elección son incluidos en todos los módulos del sistema de votación en formato XML. En el Anexo B se incluye un documento XML de ejemplo de definición de una elección.

Los datos generados durante el proceso de votación (datos públicos) son gestionados por el módulo BulletinBoard, los datos son almacenados en una base de datos relacional, cuyo modelo de datos entidad-relación se presenta en la Figura 5.20.

La base de datos es inicializada utilizando un script SQL, el cuál está incluido en el Anexo A.

## 5.3. Interfaces de usuario

#### Interfaces del módulo VerificadorExterno

La interfaz del *VerificadorExterno* ha sido implementada con Java Swing. La ventana principal de la aplicación tiene dos opciones: *Actualizar Datos* y *Verificar Todo*.

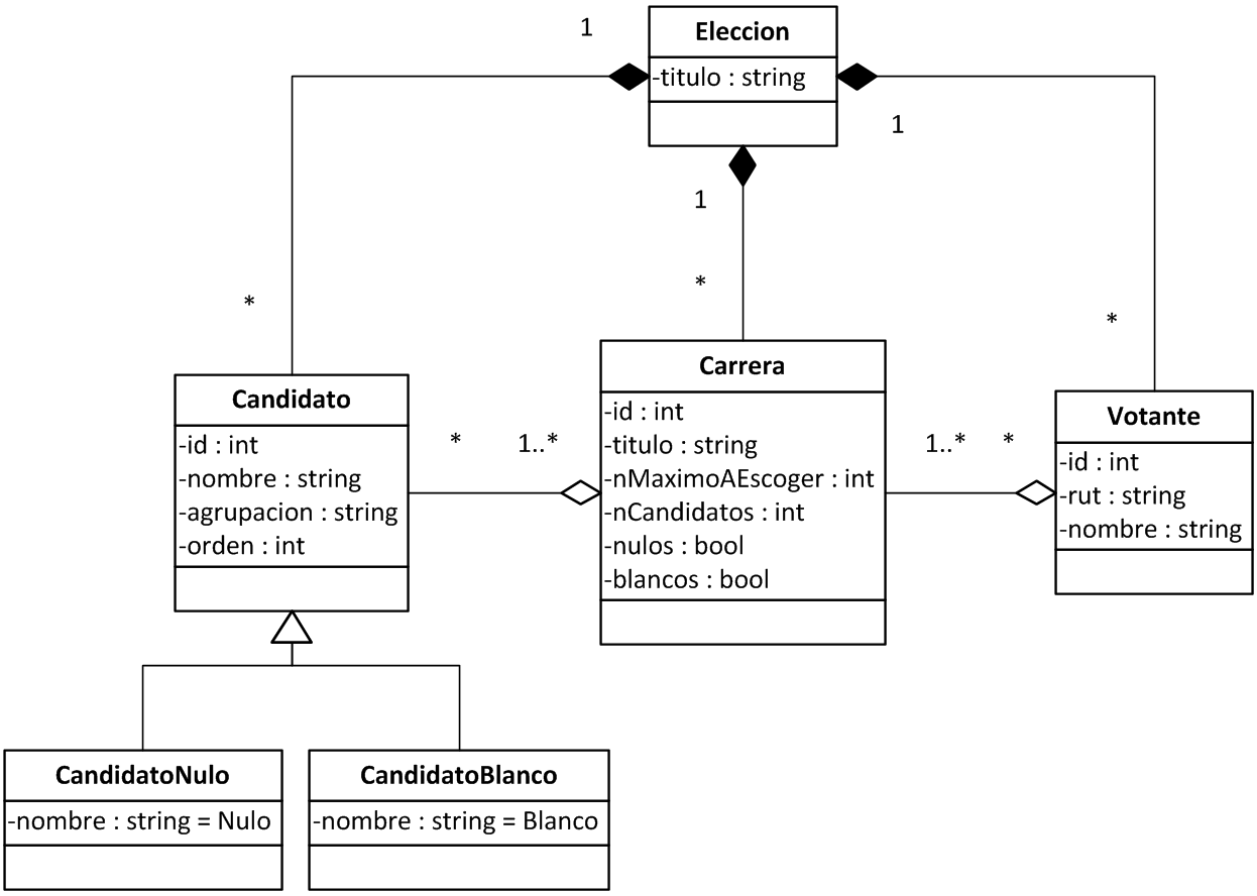


Figura 5.19: Diagrama de clases UML con la estructura de datos de definición de una elección

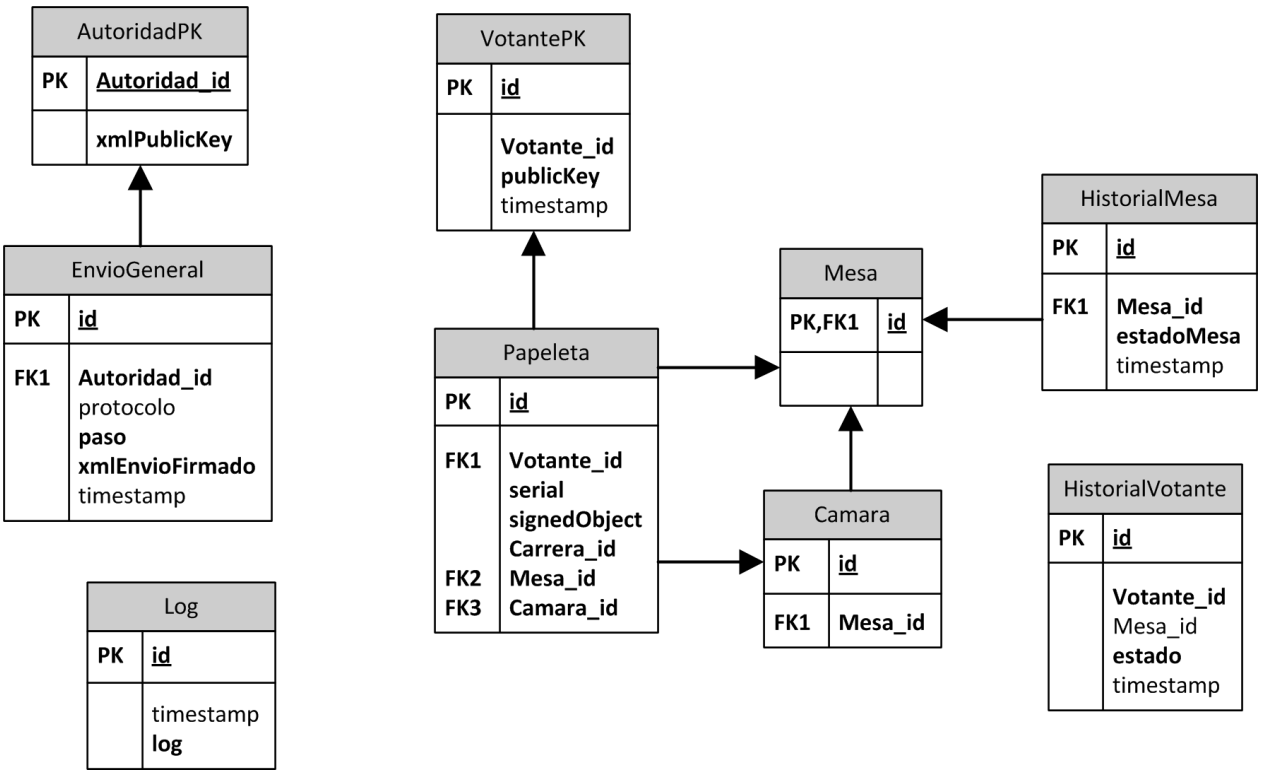


Figura 5.20: Diagrama de datos relacional de la base de datos para datos generados durante la elección

Adicionalmente la ventana cuenta con cuatro pestañas, en cada pestaña se muestra un reporte detallado de la verificación de una fase del proceso de elección.

En la Figura 5.21 se muestra el reporte generado para la verificación de la fase de generación de claves distribuida del sistema (protocolo DKG).

La Figura 5.22 muestra un reporte con todos los votos emitidos durante el proceso de elección, analizando la validez de cada voto.

En la Figura 5.23 se ve el reporte generado de la verificación del proceso de revisión y conteo de votos distribuido.

La Figura 5.24 muestra los resultados finales de la elección obtenidos durante el proceso de verificación.

## 5.4. Validación de la implementación

La implementación del sistema se validó bajo distintos escenarios, montando elecciones de prueba, con el objetivo de poner a prueba las cualidades del sistema de votación y validar el correcto comportamiento del sistema.

Se evaluaron escenarios de robustez del sistema, tolerancia a fallas y restauración de los servidores, desconección de los clientes a Internet, caída del servidor principal.

Se evaluaron escenarios con autoridades deshonestas, poniendo a prueba la eficacia de los protocolos del sistema y el sistema de Verificación de la elección, el cuál debe detectar y reportar estos eventos. Para esto se implementaron versiones alternativas de las autoridades que intentan corromper los resultados del proceso o modificar los cálculos durante los protocolos.

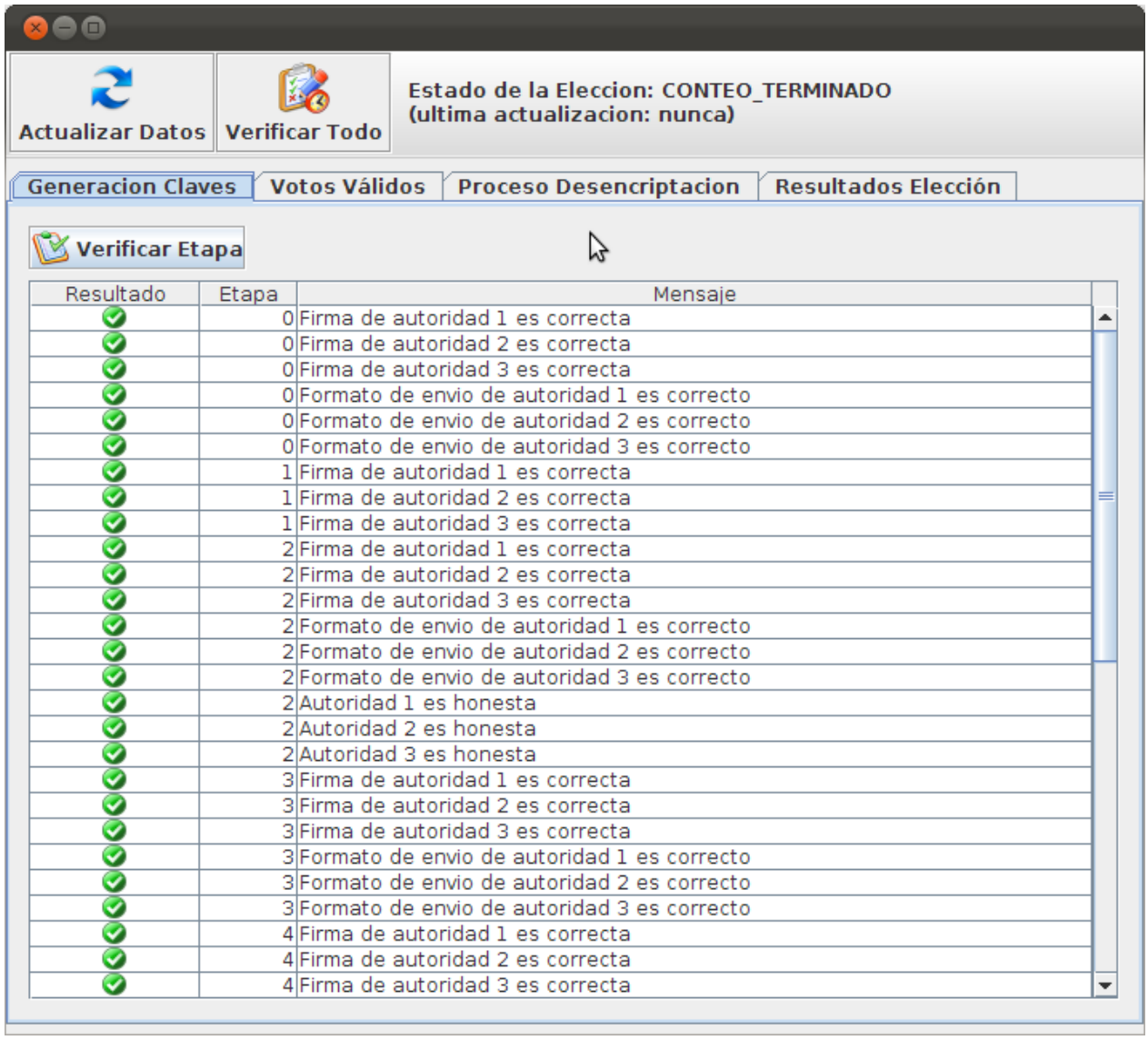


Figura 5.21: GUI VerificadorExterno, pantalla de verificación de proceso de generación de claves del sistema

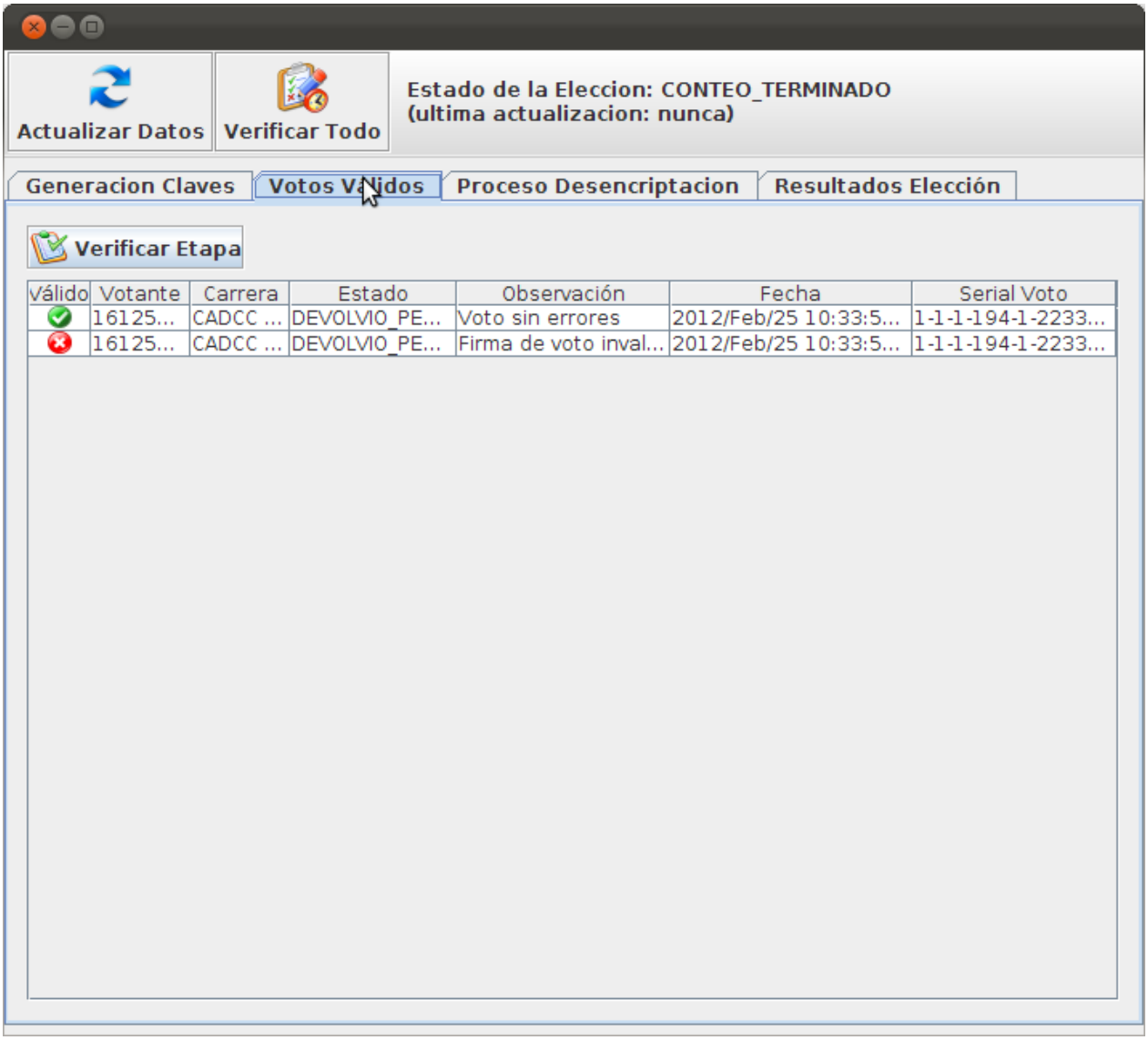


Figura 5.22: GUI VerificadorExterno, pantalla de verificación de los votos emitidos

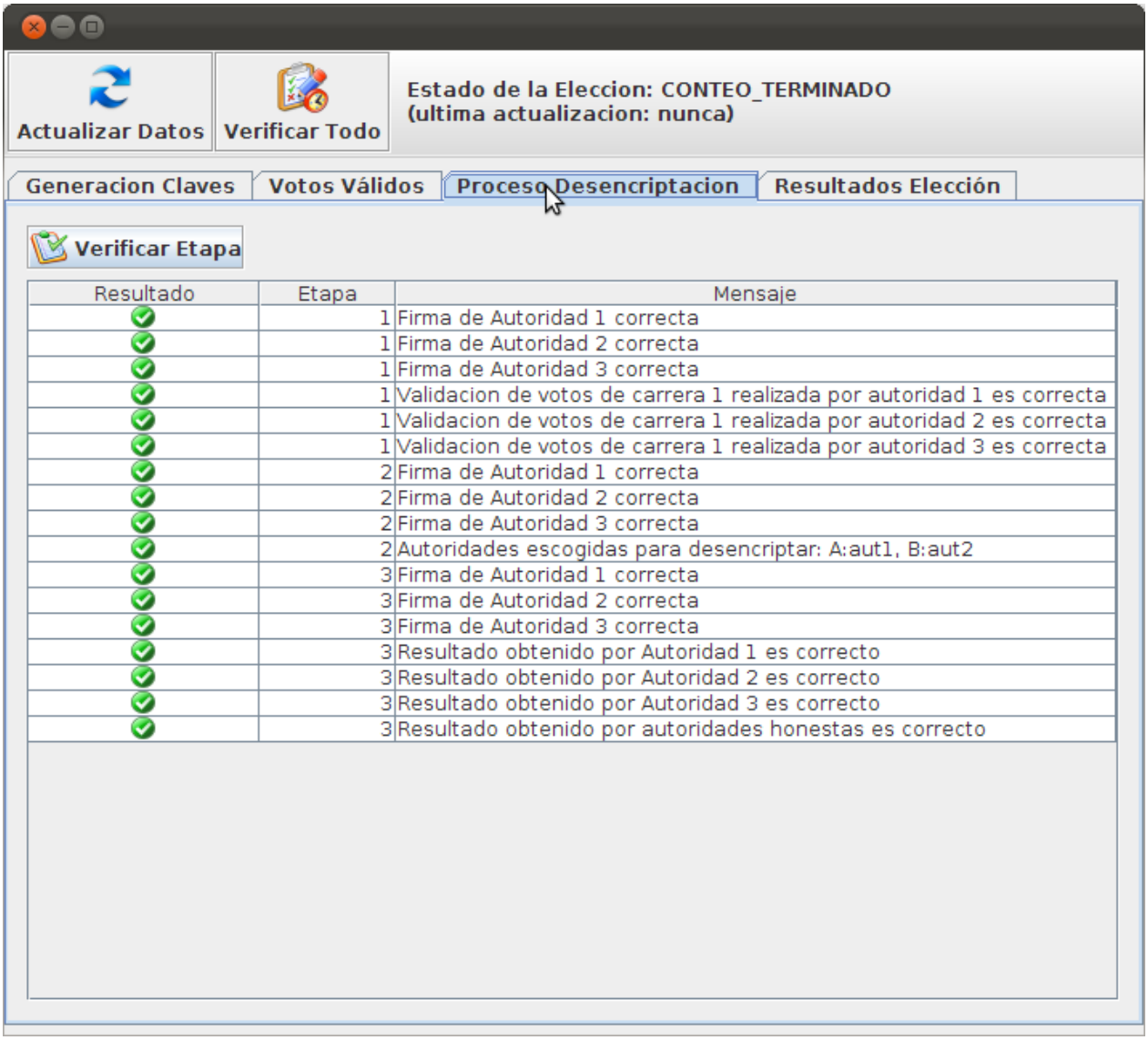


Figura 5.23: GUI VerificadorExterno, pantalla de verificación de proceso de conteo de votos

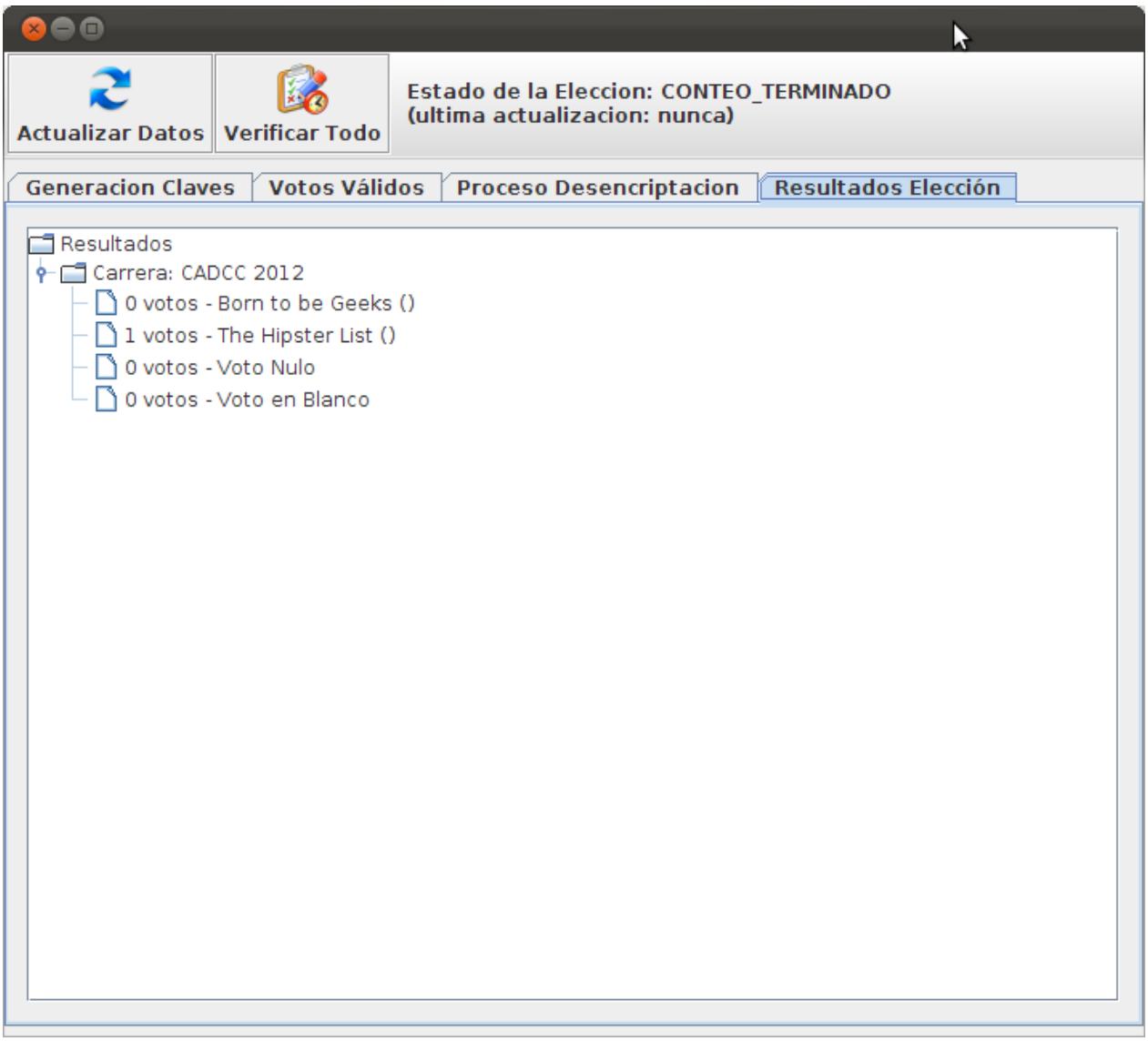


Figura 5.24: GUI VerificadorExterno, pantalla de verificación de los resultados de la elección

El montaje de los escenarios se ejecutó en dos formas:

1. **Virtualización de estaciones con VirtualBox**: Se desplegó cada estación del sistema en máquinas virtuales *VirtualBox* 4.0, utilizando una red privada (LAN) entre las estaciones.

2. **Entorno real utilizando VPN**: Se desplegaron las estaciones sobre una red pública, se montó una red VPN para aislar al sistema de votación del resto de los computadores de la red, los servidores se desplegaron como máquinas virtuales en *VMWare*, los clientes se desplegaron en computadores *Tablet PC* (ver Figura 5.25).

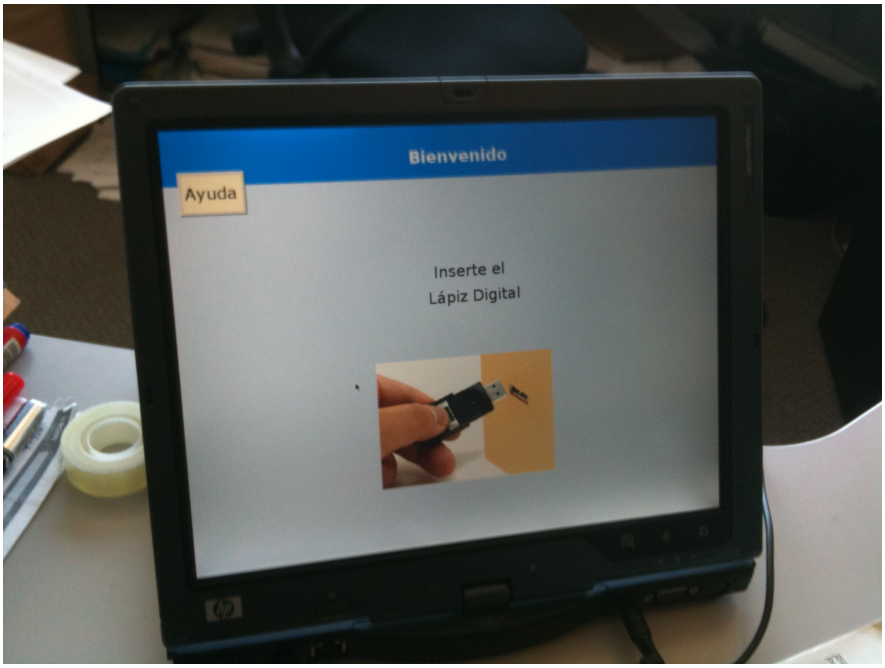


Figura 5.25: Pruebas del sistema en *laptop* con pantalla táctil

En ambos despliegues se probaron los distintos escenarios de fallas red y de autoridades deshonestas. Para todos los escenarios se obtuvo el comportamiento esperado de recuperación del sistema y término exitoso del proceso de votación. Los resultados fueron correctamente verificados por el *VerificadorExterno*, ejecutando correctamente las verificaciones y detectando los intentos de modificación o corrupción durante los protocolos criptográficos. A continuación se presenta un resumen de las pruebas realizadas y sus resultados:

- En un escenario sin fallas de ningún tipo, el probador vota usando distintos votantes. Los votos son generados y publicados correctamente en el BulletinBoard. Se mantiene el respectivo respaldo de los votos en cada cámara de votación. El recuento de votos es realizado correctamente por las autoridades (todas honestas). Los datos publicos son publicados en el servidor Web de forma periódica. El verificador externo descarga los datos públicos y genera los reportes de cada fase del sistema, comprobando los resultados correctamente y validando los resultados parciales.

- Un votante realiza la votación reiteradas veces. En este escenario se espera que por cada votante sólo se considere válido el primer voto publicado en el repositorio (asociado a la

primera clave pública del votante publicada). En esta prueba el sistema hizo lo esperado: se considera inválido todo voto publicado que no cumplía estas condiciones. Es resultado final es realizado correctamente. Todo esto se pudo observar en el software verificador, donde se identifica cada voto válido y cada voto inválido (por votante y por orden de publicación).

- Escenario con una autoridad deshonesta. Este escenario se simuló con la implementación deshonesta de una autoridad (en cada etapa de los protocolos la autoridad deshonesta realiza cálculos distintos al resto, o trata de imponer sus propios resultados). En este caso se espera que el sistema opere de forma normal, detectándo y descartando a esta autoridad de los protocolos. El resultado de esta prueba muestra el correcto funcionamiento del sistema, es decir, a pesar de la autoridad deshonesta el sistema realiza los protocolos correctamente y obtiene los resultados finales esperados. El verificador externo genera el reporte donde informa las etapas donde la autoridad deshonesta genera *outputs* inválidos, identificándola como corrupta. La verificación se realiza ignorándo a esta autoridad y validando los resultados obtenidos.

- Escenario con dos autoridades deshonestas. Bajo este escenario el resultado del sistema pierde su validez. En este caso el verificador externo detecta que hay dos autoridades deshonestas e informa de la invalidez de los resultados obtenidos.

- Fallas externas durante los protocolos. En este escenario (con múltiples sub-escenarios) se probó el comportamiento de los esquemas de restauración de los protocolos de las autoridades. Se generaron fallas que hacen detener el procesamiento de una o más autoridades, en cada una de las etapas de los protocolos. Se espera que al ejecutar nuevamente la autoridad, esta pueda retomar el proceso y el estado en el cual se encontrado, continuando el protocolo a partir de ese momento sin afectar el funcionamiento del resto, las cuales se mantienen en espera. El resultado de estas pruebas es el esperado, los protocolos son restituidos correctamente después de una falla y los resultados obtenidos del conteo de votos son correctos.

- Desconexión entre los clientes y el servidor. En este escenario se simuló la pérdida de conectividad entre los clientes (cámaras o mesas de votación) con el servidor, por lo tanto éstas no pueden publicar los votos ni las claves públicas de los votantes. Se espera en este escenario que los clientes almacenen de forma persistente estos datos utilizando caché y envien los datos una vez que recuperen conexión al servidor. El resultado de estas pruebas muestra el correcto funcionamiento del caché del sistema. No hay pérdida de datos, incluso si se apaga la máquina del cliente. Los datos son publicados correctamente una vez recuperada la conexión y el resultado final de la elección obtenido es correcto.

# Capítulo 6

# Extensiones y posibles mejoras

## BulletinBoard distribuido

Se puede incorporar un mecanismo de *broadcast* de mensajes eficiente y confiable, que permita distribuir el BulletinBoard en tres o más servidores, de esta manera se puede utilizar un protocolo donde se consideren válidos los datos cuando la mayoría de los participantes (servidores) estén de acuerdo. Al tener el BulletinBoard distribuido se elimina el punto único de fallo del sistema y no habría necesidad de depositar absoluta confianza en sólo un servidor.

## Extender la cantidad de Autoridades de votación

El sistema original fue implementado para distribuir los protocolos criptográficos de votación en tres autoridades, esto no se modificó en este trabajo ya que estaba fuera del alcance definido. Esta limitación puede ser eliminada implementando los protocolos de forma que utilicen cualquier cantidad de autoridades (impar) mayor a tres, así si se tienen $n$ autoridades, donde $n$ es impar, bastaría que sólo $\lceil n/2 \rceil$ autoridades sean honestas para tener un proceso de elección válido y correcto. Por ejemplo, si el proceso se distribuyera en 9 participantes (autoridades), incluso si 4 participantes son deshonestos o se coluden para modificar o anular la elección, el proceso seguiría siendo válido y correcto ya que habrían 5 autoridades honestas.

Con este cambio sería considerablemente más difícil invalidar una elección ya que se necesitaría que se coludan una cantidad mucho mayor de participantes (autoridades) para invalidar o modificar una elección, logrando mayor confiabilidad en el proceso.

## Utilización de servicios Web

La utilización de RMI para conectar los componentes del sistema impone la restricción de que todos los componentes del sistema deben ser implementados con Java y RMI, por lo tanto el sistema no puede interoperar con distintas tecnologías. Éste es un problema especialmente para la implementación del módulo de la *Cámara* en donde el votante emite el voto, ya que sería deseable implementar una interfaz de usuario para la *Cámara* que pueda ser utiliza en cualquier tipo de sistema o dispositivo, por ejemplo, se podría utilizar un *tablet* con pantalla táctil y sistema operativo *Android*, *Mac iOS* o *Windows*. Esto gracias a que la tecnología de servicios Web es soportada de forma estándar en gran parte de las plataformas en uso, permitiendo interoperar distintas tecnologías.

Adicionalmente se podría incorporar seguridad en las comunicaciones con una tecnología como *WS-Security*, así sería posible incluso reemplazar el uso de la red VPN por este esquema de seguridad.

## Interfaz de votación Web

Se podrían implementar las módulos de la *Mesa* y *Cámara* de votación utilizando interfaces Web, con esto se tendría una interfaz de usuario portable a múltiples dispositivos de hardware y sistemas operativos. Para implementar las interfaces Web sería necesario portar la lógica de estos módulos a un lenguaje para Web que funcione en el lado del cliente, como *JavaScript*.

Sería posible incorporar el concepto *single page Web application* el cual es utilizado por el sistema *Helios* (ver sección 2.1.1 en página 13). Con esta técnica la aplicación se carga de una vez en el navegador Web sin realizar peticiones al servidor hasta el momento de enviar la papeleta digital, reduciendo el riesgo de seguridad generado por el tráfico de datos a través de la red.

## Uso de interfaces de usuario pre-renderizadas

Las interfaces pre-renderizadas son interfaces que han sido generadas con anterioridad. Durante la ejecución de las interfaces éstas sólo se dibujan (imágenes), por lo que no se requieren librerías gráficas sofisticadas que podrían introducir posibles vulnerabilidades adicionales al sistema. Esta técnica de implementación de interfaces se propone en [12] para mejorar la seguridad del sistema y reducir las dependencias a librerías.

Esta técnica requiere un componente de software adicional para preparar y generar las interfaces antes de iniciar la elección, para que puedan ser utilizadas durante la elección.

## Uso de servicios *cloud computing*

Es posible utilizar servicios de computación en la nube como PaaS (*Platform as a Service*), por ejemplo es posible utilizar servicios de colas de mensajes robustos y de alta disponibilidad, los que además tienen costos de uso bastante económicos.

## Instalación en servidores *cloud computing*

Es posible construir el sistema de votación de forma que pueda ser desplegado en servidores virtualizados o VPS en la nube. Para esto se puede preparar un sistema operativo personalizado y preconfigurado para servidores de máquinas virtuales, por ejemplo, imágenes de disco de *VMWare*, los que pueden ser desplegados de forma simple en servicios que soporten tal plataforma.

## Extensión del módulo Verificador Externo

El módulo *Verificador Externo* puede ser extendido para que muestre reportes adicionales acerca del comportamiento y funcionamiento del sistema, por ejemplo, el sistema puede mostrar un registro detallado de los eventos ocurridos a lo largo del proceso de elección y notificar cuando se detecten situaciones anómalas o que podrían ser sospechosas.

## Reemplazo del *token* (*pendrive*) por otro dispositivo

Es posible reemplazar el *token* de votación utilizado (memoria *flash* USB) por otro tipo de dispositivos seguros, como los *tokens* criptográficos que tienen la capacidad de generar internamente claves privadas, firmar y encriptar. Esto tiene la ventaja de que las claves privadas nunca deben salir del dispositivo, haciendo muy difícil la obtención de estas claves por parte de un tercero.

## Seguridad y certificación de los ejecutables

Aunque no es posible asegurar que el software del sistema de votación funcionará en un entorno completamente aislado de *malware*, se podría implementar o utilizar un esquema de ejecutables firmados, de forma de permitir la ejecución sólo de la versión oficial del sistema y no de una versión alterada por terceros. Para esto se podría utilizar *hardware* especial que sólo ejecuta software certificado.

También sería importante analizar y abordar el problema de posibles ataques del tipo DoS o DDoS (*Distributed Denegation of Service*) y proponer posibles soluciones para enfrentarlo.

## Generación de parámetros iniciales del sistema de forma distribuida

Actualmente en el BulletinBoard se generan parámetros aleatorios al comienzo de la elección, los cuales son utilizados en los protocolos criptográficos de votación, es posible generar estos valores aleatorios de forma distribuida para asegurar que los valores sean realmente aleatorios.

Entre todos estas alternativas se recomienda con mayor prioridad extender la cantidad de autoridades que pueden participar en el sistema, junto con distribuir el BulletinBoard. Luego podría considerarse la implementación de interfaces Web y el uso de servicios *cloud computing*.

# Capítulo 7

# Conclusiones

En este trabajo de título se mejoró y extendió un prototipo de sistema de votación electrónica existente. Este prototipo utilizaba un esquema criptográfico de votación electrónica de tipo homomórfico, eficiente para múltiples candidatos. El objetivo de las mejoras fue conseguir un sistema más robusto, más seguro, tolerante a fallas, con mayor disponibilidad, que sea de fácil uso, fácil de configurar y administrar. La funcionalidad adicional más importante consistió en hacer el sistema *universalmente verificable* mediante una herramienta externa usable por cualquier usuario, atributo fundamental para dar transparencia al proceso, generando confianza en los usuarios y haciéndolo más resistente frente a colusión.

Los logros concretos de este trabajo son los siguientes:

- Se hizo una re-ingeniería del sistema, separando los componentes, e incorporando una arquitectura de capas de abstracción que separa adecuadamente las responsabilidades, haciendo el sistema más modificable y mantenible. El esquema de persistencia basado en archivos de texto plano, del sistema original, estaba implementado de una forma poco confiable o robusta. Este esquema de persistencia se reemplazó por una base de datos relacional usada de forma más robusta y confiable, que maneja concurrencia, transacciones y tolerancia a fallos. La persistencia es gestionada mediante una capa de acceso a datos usando la técnica *Object-Relational Mapping*.

- Se implementó una aplicación de escritorio, el *Verificador Externo*, que permite verificar en tiempo real todo el proceso de votación, la aplicación verifica los datos publicados durante las distintas etapas y protocolos usados por el sistema de votación, generando reportes acerca del éxito o fracaso de cada paso realizado. Esta herramienta permite a cualquier votante u observador comprobar los votos válidos, detectar que las *Autoridades de Conteo* sean honestas en cada etapa y comprobar que el resultado de la elección calculado por el sistema sea el correcto.

- Se implementó una herramienta con interfaz de línea de comandos para configurar una nueva elección. Esta aplicación recibe los datos de la elección (carreras, candidatos y votantes) en un formato de datos simple, estos datos son validados, comprobando que cumplan con las reglas de necesarias de elección, finalmente se genera un archivo XML con los datos de configuración correctos. Con esta herramienta se simplifica la configuración de una elección y se evita tener problemas durante la elección debido a la utilización de datos mal configurados o repetidos.

- Se añadió al sistema la cualidad de ser tolerante a fallas frente a fallas de hardware como cortes de electricidad. El sistema recuerda el estado en que se encuentra pudiendo restaurarse a la etapa en que se encontraba antes de la falla, así el funcionamiento del sistema no es comprometido y no se pierden los datos generados hasta ese momento.

- Se agregó a los módulos clientes (*Cámara* y *Mesa* de votación) un sistema de caché de datos persistente que permite seguir utilizando el sistema incluso en caso de que el servidor central (repositorio de datos) no se encuentre disponible. Esta persistencia se logra gracias al uso de dispositivos de memoria portable adjuntos a cada cámara y mesa de votación. De esta forma aumenta la disponibilidad del sistema de votación.

- Se incorporó el uso de una herramienta de *logging* de eventos detallada. Junto con esto se mejoró la manipulación y registro de errores o excepciones esperadas y no esperados, mejorando la auditabilidad y robustez del sistema.

- Se generaron sistemas operativos minimales, personalizados para el sistema de votación que funcionan sin necesidad de ser instalados ya que se cargan directamente en la memoria RAM. Se creó una versión para los servidores y otra para las aplicaciones cliente, basadas en Debian Linux.

- Se crearon *scripts* de instalación del sistema que permiten configurar los servidores y clientes de forma simplificada. Con ésto se logró reducir considerablemente el tiempo de despliegue y configuración del sistema desde varios días, hasta minutos. Además se reduce la posibilidad de cometer errores en el proceso, junto con mejorar la aprendibilidad y facilidad de uso del sistema.

- También se corrigieron múltiples *bugs* que no habían sido reportados previamente y se refactorizó parte de la implementación en distintos módulos del sistema. Los *bugs* se encontraban en las interfaces de la Cámara de votación, la Mesa de votación y en los protocolos distribuidos de las autoridades.

Se ha logrado implementar un prototipo de sistema de votación electrónica con múltiples cualidades que lo hacen un candidato para ser utilizado en elecciones reales, ya que aborda los problemas técnicos y prácticos que son importantes a la hora de implantar un sistema de este tipo, como la usabilidad, robustez y mantenibilidad, abarcando más allá de los protocolos criptográficos y modelos teóricos. Se puede considerar este trabajo como un acercamiento real a un producto de software para votación electrónica, por lo que el sistema podría ser utilizado como base para obtener un producto final.

Como se describió en el capítulo *Extensiones y posibles mejoras*, el sistema puede ser mejorado de diversas maneras:

- Distribuyendo el *BulletinBoard* para eliminar el único punto de falla.

- Mejorar los protocolos criptográficos para hacer el sistema más resistente a colusión.

- Utilizar tecnologías como servicios web o interfaces web que permitirían usar el sistema en distintas plataformas, mejorando la experiencia de uso del sistema. Mejorar el renderizado de interfaces de usuario con la técnica de interfaces pre-renderizadas.

- Incorporar la utilización de servicios *cloud computing* que podrían hacer mas confiable y robusto el sistema.
- Extender el *Verificador Externo* para mostrar información más detallada acerca del funcionamiento del sistema.

# Glosario

**API:** (*Application Programming Interface*) Interfaz de programación de aplicaciones o API es el conjunto de funciones y procedimientos (o métodos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

**BD:** (Base de Datos).

**CSV:** (*Comma Separated Values*) Los ficheros CSV son un tipo de documento en formato abierto sencillo para representar datos en forma de tabla, en las que las columnas se separan por comas (o punto y coma) y las filas por saltos de línea.

**DRE:** (*Direct-Recording Electronic voting machine*) Las máquinas de votar electrónicas de registro directo (DRE) son sistemas computacionales que graban los votos por medio de una papeleta de votación en forma de pantalla provista de componentes mecánicos o eléctrico-ópticos que pueden ser activados por el votante (típicamente botones o pantalla de digitación).

**IaaS:** (*Infraestructure As A Service*) Infraestructura como servicio (IaaS), también llamado en algunos casos hardware as a service, se encuentra en la capa inferior del paradigma de computación en la nube y es un medio de entregar almacenamiento básico y capacidades de cómputo como servicios estandarizados en la red.

**JDO:** (*Java Data Object*) JDO es una especificación de objetos de persistencia de Java. Una de sus características es una transparencia entre los servicios de persistencia y el modelo de dominio.

**JEE:** (*Java 2 Enterprise Edition*) Java Platform, Enterprise Edition o Java EE (anteriormente conocido como J2EE), es una plataforma de programación para desarrollar y ejecutar software de aplicaciones en el lenguaje de programación Java con arquitectura de N capas distribuidas y que se apoya ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones.

**JPA:** (*Java Persistence API*) JPA es un *framework* del lenguaje de programación Java que maneja datos relacionales en aplicaciones usando la Plataforma Java en sus ediciones Standard (Java SE) y Enterprise (Java EE).

**JSON:** (*JavaScript Object Notation*) JSON es un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.

**OOP:** (*Object-Oriented Programming*) Paradigma de programación que utiliza "objetos" - estructuras de datos que consisten en campos de datos y métodos con sus interacciones - para diseñar aplicaciones y programas de cómputo.

**ORM:** (*Object-Relational Mapping*) Es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional, utilizando un motor de persistencia.

**PaaS:** (*Platform As A Service*) Plataforma como servicio es la encapsulación de una abstracción de un ambiente de desarrollo y el empaquetamiento de una carga de servicios. Es parte del paradigma de computación en la nube.

**RMI:** (*Remote Method Invocation*) Es un mecanismo ofrecido por Java para invocar un método de manera remota. Forma parte del entorno estándar de ejecución de Java y proporciona un mecanismo simple para la comunicación de servidores en aplicaciones distribuidas basadas exclusivamente en Java.

**RPC:** (*Remote Procedure Call*) Es un protocolo que permite a un programa de ordenador ejecutar código en otra máquina remota sin tener que preocuparse por las comunicaciones entre ambos.

**RSA:** (*Rivest, Shamir y Adleman*) RSA es un sistema criptográfico de clave pública desarrollado en 1977. Es el primer y más utilizado algoritmo de este tipo y es válido tanto para cifrar como para firmar digitalmente.

**SaaS:** (*Software As A Service*) Software como servicio caracteriza una aplicación completa ofrecida como un servicio, en-demanda, vía multitenencia ?que significa una sola instancia del software que corre en la infraestructura del proveedor y sirve a múltiples organizaciones de clientes. Es parte del paradigma de computación en la nube.

**SDK:** (*Software Development Kit*) Es generalmente un conjunto de herramientas de desarrollo de software que le permite al programador crear aplicaciones para un sistema concreto.

**SQL:** (*Structured Query Language*) El lenguaje de consulta estructurado o SQL es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en estas.

**UML:** (*Unified Modeling Language*) UML es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema.

**VPN:** (*Virtual Private Network*) Una red privada virtual (o VPN), es una tecnología de red que permite una extensión de la red local sobre una red pública o no controlada.

**VPS:** (*Virtual Private Server*) Un servidor virtual privado (o VPS) es un método de particionar un servidor físico en varios servidores de tal forma que todo funcione como si se estuviese ejecutando en una única máquina. Cada servidor virtual es capaz de funcionar bajo su propio sistema operativo y además cada servidor puede ser reiniciado de forma independiente.

**XML:** (*eXtensible Markup Language*) XML es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C), no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades, de ahí que se le denomine metalenguaje. Se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas.

**XSD:** (*XML Schema Definition*) XSD especifica como describir fomalmente los elementos en un documento usando XML.

**YML (YAML):** (*YAML Ain't Another Markup Language*) YAML es un formato de serialización de datos legible por humanos inspirado en lenguajes como XML.

**ZKP:** (*Zero-Knowledge Proof*) Es un método interactivo para que un participante le pruebe a otro que una afirmación (usualmente matemática) es verdadera, sin revelar nada más que la veracidad de la afirmación.

# Bibliografía


[1] Dimitris A. Gritzalis, editor. *Secure Electronic Voting*. Kluwer Academics Publishers (2003)

[2] Richard Celeste, Dick Thornburgh, and Herbert Lin, editors. *Asking The Right Questions About Electronic Voting*. The National Academic Press, Washington D.C. (2006)

[3] Cordero, Pamela. *Implementación de un Sistema de Votación Electrónica Eficiente Para Múltiples Candidatos*. Memoria (Ingeniería civil en computación). Santiago, Chile. Universidad de Chile, Facultad de Ciencias Físicas y Matemáticas. (2008)

[4] Ka-Ping Yee. *Building Reliable Voting Machine Software*. A dissertation submitted to the Graduate Division of the University of California, Berkeley in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science (2007)

[5] Ben Adida. *Helios: Web-based Open-Audit Voting*. Harvard University (2008)

[6] Daniel Sandler, Kyle Derr, Dan S. Wallach. *VoteBox: a tamper-evident, verifiable electronic voting system*. Rice University (2008)

[7] Ryan, Bismarck, Heather, Schneider, Xia. *The Prêt à Voter Verifiable Election System*. University of Luxembourg, University of Surrey (2008)

[8] Ivan Damgard, Jens Groth and Gorm Salomonsen. *The Theory and Implementation of an Electronic Voting System*. (2002)

[9] Miranda, Segio. *Implementación y evaluación de un sistema de votación electrónica, basado en técnicas criptográficas, para una votación de pequeña escala*. Memoria (Ingeniería civil en computación). Santiago, Chile. Universidad de Chile (2008)

[10] Kenneth P. Birman. *Reliable Distributed Systems*. Springer (2005)

[11] Kazue Sako and Joe Kilian. *Receipt-free mix-type voting scheme - a practical solution to the implementation of a voting booth*. In EUROCRYPT, pages 393-403, 1995.

[12] Ka-Ping Yee. *Extending prerendered-interface voting software to support accessibility and other ballot features*. University of California, Berkeley (2007)

[13] Austin Mohr. *A Survey of Zero-Knowledge Proofs with Applications to Cryptography*. Southern Illinois University at Carbondale (2008)

[14] Shamir. *How to share a secret*. CACM: Communications of the ACM, 22, 1979.

[15] Gennaro, Jarecki, Krawezyk, and Rabin. *Secure distributed key generation for discrete-log based cryptosystems*. JCRYPTOL: Journal of cryptology, 20, 2007.

[16] Juric, Kezman, Hericko, Rozman, Vezocnik. *Java RMI, RMI Tunneling and Web Services Comparison and Performance Analysis*. University of Maribor, 2004.

[17] Rafael Navarro Marset. *REST vs Web Services*. Curso Modelado, Diseño e Implementación de Servicios Web 2007. Universidad Politécnica de Valencia.

[18] Philippe Kruchten. *Architectural Blueprints - The "4+1" View Model of Software Architecture*. Rational Software Corp (1995).

[19] Linda H. Rosenberg. *Software Re-engineering*. Software Assurance Technology Center.

[20] Martin Fowler. *Refactoring* [on-line]. http://martinfowler.com/refactoring/, abril 2012 (última visita).

[21] Maximiliano Cristiá. *Introducción al Testing de Software*. Universidad Nacional de Rosario (2009)

[22] Debian Project. *Debian live build manual* [on-line]. http://live.debian.net/manual/index.en.html, abril 2012 (última visita).

# Anexo A: Script de creación de la base de datos

A continuación se adjunta un *script* de creación de la base de datos en SQL para SQLite 3. El modelo entidad-relación correspondiente puede ser visto en la figura 5.20, página 64.

```
-- Table: Mesa
CREATE TABLE Mesa (
    id INTEGER PRIMARY KEY
               NOT NULL
);

-- Table: Papeleta
CREATE TABLE Papeleta (
    Votante_id    INTEGER NOT NULL,
    serial        VARCHAR NOT NULL,
    timestamp     INTEGER NOT NULL,
    signedObject VARCHAR NOT NULL,
    Mesa_id       INTEGER NOT NULL
                          REFERENCES Mesa ( id ) ON DELETE RESTRICT
                                                 ON UPDATE RESTRICT,
    Carrera_id    INTEGER NOT NULL,
    id            INTEGER PRIMARY KEY AUTOINCREMENT
                          NOT NULL
);

-- Table: Camara
CREATE TABLE Camara (
    id        INTEGER PRIMARY KEY
                      NOT NULL,
    Mesa_id INTEGER REFERENCES Mesa ( id ) ON DELETE RESTRICT
                                           ON UPDATE RESTRICT
);

-- Table: HistorialVotante
CREATE TABLE HistorialVotante (
    Votante_id INTEGER NOT NULL,
    Mesa_id    INTEGER,
    timestamp  INTEGER,
    estado     VARCHAR NOT NULL,
    id         INTEGER PRIMARY KEY AUTOINCREMENT
                       NOT NULL,
    FOREIGN KEY ( Mesa_id ) REFERENCES Mesa ( id ) ON DELETE RESTRICT
                                                   ON UPDATE RESTRICT
);
```

```
-- Table: Log
CREATE TABLE Log (
    id          INTEGER PRIMARY KEY AUTOINCREMENT,
    timestamp   INTEGER,
    log         VARCHAR NOT NULL
);

-- Table: HistorialMesa
CREATE TABLE HistorialMesa (
    Mesa_id     INTEGER NOT NULL
                        REFERENCES Mesa ( id ) ON DELETE RESTRICT
                                               ON UPDATE RESTRICT,
    timestamp   INTEGER,
    estadoMesa  VARCHAR NOT NULL,
    id          INTEGER PRIMARY KEY AUTOINCREMENT
);

-- Table: EnvioGeneral
CREATE TABLE EnvioGeneral (
    Autoridad_id     INTEGER,
    protocolo        VARCHAR,
    paso             INTEGER NOT NULL,
    xmlEnvioFirmado  VARCHAR NOT NULL,
    timestamp        INTEGER,
    id               INTEGER PRIMARY KEY
                             NOT NULL
);

-- Table: AutoridadPK
CREATE TABLE AutoridadPK (
    Autoridad_id INTEGER PRIMARY KEY
                         NOT NULL,
    xmlPublicKey VARCHAR NOT NULL
);

-- Table: VotantePK
CREATE TABLE VotantePK (
    Votante_id INTEGER NOT NULL,
    publicKey  VARCHAR NOT NULL,
    id         INTEGER PRIMARY KEY ASC AUTOINCREMENT,
    timestamp  INTEGER
);
```

# Anexo B: Documento XML de definición de una elección

El siguiente documento XML es un ejemplo de cómo se representan los datos de definición de una elección.

```
<?xml version="1.0" encoding="UTF-8" ?>
<eleccion titulo="CADCC 2012">
  <carreras class="java.util.ArrayList">
    <carrera id="1" titulo="CADCC 2012" nMaximoAEscoger="1"
             nCandidatos="2" nulos="true" blancos="true">
      <idCandidatos class="java.util.ArrayList">
        <integer>0</integer>
        <integer>1</integer>
        <integer>2</integer>
        <integer>3</integer>
      </idCandidatos>
    </carrera>
  </carreras>
  <candidatos class="java.util.ArrayList">
    <candidato id="0" orden="1" nombre="Born to be Geeks" agrupacion=""/>
    <candidato id="1" orden="2" nombre="The Hipster List" agrupacion=""/>
    <candidato class="evote.model.electionConfig.CandidatoNulo" id="2"
               orden="-1" nombre="Voto Nulo"/>
    <candidato class="evote.model.electionConfig.CandidatoBlanco" id="3"
               orden="-1" nombre="Voto en Blanco"/>
  </candidatos>
  <votantes class="java.util.ArrayList">
    <votante id="0" rut="11111-1" nombre="Rodrigo Porras">
      <idCarreras class="java.util.ArrayList">
        <integer>1</integer>
      </idCarreras>
    </votante>
    <votante id="2" rut="22222-2" nombre="Matías Acuña">
      <idCarreras class="java.util.ArrayList">
        <integer>1</integer>
      </idCarreras>
    </votante>
    <votante id="3" rut="33333-3" nombre="Nicolás Aguilera">
      <idCarreras class="java.util.ArrayList">
```

```
            <integer>1</integer>
          </idCarreras>
        </votante>
      </votantes>
    </eleccion>
```