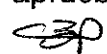


ACUERDO DEL CONSEJO GENERAL DEL INSTITUTO ELECTORAL DEL DISTRITO FEDERAL POR EL QUE SE MODIFICA EL PROGRAMA INFORMÁTICO (SOFTWARE ELECTORAL) QUE SE UTILIZARÁ EN LAS URNAS ELECTRÓNICAS EL DÍA DE LA JORNADA ELECTORAL DE 2009, APROBADO EN LA SESIÓN EXTRAORDINARIA CELEBRADA EL 29 DE ABRIL DE 2009.

CONSIDERANDO

1. Que el Instituto Electoral del Distrito Federal es autoridad en materia electoral, responsable de la función estatal de organizar las elecciones locales, independiente en sus decisiones, autónomo en su funcionamiento y profesional en su desempeño, de conformidad con lo dispuesto en los artículos 123 y 124 del Estatuto de Gobierno del Distrito Federal.
2. Que de acuerdo a lo previsto en el artículo 127 del Estatuto de Gobierno del Distrito Federal, el Instituto Electoral del Distrito Federal tiene a su cargo en forma integral y directa, entre otras, las actividades relativas a la preparación de la jornada electoral.
3. Que el 5 de enero de 1999, se publicó en la Gaceta Oficial del Distrito Federal el Código Electoral del Distrito Federal, vigente a partir del día siguiente y en cuyo Libro Tercero, Título Primero, dispone la creación del Instituto Electoral del Distrito Federal.
4. Que el Código Electoral del Distrito Federal publicado el 5 de enero de 1999 en la Gaceta Oficial del Distrito Federal fue abrogado por el nuevo Código Electoral del Distrito Federal aprobado el 20 de noviembre de 2007 por la Asamblea Legislativa del Distrito Federal y publicado el 10 de enero de 2008.
5. Que de conformidad con lo dispuesto por el artículo 2° párrafo tercero del Código Electoral del Distrito Federal, en relación con el artículo 86, párrafo segundo, fracción V del mismo ordenamiento, el Instituto Electoral del Distrito Federal, tiene otorgadas diversas atribuciones para el debido cumplimiento de sus fines y acciones, entre las que se encuentra "Preservar la autenticidad y efectividad del sufragio", se regirá por los principios de certeza, legalidad, independencia, imparcialidad, objetividad y equidad.
6. Que de acuerdo con lo establecido por el artículo 3°, párrafo primero del Código Electoral del Distrito Federal, la autoridad electoral y los procedimientos electorales garantizarán el voto universal, libre, secreto, personal e intransferible.
7. Que el Consejo General es el órgano superior de dirección del Instituto Electoral del Distrito Federal, de conformidad con lo dispuesto por el artículo 89, párrafo primero del Código Electoral del Distrito Federal. 

8. Que con fundamento en el artículo 95, fracciones XXII y XXIII del Código Electoral del Distrito Federal, el Consejo General del Instituto Electoral del Distrito Federal tiene, entre sus atribuciones, la de aprobar el modelo y los formatos de la documentación, materiales electorales, medios electrónicos y sistemas relativos al ejercicio del voto, a través de instrumentos electrónicos para el proceso electoral, así como aprobar las características de los elementos que permitan la utilización de instrumentos electrónicos para el ejercicio del voto, entre los que deberá considerar el programa informático (software electoral) y el instrumento que permita la secrecía del sufragio.
9. Que en términos de lo dispuesto por el artículo 96, párrafos primero y tercero del Código Electoral del Distrito Federal, el Consejo General cuenta con Comisiones Permanentes para el desempeño de sus atribuciones y supervisión del adecuado desarrollo de las actividades de los órganos ejecutivos y técnicos del Instituto Electoral del Distrito Federal, que auxiliará al Consejo General en lo relativo a su área de actividades; también prevé que los Secretarios, Ejecutivo y Administrativo y los órganos ejecutivos y técnicos prestarán todo el apoyo y la información que éstas requieran para el cumplimiento de sus atribuciones o de las tareas que se les hayan encomendado.
10. Que el artículo 97, fracción III del Código Electoral del Distrito Federal señala que entre las Comisiones Permanentes del Consejo General del Instituto Electoral del Distrito Federal se encuentra la Comisión de Organización y Geografía Electoral.
11. Que el artículo 102, fracciones II y V del Código Electoral del Distrito Federal, le confiere a la Comisión de Organización y Geografía Electoral, entre otras atribuciones, la de conocer los diseños y modelos de la documentación y materiales electorales, así como de los sistemas para el ejercicio del voto a través de instrumentos electrónicos, elaborados por órgano ejecutivo en materia de Organización y Geografía Electoral con apoyo, en su caso, de las instancias necesarias al efecto.
12. Que el artículo 213, párrafos primero, segundo y tercero del Código Electoral del Distrito Federal dicen que el Instituto Electoral del Distrito Federal podrá hacer uso de sistemas electrónicos de votación en los procesos electorales y de participación ciudadana, los cuales deberán garantizar la efectividad y autenticidad del sufragio. Para el ejercicio de la potestad a que hace referencia el párrafo anterior, el Consejo General del Instituto Electoral del Distrito Federal aprobará los Programas y Proyectos específicos, así como el presupuesto respectivo para la incorporación paulatina o gradual de instrumentos electrónicos para el ejercicio del voto; y todas las disposiciones que se hagan necesarias al efecto. Para la votación electrónica se establecerá un sistema que incluya, cuando menos, los siguientes elementos: I. Los instrumentos electrónicos para el ejercicio del voto; II. El programa informático electoral (software electoral); y III. El instrumento que garantice la secrecía del voto.
13. Que el artículo 214, párrafos primero, segundo y cuarto, del Código Electoral del Distrito Federal, confiere que cuando el Consejo General del Instituto apruebe las



secciones electorales en que se considere utilizar instrumentos electrónicos para el ejercicio del voto, deberá aprobar el diseño de los anteproyectos de sistemas y elementos para el ejercicio del voto, en particular del programa informático o software electoral.

Además, se define al software electoral del instrumento electrónico para la recepción del voto, como el conjunto de programas informáticos que permitan realizar, conforme lo previsto al Código Electoral del Distrito Federal, la habilitación e inhabilitación del instrumento electrónico de recepción del voto y el desarrollo de la votación electrónica y el cómputo de casillas.

También se establece que el software electoral deberá hacerse público por los medios y con la anticipación que el Consejo General del Instituto determine, con fines de transparencia y, además, deberá publicarse de manera permanente en el sitio de Internet del Instituto Electoral del Distrito Federal. Lo anterior, a efecto de garantizar que el software utilizado por los sistemas electrónicos de votación del día de la elección se corresponde plenamente con el publicado en el sitio oficial de Internet del Instituto Electoral local. Dicho software se firmará electrónicamente, a más tardar treinta días naturales previos al proceso electivo en que habrá de utilizarse.

14. Que el 9 de noviembre de 2006 en acuerdo ACU-332-06, el Consejo General del Instituto Electoral del Distrito Federal aprobó los Programas Generales del Instituto, entre los que se encuentran los relativos a la innovación tecnológica y de organización de los procesos electorales y de participación ciudadana.

Que en esa misma sesión del 9 de noviembre de 2006 en acuerdo ACU-333-06, el órgano superior de dirección del Instituto aprobó el Plan General de Desarrollo del IEDF 2006-2009, en el cual se consideran líneas estratégicas, entre las que están, la modernización tecnológica, que tiene por objeto incrementar la capacidad de innovación tecnológica en la organización electoral; la organización y capacitación electoral, que tiene como objetivo implementar los proyectos relacionados con la organización y capacitación electoral, buscando perfeccionar los métodos y herramientas que se emplean en los procesos electorales y de participación ciudadana, para asegurar a los ciudadanos el ejercicio de sus derechos políticos y electorales.

15. Que mediante los acuerdos ACU-696-03 y ACU-018-04, de fechas 30 de octubre de 2003 y 23 de marzo de 2004, respectivamente, el Consejo General del Instituto Electoral del Distrito Federal ordenó a la Comisión de Organización y Geografía Electoral que, con el apoyo de la Dirección Ejecutiva de Organización y Geografía Electoral, de la entonces Unidad de Informática y de las demás áreas del Instituto, llevara a cabo las acciones necesarias que permitieran el diseño de una urna electrónica para el ejercicio del voto de los ciudadanos.

16. Que en cumplimiento del artículo 69, fracciones II y IV del Código Electoral del Distrito Federal vigente en los años 2004 y 2005, y de los acuerdos señalados en el

 6.

numeral anterior, la Comisión de Organización y Geografía Electoral llevó a cabo diversas acciones para el diseño y producción semi-industrial de un modelo institucional de urna electrónica, con el apoyo de la Dirección Ejecutiva de Organización y Geografía Electoral, de la otrora Unidad de Informática y de las demás áreas del Instituto, con la participación de la Universidad Nacional Autónoma de México, del Instituto Politécnico Nacional, de la Universidad Autónoma Metropolitana y del Instituto Tecnológico y de Estudios Superiores de Monterrey, campus Ciudad de México.

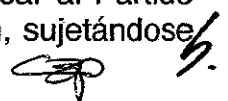
17. Que de conformidad con los convenios específicos de apoyo y colaboración, el primero celebrado entre el Instituto Politécnico Nacional (IPN) y este Instituto, y otro similar celebrado entre la Universidad Autónoma Metropolitana (UAM) y este Instituto, ambos de fecha 12 de mayo de 2004, se establece que tienen por objeto la elaboración de las especificaciones técnicas y construcción de un prototipo de una urna electrónica para futuros procesos electorales en el Distrito Federal y hacen referencia a las rutinas que se ejecutarán en el prototipo de urna electrónica.
18. Que con base en los prototipos desarrollados por las instituciones de educación superior en el año 2004 la otrora Unidad de Informática, con el apoyo de los grupos de trabajo, procedió a integrar una propuesta de diseño institucional de urna electrónica.
19. Que en su sesión del 25 de febrero de 2005, la Comisión de Organización y Geografía Electoral acordó darse por enterada de las características técnicas del diseño base de urna electrónica institucional, elaborados por la otrora Unidad de Informática, a partir de la integración de las propuestas presentadas por las instituciones de educación superior. Asimismo, aprobó informar a este órgano superior de dirección del avance en los trabajos desarrollados en el proyecto de diseño de la urna electrónica, señalados en los acuerdos ACU-696-03 y ACU-018-04 de fechas 30 de octubre de 2003 y 23 de marzo de 2004, respectivamente.
20. Que la Comisión de Organización y Geografía Electoral el 25 de febrero de 2005, adoptó el acuerdo 36-E2-05, por medio del cual autorizó a la entonces Unidad de Informática iniciar con la producción de hasta 60 urnas electrónicas con las características del diseño base presentado por dicha área técnica, en colaboración con las instituciones educativas.
21. Que en reunión de trabajo realizada el 8 de marzo de 2007, los consejeros electorales integrantes del Consejo General del Instituto Electoral del Distrito Federal se manifestaron a favor de utilizar urnas electrónicas con efectos vinculantes en la elección local del 2009 en el Distrito Federal.
22. Que en su sesión del 30 de octubre de 2007, el Consejo General recibió el Informe final de la Comisión de Organización y Geografía Electoral sobre el cumplimiento de los acuerdos del Consejo General ACU-696-03 y ACU-018-04, por los que se ordenó a la Comisión de Organización y Geografía Electoral para que, con apoyo de la Dirección Ejecutiva de Organización y Geografía Electoral y de la otrora Unidad

cap

de Informática, procediera a realizar acciones necesarias que permitieran el diseño de una urna electrónica para el ejercicio de los votos de los ciudadanos.

23. Que el 11 de enero de 2008, entró en vigor el nuevo Código Electoral del Distrito Federal, el cual, entre otras disposiciones, otorgó la atribución del Consejo General del Instituto Electoral del Distrito Federal para aprobar los sistemas relativos al ejercicio del voto a través de instrumentos electrónicos de votación en los procesos electorales y de participación ciudadana en el Distrito Federal, de conformidad con su artículo 95, fracción XXII.
24. Que la Dirección Ejecutiva de Organización y Geografía Electoral, de conformidad con lo señalado en el artículo 116, fracción II del Código Electoral del Distrito Federal, tiene entre sus atribuciones la de presentar a la Comisión de Organización y Geografía Electoral los anteproyectos de los diseños y modelos de la documentación y materiales electorales de los procesos electorales y de participación ciudadana y, en su caso, la documentación, materiales, elementos y demás sistemas que sean necesarios para el ejercicio del voto a través de instrumentos electrónicos.
25. Que de conformidad con el artículo 116, fracción II del Código Electoral del Distrito Federal en correlación con el artículo 59, fracciones V y IX del Reglamento Interior del Instituto Electoral del Distrito Federal a la Unidad Técnica de Servicios Informáticos conforme a las características de equipo, soporte y demás paqueterías; así como investigar y analizar permanentemente, nuevas tecnologías en informática y comunicaciones, para proponer su aplicación en el Instituto.
26. Que el Grupo de Desarrollo Procedimental y Normativo en reunión de trabajo realizada el 25 de febrero de 2009, aprobó el documento denominado "Procedimientos para la organización de procesos electorales y de participación ciudadana mediante el uso de instrumentos electrónicos para el ejercicio del voto", el cual establece los procedimientos técnicos y operativos para llevar a cabo la instrumentación del uso de urnas electrónicas para la emisión y cómputo de votos en los procesos electorales y de participación ciudadana en el Distrito Federal, mismo que fue conocido por la Comisión de Organización y Geografía Electoral en su sesión del 9 de marzo de 2009.
27. Que la Comisión de Organización y Geografía Electoral conoció y emitió una opinión favorable en su cuarta sesión ordinaria del 23 de abril de 2009, sobre el presente acuerdo, razón por la cual acordó remitir el documento y los anexos que lo describen al Secretario Ejecutivo para que sirva de conducto a fin de ponerlo a consideración del Consejo General en una próxima sesión.
28. Que en la Comisión de Organización y Geografía Electoral en su cuarta sesión ordinaria del 23 de abril de 2009, fue presentado a su consideración el programa informático (software electoral) conforme lo establecido en el artículo 214, primer párrafo del Código Electoral del Distrito Federal.  

29. Que el software electoral presentado es compatible con los sistemas del Instituto en materia de cómputo distrital, por tipo de elección y los demás sistemas para el proceso electoral, atendiendo a lo establecido en el artículo 214, quinto párrafo del Código Electoral del Distrito Federal.
30. Que el software electoral contiene los siguientes elementos: fecha de la jornada electoral; distrito electoral y delegación, en su caso; sección electoral; tipo de casilla; tipo de cargo a elegir (Jefe Delegacional o Diputado a la Asamblea Legislativa del Distrito Federal por el principio de mayoría relativa); número total de electores de la lista nominal; número de votantes; número de votos emitidos a favor de cada uno de los candidatos por Partido Político; apellido paterno, apellido materno y nombre completo del candidato o candidatos, según el cargo a elegir; emblema del partido Político; fórmula de candidatos (propietario y suplente); las firmas electrónicas de la Consejera Presidenta del Consejo General y del Secretario Ejecutivo del Instituto; el orden de los Partidos Políticos aparece de acuerdo a la antigüedad de su registro. Respecto a la emisión del comprobante impreso por cada voto, contiene: clave única que permite asociar de manera indubitable al comprobante impreso con la urna electrónica que emitió ese comprobante; el tipo de elección que corresponda al voto emitido y las siglas del Partido Político, de acuerdo con lo establecido en el artículo 214, quinto párrafo del Código Electoral del Distrito Federal.
31. Que el software electoral atiende los lineamientos señalados en los artículos 214 y 215 del Código Electoral del Distrito Federal. En ese sentido, el software electoral contiene la lista de candidatos a diputados por el principio de representación proporcional, misma que, adicionalmente, será colocada en la mampara de la urna electrónica; ello en virtud de que resulta operativamente inviable incluir en la pantalla de la urna electrónica la lista de representación proporcional, pues los nombres aparecerían demasiado pequeños para su lectura, lo que conllevaría a incrementar el tiempo de votación por ciudadano, situación que comprometería el flujo de la votación.
32. Que el artículo 243, fracciones II y III del Código Electoral del Distrito Federal, señala los plazos para recibir las solicitudes de registro de las candidaturas en el año de la elección para Diputados electos por el principio de mayoría relativa y Jefes Delegacionales, serán del 10 al 20 de abril, por lo que el código del programa informático (software electoral) deberá incorporar los nombres de dichos candidatos.
33. Que de conformidad con el artículo 246 del Código Electoral del Distrito Federal para la sustitución de candidatos, los Partidos Políticos o Coaliciones, lo solicitarán por escrito al Consejo General, observando las siguientes disposiciones: I. Dentro del plazo establecido para el registro de candidatos podrán sustituirlos libremente; II. Vencido el plazo a que se refiere la fracción anterior, exclusivamente podrán sustituirlos por causas de fallecimiento, inhabilitación decretada por autoridad competente, incapacidad declarada judicialmente; y III. En los casos de renuncia del candidato, la sustitución podrá realizarse siempre que ésta se presente a más tardar 30 días antes de la elección. En este caso el candidato deberá notificar al Partido Político o Coalición que lo registró, para que proceda, a su sustitución, sujetándose



a lo dispuesto por este Código para el registro de candidatos. En atención a lo anterior el código del programa informático (software electoral) podría sufrir cambios.

34. Que de conformidad con el artículo 248 párrafos primero, segundo, tercer y cuarto dicen que las boletas serán impresas dentro de los treinta días posteriores al registro de candidatos. En caso de cancelación del registro o sustitución de uno o más candidatos, no habrá modificación a las boletas si éstas ya estuvieran impresas; en todo caso, los votos contarán para los Partidos Políticos, las Coaliciones y los candidatos que estuviesen legalmente registrados ante los Consejos General o Distrital correspondientes. En caso de que el Instituto haya determinado la utilización de instrumentos electrónicos para la recepción del voto, y oportunamente haya aprobado las secciones electorales en que se habrán de operar, el software electoral a utilizar en las elecciones respectivas deberá cargarse o integrarse en los respectivos instrumentos electrónicos para la recepción del voto, configurándose además con los sistemas y bases de datos necesarios para su funcionamiento, previamente aprobados por el propio Instituto. Con la participación de los Partidos Políticos. Lo anterior se realizará dentro de los treinta días siguientes a la aprobación de los registros de candidatos. Las actividades antes citadas se realizarán en los órganos desconcentrados, para las cuales se invitará a los integrantes de los Consejos Distritales correspondientes. En caso de nulidad del registro decretada por los órganos jurisdiccionales electorales y/o sustitución de uno o más candidatos, no podrá modificarse el software electoral en lo relativo a las boletas virtuales, si éstas ya estuvieran cargadas y configuradas en los respectivos instrumentos electrónicos que se utilicen en la elección, en este caso para el cómputo de los votos se estará a la última parte del primer párrafo de este artículo.
35. Que en sesión extraordinaria celebrada el 29 de abril de 2009, el Consejo General del Instituto Electoral del Distrito Federal aprobó el Acuerdo ACU-425-09, "Acuerdo del Consejo General del Instituto Electoral del Distrito Federal por el que se aprueba el programa informático (software electoral) que se utilizará en las urnas electrónicas el día de la jornada electoral de 2009", mismo que se hizo del conocimiento del Tribunal Electoral del Distrito Federal.
36. Que en esa misma sesión, fue aprobado el ACU-426-09 "Acuerdo del Consejo General del Instituto Electoral del Distrito Federal por el que se aprueba publicar en el sitio oficial de Internet del IEDF, el modelo de software electoral que será firmado electrónicamente".
37. Que el Tribunal Electoral del Distrito Federal aprobó el 5 de mayo de 2009, el Acuerdo 024/2009, en virtud del cual fueron expedidas las "Bases y Criterios con apoyo en los cuales el Tribunal Electoral del Distrito Federal, aplicará lo relativo a las nulidades establecidas en la Ley Procesal para el Distrito Federal, por la utilización de dispositivos electrónicos para la recepción de votación, de acuerdo con lo establecido con el artículo 94 de dicho ordenamiento".
38. Que en las Bases y Criterios citados en el considerando anterior, se estipula como causal de nulidad de la votación "cuando las boletas electrónicas puestas a

cap 5.

disposición de los electores por el Sistema Electrónico de Votación para el ejercicio del voto, no cumplan con las medidas de certeza y con los requisitos previstos en los artículos 214, párrafo quinto, fracción IV, y 247 del Código Electoral del Distrito Federal que resulten aplicables en lo conducente”.

39. Que, de manera adicional, resulta indispensable modificar las pantallas de las boletas virtuales, a fin de otorgar mayor espacio entre los emblemas de los partidos políticos participantes en el proceso electoral, a efecto de que puedan ser utilizadas las mascarillas Braille para el ejercicio del voto electrónico de los ciegos y débiles visuales.
40. Que durante el desarrollo de la quinta sesión ordinaria de la Comisión de Organización y Geografía Electoral, mediante el acuerdo 137/5ªOrd./09. fue conocido y remitido a este órgano superior de dirección, el “Proyecto de Acuerdo del Consejo General del Instituto Electoral del Distrito Federal por el que se modifica el programa informático (software electoral) que se utilizará en las urnas electrónicas el día de la jornada electoral de 2009.

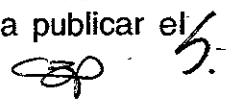
Por lo antes expuesto y con fundamento en los artículos 123; 124 y 127 del Estatuto de Gobierno del Distrito Federal; 2º, párrafo tercero; 3º, párrafo primero; 31, párrafo primero; 34, párrafo segundo; 86, fracción V; 89, párrafo primero; 95, fracciones XXII y XXIII; 96; 97, fracción III; 102, fracciones II y V; 116, fracción II; 213, párrafos primero, segundo y tercero; 214, párrafos primero, segundo y cuarto; 243, fracciones II y ; 246 y 248, párrafos primero, segundo, tercero y cuarto del Código Electoral del Distrito Federal; 59, fracciones V y IX del Reglamento Interior del Instituto Electoral del Distrito Federal, así como en los acuerdos ACU-425-09 y ACU-426-09 del Consejo General del Instituto Electoral del Distrito Federal, y de los acuerdos 103/4ª Ord/09 y 137/5ªOrd/09 de la Comisión de Organización y Geografía Electoral, el Consejo General del Instituto Electoral del Distrito Federal emite el siguiente:

ACUERDO

PRIMERO. Se aprueban las modificaciones al programa informático (software electoral) que se utilizará en las urnas electrónicas en la jornada electoral del cinco de julio de 2009, de conformidad con el documento que, como anexo, forma parte integral del presente Acuerdo, y con lo que establecen los considerandos 37, 38 y 39 del mismo.

SEGUNDO. Se instruye a la Secretaría Ejecutiva, para que, con apoyo de la Unidad Técnica de Servicios Informáticos, y demás áreas del Instituto involucradas, se lleven a cabo las acciones necesarias para la modificación de dicho software electoral en los términos del Acuerdo primero.

TERCERO. Se instruye a la Secretaría Ejecutiva para que, con el apoyo de las áreas involucradas del Instituto, se lleven a cabo las acciones necesarias para publicar el



software modificado, de manera permanente, en la página de Internet www.iedf.org.mx, concretamente en el micrositio denominado "Urna Electrónica".

CUARTO. El presente Acuerdo entrará en vigor al momento de su aprobación.

QUINTO. Publíquese el presente Acuerdo y sus anexos en los estrados del Instituto Electoral del Distrito Federal, en la Gaceta Oficial del Distrito Federal y en la página de Internet, www.iedf.org.mx

Así lo aprobaron por unanimidad de votos los CC. Consejeros Electorales integrantes del Consejo General del Instituto Electoral del Distrito Federal, en la sesión pública de fecha uno de junio de dos mil nueve, firmando al calce, la Consejera Presidenta y el Secretario del Consejo General del Instituto Electoral del Distrito Federal, con fundamento en los artículos 105, fracción VI y 110, fracción XIII del Código Electoral del Distrito Federal, doy fe.

La Consejera Presidenta



Mtra. Beatriz Claudia Zavala Pérez

El Secretario Ejecutivo



Lic. Sergio Jesús González Muñoz

**PANTALLAS DE OPERACIÓN DEL
 PROGRAMA INFORMÁTICO (SOFTWARE ELECTORAL)
 QUE SE UTILIZARÁ EN LAS URNAS ELECTRÓNICAS
 EL DÍA DE LA JORNADA ELECTORAL DE 2009.**

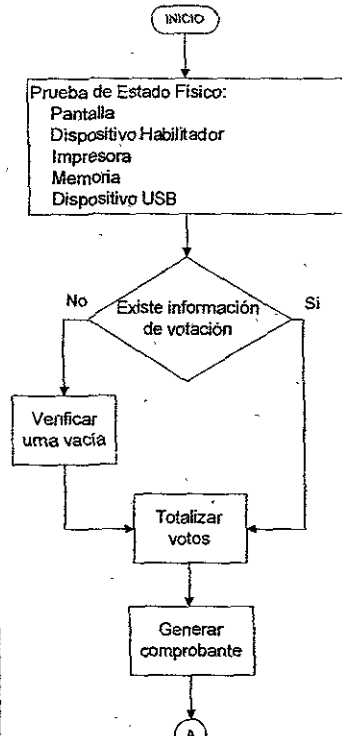
El programa informático (software electoral) en su operación, contempla básicamente tres procesos consecutivos:

1. Proceso de apertura de casilla,
2. Proceso de votación y
3. Proceso de cierre de casilla.

En el que se destacan las acciones descritas en cada uno de estos:

1. Proceso de apertura de casilla: realiza la prueba de estado físico de los componentes de la urna electrónica, verificación de información de votación, totalizador de votos generación de comprobante.
2. Proceso de votación: permite la votación en dos boletas virtuales predefinidas y almacena dicha votación.
3. Proceso de cierre de casilla: una vez presionado el botón derecho de la botonera en relieve, el software de urna electrónica solicita la contraseña para realizar el cierre de la votación, realiza la totalización de votos, codifica archivos de votos y bitácora, y genera el comprobante de escrutinio y cómputo de la votación.

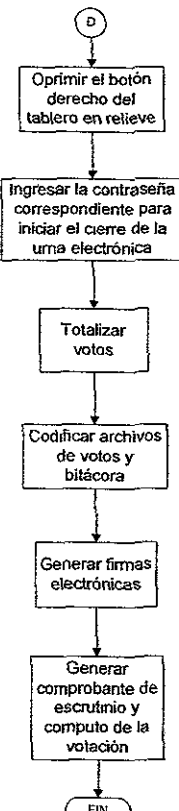
Tabla de diagrama de flujo de los procesos de operación del programa informático (software electoral) de urna electrónica:

No.	Descripción	Diagrama de flujo
1.	Este proceso tiene correspondencia desde la pantalla número 1 hasta la número 21.	PROCESO DE APERTURA DE CASILLA  <pre> graph TD INICIO([INICIO]) --> Prueba[Prueba de Estado Físico: Pantalla Dispositivo Habilitador Impresora Memoria Dispositivo USB] Prueba --> Decision{Existe información de votación} Decision -- No --> Verificar[Verificar urna vacía] Decision -- Si --> Totalizar[Totalizar votos] Verificar --> Totalizar Totalizar --> Generar[Generar comprobante] Generar --> A((A)) </pre>

**PANTALLAS DE OPERACIÓN DEL
 PROGRAMA INFORMÁTICO (SOFTWARE ELECTORAL)
 QUE SE UTILIZARÁ EN LAS URNAS ELECTRÓNICAS
 EL DÍA DE LA JORNADA ELECTORAL DE 2009.**

No.	Descripción	Diagrama de flujo
2.	Este proceso tiene correspondencia desde la pantalla número 22 hasta la número 30.	PROCESO DE VOTACIÓN <pre> graph TD A((A)) --> U1[Urna Deshabilitada] U1 --> H[Habilitar la Urna con el Dispositivo Habilitador] H --> V1[Votar en la primera boleta virtual] V1 --> C1{Confirma el voto} C1 -- No --> V1 C1 -- Si --> G1[Generar voto] G1 --> A1[Almacena voto en dispositivos] A1 --> B1((B)) B1 --> V2[Votar en la segunda boleta virtual] V2 --> C2{Confirma el voto} C2 -- No --> V2 C2 -- Si --> G2[Generar voto] G2 --> A2[Almacena voto en dispositivos] A2 --> C3((C)) C3 --> V1 A2 --> D((D)) </pre>

**PANTALLAS DE OPERACIÓN DEL
 PROGRAMA INFORMÁTICO (SOFTWARE ELECTORAL)
 QUE SE UTILIZARÁ EN LAS URNAS ELECTRÓNICAS
 EL DÍA DE LA JORNADA ELECTORAL DE 2009.**

No.	Descripción	Diagrama de flujo
3.	Este proceso tiene correspondencia desde la pantalla número 31 hasta la número 34.	PROCESO DE CIERRE DE CASILLA  <pre> graph TD D((D)) --> A[Oprimir el botón derecho del tablero en relieve] A --> B[Ingresar la contraseña correspondiente para iniciar el cierre de la urna electrónica] B --> C[Totalizar votos] C --> D1[Codificar archivos de votos y bitácora] D1 --> E[Generar firmas electrónicas] E --> F[Generar comprobante de escrutinio y computo de la votación] F --> FIN((FIN)) </pre>

Cap

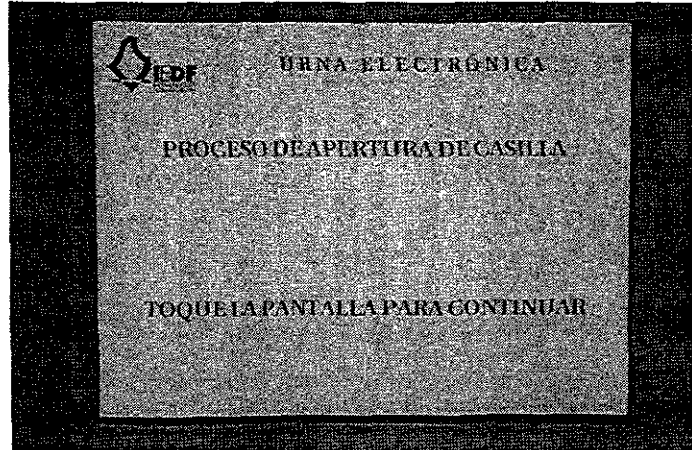
h.

**PANTALLAS DE OPERACIÓN DEL
PROGRAMA INFORMÁTICO (SOFTWARE ELECTORAL)
QUE SE UTILIZARÁ EN LAS URNAS ELECTRÓNICAS
EL DÍA DE LA JORNADA ELECTORAL DE 2009.**

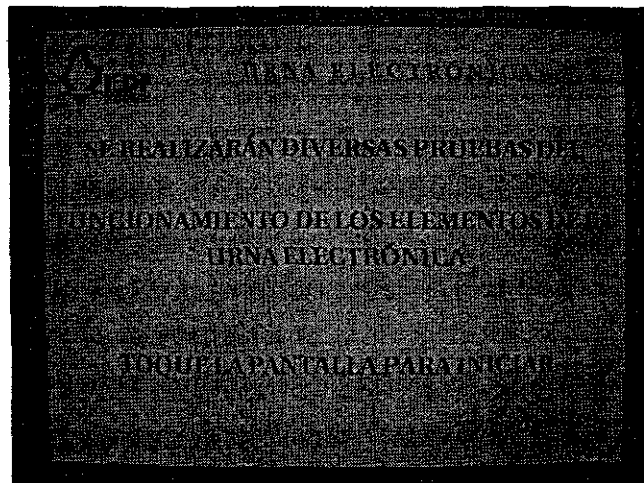
Tabla de pantallas de operación del programa informático (software electoral) de urna electrónica.

NO.	DESCRIPCIÓN	PANTALLAS
-----	-------------	-----------

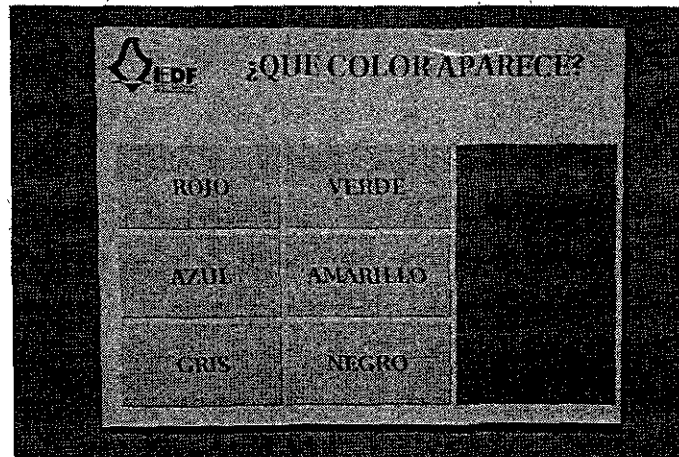
1. Pantalla de apertura de casilla.



2. Pantalla de espera, para preparación de componentes.

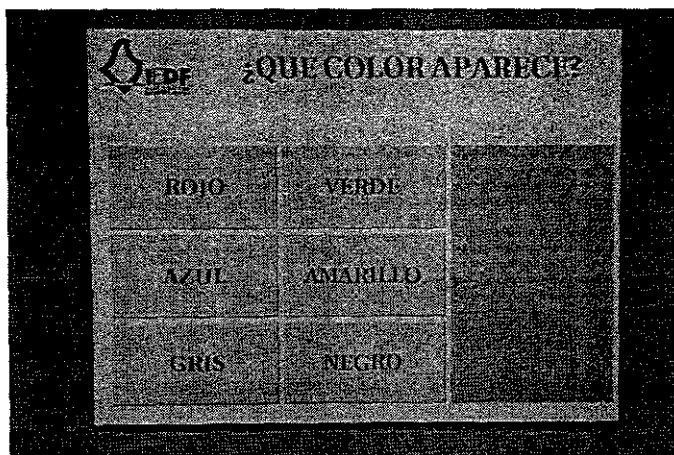


3. Pantalla de pruebas de colores: rojo (1 de 6).

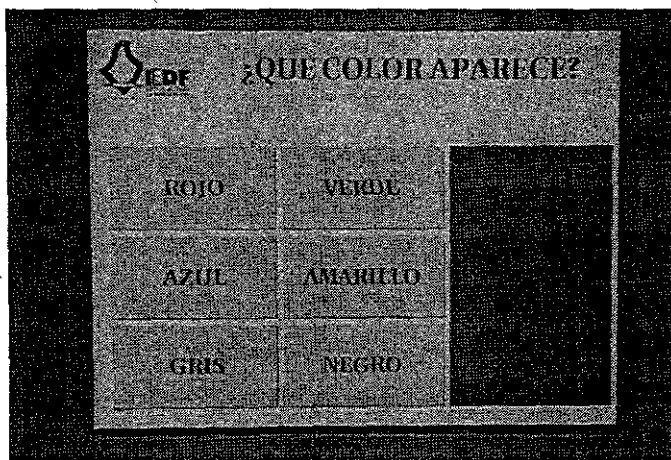


NO.	DESCRIPCIÓN	PANTALLAS
-----	-------------	-----------

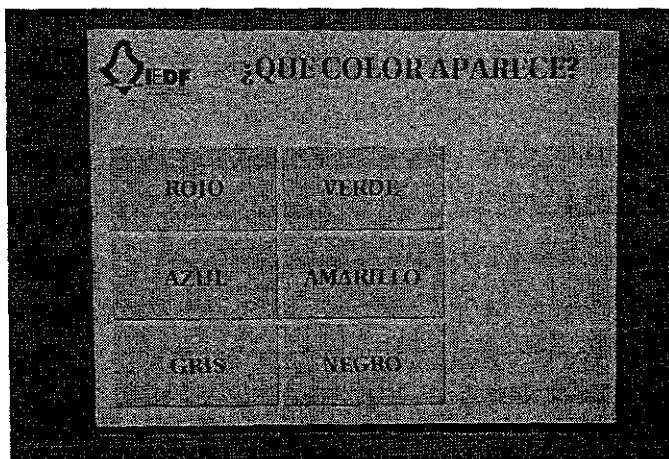
4. Pantalla de pruebas de colores: verde (2 de 6).



5. Pantalla de pruebas de colores: azul (3 de 6).



6. Pantalla de pruebas de colores: amarillo (4 de 6).



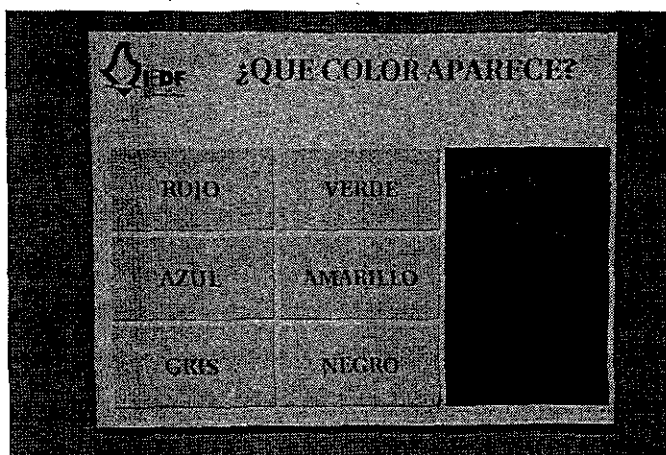
Cap
4.

NO	DESCRIPCIÓN	PANTALLAS
----	-------------	-----------

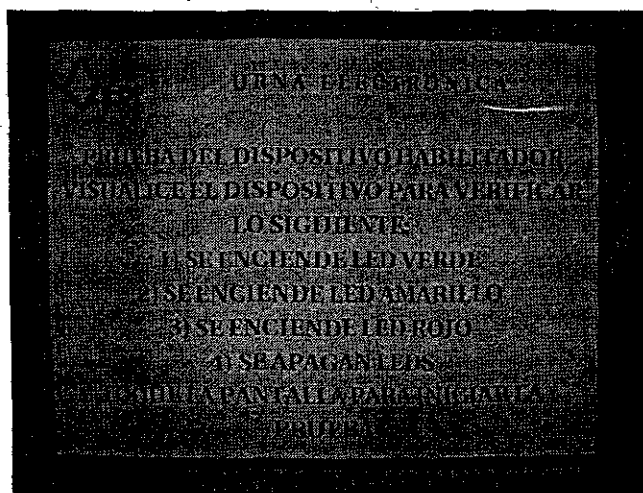
7. Pantalla de pruebas de colores: gris (5 de 6).



8. Pantalla de pruebas de colores: negro (6 de 6).

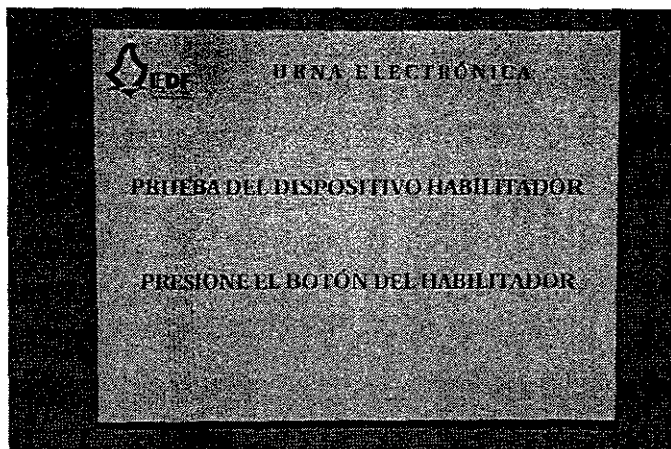


9. Pantalla del dispositivo habilitador: (luces)

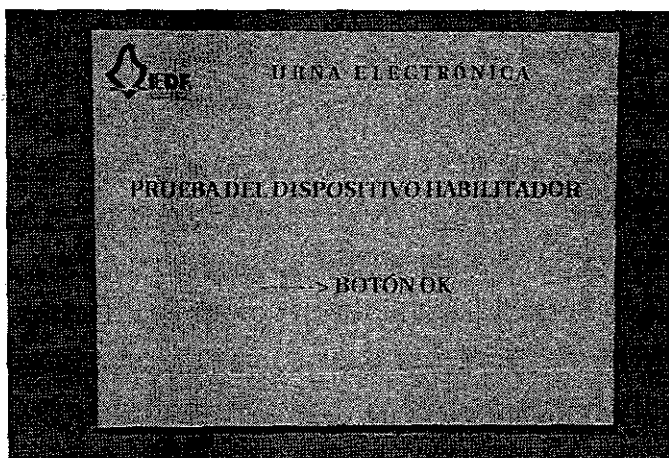


NO.	DESCRIPCIÓN	PANTALLAS
-----	-------------	-----------

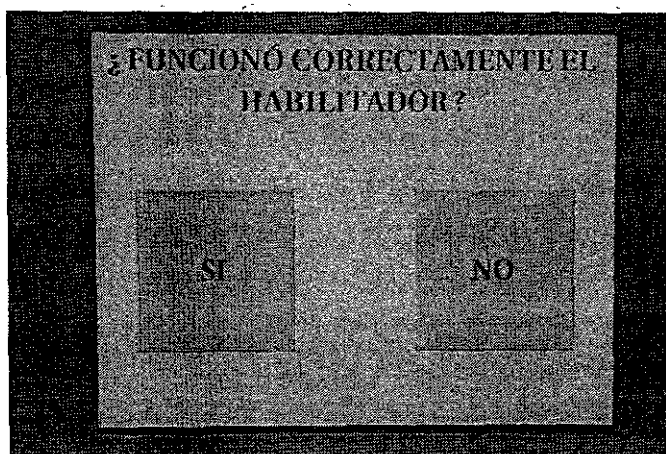
10. Pantalla del dispositivo habilitador: (botón)



11. Pantalla del dispositivo habilitador: (botón)



12. Pantalla de confirmación de funcionamiento del dispositivo habilitador.



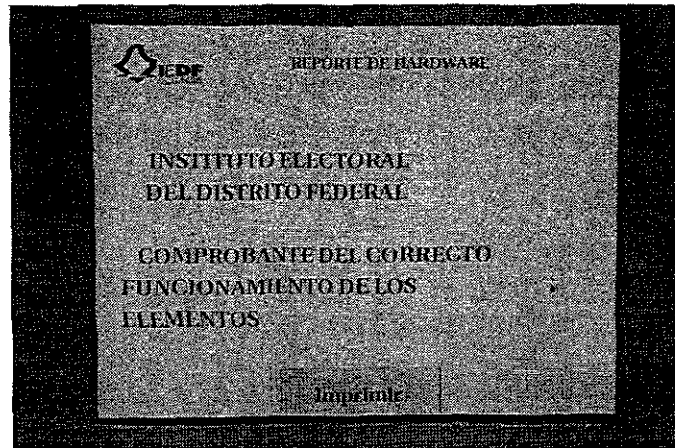
6.

cap

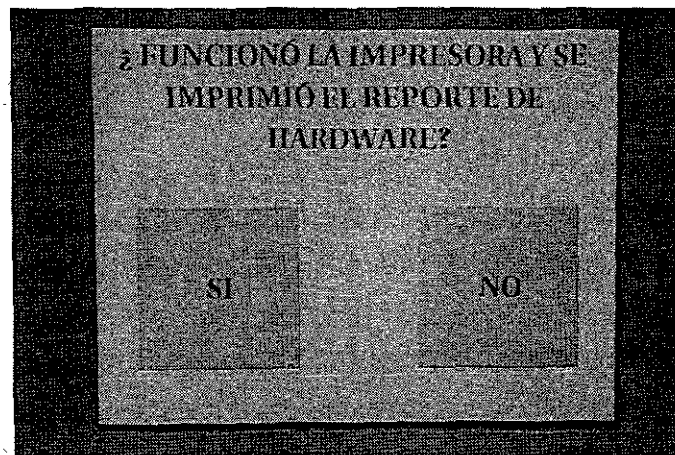
**PANTALLAS DE OPERACIÓN DEL
 PROGRAMA INFORMÁTICO (SOFTWARE ELECTORAL)
 QUE SE UTILIZARÁ EN LAS URNAS ELECTRÓNICAS
 EL DÍA DE LA JORNADA ELECTORAL DE 2009.**

NO	DESCRIPCIÓN	PANTALLAS
----	-------------	-----------

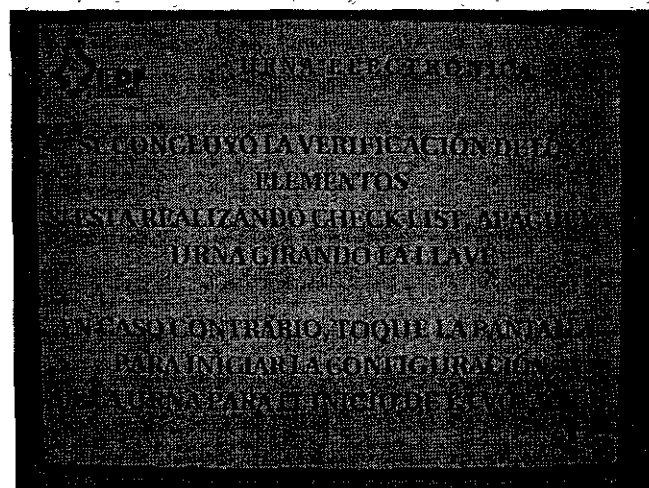
13. Pantalla de funcionamiento de los elementos.



14. Pantalla de funcionamiento de impresora.



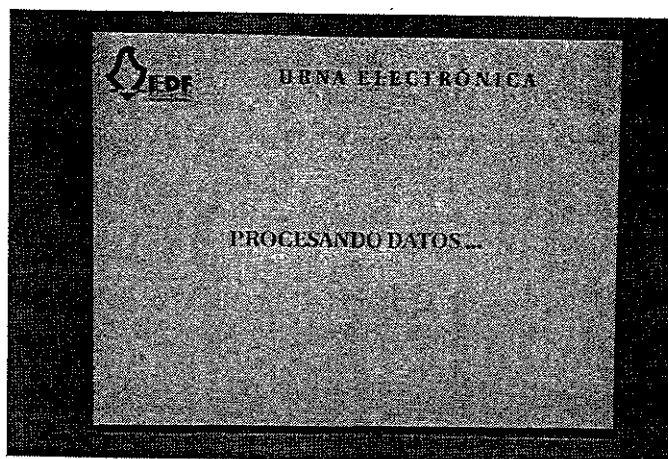
15. Pantalla de conclusión de verificación de componentes.



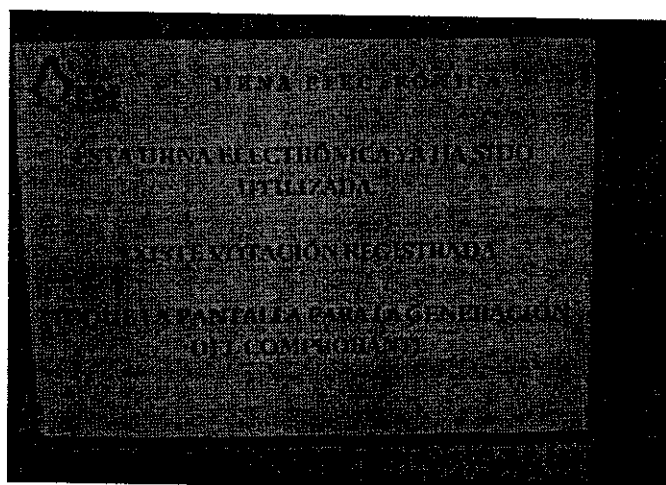
**PANTALLAS DE OPERACIÓN DEL
 PROGRAMA INFORMÁTICO (SOFTWARE ELECTORAL)
 QUE SE UTILIZARÁ EN LAS URNAS ELECTRÓNICAS
 EL DÍA DE LA JORNADA ELECTORAL DE 2009.**

NO.	DESCRIPCIÓN	PANTALLAS
-----	-------------	-----------

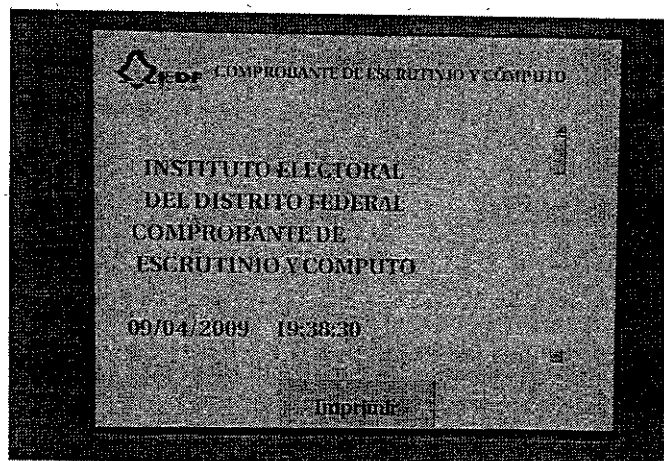
16. Pantalla de espera, para procesar datos.



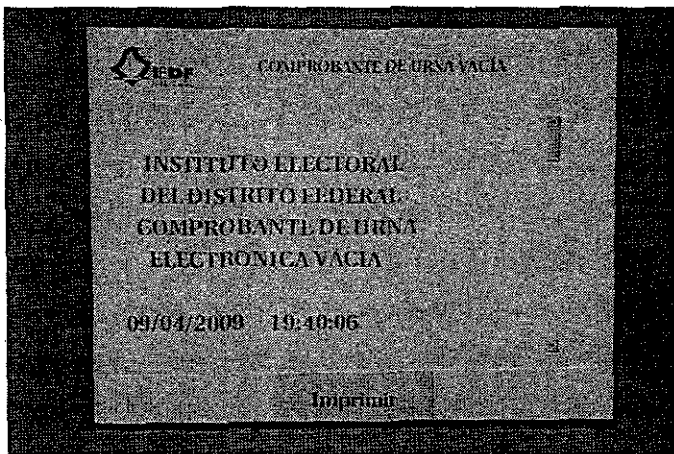
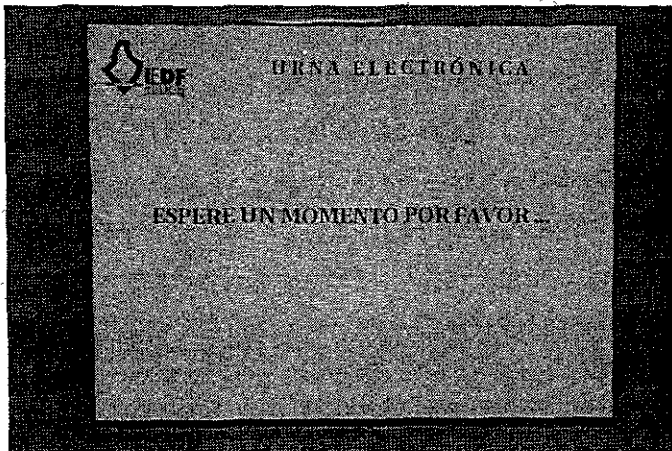
17. Pantalla de existencia de votación registrada.



18. Pantalla de comprobante de escrutinio y cómputo.



**PANTALLAS DE OPERACIÓN DEL
 PROGRAMA INFORMÁTICO (SOFTWARE ELECTORAL)
 QUE SE UTILIZARÁ EN LAS URNAS ELECTRÓNICAS
 EL DÍA DE LA JORNADA ELECTORAL DE 2009.**

NO.	DESCRIPCIÓN	PANTALLAS
19.	Pantalla para la impresión del comprobante de urna vacía.	
20.	Pantalla del comprobante de urna vacía.	
21.	Pantalla de espera, para iniciar el proceso de votación.	

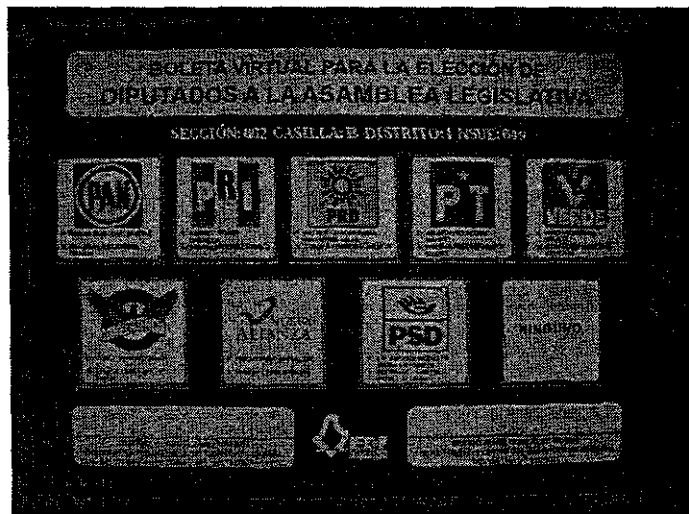
5.

NO.	DESCRIPCIÓN	PANTALLAS
-----	-------------	-----------

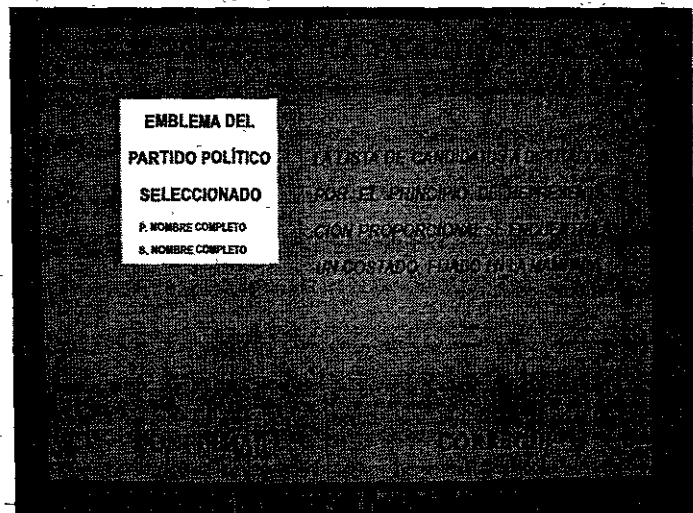
- 22 Pantalla de aviso de urna deshabilitada.



- 23 Pantalla de boleta virtual de Diputados a la Asamblea Legislativa.

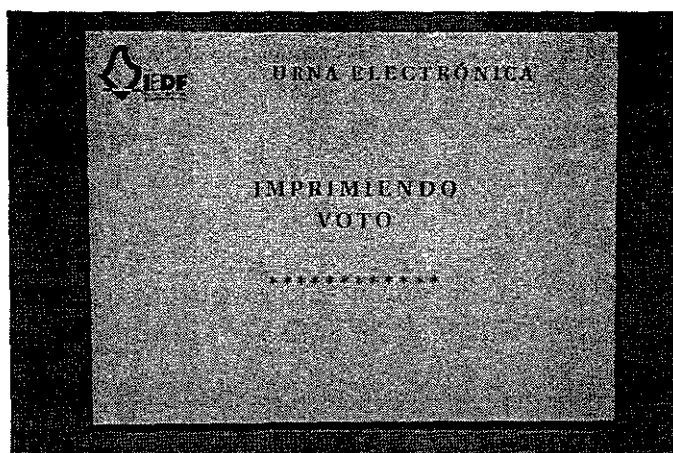


- 24 Pantalla de confirmación de voto.

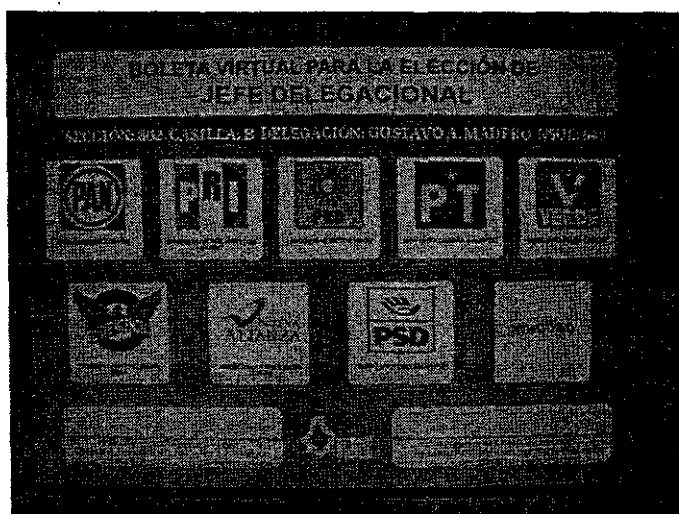


NO.	DESCRIPCIÓN	PANTALLAS
-----	-------------	-----------

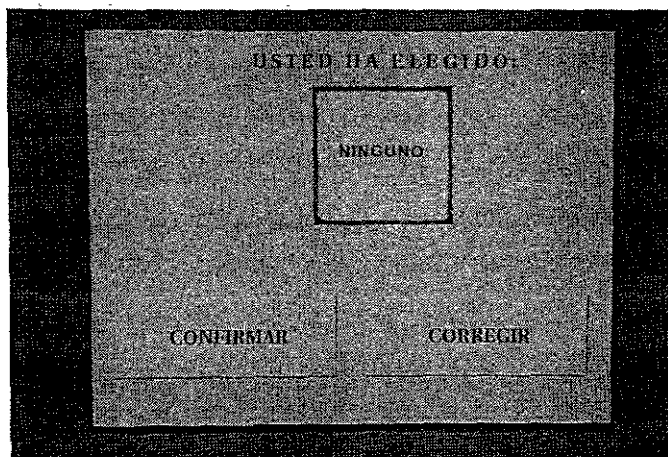
- 25 Pantalla de aviso de impresión de voto.



- 26 Pantalla de la boleta virtual de Jefe Delegacional.

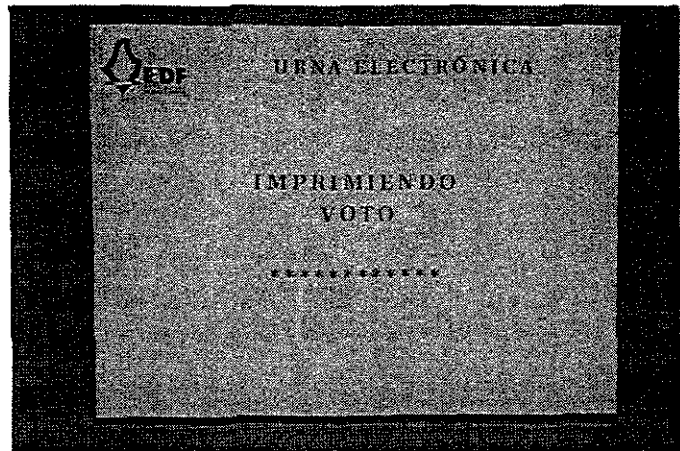


- 27 Pantalla de confirmación de voto.

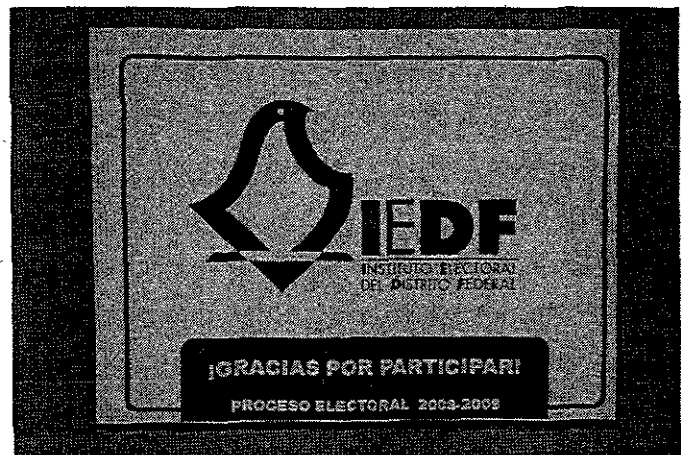


NO.	DESCRIPCIÓN	PANTALLAS
-----	-------------	-----------

- 28 Pantalla de aviso de impresión de voto.



- 29 Pantalla de aviso "Gracias por participar".



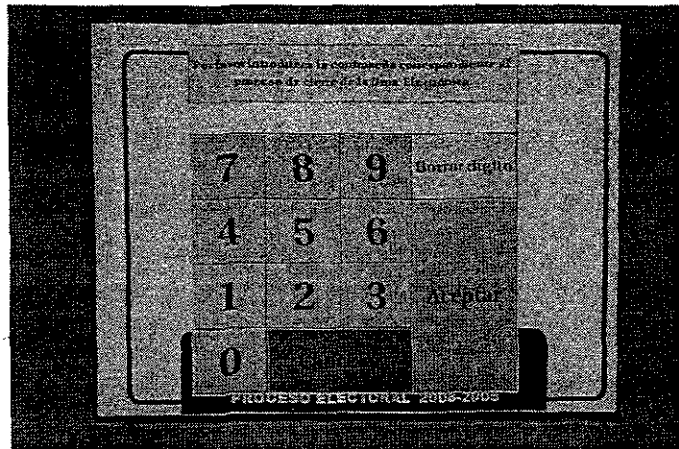
- 30 Pantalla de aviso "urna deshabilitada".



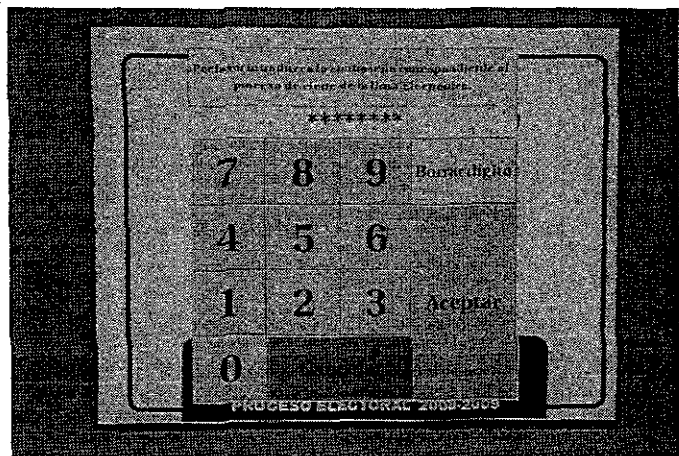
**PANTALLAS DE OPERACIÓN DEL
 PROGRAMA INFORMÁTICO (SOFTWARE ELECTORAL)
 QUE SE UTILIZARÁ EN LAS URNAS ELECTRÓNICAS
 EL DÍA DE LA JORNADA ELECTORAL DE 2009.**

NO.	DESCRIPCIÓN	PANTALLAS
-----	-------------	-----------

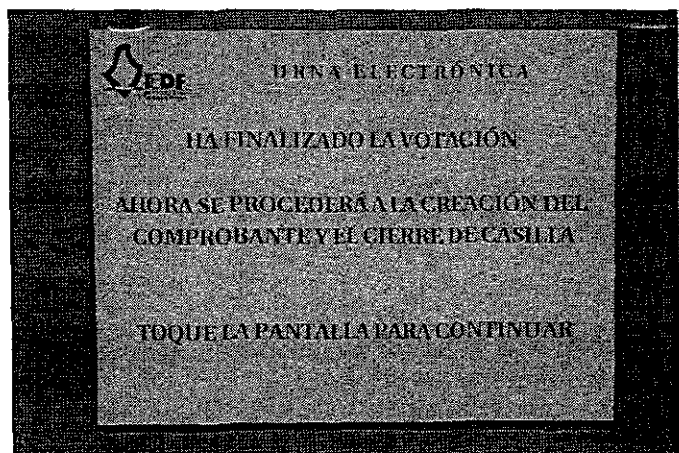
- 31 Pantalla de recepción de contraseña para el proceso de cierre.



- 32 Pantalla con contraseña digitada para el proceso de cierre.



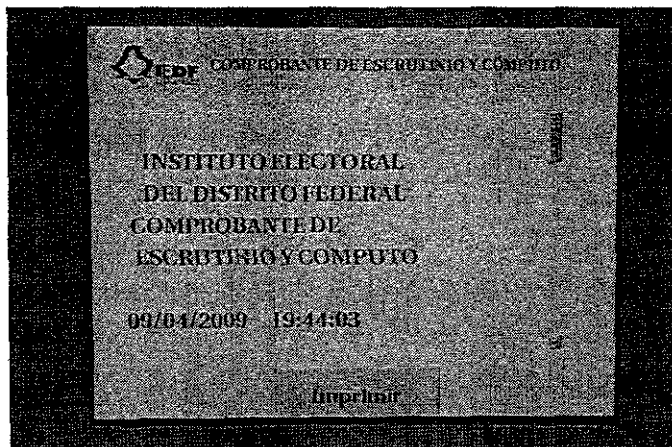
- 33 Pantalla para generar el comprobante de escrutinio y cómputo de la votación.



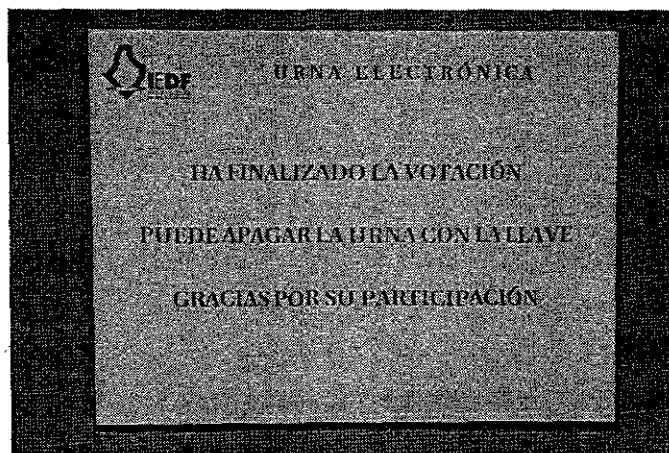
**PANTALLAS DE OPERACIÓN DEL
 PROGRAMA INFORMÁTICO (SOFTWARE ELECTORAL)
 QUE SE UTILIZARÁ EN LAS URNAS ELECTRÓNICAS
 EL DÍA DE LA JORNADA ELECTORAL DE 2009.**

NO.	DESCRIPCIÓN	PANTALLAS
-----	-------------	-----------

- 34 Pantalla de comprobante de escrutinio y cómputo de la votación.



- 35 Pantalla de aviso "finalización de votación"




 4.

ARCHIVO CFRMCONFIGURAURNA.CPP

```
/*
 * ** Nombre del Programa : Fuentes/CfrmConfiguraUrna.cpp
 * ** Descripción : Archivo fuente que implementa las funciones encargadas de
 leer archivos binarios genéricos correspondientes a la casilla, además de leer
 los datos de seleccionados.
 */
```

Clase que configura la Urna Electrónica de acuerdo a la sección y tipo de casilla que se trate. Los datos se obtiene del archivo de configuración contenido en el dispositivo USB con el que se abrió la Urna.

```
#include <CfrmConfiguraUrna.h>
#define __CFRMCONFIGURAURNA_MSG__
#include <errno.h>
#include <common.h>
#include <stdlib.h>
#include <string.h>
#include <SysLogExa.h>

CfrmConfiguraUrna::CfrmConfiguraUrna( int pintPID, stcPaths *pstcPathsPtrDat,
int pintSeccion, const char *pchrPtrTipoCasilla )/*{{{*/
{
    : intPID( pintPID ), intError( -1 ), intSeccion( pintSeccion ),
chrPtrError( NULL ), chrPtrTipoCasilla( pchrPtrTipoCasilla ), stcPathsPtrDat(
pstcPathsPtrDat )
{
    if ( !pintSeccion )
    {
        UrnaLog( intPID, "La seccion no es valida", stcPathsPtrDat );
        chrPtrError = gchrArrPtrConfUErrors[intError = 0];
        return;
    }
    if ( !pchrPtrTipoCasilla )
    {
        UrnaLog( intPID, "El tipo de casilla no es valido", stcPathsPtrDat
);
        chrPtrError = gchrArrPtrConfUErrors[intError = 1];
        return;
    }
    readFile();
    getConfiguracionCasilla();
    orderData();
}}{}}*/

CfrmConfiguraUrna::~CfrmConfiguraUrna()/*{{{*/
{
}}{}}*/

const char *CfrmConfiguraUrna::getError()/*{{{*/
{
    return chrPtrError;
}}{}}*/
```

```
int CfrmConfiguraUrna::getNumParticipantes()/*{{{*/Esta función obtiene el número de
Participantes (Partidos Políticos) asociados a esta Urna
{
    return intArrRegistros[3];
}}}}*/

int CfrmConfiguraUrna::getNumCandidatosJD()/*{{{*/ Esta función obtiene el número
de candidatos a Jefe Delegacional asociados a esta Urna
{
    return intArrRegistros[5] + 1;
}}}}*/

int CfrmConfiguraUrna::getNumCandidatosMR()/*{{{*/ Esta función obtiene el número
de candidatos a Diputados de Mayoría Relativa asociados a esta Urna
{
    return intArrRegistros[6] + 1;
}}}}*/

int CfrmConfiguraUrna::getNumCandidatosRP()/*{{{*/ Esta función obtiene el
número de candidatos a Diputados de Representación Proporcional asociados a
esta Urna
{
    return intArrRegistros[7] + 1;
}}}}*/

stcCasilla CfrmConfiguraUrna::getCasilla()/*{{{*/ Esta función regresa la sección y
tipo de casilla asociada a esta Urna
{
    return *stcCasPtrDat;
}}}}*/

stcDistrito CfrmConfiguraUrna::getDistrito()/*{{{*/Esta función regresa los datos del
distrito asociado a esta Urna
{
    return *stcDisPtrDat;
}}}}*/

stcDelegacion CfrmConfiguraUrna::getDelegacion()/*{{{*/Esta función regresa los datos
de la delegación asociada a esta Urna
{
    return *stcDelPtrDat;
}}}}*/

stcCandidato_JD *CfrmConfiguraUrna::getCandidatosJD()/*{{{*/ Esta función regresa
los candidatos a Jefe Delegacional asociados a esta esta Urna
{
    return stcCanJDPtrDat;
}}}}*/

stcCandidato_MR *CfrmConfiguraUrna::getCandidatosMR()/*{{{*/ Esta función regresa los
candidatos a Diputado de Mayoría Relativa asociados a esta esta Urna
{
    return stcCanMRPtrDat;
}}}}*/

stcCandidato_RP *CfrmConfiguraUrna::getCandidatosRP()/*{{{*/ Esta función regresa
los candidatos a Diputado de Representación Proporcional asociados a esta esta Urna
{
    return stcCanRPPtrDat;
```

```
}}/*}}*/
```

```
stcParticipante *CfrmConfiguraUrna::getParticipantes()/*{{{*/ Esta función regresa  
los participantes (partidos Políticos) asociados a esta Urna
```

```
{
    return stcParPtrDat;
}}/*}}*/
```

```
//////////////////////////////////// PRIVATE //////////////////////////////////////
```

```
void CfrmConfiguraUrna::readFile()/*{{{*/Se utiliza para leer el archivo binario
```

```
{
    void *lvidPtrData[8];
    size_t lsztArrSize[8];

    if ( readBinaryFile( stcPathsPtrDat->chrArrBinary_File, lvidPtrData,
lsztArrSize ) != -1 )
    {
        UrnaLog( intPID, "Lectura del archivo con los datos genericos  
realizada satisfactoriamente", stcPathsPtrDat );
        stcCasPtrDat = (stcCasilla *)lvidPtrData[3];
        intArrRegistros[0] = lsztArrSize[3];
        stcDisPtrDat = (stcDistrito *)lvidPtrData[1];
        intArrRegistros[1] = lsztArrSize[1];
        stcDelPtrDat = (stcDelegacion *)lvidPtrData[2];
        intArrRegistros[2] = lsztArrSize[2];
        stcParPtrDat = (stcParticipante *)lvidPtrData[0];
        intArrRegistros[3] = lsztArrSize[0];
        stcCanJDPtrDat = (stcCandidato_JD *)lvidPtrData[5];
        intArrRegistros[5] = lsztArrSize[5];
        stcCanMRPtrDat = (stcCandidato_MR *)lvidPtrData[6];
        intArrRegistros[6] = lsztArrSize[6];
        stcCanRPPtrDat = (stcCandidato_RP *)lvidPtrData[7];
        intArrRegistros[7] = lsztArrSize[7];
    }
    else
    {
        UrnaLog( intPID, "Error en la lectura del archivo con los datos  
genericos", stcPathsPtrDat );
        chrPtrError = gchrArrPtrConfUErrors[intError = 2];
        return;
    }
}}/*}}*/
```

```
void CfrmConfiguraUrna::orderData()/*{{{*/Para ordenar los datos según su ID
```

```
{
    UrnaLog( intPID, "Se procedera al ordenamiento de los datos  
seleccionados", stcPathsPtrDat );
    orderCandidatoJD();
    orderCandidatoMR();
    orderCandidatoRP();
    orderParticipantes();
}}/*}}*/
```

Cap

6.

void CfrmConfiguraUrna::orderCandidatoJD()/*{{{*/Ordena los candidatos a Jefe Delegacional de acuerdo al campo prelación

```
{
    stcCandidato_JD lstcCanJDPtrDat;
    int lintJ, lintK;

    for ( lintJ = 0; lintJ < intArrRegistros[5] - 1; lintJ++ )
        for ( lintK = lintJ + 1; lintK < intArrRegistros[5]; lintK++ )
            if ( (stcCanJDPtrDat + lintJ)->int_Prelacion > (stcCanJDPtrDat
+ lintK)->int_Prelacion )
                {
                    lstcCanJDPtrDat = *(stcCanJDPtrDat + lintJ);
                    *(stcCanJDPtrDat + lintJ) = *(stcCanJDPtrDat + lintK);
                    *(stcCanJDPtrDat + lintK) = lstcCanJDPtrDat;
                }

    UrnaLog( intPID, "Datos de Candidato a Jefe de delegacional ordenados
satisfactoriamente", stcPathsPtrDat );
}/*}}}]*/
```

void CfrmConfiguraUrna::orderCandidatoMR()/*{{{*/Esta función ordena los candidatos a Diputado de Mayoría Relativa de acuerdo al campo prelación

```
{
    stcCandidato_MR lstcCanMRPtrDat;
    int lintJ, lintK;

    for ( lintJ = 0; lintJ < intArrRegistros[6] - 1; lintJ++ )
        for ( lintK = lintJ + 1; lintK < intArrRegistros[6]; lintK++ )
            if ( (stcCanMRPtrDat + lintJ)->int_Prelacion > (stcCanMRPtrDat
+ lintK)->int_Prelacion )
                {
                    lstcCanMRPtrDat = *(stcCanMRPtrDat + lintJ);
                    *(stcCanMRPtrDat + lintJ) = *(stcCanMRPtrDat + lintK);
                    *(stcCanMRPtrDat + lintK) = lstcCanMRPtrDat;
                }

    UrnaLog( intPID, "Datos de Candidato a Diputado por el principio de
mayoría relativa ordenados satisfactoriamente", stcPathsPtrDat );
}/*}}}]*/
```

void CfrmConfiguraUrna::orderCandidatoRP()/*{{{*/Esta función ordena los candidatos a Diputado de Representación Proporcional de acuerdo al campo prelación

```
{
    stcCandidato_RP lstcCanRPPtrDat;
    int lintJ, -lintK;

    for ( lintJ = 0; lintJ < intArrRegistros[7] - 1; lintJ++ )
        for ( lintK = lintJ + 1; lintK < intArrRegistros[7]; lintK++ )
            if ( (stcCanRPPtrDat + lintJ)->int_Prelacion > (stcCanRPPtrDat
+ lintK)->int_Prelacion )
                {
                    lstcCanRPPtrDat = *(stcCanRPPtrDat + lintJ);
                    *(stcCanRPPtrDat + lintJ) = *(stcCanRPPtrDat + lintK);
                    *(stcCanRPPtrDat + lintK) = lstcCanRPPtrDat;
                }

    UrnaLog( intPID, "Datos de Candidato a Diputado por el principio de
representacion proporcional ordenados satisfactoriamente", stcPathsPtrDat );
}
```

```
}}/}}}}*/
```

```
void CfrmConfiguraUrna::orderParticipantes()/*{{{*/Esta función ordena los  
Participantes (Partidos Políticos)
```

```
{
    stcParticipante lstcParPtrDat;
    int lintJ, lintK;

    for ( lintJ = 0; lintJ < intArrRegistros[3] - 1; lintJ++ )
        for ( lintK = lintJ + 1; lintK < intArrRegistros[3]; lintK++ )
            if ( (stcParPtrDat + lintJ)->int_Id_Partido > (stcParPtrDat +
lintK)->int_Id_Partido )
            {
                lstcParPtrDat = *(stcParPtrDat + lintJ);
                *(stcParPtrDat + lintJ) = *(stcParPtrDat + lintK);
                *(stcParPtrDat + lintK) = lstcParPtrDat;
            }

    UrnaLog( intPID, "Datos de partidos politicos(Participantes) ordenados  
satisfactoriamente", stcPathsPtrDat );
}}/}}}}*/
```

```
void CfrmConfiguraUrna::getConfiguracionCasilla()/*{{{*/Esta función es utilizada  
internamente para llamar a las funciones para obtener datos de casilla, y candidatos
```

```
{
    int lintX = 0;
    stcCasilla lstcCasDat = { 0, 0, 0, 0, "", 0 };

    if ( intError != -1 )
        return;

    UrnaLog( intPID, "Se procedera a la obtencion de los datos de la casilla",
stcPathsPtrDat );

    for ( lintX = 0; lintX < intArrRegistros[0]; lintX++ )
        if ( (stcCasPtrDat + lintX)->int_Id_Seccion == intSeccion &&
(!strcmp( (stcCasPtrDat + lintX)->chr_Id_Casilla, chrPtrTipoCasilla, 2 ) ||
!strcmp( (stcCasPtrDat + lintX)->chr_Id_Casilla, chrPtrTipoCasilla, 1 ) ) )
        {
            lstcCasDat = *(stcCasPtrDat + lintX);
            free ( stcCasPtrDat );
            if ( getMemory( sizeof( stcCasilla ), 1, (void
**)&stcCasPtrDat ) )
            {
                chrPtrError = gchrArrPtrConfUErrors[intError = 3];
                UrnaLog( intPID, "Error al generar memoria para  
almacenar los datos de la casilla", stcPathsPtrDat );
                return;
            }
            *stcCasPtrDat = lstcCasDat;
            intArrRegistros[0] = 1;
            break;
        }

    if ( lintX == intArrRegistros[0] && lintX != 1 )
    {
        UrnaLog( intPID, "No se encontro la casilla seleccionada.",
stcPathsPtrDat );
        chrPtrError = gchrArrPtrConfUErrors[intError = 4];
        return;
    }

    UrnaLog( intPID, "Casilla seleccionada satisfactoriamente", stcPathsPtrDat
);
    if ( findDistrito( stcCasPtrDat->int_Id_Distrito ) )
```




```

{
    UrnaLog( intPID, "Error al buscar el Distrito", stcPathsPtrDat );
    chrPtrError = gchrArrPtrConfUErrors[intError = 4];
    return;
}
UrnaLog( intPID, "Distrito seleccionado satisfactoriamente",
stcPathsPtrDat );
if ( findDelegacion( stcCasPtrDat->int_Id_Delegacion ) )
{
    UrnaLog( intPID, "Error al buscar la Delegacion", stcPathsPtrDat );
    chrPtrError = gchrArrPtrConfUErrors[intError = 5];
    return;
}
UrnaLog( intPID, "Delegacion seleccionada satisfactoriamente",
stcPathsPtrDat );
if ( findCand_Delegado( stcCasPtrDat->int_Id_Delegacion ) )
{
    UrnaLog( intPID, "Error al buscar los Candidatos a Jefe
Delegacional", stcPathsPtrDat );
    chrPtrError = gchrArrPtrConfUErrors[intError = 6];
    return;
}
UrnaLog( intPID, "Candidatos a Jefe delegacional seleccionados
satisfactoriamente", stcPathsPtrDat );
if ( findCand_Diputado_MR( stcCasPtrDat->int_Id_Distrito ) )
{
    UrnaLog( intPID, "Error al buscar los candidatos a diputados por el
principio de mayoria relativa", stcPathsPtrDat );
    chrPtrError = gchrArrPtrConfUErrors[intError = 7];
    return;
}
UrnaLog( intPID, "Candidatos a diputados por el principio de mayoria
relativa seleccionados satisfactoriamente", stcPathsPtrDat );
}/*}}}*/

bool CfrmConfiguraUrna::findDistrito( int pintId_Distrito )/*{{{*/ Esta función
obtiene el distrito en que se encuentra la Urna
{
    stcDistrito *lstcDisPtrDat;

    for ( int lintX = 0; lintX < intArrRegistros[2]; lintX++ )
        if ( (stcDisPtrDat + lintX)->int_Id_Distrito == pintId_Distrito )
        {
            if ( getMemory( sizeof( stcDistrito ), 1, (void
**) &lstcDisPtrDat ) )
                return true;
            *lstcDisPtrDat = *(stcDisPtrDat + lintX);
            free( stcDisPtrDat );
            stcDisPtrDat = lstcDisPtrDat;
            intArrRegistros[2] = 1;
            break;
        }
    return false;
}/*}}}*/

bool CfrmConfiguraUrna::findDelegacion( int pintId_Delegacion )/*{{{*/ Esta
función obtiene la delegación en que se encuentra la Urna
{
    stcDelegacion *lstcDelPtrDat;

    for ( int lintX = 0; lintX < intArrRegistros[1]; lintX++ )

```

```

    if ( (stcDelPtrDat + lintX)->int_Id_Delegacion == pintId_Delegacion
)
    {
        if ( getMemory( sizeof( stcDelegacion ), 1, (void
**) &lstcDelPtrDat ) )
            return true;
        *lstcDelPtrDat = *(stcDelPtrDat + lintX);
        free( stcDelPtrDat );
        stcDelPtrDat = lstcDelPtrDat;
        intArrRegistros[1] = 1;
        break;
    }
    return false;
}/*}}*/

```

bool CfrmConfiguraUrna::findCand_Delegado(int pintId_Delegacion)/*{{{*/ Esta función obtiene los candidatos para la elección a Jefe Delegacional

```

{
    int lintReg = 0;
    stcCandidato_JD *lstcCanJDPtrDat;

    for ( int lintX = 0; lintX < intArrRegistros[5]; lintX++ )
        if ( (stcCanJDPtrDat + lintX)->int_Id_Delegacion ==
pintId_Delegacion )
            lintReg++;
    if ( getMemory( sizeof( stcCandidato_JD ), lintReg, (void
**) &lstcCanJDPtrDat ) )
        return true;

    for ( int lintX = 0, lintReg = 0; lintX < intArrRegistros[5]; lintX++ )
        if ( (stcCanJDPtrDat + lintX)->int_Id_Delegacion ==
pintId_Delegacion )
            *(lstcCanJDPtrDat + lintReg++) = *(stcCanJDPtrDat + lintX);
    free ( stcCanJDPtrDat );
    stcCanJDPtrDat = lstcCanJDPtrDat;
    intArrRegistros[5] = lintReg;
    return false;
}/*}}*/

```

bool CfrmConfiguraUrna::findCand_Diputado MR(int pintId_Distrito)/*{{{*/ Esta función obtiene los candidatos para la elección a Diputado de Mayoría Relativa

```

{
    int lintReg = 0;
    stcCandidato_MR *lstcCanMRPtrDat;

    for ( int lintX = 0; lintX < intArrRegistros[6]; lintX++ )
        if ( (stcCanMRPtrDat + lintX)->int_Id_Distrito == pintId_Distrito )
            lintReg++;
    if ( getMemory( sizeof( stcCandidato_MR ), lintReg, (void
**) &lstcCanMRPtrDat ) )
        return true;

    for ( int lintX = 0, lintReg = 0; lintX < intArrRegistros[6]; lintX++ )
        if ( (stcCanMRPtrDat + lintX)->int_Id_Distrito == pintId_Distrito )
            *(lstcCanMRPtrDat + lintReg++) = *(stcCanMRPtrDat + lintX);
    free ( stcCanMRPtrDat );
    stcCanMRPtrDat = lstcCanMRPtrDat;
    intArrRegistros[6] = lintReg;
    return false;
}/*}}*/

```

ARCHIVO CFRMJORNADALECTORAL.CPP

```
/*  
** Nombre del Programa : Fuentes/CfrmJornadaElectoral.cpp  
** Descripción : Archivo fuente que implementa las funciones para administrar  
el registro de votos de la sesión electoral, desde la validación y selección de  
boletas a utilizar hasta el registro de los votos.  
*/
```

Esta clase implementa la sesión del voto electrónico y comprende todos los tipos de elecciones para los cuales ha sido configurada la Urna.

```
#include <CfrmJornadaElectoral.h>  
#include <CfrmBoleta.h>  
#include <CfrmMensajes.h>  
#include <CfrmSeleccion.h>  
#include <CfrmDespedida.h>  
#include <CfrmEsperaHabilitacion.h>  
#include <qevent.h>  
#include <qpoint.h>  
#include <qtimer.h>  
#include <NasSound.h>  
#include <qapplication.h>  
#include <stdlib.h>  
#include <string.h>  
#include <common.h>  
#include <BarCode.h>  
#include <omarssio.h>  
#include <SysLogExa.h>  
#include <SerialPrinter.h>
```

```
// Variables para tomar el tiempo que se lleva la respuesta a cada pregunta  
extern bool Tiempo;  
extern time_t T_Inicio;  
extern time_t T_Fin;  
//extern struct stcRespuestas stcResp;  
struct stcRespuestas stcResp;  
struct stcTotales stcTotal[25];
```

```
//variables externas con datos  
extern int int_NumDistrito;  
extern int int_ListaNom;  
extern char chrrArr_Serie_Urna[40];  
extern int int_NumSeccion;  
extern char chr_Tipo_Casilla[2];  
extern struct stcVotos stcMatrizVotos[766][2];  
int contador;
```

```
CfrmJornadaElectoral::CfrmJornadaElectoral( int pintPID, int pintWidth, int  
pintHeight, stcPaths *pstcPathsPtrDat, QWidget *pwdgParent, const char  
*pchrPtrName, WFlags pwflFlags )/*{{{*/  
    : QWidgetStack( pwdgParent, pchrPtrName, pwflFlags ),  
      stcPathsPtrDat( pstcPathsPtrDat ), stcBarPtrACodeA( NULL ), bolEspecial(  
FALSE ), bolActivacion( FALSE ), bolTimerFinal( FALSE ), bolSesionActiva( FALSE  
) , wflFlags( pwflFlags ), intPID( pintPID ), intStep( 0 ), intStepE( 0 ),  
intVotos( 0 ), intWidth( pintWidth ), intHeight( pintHeight )
```

```
{
    setMinimumSize( intWidth, intHeight );
    setMaximumSize( intWidth, intHeight );
}/*}}*/

CfrmJornadaElectoral::~CfrmJornadaElectoral()/*{{{*/
{
}/*}}*/

void CfrmJornadaElectoral::inicializaForma()/*{{{*/Esta función crea las pantallas
correspondientes a la Jornada Electoral y las ordena
{
    raiseWidget( 1 );
    qApp->processEvents();

    int lintWidget = 2;
    int k=1;
    int i;
    int j,p,q;
    // Se borra la matriz de votos Pregunta 1
    for(i=1; i < 10;i++) {
        stcTotal[k].int_Id_Pregunta=1;
        stcTotal[k].int_Id_Opcion=i;
        stcTotal[k].int_Total=0;
        k++;
    }

    // Se borra la matriz de votos Pregunta 2
    for(i=1; i < 10;i++) {
        stcTotal[k].int_Id_Pregunta=2;
        stcTotal[k].int_Id_Opcion=i;
        stcTotal[k].int_Total=0;
        k++;
    }

    char buffer[100];

    sprintf(buffer, "13. Total de lista nom: %d", int_ListaNom);
    UrnaLog( intPID, buffer, stcPathsPtrDat );
    for (i=0; i< int_ListaNom;i++){
        for (j=0;j<2;j++){

            stcMatrizVotos[i][j].intBusy=0;

        }
    }
    UrnaLog( intPID, "13. Se limpió la matriz de votos", stcPathsPtrDat );

    //Busca si la urna ya fue utilizada
    FILE *pf;
    pf= fopen( "/etc/asvel/log/VeVector.dat", "rt");
```

```

if (pf!=NULL) {
    fgets(buffer,sizeof(buffer),pf);
    k=0;
    while (!feof(pf)) {

        p=strtok(buffer,"|");
        q=strtok(NULL,"|");
        if (atoi(p)==1 || atoi(p)==2) {

            stcMatrizVotos[k][atoi(p)-1].intBusy=1;
            stcMatrizVotos[k][atoi(p)-1].int_Seccion=int_NumSeccion;
            stcMatrizVotos[k][atoi(p)-1].int_Id_Participante=atoi(q);
            stcMatrizVotos[k][atoi(p)-1].int_Id_Eleccion=atoi(p);
        }

        k++;
        if (k==int_ListaNom) k=0;
        fgets(buffer,sizeof(buffer),pf);
    }
    fclose(pf);
}

tmrPtrRetardo = new QTimer( this );
connect( tmrPtrRetardo, SIGNAL( timeout() ), this, SLOT(
slotTiempoAgotado() ) );

tmrPtrHoraFinal = new QTimer( this );
connect( tmrPtrHoraFinal, SIGNAL( timeout() ), this, SLOT(
slotTerminaJornadaElectoral() ) );

sprPtrPrinter = new SerialPrinter();

((CfrmEsperaHabilitacion *)wdgPtrEsperaHabilitacionDat)-
>setDistritoYDelegacion( stcConfCasDat.stcDisDat.int_Id_Distrito,
stcConfCasDat.stcDelDat.chrArr_Delegacion_De );
((CfrmEsperaHabilitacion *)wdgPtrEsperaHabilitacionDat)-
>modeEsperaVotantes();
((CfrmEsperaHabilitacion *)wdgPtrEsperaHabilitacionDat)-
>habilitaKeyEvent(FALSE);
UrnaLog( intPID, "13. Se ha creado la pantalla de espera de votantes",
stcPathsPtrDat );

CfrmBoleta *lCfrmBoletaMR = new CfrmBoleta( intPID, intWidth, intHeight,
stcPathsPtrDat, this, "BoletaMR", wflFlags );

lCfrmBoletaMR->setSound( sndPtrSound );
lCfrmBoletaMR-
>setDatosDeCasilla(stcConfCasDat.stcCasDat.int_Id_Seccion,
stcConfCasDat.stcCasDat.chr_Id_Casilla,
stcConfCasDat.stcDelDat.chrArr_Delegacion_De,
stcConfCasDat.stcDisDat.chrArr_Romano, chrrArr_Serie_Urna,2);

lCfrmBoletaMR->setParticipantes( stcConfCasDat.intNumPar,
stcConfCasDat.stcParPtrDat );
lCfrmBoletaMR->setCandidatos( 3, stcConfCasDat.intNumCanMR,
stcConfCasDat.stcCanMRPtrDat );

```

SP
h.

```
lCfrmBoletaMR->disableButtons(FALSE);

connect( lCfrmBoletaMR, SIGNAL( signalSeleccion() ), SLOT(
slotAnalizaSecuencia() ) );
connect (lCfrmBoletaMR,SIGNAL (signalElige( int, int
)),SLOT(slotEligeParticipante(int, int )));

addWidget( lCfrmBoletaMR, lintWidget++ );
UrnaLog( intPID, "13. Se ha creado la pantalla para la primera
boleta", stcPathsPtrDat );

CfrmBoleta *lCfrmBoletaJD = new CfrmBoleta( intPID, intWidth,
intHeight, stcPathsPtrDat, this, "BoletaJD", wflFlags );
lCfrmBoletaJD->setSound( sndPtrSound );
lCfrmBoletaJD-
>setDatosDeCasilla(stcConfCasDat.stcCasDat.int_Id_Seccion,
stcConfCasDat.stcCasDat.chr_Id_Casilla,
stcConfCasDat.stcDelDat.chrArr_Delegacion_De,
stcConfCasDat.stcDisDat.chrArr_Romano, chrArr_Serie_Urna,1);

lCfrmBoletaJD->setParticipantes( stcConfCasDat.intNumPar,
stcConfCasDat.stcParPtrDat );
lCfrmBoletaJD->setCandidatos( 2, stcConfCasDat.intNumCanJD,
stcConfCasDat.stcCanJDPtrDat );

lCfrmBoletaJD->disableButtons(FALSE);

connect( lCfrmBoletaJD, SIGNAL( signalSeleccion() ), SLOT(
slotAnalizaSecuencia() ) );
connect (lCfrmBoletaJD,SIGNAL (signalElige( int, int
)),SLOT(slotEligeParticipante(int, int )));

addWidget( lCfrmBoletaJD, lintWidget++ );
UrnaLog( intPID, "13. Se ha creado la pantalla para la segunda
boleta", stcPathsPtrDat );

//Por compatibilidad, se crean boletas aunque no se visualicen
CfrmBoleta *lCfrmBoletaJG = new CfrmBoleta( intPID, intWidth,
intHeight, stcPathsPtrDat, this, "BoletaJG", wflFlags );

lCfrmBoletaJG->setSound( sndPtrSound );

lCfrmBoletaJG->setDatosDeCasilla(stcConfCasDat.stcCasDat.int_Id_Seccion,
stcConfCasDat.stcCasDat.chr_Id_Casilla,
stcConfCasDat.stcDelDat.chrArr_Delegacion_De,
stcConfCasDat.stcDisDat.chrArr_Romano,
chrArr_Serie_Urna,1);
lCfrmBoletaJG->setParticipantes( stcConfCasDat.intNumPar,
stcConfCasDat.stcParPtrDat );
lCfrmBoletaJG->setCandidatos( 1, stcConfCasDat.intNumCanJG,
stcConfCasDat.stcCanJGPtrDat );
```

```
lCfrmBoletaJG->disableButtons(TRUE);
lCfrmBoletaJG->setInputMethodEnabled ( FALSE );
connect( lCfrmBoletaJG, SIGNAL( signalSeleccion() ), SLOT(
slotAnalizaSecuencia() ) );
connect (lCfrmBoletaJG,SIGNAL (signaleElige( int , int
)),SLOT(slotEligeParticipante(int , int )));
addWidget( lCfrmBoletaJG, lintWidget++ );

CfrmBoleta *lCfrmBoletaRP = new CfrmBoleta( intPID, intWidth,
intHeight, stcPathsPtrDat, this, "BoletaRP", wflFlags );
addWidget( lCfrmBoletaRP, lintWidget++ );

//Pantalla de gracias por participar
CfrmDespedida *lCfrmDespedida = new CfrmDespedida( intWidth, intHeight,
stcPathsPtrDat, this, "Despedida", wflFlags );
connect( lCfrmDespedida, SIGNAL( signalClose() ), SLOT(
slotAnalizaSecuencia() ) );
addWidget( lCfrmDespedida, lintWidget++ );
UrnaLog( intPID, "13. Se ha creado la pantalla de Despedida en el stack
Jornada Electoral", stcPathsPtrDat );

//pantalla de imprime voto
((CfrmMensajes *)wdgPtrMensajesDat)->setMessage( "I M P R I M I E N D O\n
V O T O \n\n* * * * * " );
((CfrmMensajes *)wdgPtrMensajesDat)->disableClicked( FALSE );

((CfrmEsperaHabilitacion *)wdgPtrEsperaHabilitacionDat)->habilitaKeyEvent(
TRUE );
intEndTimer = startTimer( 1000 );

//Pantalla de selección
wdgPtrSeleccionDat = new CfrmSeleccion( intPID, intWidth, intHeight,
stcPathsPtrDat, 0, "Seleccion", wflFlags );
((CfrmSeleccion *)wdgPtrSeleccionDat)->setSound( sndPtrSound );
((CfrmSeleccion *)wdgPtrSeleccionDat)->setSerialPrinter( sprPtrPrinter );
((CfrmSeleccion *)wdgPtrSeleccionDat)->setCasilla( stcConfCasDat.stcCasDat
);
((CfrmSelección *)wdgPtrSeleccionDat)->setCfrmMensajes( wdgPtrMensajesDat
);

connect( (CfrmSeleccion *)wdgPtrSeleccionDat, SIGNAL( signalClose( int )
), SLOT( slotSeleccionConfirmada( int ) ));
connect( (CfrmSeleccion *)wdgPtrSeleccionDat, SIGNAL( signalCorregir( int
)), SLOT( slotSeleccionCorregir( int ) ));
UrnaLog( intPID, "13. Se ha creado la pantalla de seleccion",
stcPathsPtrDat );
addWidget( wdgPtrSeleccionDat, lintWidget );

//Empieza con la primera pantalla
raiseWidget( 0 );
qApp->processEvents();
```

```

    stcBarPtrACode = stcBarPtrDat + ((CfrmEsperaHabilitacion *)widget( 0 ))-
>code();
    stcBarPtrACode->intJg = 0;
    stcBarPtrACode->intJd = 0;
    stcBarPtrACode->intMr = 0;
    stcBarPtrACode->intRp = 0;

    QTimer *timer=new QTimer(this);
    connect(timer, SIGNAL(timeout()), this, SLOT(slotTiempo()));
    timer->start(3600000, TRUE);
    UrnaLog( intPID, "13. Se ha iniciado el contador de tiempo para copiar archivos
de auditoría", stcPathsPtrDat );

    // Con esto espera el boton habilitador
    ((CfrmEsperaHabilitacion *)wdgPtrEsperaHabilitacionDat)-
>EsperaHabilitacion();

}/*}}}}*/

void CfrmJornadaElectoral::setConfiguracionCasilla( stcConfCasilla
pstcConfCasDat )/*{{{*/ Esta función establece la sección y casilla
{
    stcConfCasDat = pstcConfCasDat;
    UrnaLog( intPID, "Se obtuvo la configuración de la casilla del stack
principal", stcPathsPtrDat );
}/*}}}}*/

void CfrmJornadaElectoral::setCfrmEsperaHabilitacion( QWidget
*pwdgPtrEsperaHabilitacionDat )/*{{{*/ Esta función pasa la clase CfrmJornadaElectoral
para que pueda ser utilizada
{
    addWidget( pwdgPtrEsperaHabilitacionDat, 0 );
    wdgPtrEsperaHabilitacionDat = pwdgPtrEsperaHabilitacionDat;
    stcBarPtrDat = ((CfrmEsperaHabilitacion *)wdgPtrEsperaHabilitacionDat)-
>barCodes();
    connect( (CfrmEsperaHabilitacion *)wdgPtrEsperaHabilitacionDat, SIGNAL(
signalHabilitar() ), this, SLOT( slotAnalizaSecuencia() ) );
    UrnaLog( intPID, "Se ha agregado la pantalla de Espera de Habilitacion al
stack de Jornada Electoral", stcPathsPtrDat );
}/*}}}}*/

void CfrmJornadaElectoral::apagaUrna()/*{{{*/ Esta función se encarga de apagar la
Urna en caso de error
{
#ifdef __SERIAL_PRINTER__
    sprPtrPrinter->cutPaper( 127 );
#endif
    UrnaLog( intPID, "Ocurrio un error irrecoverable. Se apagara la urna",
stcPathsPtrDat );
    ((CfrmMensajes *)wdgPtrMensajesDat)->setMessage( "HA OCURRIDO UN ERROR POR
LO QUE SE APAGARA LA URNA\n\n\nTOQUE LA PANTALLA PARA TERMINAR" );
    tmrPtrRetardo->stop();
    killTimer( intEndTimer );
    raiseWidget( 1 );

```

```
intStep = 20;
}/*}}}}*/
```

```
void CfrmJornadaElectoral::errorBattery()/*{{{*/Esta función muestra un mensaje de
error cuando esta fallando la bateria de la Urna
{
    UrnaLog( intPID, "Ocurrio un error con la bateria. El volataje es muy
bajo. Se apagara la urna", stcPathsPtrDat );
    ((CfrmMensajes *)wdgPtrMensajesDat)->setMessage( "HA OCURRIDO UN ERROR CON
LA BATERIA\n\n\nTOQUE LA PANTALLA PARA APAGAR LA URNA" );
    if ( bolSesionActiva )
    {
        registraCodigo( stcPathsPtrDat->chrArrBarCode_File, stcBarPtrACode
);
        registraCodigo( stcPathsPtrDat->chrArrBarCodeR_File, stcBarPtrACode
);
#ifdef __SERIAL_PRINTER__
        if ( sprPtrPrinter->cutPaper( 127 ) == -1 )
        {
            sprPtrPrinter->cutPaper( 127 );
            apagaUrna();
            return;
        }
#endif
        tmrPtrRetardo->stop();
        killTimer( intEndTimer );
        raiseWidget( 1 );
        intStep = 20;
    }/*}}}}*/

void CfrmJornadaElectoral::errorRegVotos()/*{{{*/
{
#ifdef __SERIAL_PRINTER__
    if ( sprPtrPrinter->cutPaper( 127 ) == -1 )
    {
        sprPtrPrinter->cutPaper( 127 );
        apagaUrna();
        return;
    }
#endif
    UrnaLog( intPID, "Ocurrio un error en la operacion de transaccion. Se
apagara la urna", stcPathsPtrDat );
    if ( intEVotos == -1 )
        ((CfrmMensajes *)wdgPtrMensajesDat)->setMessage( "HA OCURRIDO UN
ERROR DURANTE\nEL REGISTRO DEL VOTO\n\nEN LA MEMORIA INTERNA\n\n\nTOQUE LA
PANTALLA PARA TERMINAR" );
    if ( intEVotos == -2 )
        ((CfrmMensajes *)wdgPtrMensajesDat)->setMessage( "HA OCURRIDO UN
ERROR DURANTE\nEL REGISTRO DEL VOTO\n\nEL DISPOSITIVO DE RESPALDO\n\n\nTOQUE LA
PANTALLA PARA TERMINAR" );
    tmrPtrRetardo->stop();
    killTimer( intEndTimer );
    raiseWidget( 1 );
}
```

```

        intStep = 20;
    }/*}}}}*/

QString CfrmJornadaElectoral::revisaVotacion()/*{{{*/
{
    char *lchrArrPtrTipo[2] = { "no emitido", "emitido" };
    QString lstrVotacion;

    if ( strcmp( stcConfCasDat.stcCasDat.chr_Id_Casilla, "E" ) )
        lstrVotacion.sprintf( "ESTADO DE LA ULTIMA SESION\n\nVOTO POR JG:
%s\nVOTO POR JD: %s\nVOTO POR DIP. DE MR: %s\n\n\nTOQUE LA PANTALLA PARA
CONTINUAR", lchrArrPtrTipo[stcBarPtrACodeA->intJg],
lchrArrPtrTipo[stcBarPtrACodeA->intJd], lchrArrPtrTipo[stcBarPtrACodeA->intMr]
);
    else
        lstrVotacion.sprintf( "ESTADO DE LA ULTIMA SESION\n\nVOTO POR JG:
%s\nVOTO POR DIP. DE RP: %s\n\n\nTOQUE LA PANTALLA PARA CONTINUAR",
lchrArrPtrTipo[stcBarPtrACodeA->intJg], lchrArrPtrTipo[stcBarPtrACodeA->intRp]
);
    return lstrVotacion;
}/*}}}}*/

void CfrmJornadaElectoral::slotTiempoAgotado()/*{{{*/ Este slot se activa cuando el
tiempo para emitir votos (sesión de votación) se ha agotado, deshabilita las boletas y envía un
mensaje para que el votante sepa que el tiempo se ha agotado
{
#ifdef __SERIAL_PRINTER__
    if ( sprPtrPrinter->cutPaper( 127 ) == -1 )
    {
        sprPtrPrinter->cutPaper( 127 );
        apagaUrna();
        return;
    }
#endif
    ((CfrmMensajes *)wdgPtrMensajesDat)->setMessage( "SU TIEMPO PARA VOTAR HA
TERMINADO\n\nSOLICITE AL PRESIDENTE\nLA REACTIVACION DE LA URNA" );
    ((CfrmMensajes *)wdgPtrMensajesDat)->disableClicked( TRUE );
    raiseWidget( 1 );
    stcBarPtrACodeA = stcBarPtrACode;
    if ( registraCodigo( stcPathsPtrDat->chrArrBarCode_File, stcBarPtrACode )
    )
        apagaUrna();
    if ( registraCodigo( stcPathsPtrDat->chrArrBarCodeR_File, stcBarPtrACode )
    )
        apagaUrna();
    ((CfrmMensajes *)wdgPtrMensajesDat)->startTimer();
    UrnaLog( intPID, "El tiempo para emitir el voto ha terminado",
stcPathsPtrDat );
    intStep = 8;
}/*}}}}*/

void CfrmJornadaElectoral::slotAnalizaSecuencia()/*{{{*/ Este slot se encarga de
analizar la secuencia que sigue la Jornada Electoral de la Urna y ejecuta la acción pertinente.
{
    char buffer[150];
    int lintDesp;

    try{

```

```
// deshabilita todos los botones
    sprintf(buffer,"%s%d","13. CfrmJornadaElectoral intStep=",intStep);
    UrnaLog( intPID,buffer, stcPathsPtrDat );

    sprintf(buffer,"%s%d","13. Valor de stcBarPtrACode->intJg
=",stcBarPtrACode->intJg);
    UrnaLog( intPID,buffer, stcPathsPtrDat );

    sprintf(buffer,"%s%d","13. Valor de stcBarPtrACode->intJd
=",stcBarPtrACode->intJd);
    UrnaLog( intPID,buffer, stcPathsPtrDat );

    sprintf(buffer,"%s%d","13. Valor de stcBarPtrACode->intMr
=",stcBarPtrACode->intMr);
    UrnaLog( intPID,buffer, stcPathsPtrDat );

//    sprintf(buffer,"%s%d","Valor de stcBarPtrACode->intRp =",stcBarPtrACode-
>intRp);
//    UrnaLog( intPID,buffer, stcPathsPtrDat );
if (contador=int_ListaNom){
    intStep=20;
}
switch ( intStep++ )
{
    case 0 : tmrPtrRetardo->start( intVoteTime, TRUE );
        if ( bolTimerFinal )
        {
            bolTimerFinal = FALSE;
            tmrPtrHoraFinal->stop();
        }

        bolSesionActiva = TRUE;

        stcBarPtrACode->intJg = 0;
        stcBarPtrACode->intJd = 0;
        stcBarPtrACode->intMr = 0;
//        stcBarPtrACode->intRp = 0;

        sprintf(stcResp.chrArr_Id_SerialUrna,'\0');
        stcResp.int_Distrito = 0;
        stcResp.int_Id_Pregunta = 0;
        stcResp.int_Id_Opcion = 0;
        sprintf(stcResp.chrArr_T_Inicio, '\0');
        stcResp.doub_T_Total=0;
        stcResp.doub_T_Total_G=0;

        ((CfrmBoleta *) widget( 2 ))->bolDesSeleccion = TRUE;
        ((CfrmBoleta *) widget( 3 ))->bolDesSeleccion = TRUE;
        ((CfrmBoleta *) widget( 4 ))->bolDesSeleccion = TRUE;
        //((CfrmBoleta *) widget( 5 ))->bolDesSeleccion = TRUE;

        ((CfrmEsperaHabilitacion *)wdgPtrEsperaHabilitacionDat)-
>habilitaKeyEvent( FALSE );
        //TIEMPOS
        if (!Tiempo) {
            Tiempo=TRUE;
            time(&T_Inicio);
        }
    }
}
```

```

stcPathsPtrDat );

    UrnaLog( intPID,"13. inicio de votacion boleta 1",
    sprintf(buffer,"%s",asctime(localtime(&T_Inicio)));
    UrnaLog( intPID,buffer, stcPathsPtrDat );
    }

qApp->removePostedEvents ( widget( 7) );
//qApp->flushX();
//

((CfrmBoleta *) widget( 2))->bolDesSeleccion = TRUE;
// ((CfrmBoleta *) widget(2))->blockSignals(FALSE);
((CfrmBoleta *) widget( 2))->disableButtons(TRUE);
    if (qApp->hasPendingEvents())
qApp->removePostedEvents ( widget( 2) );

    raiseWidget( 2 );

qApp->processEvents();
//slotAnalizaSecuencia();

sndPtrSound[8]->stop();
sndPtrSound[3]->play();
//((CfrmBoleta *) widget( 2))->EsperaBraille();
break;
case 1 :
    if (qApp->hasPendingEvents())
        qApp->removePostedEvents ( widget( 1) );

        raiseWidget( 1 );
        ((CfrmMensajes *)wdgPtrMensajesDat)->startTimer();
        qApp->processEvents();
        //antes en seleccion
        guardaRespuestas();
        ImprimeTicket(1);
        stcBarPtrACode->intJd = 1;

        //sndPtrSound[8]->stop();
        //sndPtrSound[9]->play();
        break;
case 2 : // Opcion para jefe delegacional
        if ( stcBarPtrACode->intJd != 0 )
        {
            intStep++;
            intVotos++;
            slotAnalizaSecuencia();
            break;
        }
        ((CfrmBoleta *) widget( 2))->disableButtons(FALSE);
        ((CfrmBoleta *) widget( 3))->disableButtons(TRUE);
        //qApp->flushX();
        ((CfrmBoleta *) widget(3))->bolDesSeleccion = TRUE;
        ((CfrmBoleta *) widget(3))->blockSignals(FALSE);

        if (qApp->hasPendingEvents())
            qApp->removePostedEvents ( widget( 3)

```

Handwritten signature

Handwritten mark

```

        raiseWidget( 3 );
        qApp->processEvents();
        sndPtrSound[8]->stop();
        sndPtrSound[4]->play();
        //TIEMPOS mgg
        if (!Tiempo) {
            Tiempo=TRUE;
            UrnaLog( intPID,"13. inicio de
votacion boleta 2", stcPathsPtrDat );
            time(&T_Inicio);

            sprintf(buffer,"%s",asctime(localtime(&T_Inicio)));
            UrnaLog( intPID,buffer, stcPathsPtrDat
);
        }
        break;
    case 3 :
        if (qApp->hasPendingEvents())
            qApp->removePostedEvents ( widget( 1 ) );
        raiseWidget( 1 );
        ((CfrmMensajes *)wdgPtrMensajesDat)->startTimer();
        qApp->processEvents();
        stcBarPtrACode->intMr= 1;
        guardaRespuestas();
        ImprimeTicket(2);
        //sndPtrSound[8]->stop();
        //sndPtrSound[9]->play();
        break;
    case 4 :
        if ( stcBarPtrACode->intJd != 0 )
        {
            intStep++;
            slotAnalizaSecuencia();
            break;
        }

        if (qApp->hasPendingEvents())
            qApp->removePostedEvents ( widget( 3)
);

        raiseWidget( 3 );
        qApp->processEvents();
        sndPtrSound[8]->stop();
        sndPtrSound[4]->play();
        //TIEMPOS
        if (!Tiempo) {
            Tiempo=TRUE;
            UrnaLog( intPID,"13. inicio de
votacion pregunta 2", stcPathsPtrDat );
            time(&T_Inicio);

            sprintf(buffer,"%s",asctime(localtime(&T_Inicio)));
            UrnaLog( intPID,buffer, stcPathsPtrDat
);

            //Removing tirtth question ARP 031609
            //
            //raiseWidget( 4 );
            //uApp->processEvents();
            //intStep=8;

```

SP
5.

```

//slotAnalizaSecuencia();
//
//
//sndPtrSound[5]->play();
//TIEMPOS
if (!Tiempo) {
    Tiempo=TRUE;
    time(&T_Inicio);

    sprintf(buffer,"%s",asctime(localtime(&T_Inicio)));
    UrnaLog( intPID,buffer, stcPathsPtrDat );
}
break;
case 5 :
if (qApp->hasPendingEvents())
    qApp->removePostedEvents ( widget( 1 ) );

    raiseWidget( 1 );
    ((CfrmMensajes *)wdgPtrMensajesDat)->startTimer();
    qApp->processEvents();

    guardaRespuestas();
    ImprimeTicket(1);
    stcBarPtrACode->intJd= 1;

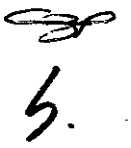
    break;
case 6 :

    if ( stcBarPtrACode->intJd != 0 )
    {
        intStep++;
        intVotos++;
        slotAnalizaSecuencia();
        break;
    }

    break;
case 7 :
    intStep=8;
    slotAnalizaSecuencia();

    break;
case 8:
    tmrPtrRetardo->stop();
    intVotos = 0;
    //qApp->flushX();
    if (qApp->hasPendingEvents())
        qApp->removePostedEvents ( widget( 6 ) );
    raiseWidget( 6 );
    qApp->processEvents();
    ((CfrmDespedida *)widget( 6 ))->iniciaCuenta();
    UrnaLog( intPID, "13: Se ha terminado una sesion de
votacion", stcPathsPtrDat );
    contador++;
    break;
case 9:
    if (qApp->hasPendingEvents())

```



```

        qApp->removePostedEvents ( widget ( 0 ) );
        raiseWidget ( 0 );
        qApp->processEvents();
        intStep = 0;
//        qApp->syncX();
        stcBarPtrACode->intJg = 0;
        stcBarPtrACode->intJd = 0;
        stcBarPtrACode->intMr = 0;
        stcBarPtrACode->intRp = 0;

        ((CfrmBoleta *) widget ( 2 ))->bolDesSeleccion = TRUE;
        ((CfrmBoleta *) widget ( 3 ))->bolDesSeleccion = TRUE;
        ((CfrmBoleta *) widget ( 4 ))->bolDesSeleccion = TRUE;
        (((CfrmBoleta *) widget ( 5 ))->bolDesSeleccion = TRUE;
        ((CfrmBoleta *) widget ( 2 ))->disableButtons(TRUE);

        //qApp->flushX();
        habilitador(Todos, Apaga);
        habilitador(Verde, Prende);
        // bolActivacion = FALSE;
        bolSesionActiva = FALSE;
        ((CfrmEsperaHabilitacion *)wdgPtrEsperaHabilitacionDat)-
>EsperaHabilitacion();
        break;
        case 20:
#ifdef __BITSY__
            if ( haltSistem() )
            {
                qApp->exit( 0 );
                habilitador(Todos, Apaga);
            }
        #else
            qApp->exit( 0 );
        #endif
        break;
    }
}/*}}}}*/

void CfrmJornadaElectoral::slotTerminaJornadaElectoral()/*{{{ Este slot da de
baja las pantallas que se utilizarón durante la Jornada Electoral. Borra las boletas utilizadas y emite
una señal que indica que la Jornada Electoral ha terminado.

{
    QWidget *lwdgPtrTmp;

    if ( bolSesionActiva )
        return;

    delete tmrPtrRetardo;
    delete tmrPtrHoraFinal;
    disconnect( (CfrmMensajes *)wdgPtrMensajesDat, SIGNAL( signalTimeout() ),
this, SLOT( slotAnalizaSecuencia() ) );
    disconnect( (CfrmMensajes *)wdgPtrMensajesDat, SIGNAL( signalClicked() ),
this, SLOT( slotAnalizaSecuencia() ) );
    disconnect( (CfrmEsperaHabilitacion *)wdgPtrEsperaHabilitacionDat, SIGNAL(
signalHabilitar() ), this, SLOT( slotAnalizaSecuencia() ) );
    removeWidget( wdgPtrMensajesDat );
    removeWidget( wdgPtrEsperaHabilitacionDat );
    delete wdgPtrSeleccionDat;

```

```

removeWidget( lwdgPtrTmp = widget( 0 ) );
delete lwdgPtrTmp;
removeWidget( lwdgPtrTmp = widget( 2 ) );
delete lwdgPtrTmp;
removeWidget( lwdgPtrTmp = widget( 3 ) );
delete lwdgPtrTmp;
removeWidget( lwdgPtrTmp = widget( 4 ) );
delete lwdgPtrTmp;
if ( !bolEspecial )
{
    removeWidget( lwdgPtrTmp = widget( 5 ) );
    delete lwdgPtrTmp;
}
#ifdef __SERIAL_PRINTER__
    delete sprPtrPrinter;
#endif
killTimer( intEndTimer );
UrnaLog( intPID, "Ha terminado la fase \"Jornada de Votacion\"",
stcPathsPtrDat );
    emit signalClose();
}/*}}}}*/

void CfrmJornadaElectoral::timerEvent( QTimerEvent *e )/*{{{*/
{
    if ( e->timerId() == intEndTimer )
    {
        if ( !checkPower( 10.5, &fltVoltaje ) )
        {
            errorBattery();
            return;
        }
        if ( QTime::currentTime() >= QTime::fromString( stcPathsPtrDat-
>chrArrCierre ) && !bolTimerFinal && !bolSesionActiva )
        {
            bolTimerFinal = TRUE;
            tmrPtrHoraFinal->start( intTCierre, TRUE );
            UrnaLog( intPID, "El tiempo de tolerancia de cierre se ha
activado.", stcPathsPtrDat );
        }
        else if ( QTime::currentTime() >= QTime::fromString( stcPathsPtrDat-
>chrArrCierreFinal ) && !bolSesionActiva )
        {
            UrnaLog( intPID, "El tiempo de votacion ha excedido el tiempo,
se terminara esta fase.", stcPathsPtrDat );
            slotTerminaJornadaElectoral();
        }
    }
}/*}}}}*/

```

Handwritten signature and the number 5.

ARCHIVO CFRMTOTALIZAVOTACION.CPP

Esta clase realiza la sumatoria de los votos recibidos en la Urna Electrónica, así como la creación de las actas correspondientes. La funciones creaActaJD(), creaActaMR(), creaActaRP() crean los archivos de las actas Jefe delegacional, Diputado Representación Proporcional y Diputado de Mayoría Relativa, respectivamente.

```
#include <CfrmTotalizaVotacion.h>
#define __CFRMTOTALIZAVOTACION_MSG__
#include <time.h>
#include <errno.h>
#include <fcntl.h>
#include <common.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <SysLogExa.h>

extern int int_NumDistrito;
extern char chrrArr_Serie_Urna[40];
extern char chr_Tipo_Casilla[2];
extern int int_Delegacion;
extern int int_NumSeccion;
extern int int_ListaNom;
extern struct stcTotales stcTotal[25];
extern struct stcRespuestas stcResp;

CfrmTotalizaVotacion::CfrmTotalizaVotacion( int pintPID, stcPaths
*pstcPathsPtrDat, int pintNumPar, stcParticipante *pstcParPtrDat )/*{{{*/
: stcVotPtrDat( NULL ), stcPathsPtrDat( pstcPathsPtrDat ), stcParPtrDat(
pstcParPtrDat ),
intPID( pintPID ), intError( -1 ), intNumPar( pintNumPar ), intNumCanJG(
0 ), intNumCanJD( 0 ), intNumCanMR( 0 ), intNumCanRP( 0 ), chrPtrError( NULL )
{
    UrnaLog( intPID, "Se iniciara la totalizacion de los votos",
stcPathsPtrDat );
}/*}}}*/

CfrmTotalizaVotacion::~CfrmTotalizaVotacion()/*{{{*/
{
}/*}}}*/

void CfrmTotalizaVotacion::writeFile()/*{{{*/ Esta función guarda los datos de la
votación en la bitácora del sistema(log) generados durante la jornada electoral. los datos también
son guardados en el dispositivo de almacenamiento USB.
{
    int lintFileDes;
    stcConfiguracion lstcConfEncabezado;

    if ( intError != -1 )
        return;
```

```

    if ( (lintFileDes = open( stcPathsPtrDat->chrArrTotalizacion_File, O_CREAT
| O_RDWR, S_IRWXU | S_IRWXG )) == -1 )
    {
        chrPtrError = gchrArrPtrTotVotosErrors[intError = 10];
        return;
    }

    lstcConfEncabezado.int_Id_Seccion = stcCasDat.int_Id_Seccion;
    strcpy( lstcConfEncabezado.chr_Id_Casilla, stcCasDat.chr_Id_Casilla );

    if ( write( lintFileDes, &lstcConfEncabezado, sizeof( stcConfiguracion ) )
== -1 )
    {
        chrPtrError = gchrArrPtrTotVotosErrors[intError = 10];
        return;
    }
    if ( (int)write( lintFileDes, stcVotPtrDat, sizeof( stcVotos ) *
stcCasDat.int_Lista_Nom ) == -1 )
    {
        chrPtrError = gchrArrPtrTotVotosErrors[intError = 10];
        return;
    }

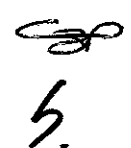
    if ( (int)write( lintFileDes, stcResVotPtrVotosJG, sizeof(
stcResultadoCasilla ) * intNumCanJG ) == -1 )
    {
        chrPtrError = gchrArrPtrTotVotosErrors[intError = 10];
        return;
    }

    if ( !strcmp( stcCasDat.chr_Id_Casilla, "E" ) )
    {
        if ( (int)write( lintFileDes, stcResVotPtrVotosRP, sizeof(
stcResultadoCasilla ) * intNumCanRP ) == -1 )
        {
            chrPtrError = gchrArrPtrTotVotosErrors[intError = 10];
            return;
        }
    }
    else
    {
        if ( (int)write( lintFileDes, stcResVotPtrVotosJD, sizeof(
stcResultadoCasilla ) * intNumCanJD ) == -1 )
        {
            chrPtrError = gchrArrPtrTotVotosErrors[intError = 10];
            return;
        }
        if ( (int)write( lintFileDes, stcResVotPtrVotosMR, sizeof(
stcResultadoCasilla ) * intNumCanMR ) == -1 )
        {
            chrPtrError = gchrArrPtrTotVotosErrors[intError = 10];
            return;
        }
    }

    close( lintFileDes );
}/*}}*/

```

void CfrmTotalizaVotacion::createActa(int pintOpc)/*{/* Esta función crea las
actas a partir del archivo de "log" de los votos



```
{
    readFile();
    countVotos();
    createActas( pintOpc );
}/*}}*/

const char *CfrmTotalizaVotacion::getError()/*{{{*/
{
    return chrPtrError;
}/*}}*/

void CfrmTotalizaVotacion::setCandidatoJD( int pintNumCan, stcCandidato_JD
*pstcCanJDPtrDat )/*{{{*/ Esta función establece los candidatos a Jefe Delegacional
{
    intNumCanJD = pintNumCan;

    if ( intError != -1 )
        return;
    if ( !(stcResVotPtrVotosJD = (stcResultadoCasilla *)malloc( sizeof(
stcResultadoCasilla ) * pintNumCan )) )
    {
        chrPtrError = gchrArrPtrTotVotosErrors[intError = 1];
        return;
    }
    for ( int lintX = 0; lintX < pintNumCan - 1; lintX++ )
    {
        (stcResVotPtrVotosJD + lintX)->int_Id_Participante =
(pstcCanJDPtrDat + lintX)->int_Id_Participante;
        (stcResVotPtrVotosJD + lintX)->int_Seccion =
stcCasDat.int_Id_Seccion;
        strcpy( (stcResVotPtrVotosJD + lintX)->chr_Id_Casilla,
stcCasDat.chr_Id_Casilla );
        (stcResVotPtrVotosJD + lintX)->int_Id_Eleccion = 2;
        (stcResVotPtrVotosJD + lintX)->int_TotalVotos = 0;
    }
    (stcResVotPtrVotosJD + pintNumCan - 1)->int_Id_Participante = 0;
    (stcResVotPtrVotosJD + pintNumCan - 1)->int_Seccion =
stcCasDat.int_Id_Seccion;
    strcpy( (stcResVotPtrVotosJD + pintNumCan - 1)->chr_Id_Casilla,
stcCasDat.chr_Id_Casilla );
    (stcResVotPtrVotosJD + pintNumCan - 1)->int_Id_Eleccion = 2;
    (stcResVotPtrVotosJD + pintNumCan - 1)->int_TotalVotos = 0;
}/*}}*/

void CfrmTotalizaVotacion::setCandidatoMR( int pintNumCan, stcCandidato_MR
*pstcCanMRPtrDat )/*{{{*/ Esta función establece los candidatos a Diputado por Mayoría Relativa
{
    intNumCanMR = pintNumCan;

    if ( intError != -1 )
        return;
    if ( !(stcResVotPtrVotosMR = (stcResultadoCasilla *)malloc( sizeof(
stcResultadoCasilla ) * pintNumCan )) )
    {
        chrPtrError = gchrArrPtrTotVotosErrors[intError = 1];
        return;
    }
    for ( int lintX = 0; lintX < pintNumCan - 1; lintX++ )
```

Cap
5.

```

    {
        (stcResVotPtrVotosMR + lintX)->int_Id_Participante =
        (pstcCanMRPtrDat + lintX)->int_Id_Participante;
        (stcResVotPtrVotosMR + lintX)->int_Seccion =
        stcCasDat.int_Id_Seccion;
        strcpy( (stcResVotPtrVotosMR + lintX)->chr_Id_Casilla,
        stcCasDat.chr_Id_Casilla );
        (stcResVotPtrVotosMR + lintX)->int_Id_Eleccion = 3;
        (stcResVotPtrVotosMR + lintX)->int_TotalVotos = 0;
    }
    (stcResVotPtrVotosMR + pintNumCan - 1)->int_Id_Participante = 0;
    (stcResVotPtrVotosMR + pintNumCan - 1)->int_Seccion =
    stcCasDat.int_Id_Seccion;
    strcpy( (stcResVotPtrVotosMR + pintNumCan - 1)->chr_Id_Casilla,
    stcCasDat.chr_Id_Casilla );
    (stcResVotPtrVotosMR + pintNumCan - 1)->int_Id_Eleccion = 3;
    (stcResVotPtrVotosMR + pintNumCan - 1)->int_TotalVotos = 0;
}/*}}}*/

void CfrmTotalizaVotacion::setCandidatoRP( int pintNumCan, stcCandidato_RP
*pstcCanRPPtrDat )/*{{{*/ Esta función establece los candidatos a diputados por Mayoría
Relativa
{
    intNumCanRP = pintNumCan;

    if ( intError != -1 )
        return;
    if ( !(stcResVotPtrVotosRP = (stcResultadoCasilla *)malloc( sizeof(
    stcResultadoCasilla ) * pintNumCan.)) )
    {
        chrPtrError = gchrArrPtrTotVotosErrors[intError = 1];
        return;
    }
    for ( int lintX = 0; lintX < pintNumCan - 1; lintX++ )
    {
        (stcResVotPtrVotosRP + lintX)->int_Id_Participante =
        (pstcCanRPPtrDat + lintX)->int_Id_Participante;
        (stcResVotPtrVotosRP + lintX)->int_Seccion =
        stcCasDat.int_Id_Seccion;
        strcpy( (stcResVotPtrVotosRP + lintX)->chr_Id_Casilla,
        stcCasDat.chr_Id_Casilla );
        (stcResVotPtrVotosRP + lintX)->int_Id_Eleccion = 4;
        (stcResVotPtrVotosRP + lintX)->int_TotalVotos = 0;
    }
    (stcResVotPtrVotosRP + pintNumCan - 1)->int_Id_Participante = 0;
    (stcResVotPtrVotosRP + pintNumCan - 1)->int_Seccion =
    stcCasDat.int_Id_Seccion;
    strcpy( (stcResVotPtrVotosRP + pintNumCan - 1)->chr_Id_Casilla,
    stcCasDat.chr_Id_Casilla );
    (stcResVotPtrVotosRP + pintNumCan - 1)->int_Id_Eleccion = 4;
    (stcResVotPtrVotosRP + pintNumCan - 1)->int_TotalVotos = 0;
}/*}}}*/

void CfrmTotalizaVotacion::setCasDisDel( stcCasilla pstcCasDat, stcDistrito
pstcDisDat, stcDelegacion pstcDelDat )/*{{{*/ Esta función establece los datos de la
casilla, delegación y distrito de este objeto
{
    stcCasDat = pstcCasDat;
    stcDisDat = pstcDisDat;
    stcDelDat = pstcDelDat;
}

```

```

        if ( intError != -1 )
            return;
        if ( !(stcVotPtrDat = (stcVotos *)malloc( sizeof( stcVotos ) *
stcCasDat.int_Lista_Nom )) )
        {
            UrnaLog( intPID, "Hubo un error al intentar generar memoria para la
totalizacion de votos", stcPathsPtrDat );
            chrPtrError = gchrArrPtrTotVotosErrors[intError = 0];
            return;
        }
        UrnaLog( intPID, "Se han obtenido los datos de casilla", stcPathsPtrDat );
    }/*}}}}*/

```

void CfrmTotalizaVotacion::readFile()/*{{{ Esta función lee el archivo donde se guardan los votos y obtiene los datos para la contabilización de los votos

```

{
    FILE *lfilePtrVotosFile = NULL;

    if ( intError != -1 )
        return;

    if ( !(lfilePtrVotosFile = fopen( stcPathsPtrDat->chrArrRegVotos_File, "r"
)) )
    {
        UrnaLog( intPID, "Ocurrió un error al intentar abrir el archivo de
registro de votos", stcPathsPtrDat );
        chrPtrError = gchrArrPtrTotVotosErrors[intError = 2];
        return;
    }
    if ( (int)fread( stcVotPtrDat, sizeof( stcVotos ),
stcCasDat.int_Lista_Nom, lfilePtrVotosFile ) == -1 )
    {
        UrnaLog( intPID, "Ocurrió un error al intentar leer el archivo de
registro de votos", stcPathsPtrDat );
        chrPtrError = gchrArrPtrTotVotosErrors[intError = 3];
        return;
    }
    fclose( lfilePtrVotosFile );
}/*}}}}*/

```

void CfrmTotalizaVotacion::countVotos{ Se encarga de elegir que tipo de función va a ocupar, dependiendo del tipo de casilla que se trate

```

    if ( intError != -1 )
        return;
    if ( !strcmp( stcCasDat.chr_Id_Casilla, "E" ) )
        countSpecial();
    else
        countBasic();
}/*}}}}*/

```

void CfrmTotalizaVotacion::countBasic()/*{{{ Esta función realiza la contabilización de votos de una casilla básica

```

{
    if ( intError != -1 )
        return;
    for ( int lintX = 0; lintX < stcCasDat.int_Lista_Nom; lintX++ )
        switch( (stcVotPtrDat + lintX)->int_Id_Eleccion )
        {

```

SP
h.

```

        case 1 : for ( int lintY = 0; lintY < intNumCanJG; lintY++ )
            if ( (stcResVotPtrVotosJG + lintY) -
>int_Id_Participante == (stcVotPtrDat + lintX)->int_Id_Participante )
            {
                (stcResVotPtrVotosJG + lintY) -
>int_TotalVotos++;
                break;
            }
        break;
        case 2 : for ( int lintY = 0; lintY < intNumCanJD; lintY++ )
            if ( (stcResVotPtrVotosJD + lintY) -
>int_Id_Participante == (stcVotPtrDat + lintX)->int_Id_Participante )
            {
                (stcResVotPtrVotosJD + lintY) -
>int_TotalVotos++;
                break;
            }
        break;
        case 3 : for ( int lintY = 0; lintY < intNumCanMR; lintY++ )
            if ( (stcResVotPtrVotosMR + lintY) -
>int_Id_Participante == (stcVotPtrDat + lintX)->int_Id_Participante )
            {
                (stcResVotPtrVotosMR + lintY) -
>int_TotalVotos++;
                break;
            }
        break;
    }
}/*}}}}*/

```

void CfrmTotalizaVotacion::countSpecial()/*{{{*/ Esta función realiza la contabilización de votos de una casilla especial

```

{
    if ( intError != -1 )
        return;
    for ( int lintX = 0; lintX < stcCasDat.int_Lista_Nom; lintX++ )
        switch( (stcVotPtrDat + lintX)->int_Id_Eleccion )
        {
            case 1 : for ( int lintY = 0; lintY < intNumCanJG; lintY++ )
                if ( (stcResVotPtrVotosJG + lintY) -
>int_Id_Participante == (stcVotPtrDat + lintX)->int_Id_Participante )
                {
                    (stcResVotPtrVotosJG + lintY) -
>int_TotalVotos++;
                    break;
                }
            break;
            case 4 : for ( int lintY = 0; lintY < intNumCanRP; lintY++ )
                if ( (stcResVotPtrVotosRP + lintY) -
>int_Id_Participante == (stcVotPtrDat + lintX)->int_Id_Participante )
                {
                    (stcResVotPtrVotosRP + lintY) -
>int_TotalVotos++;
                    break;
                }
            break;
        }
}/*}}}}*/

```

void CfrmTotalizaVotacion::createActas(int pintOpc)/*{{{*/

CSF

h.



5.

[illegible]

```
void CfrmTotalizaVotacion::creaActaJD( FILE *pfilePtrFile )/*{{{*/ Esta función crea el acta de cómputo de votos para la elección de Jefe Delegacional
```

```
int lintTotal = 0;
char chrArrBuffer[150];
time_t now = time(NULL);
int tmpVotos;
```

```
int i=1,j=1,k=0;
FILE *pf;
FILE *pf2;
char buffer[100];
char buffer2[100];
int p,q;
```

```
for (i=0; i < 25; i++)
{
    stcTotal[k].int_Id_Pregunta=0;
    stcTotal[k].int_Id_Opcion=0;
    stcTotal[k].int_Total=0;
    k++;
}
```

5.

```
k=1;
pf= fopen( "/etc/asvel/log/Vector.dat", "rt");
fgets(buffer,sizeof(buffer),pf);
while (!feof(pf)) {
    p= strtok(buffer,"|");
    q= strtok(NULL,"|");

    switch(atoi(p))
    {
        case 1:
            k++;
            switch(atoi(q))
            {
                case 1:

                    stcResp.int_Id_Opcion=1;
                    stcTotal[1].int_Total=stcTotal[1].int_Total+1;
                    break;
                case 2:

                    stcResp.int_Id_Opcion=2;
                    stcTotal[2].int_Total=stcTotal[2].int_Total+1;
                    break;
                case 3:

                    stcResp.int_Id_Opcion=3;
                    stcTotal[3].int_Total=stcTotal[3].int_Total+1;
                    break;
                case 4:

                    stcResp.int_Id_Opcion=4;
                    stcTotal[4].int_Total=stcTotal[4].int_Total+1;
                    break;
                case 5:

                    stcResp.int_Id_Opcion=5;
                    stcTotal[5].int_Total=stcTotal[5].int_Total+1;
                    break;
                case 6:

                    stcResp.int_Id_Opcion=6;
                    stcTotal[6].int_Total=stcTotal[6].int_Total+1;
                    break;
                case 7:

                    stcResp.int_Id_Opcion=7;
                    stcTotal[7].int_Total=stcTotal[7].int_Total+1;
                    break;
                case 8:

                    stcResp.int_Id_Opcion=8;
                    stcTotal[8].int_Total=stcTotal[8].int_Total+1;
                    break;
                case 9:

                    stcResp.int_Id_Opcion=9;
```

```
        stcTotal[9].int_Total=stcTotal[9].int_Total+1;
        break;
    }

    break;
    case 2:

    k++;

    switch(atoi(q))
    {
        case 10:

            stcResp.int_Id_Opcion=1;
            stcTotal[10].int_Total=stcTotal[10].int_Total+1;
            break;
            case 11:

                stcResp.int_Id_Opcion=2;
                stcTotal[11].int_Total=stcTotal[11].int_Total+1;
                break;
            case 12:

                stcResp.int_Id_Opcion=3;
                stcTotal[12].int_Total=stcTotal[12].int_Total+1;
                break;
            case 13:

                stcResp.int_Id_Opcion=4;
                stcTotal[13].int_Total=stcTotal[13].int_Total+1;
                break;
            case 14:

                stcResp.int_Id_Opcion=5;
                stcTotal[14].int_Total=stcTotal[14].int_Total+1;
                break;
            case 15:

                stcResp.int_Id_Opcion=6;
                stcTotal[15].int_Total=stcTotal[15].int_Total+1;
                break;
            case 16:

                stcResp.int_Id_Opcion=7;
                stcTotal[16].int_Total=stcTotal[16].int_Total+1;
                break;
            case 17:

                stcResp.int_Id_Opcion=8;
                stcTotal[17].int_Total=stcTotal[17].int_Total+1;
                break;
            case 18:

                stcResp.int_Id_Opcion=9;
                stcTotal[18].int_Total=stcTotal[18].int_Total+1;
                break;
        }

    break;
```

ESP
5.

```

    }
    fgets(buffer, sizeof(buffer), pf);

    }
    fclose(pf);
    // Lectura file

tmpVotos = 0;

if ( (pf2= fopen( "/etc/asvel/log/total.dat", "wt" ) ) == NULL )
    return -1;
    for(i=1; i <=18; i ++) {
        if (i<=9)
            {j=1;}
        else
            {j=2;}
        sprintf(buffer2, "%d%c%d%c%d%c%c", j, '|', i, '|', stcTotal[i].int_Total, 13, 10);
        fputs( buffer2, pf2);
    }
    fclose ( pf2);

    // Inicio de creacion de archivo
    fprintf( pflePtrFile, "\n          INSTITUTO ELECTORAL\n          DEL
DISTRITO FEDERAL\n\n" );
    fprintf( pflePtrFile, "  COMPROBANTE DE ESCRUTINIO Y\n
COMPUTO\n" );
    fprintf( pflePtrFile, "  PROCESO ELECTORAL 2008-2009\n" );

    strftime( chrArrBuffer, sizeof( chrArrBuffer ) - 1, " %d/%m/%Y
%X \n\n", localtime( &now ) );
    fprintf( pflePtrFile, "%s", chrArrBuffer );
    char *chr_Distrito=regresaRomano(int_NumDistrito);
    char *chr_Delegacion=regresaDelegacion(int_Delegacion);
    sprintf(chrArrBuffer, "DISTRITO %s \nDELEGACION %s\nSECCION %d
CASILLA %s\nELECTORES EN LISTA NOMINAL %d\nNSUE=%s\n", chr_Distrito,
chr_Delegacion, int_NumSeccion, chr_Tipo_Casilla, int_ListaNom,
chrArr_Serie_Urna);
    fprintf( pflePtrFile, "%s\n", chrArrBuffer);

    fprintf( pflePtrFile, "\n\nELECCION DE DIPUTADOS A LA ALDF\n
POR EL PRINCIPIO DE MAYORIA\n          RELATIVA\n\n");
    //fprintf( pflePtrFile, "DE MEXICO? \n");
    fprintf( pflePtrFile, "%s%d\n", "PAN",
stcTotal[10].int_Total );
    fprintf( pflePtrFile, "%s%d\n", "PRI",
stcTotal[11].int_Total );
    fprintf( pflePtrFile, "%s%d\n", "PRD",
stcTotal[12].int_Total );
    fprintf( pflePtrFile, "%s%d\n", "PT",
stcTotal[13].int_Total );
    fprintf( pflePtrFile, "%s%d\n", "PVEM",
stcTotal[14].int_Total );
    fprintf( pflePtrFile, "%s%d\n", "CONVERGENCIA",
stcTotal[15].int_Total );
    fprintf( pflePtrFile, "%s%d\n", "NUEVA ALIANZA",
stcTotal[16].int_Total );

```

```
fprintf( pfilePtrFile, "%s%d\n", "PSD",
stcTotal[17].int_Total );
fprintf( pfilePtrFile, "%s%d\n", "VOTOS NULOS",
stcTotal[18].int_Total );

        fprintf( pfilePtrFile, "\nELECCION DE JEFE DELEGACIONAL\n" );
fprintf( pfilePtrFile, "%s%d\n", "PAN", stcTotal[1].int_Total
);
fprintf( pfilePtrFile, "%s%d\n", "PRI", stcTotal[2].int_Total
);
fprintf( pfilePtrFile, "%s%d\n", "PRD", stcTotal[3].int_Total
);
fprintf( pfilePtrFile, "%s%d\n", "PT", stcTotal[4].int_Total
);
fprintf( pfilePtrFile, "%s%d\n", "PVEM", stcTotal[5].int_Total
);
fprintf( pfilePtrFile, "%s%d\n", "CONVERGENCIA", stcTotal[6].int_Total
);
fprintf( pfilePtrFile, "%s%d\n", "NUEVA ALIANZA", stcTotal[7].int_Total
);
fprintf( pfilePtrFile, "%s%d\n", "PSD", stcTotal[8].int_Total
);
fprintf( pfilePtrFile, "%s%d\n", "VOTOS NULOS", stcTotal[9].int_Total
);
```

```
lintTotal=k/2;
        fprintf( pfilePtrFile, "\n NUMERO DE VOTANTES \n\n %d\n",
lintTotal );
}/*}}}}*/
```

void CfrmTotalizaVotacion::creaActaMR(FILE *pfilePtrFile)/*{{{*/ Esta función crea el acta de cómputo de votos para la elección de Diputado por Mayoría Relativa

```
{
    int lintTotal = 0;
    char chrArrBuffer[150];
    time_t now = time(NULL);
    int tmpVotos;
    //mgg
    int i=1,j=1,k=0;
    FILE *pf;
    char buffer[100];
    int p,q;
    //mgg

    for (i=0; i < 25; i++)
    {
        stcTotal[k].int_Id_Pregunta=0;
        stcTotal[k].int_Id_Opcion=0;
        stcTotal[k].int_Total=0;
        k++;
```

SP
5.

```

    }

    k=1;
    pf= fopen( "/etc/asvel/log/Vector.dat", "rt");
    fgets(buffer,sizeof(buffer),pf);
    while (!feof(pf)) {
        p= strtok(buffer,"|");
        q= strtok(NULL,"|");

        switch(atoi(p))
        {
            case 1:
                k++;
                switch(atoi(q))
                {
                    case 1:
                        stcTotal[1].int_Id_Pregunta=1;
                        stcTotal[1].int_Id_Opcion=1;
                        stcResp.int_Id_Opcion=1;
                        stcTotal[1].int_Total=stcTotal[1].int_Total+1;
                    break;
                    case 2:
                        stcTotal[2].int_Id_Pregunta=1;
                        stcTotal[2].int_Id_Opcion=2;
                        stcResp.int_Id_Opcion=2;
                        stcTotal[2].int_Total=stcTotal[2].int_Total+1;
                    break;
                    case 3:
                        stcTotal[3].int_Id_Pregunta=1;
                        stcTotal[3].int_Id_Opcion=3;
                        stcResp.int_Id_Opcion=3;
                        stcTotal[3].int_Total=stcTotal[3].int_Total+1;
                    break;
                    case 4:
                        stcTotal[4].int_Id_Pregunta=1;
                        stcTotal[4].int_Id_Opcion=4;
                        stcResp.int_Id_Opcion=4;
                        stcTotal[4].int_Total=stcTotal[4].int_Total+1;
                    break;
                    case 5:
                        stcTotal[5].int_Id_Pregunta=1;
                        stcTotal[5].int_Id_Opcion=5;
                        stcResp.int_Id_Opcion=5;
                        stcTotal[5].int_Total=stcTotal[5].int_Total+1;
                    break;
                    case 6:
                        stcTotal[6].int_Id_Pregunta=1;
                        stcTotal[6].int_Id_Opcion=6;
                        stcResp.int_Id_Opcion=6;
                        stcTotal[6].int_Total=stcTotal[6].int_Total+1;
                    break;
                    case 7:
                        stcTotal[7].int_Id_Pregunta=1;
                        stcTotal[7].int_Id_Opcion=7;
                        stcResp.int_Id_Opcion=7;
                        stcTotal[7].int_Total=stcTotal[7].int_Total+1;
                    break;
                    case 8:
                        stcTotal[8].int_Id_Pregunta=1;

```

```
        stcTotal[8].int_Id_Opcion=8;
        stcResp.int_Id_Opcion=8;
        stcTotal[8].int_Total=stcTotal[8].int_Total+1;
    break;
    case 9:
        stcTotal[9].int_Id_Pregunta=1;
        stcTotal[9].int_Id_Opcion=9;
        stcResp.int_Id_Opcion=9;
        stcTotal[9].int_Total=stcTotal[9].int_Total+1;
    break;
}

break;
case 2:

k++;

    switch(atoi(q))
    {
        case 10:
            stcTotal[10].int_Id_Pregunta=2;
            stcTotal[10].int_Id_Opcion=1;
            stcResp.int_Id_Opcion=1;
            stcTotal[10].int_Total=stcTotal[10].int_Total+1;
        break;
        case 11:
            stcTotal[11].int_Id_Pregunta=2;
            stcTotal[11].int_Id_Opcion=2;
            stcResp.int_Id_Opcion=2;
            stcTotal[11].int_Total=stcTotal[11].int_Total+1;
        break;
        case 12:
            stcTotal[12].int_Id_Pregunta=2;
            stcTotal[12].int_Id_Opcion=3;
            stcResp.int_Id_Opcion=3;
            stcTotal[12].int_Total=stcTotal[12].int_Total+1;
        break;
        case 13:
            stcTotal[13].int_Id_Pregunta=2;
            stcTotal[13].int_Id_Opcion=4;
            stcResp.int_Id_Opcion=4;
            stcTotal[13].int_Total=stcTotal[13].int_Total+1;
        break;
        case 14:
            stcTotal[14].int_Id_Pregunta=2;
            stcTotal[14].int_Id_Opcion=5;
            stcResp.int_Id_Opcion=5;
            stcTotal[14].int_Total=stcTotal[14].int_Total+1;
        break;
        case 15:
            stcTotal[15].int_Id_Pregunta=2;
            stcTotal[15].int_Id_Opcion=6;
            stcResp.int_Id_Opcion=6;
            stcTotal[15].int_Total=stcTotal[15].int_Total+1;
        break;
        case 16:
            stcTotal[16].int_Id_Pregunta=2;
            stcTotal[16].int_Id_Opcion=7;
            stcResp.int_Id_Opcion=7;
            stcTotal[16].int_Total=stcTotal[16].int_Total+1;
```

GP
5.

```

        break;
        case 17:
            stcTotal[17].int_Id_Pregunta=2;
            stcTotal[17].int_Id_Opcion=8;
            stcResp.int_Id_Opcion=8;
            stcTotal[17].int_Total=stcTotal[17].int_Total+1;
        break;
        case 18:
            stcTotal[18].int_Id_Pregunta=2;
            stcTotal[18].int_Id_Opcion=9;
            stcResp.int_Id_Opcion=9;
            stcTotal[18].int_Total=stcTotal[18].int_Total+1;
        break;
    }

    break;

}

fgets(buffer,sizeof(buffer),pf);

}
fclose(pf);
// Lectura file

tmpVotos = 0;

// Inicio de creacion de archivo
fprintf( pflePtrFile, "\n          INSTITUTO ELECTORAL\n          DEL
DISTRITO FEDERAL\n\n" );
fprintf( pflePtrFile, "  COMPROBANTE DE ESCRUTINIO Y\n
COMPUTO\n" );
fprintf( pflePtrFile, "  PROCESO ELECTORAL 2008-2009\n" );

strftime( chrArrBuffer, sizeof( chrArrBuffer ) - 1, "  %d/%m/%Y
%X \n\n", localtime( &now ) );
fprintf( pflePtrFile, "%s", chrArrBuffer );

char *chr_Distrito=regresaRomano(int_NumDistrito);
char *chr_Delegacion=regresaDelegacion(int_Delegacion);
sprintf(chrArrBuffer, "DISTRITO %s \nDELEGACION %s\nSECCION %d
CASILLA %s\nELECTORES EN LISTA NOMINAL %d\nNSUE=%s\n",chr_Distrito,
chr_Delegacion, int_NumSeccion, chr_Tipo_Casilla,int_ListaNom,
chrrArr_Serie_Urna);
fprintf( pflePtrFile, "%s\n", chrArrBuffer);

fprintf( pflePtrFile, "\n\nELECCION DE DIPUTADOS A LA ALDF\n
POR EL PRINCIPIO DE MAYORIA\n          RELATIVA\n\n");

fprintf( pflePtrFile, "%s%d\n", "PAN",
stcTotal[10].int_Total );
fprintf( pflePtrFile, "%s%d\n", "PRI",
stcTotal[11].int_Total );
fprintf( pflePtrFile, "%s%d\n", "PRD",
stcTotal[12].int_Total );

```

```
fprintf( pfilePtrFile, "%s%d\n", "PT",
stcTotal[13].int_Total );
fprintf( pfilePtrFile, "%s%d\n", "PVEM",
stcTotal[14].int_Total );
fprintf( pfilePtrFile, "%s%d\n", "CONVERGENCIA",
stcTotal[15].int_Total );
fprintf( pfilePtrFile, "%s%d\n", "NUEVA ALIANZA",
stcTotal[16].int_Total );
fprintf( pfilePtrFile, "%s%d\n", "PSD",
stcTotal[17].int_Total );
fprintf( pfilePtrFile, "%s%d\n", "VOTOS NULOS",
stcTotal[18].int_Total );
```

```
fprintf( pfilePtrFile, "\nELECCION DE JEFE DELEGACIONAL\n" );
```

```
fprintf( pfilePtrFile, "%s%d\n", "PAN", stcTotal[1].int_Total );
fprintf( pfilePtrFile, "%s%d\n", "PRI", stcTotal[2].int_Total );
fprintf( pfilePtrFile, "%s%d\n", "PRD", stcTotal[3].int_Total );
fprintf( pfilePtrFile, "%s%d\n", "PT", stcTotal[4].int_Total );
fprintf( pfilePtrFile, "%s%d\n", "PVEM", stcTotal[5].int_Total );
fprintf( pfilePtrFile, "%s%d\n", "CONVERGENCIA", stcTotal[6].int_Total );
fprintf( pfilePtrFile, "%s%d\n", "NUEVA ALIANZA", stcTotal[7].int_Total );
fprintf( pfilePtrFile, "%s%d\n", "PSD", stcTotal[8].int_Total );
fprintf( pfilePtrFile, "%s%d\n", "VOTOS NULOS", stcTotal[9].int_Total );
```

```
lintTotal=k/2;
```

```
fprintf( pfilePtrFile, "\n NUMERO DE VOTANTES \n\n %d\n",
lintTotal );
```

```
/*}}*/
```

void CfrmTotalizaVotacion::creaActaRP(FILE *pfilePtrFile)/*{{*/ Esta función crea el acta de cómputo de votos para la elección de Diputado por Representación Proporcional

```
{
    int lintTotal = 0;
    char chrArrBuffer[150];
    time_t now = time(NULL);
    int tmpVotos;
    //mgg
    int i=1,j=1,k=0;
    char buffer[100];
    char *p;
    //mgg
```

```
static char *Romano_Dis[]={
```

```

"Urna de Contingencia",

"I","II","III","IV","V","VI","VII","VIII","IX","X","XI","XII","XIII","XIV"
,"XV","XVI","XVII","XVIII",

"XIX","XX","XXI","XXII","XXIII","XXIV","XXV","XXVI","XXVII","XXVIII","XXIX"
,"XXX","XXXI","XXXII",
"XXXIII","XXXIV","XXXV","XXXVI","XXXVII","XXXVIII","XXXIX","XL"
};

for (i=0; i < 25; i++)
{
    stcTotal[k].int_Id_Pregunta=0;
    stcTotal[k].int_Id_Opcion=0;
    stcTotal[k].int_Total=0;
    k++;
}

// Inicio de creacion de archivo
fprintf( pfilePtrFile, "\n          INSTITUTO ELECTORAL\n          DEL
DISTRITO FEDERAL\n\n" );
fprintf( pfilePtrFile, "          COMPROBANTE DE URNA \n
ELECTRONICA VACIA\n" );
fprintf( pfilePtrFile, " PROCESO ELECTORAL 2008-2009\n" );

strftime( chrArrBuffer, sizeof( chrArrBuffer ) - 1, " %d/%m/%Y
%X \n\n", localtime( &now ) );
fprintf( pfilePtrFile, "%s", chrArrBuffer );

char *chr_Distrito=regresaRomano(int_NumDistrito);
char *chr_Delegacion=regresaDelegacion(int_Delegacion);
sprintf(chrArrBuffer, "DISTRITO %s \nDELEGACION %s\nSECCION %d
CASILLA %s\nELECTORES EN LISTA NOMINAL %d\nNSUE=%s\n",chr_Distrito,
chr_Delegacion, int_NumSeccion, chr_Tipo_Casilla,int_ListaNom,
chrArr_Serie_Urna);
fprintf( pfilePtrFile, "%s\n", chrArrBuffer);

fprintf( pfilePtrFile, "\n\nELECCION DE DIPUTADOS A LA ALDF\n
POR EL PRINCIPIO DE MAYORIA\n          RELATIVA\n\n");

fprintf( pfilePtrFile, "%s%d\n", "PAN          ", stcTotal[10].int_Total
);
fprintf( pfilePtrFile, "%s%d\n", "PRI          ", stcTotal[11].int_Total
);
fprintf( pfilePtrFile, "%s%d\n", "PRD          ", stcTotal[12].int_Total
);
//
fprintf( pfilePtrFile, "%s%d\n", "PT          ", stcTotal[13].int_Total
);

```

SP
5.

```
fprintf( pfilePtrFile, "%s%d\n", "PVEM", stcTotal[14].int_Total
);

fprintf( pfilePtrFile, "%s%d\n", "CONVERGENCIA", stcTotal[15].int_Total
);

fprintf( pfilePtrFile, "%s%d\n", "NUEVA ALIANZA", stcTotal[16].int_Total
);
fprintf( pfilePtrFile, "%s%d\n", "PSD", stcTotal[17].int_Total
);
fprintf( pfilePtrFile, "%s%d\n", "VOTOS NULOS", stcTotal[18].int_Total
);

                fprintf( pfilePtrFile, "\n\n\nELECCION DE JEFE
DELEGACIONAL\n\n\n" );

fprintf( pfilePtrFile, "%s%d\n", "PAN", stcTotal[1].int_Total
);
fprintf( pfilePtrFile, "%s%d\n", "PRI", stcTotal[2].int_Total
);
fprintf( pfilePtrFile, "%s%d\n", "PRD", stcTotal[3].int_Total
);
fprintf( pfilePtrFile, "%s%d\n", "PT", stcTotal[4].int_Total
);
fprintf( pfilePtrFile, "%s%d\n", "PVEM", stcTotal[5].int_Total
);
fprintf( pfilePtrFile, "%s%d\n", "CONVERGENCIA", stcTotal[6].int_Total
);
fprintf( pfilePtrFile, "%s%d\n", "NUEVA ALIANZA", stcTotal[7].int_Total
);
fprintf( pfilePtrFile, "%s%d\n", "PSD", stcTotal[8].int_Total
);
fprintf( pfilePtrFile, "%s%d\n", "VOTOS NULOS", stcTotal[9].int_Total
);

        fprintf( pfilePtrFile, "\n  NUMERO DE VOTANTES  \n\n      %d\n",
lintTotal );

/*}}}}*/

int  CfrmTotalizaVotacion::verifyFile( char *pchrPtrFile )/*{{{*/
{
    struct stat stcStatDat;

    if ( stat( pchrPtrFile, &stcStatDat ) == -1 )
        if ( errno == ENOENT )
            return 0;

    return 1;
/*}}}}*/

char* CfrmTotalizaVotacion::regresaRomano(int int_Distrito){
    char *Romano_Dis[]={

        "I","II","III","IV","V","VI","VII","VIII","IX","X","XI","XII","XIII","XIV",
        "XV","XVI","XVII","XVIII",

        "XIX","XX","XXI","XXII","XXIII","XXIV","XXV","XXVI","XXVII","XXVIII","XXIX",
        "XXX","XXXI","XXXII",
```

```
"XXXIII", "XXXIV", "XXXV", "XXXVI", "XXXVII", "XXXVIII", "XXXIX", "XL"
};
return Romano_Dis[int_Distrito-1];

}

char* CfrmTotalizaVotacion::regresaDelegacion(int int_Delegacion){
    char *Delegacion[]={"AZCAPOTZALCO", "COYOACAN", "CUAJIMALPA",
    "GUSTAVO A. MADERO", "IZTACALCO", "IZTAPALAPA", "MAGDALENA CONTRERAS", "MILPA
    ALTA", "ALVARO OBREGON", "TLAHUAC", "TLALPAN", "XOCHIMILCO", "BENITO JUAREZ",
    "CUAUHTEMOC", "MIGUEL HIDALGO", "VENUSTIANO CARRANZA"
    };
    return Delegacion[int_Delegacion-2];
}

char *CfrmTotalizaVotacion::findParticipante( int pint_Id_Participante )/*{{{*/
Esta función regresa el nombre de los participantes (Partidos políticos) asociado al identificador
{
    for ( int lintX = 0; lintX < intNumPar; lintX++ )
        if ( pint_Id_Participante == (stcParPtrDat + lintX)->int_Id_Partido
    )
        return (stcParPtrDat + lintX)->chrArr_Siglas;
    return 0;
}/*}}}*/
```